A Formalism and an Algorithm for Computing Pragmatic Inferences and Detecting Infelicities

by

Daniel Marcu

Department of Computer Science
University of Toronto
Toronto, Canada
September 1994

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Copyright © 1994 by Daniel Marcu

Abstract

Since Austin introduced the term *infelicity*, the linguistic literature has been flooded with its use. Today, not only performatives that fail are considered infelicitous but also utterances that are syntactically, semantically, or pragmatically ill-formed. However, no formal or computational explanation has been given for infelicity. This thesis provides one for those infelicities that occur when a pragmatic inference is cancelled.

We exploit a well-known difference between pragmatic and semantic information: since implicatures and presuppositions, i.e., the carriers of pragmatic information, are not specifically uttered, pragmatic inferences are defeasible, while most of semantic inferences are indefeasible. Our contribution assumes the existence of a finer grained taxonomy with respect to pragmatic inferences. It is shown that if one wants to account for the natural language expressiveness, she should distinguish between pragmatic inferences that are felicitous to defeat and pragmatic inferences that are infelicitously defeasible. Thus, it is shown that one should consider at least three types of information: indefeasible, felicitously defeasible, and infelicitously defeasible. The cancellation of the last of these determines the pragmatic infelicities.

A new formalism has been devised to accommodate the three levels of information, called *stratified logic*. Within it, we are able to express formally notions such as *utterance u presupposes p* or *utterance u is infelicitous*. Special attention is paid to the implications that our work has in solving some well-known existential philosophical puzzles. The formalism yields an algorithm for computing interpretations for utterances, for determining their associated presuppositions, and for signalling infelicities. Its implementation is a Lisp program that takes as input a set of stratified formulas that constitute the necessary semantic and pragmatic knowledge and the logical translation of an utterance or set of utterances and that computes a set of *optimistic* interpretations for the given utterances. The program computes for each set of utterances the associated presuppositions and signals when an infelicitous sentence has been uttered.

Acknowledgments

First, I thank my supervisor, Graeme Hirst, for his competent guidance, patience, and humor in explaining and re-explaining so many things; and confidence that I can delineate and solve problems on my own. I also thank him for the way he taught me to weight and glue words into sentences, sentences into paragraphs, to put down ideas in a way that other people can understand them. The flaws in this thesis show that I still have a long way to go until I come to know well everything he taught me.

I want to thank Hector Levesque, my second reader, for the helpful comments he gave me. I thank Hector Levesque and Ray Reiter for teaching me how to tackle a logical problem and how to find formal explanations for the phenomena around us. I thank Jeff Siskind for reminding me that the ultimate proof is a program that *does* something.

I thank my colleagues in the natural language, cognitive robotics, and knowledge representation groups who shared with me their questions, problems, opinions, and thoughts.

I am grateful for the financial support I have received from the University of Toronto and from the Natural Sciences and Engineering Research Council of Canada.

Because a thesis is not only words, programs, or formulas I thank Ed Klajman for being the friend with whom I was able to get into a world free of syntactic, semantic, or pragmatic constraints.

I thank my parents for everything they have taught me, for the understanding and trust they have shown me every day since I have started this work.

But most of all, I want to thank my wife, Oana, for her endless love, support, and strength in believing that one day, we will be again together, in a better world, every single minute.

Contents

List of Figures

Chapter 1

Pragmatic inferences, infelicities, and logic

Much of the research done in linguistic pragmatics makes extensive use of the term *infelicity*. However, it seems that no approach offers a formal or computational definition for this phenomenon. It is the goal of this thesis to provide one. Unfortunately, the scope of the phenomenon of infelicity is extremely large and not well-defined. Apparently, the word was introduced by Austin [1962] to characterize performative utterances that fail. But since then, it has become fashionable to characterize anything that goes wrong as infelicitous. If a sentence is not syntactically or semantically well-formed it is often said to be infelicitous; if an utterance is not pragmatically well-formed it is said to be infelicitous too. This thesis is about pragmatic infelicities: those that occur when some Gricean inferences are cancelled.

In this chapter we review the relevant work in implicature and presupposition from a perspective concerned primarily with aspects of felicity. The aim of the first two sections is to familiarize the reader with the vocabulary and to give her a *feeling* of what pragmatic infelicities are. The third section analyzes pragmatic inferences in detail and shows why it is inappropriate to equate infelicity to inconsistency. The last section studies representative approaches to default reasoning and shows that none of them is able to formalize adequately the infelicity phenomenon.

In order to capture pragmatic infelicities, we introduce a new formalism called *stratified logic*. Chapter 2 contains the informal intuitions that lie behind this new mathematical tool, and a characterization of its syntax and semantics for the propositional and first-order case.

Chapter 3 sheds some light on a fundamental difference between the notions of lexical semantics and lexical pragmatics. It is shown that an appropriate semantic and pragmatic characterization of the lexical items and syntactical constructs in terms of stratified logic not only provides a formal characterization for pragmatic infelicities but also an adequate explanation for cancellations that implicatures and presuppositions are subject to. Chapter 4 extends this work and shows how the solution developed for simple utterances is appropriate without any modifications for handling complex utterances. Thus, we obtain a formalism and a computational method that signal infelicitous utterances and determine the presuppositions that are associated with an utterance independent of its complexity. Our method works even for sequences of utterances, a case that is omitted by previous research in the area.

The existential presuppositions carried by definite references are one of the most problematic linguistic and philosophical issues. Chapter 5 is dedicated to a review of the work done in this area. It is shown that most of the philosophical approaches are more concerned in dealing with nonexistence and they fail to capture the existential presuppositions, while most of the linguistic approaches are incapable of accommodating nonexistent entities. We use a rich ontology à la Hirst and a set of methodological principles that embed the essence of Meinong's philosophy and Grice's conversational principles into a stratified logic, under an unrestricted interpretation of the quantifiers. The result is a logical formalism that yields a tractable computational method that uniformly calculates all the presuppositions of a given utterance, including the existential ones. A comparison with Parsons's, Russell's, and Hobbs's work emphasizes the superiority of our approach.

Chapter 6 is concerned with some implementation issues. Two annexes are provided: one that contains the Lisp code of our implementation, and one that contains the results computed by our program on a representative set of examples.

1.1 Implicatures

Although Grice [1975, 1978] sketched his theory of logic and conversation only briefly, his ideas had an enormous impact on the research on pragmatics. Grice wanted to avoid any ambiguity between what is said in an utterance and the implications, intentions, or beliefs that are used to interpret the utterance, or that are conveyed by it. The beauty of his theory emanates from a very small set of conversational principles or maxims that are the linguistic expression of a more general form of human cooperative behavior [Grice, 1975, p. 47]. Here follow these principles:

<u>Maxims of quality</u>

Try to make your contribution one that is true.

1. Do not say what you believe to be false.

2. Do not say that for which you lack adequate evidence.

Maxims of quantity

1. Make your contribution as informative as is required (for the current purposes of the exchange).
2. Do not make your contribution more informative than is required[1].

Maxim of relation

1. Be relevant.

Maxims of manner

1. Avoid obscurity of expression.
2. Avoid ambiguity.
3. Be brief (avoid unnecessary prolixity).
4. Be orderly.

By observing these maxims, one is able to explain inferences that do not belong to the conventional meaning of an utterance. These inferences are called *conversational implicatures*. It is important to notice that Gricean maxims are not intended to be prescriptive rules that conversants must obey. Rather, they are default rules of the conversants' intentions or knowledge. At the end of his paper, Grice [1975, pp. 57–58] provides a list of the features he attributes to conversational implicatures:

1. They are cancellable.
2. They are non-detachable, i.e., it will be impossible to find a way of saying the same thing that lacks the corresponding implicature.
3. They are not part of the semantic meaning of the linguistic expressions to which they attach.
4. They are carried by the saying of what is said, rather than by what is said.
5. Usually, each implicature is a disjunction of possible specific explanations in a given context.

[1]The validity of this maxim is disputable, because one may be enticed to subsume it under the maxim of relevance; this is admitted even by Grice himself. General inquiries into this matter are made by Green [1989]. A study related to its impact on redundancy in collaborative dialogues is elaborated by Walker [1992].

The first and third property are those that most favours their cancelability or defeasibility. But if one examines implicatures more carefully, she will notice that sometimes it is *felicitous* to cancel or defeat them, while at other times it is not.

Following Grice and Levinson [1983], one may distinguish four categories of implicatures: generalized, particularized, floating, and conventional implicatures.

Generalized implicatures are ones which do not require a particular context in order to be inferred. The following generalized implicatures can be inferred if the corresponding maxim is observed.

Quality

(**1.1**) John bought a new car.

(**1.2**) $\triangleright$ I believe he did it and I have adequate evidence for this.

Notice that cancelling a quality implicature is infelicitous:

(**1.3**) $\star$ John bought a new car, but I don't believe it.

Quantity

(**1.4**) Jane has three exams this term.

(**1.5**) $\triangleright$ Jane has *only* three exams this term.

Although it would be compatible with the truth of the above utterance that Jane has four exams, the implicature is that Jane has *only* three exams because if she had had four (or more), then by the maxim of quantity (say as much as is required) one should have said so. However, one can felicitously defeat the implicature if she utters:

(**1.6**) Jane has three exams this term; in fact, she has four.

Relevance

(**1.7**) Give me that book, please.

(**1.8**) $\triangleright$ Give me that book now.

In this case, the implicature can be felicitously defeated if one utters:

(**1.9**) Give me that book when you finish it.

Manner (order)

(**1.10**) Ed went to the airport and flew to California.

(**1.11**) $\triangleright$ Ed performed the above actions in the given order.

Cancelling the above implicature is infelicitous:

(**1.12**) $\star$ Ed went to the airport and flew to California, but he went to California first.

Particularized implicatures are valid only in a particular context. For example, one would normally infer

(**1.13**) $\triangleright$ Perhaps the dog ate the roast beef.

from

(**1.14**) The dog looks happy.

only in a context where (1.14) was uttered as an answer to

(**1.15**) Where is the roast beef? [Horton, 1987, p. 2]

This particularized implicature can be felicitously cancelled if instead of (1.14), one utters:

(**1.16**) The dog looks happy, but it was here all the time.

Other kinds of implicature come about by blatantly not following some maxims. These give the *floating implicatures*. For example, by answering with

(**1.17**) **A:** My car's not working.

to a question such as

(**1.18**) **B:** Can I get a ride with you? [Green, 1989, p. 92]

one presumably implicates that because his car is not working, he cannot provide a ride for **B**. However, the implicature is felicitously defeasible if **A** utters:

(**1.19**) **A:** My car's not working, but I'll take my wife's.

Further work in the field of quantity implicatures [Horn, 1972, Gazdar, 1979] gave rise to two important sub-cases of what today are called *conventional implicatures*.

The first is *scalar implicatures*. For any linguistic scale arranged in a linear order by degree of informativeness or semantic strength $\langle e_1, e_2, \ldots, e_n \rangle$, the utterance of a well-formed sentence $A(e_i)$ will implicate $\neg A(e_{i-1})$, $\neg A(e_{i-2})$, ..., $\neg A(e_1)$. For example on the scale $\langle all, most, many, some, few \rangle$, the utterance of

(**1.20**) John says that some of the boys went to the theatre.

will implicate the following conventional implicatures:

(1.21) $\triangleright$ Not many/most/all of the boys went to the theatre.

Normally, conventional implicatures are felicitously defeasible. Consider the following repair for the previous example:

(1.22) John says that some of the boys went to the theatre. John, Mike, and Jeff were there. Fred was there too. In fact *all* of them went to the theatre.

The second type of conventional implicature, the *clausal implicature* was formulated by Gazdar [1979]. For example, if one utters

(1.23) John believes Margaret to be unfaithful.

the clausal implicatures are:

(1.24) $\triangleright$ It is possible for Margaret to be unfaithful.

and

(1.25) $\triangleright$ It is possible for Margaret to be faithful.

It seems that cancelling the clausal implicatures is felicitous:

(1.26) John believes Margaret to be unfaithful, but it is impossible for her to be like this.

There are a number of attempts to accommodate Grice's conversational principles with syntactic and semantic theories. For example, Gordon and Lakoff [1975] try to formalize the conversational principles, to incorporate them into a theory of generative semantics, and to determine if there are rules of grammar that depend on these conversational principles. The conversational implicatures are defined in terms of logical entailment derived from a logical translation of the Gricean maxims, a class of contexts, and the given utterance. Logical entailment is essentially monotonic. Therefore it cannot explain the cancelability of some implicatures. However, Gordon and Lakoff's suggestion that an appropriate formalism should distinguish between relations such as entailment, equivalence, presupposition, assumption, and conversational entailment would be valuable for our purposes.

Hirschberg is interested in a theory of conversational implicature and provides a set of criteria for recognizing them [1985, p. 38]. Green [1990, 1992] uses Hirschberg's theory as the foundation for the introduction of another type of implicature, *normal state implicature*, and for the study of implicature in indirect replies. Normal state implicature is used to offer an explanation for the circumscription and qualification problems on linguistic grounds, but her solution circumvents many important aspects, such as the representation of the preconditions of a plan, or the recognition by the hearer of the speaker's plans. We believe that in most of the cases, is unreasonable to assume that a hearer is aware not only of the Gricean conversational principles and the conversants' beliefs but also of the speaker's plans in order to infer the expected implicatures.

A good analysis of the interaction between the conversational principles and a natural language generator is presented by Reiter [1990]. His concern is the generation of natural language utterances that are free of false implicatures, i.e., those that are maximal under the rules of local brevity, unnecessary components, and lexical preference. The proposed algorithms prevent the generation of a sentence such as *Sit by the brown wooden table* when *Sit by the table* would suffice. Complexity results for the various algorithms are also established.

Dealing with implicatures is not purely a theoretical issue: understanding them can substantially improve many day-to-day tasks, such as interrogating a database. A computational account of Grice's conversational principles with respect to a natural language database interface is presented by Kaplan [1982]. The interface is supposed to provide co-operative responses for the queries. For questions such as *Which students got a grade of F in CS105 in Spring 1980?* there may be situations in which it will be better to answer *CS105 was not given in Spring 1980* instead of *nil.*

The examples we have given show a clear delimitation between the implicatures that are felicitous to cancel and those that are not. However, we are not aware of any research concerning implicatures that tries to exploit this difference. Such an approach seems interesting to explore because the same dichotomy can be found when one studies presuppositions: some of them are felicitous to defeat, while others are not.

1.2 Presuppositions

1.2.1 What are presuppositions?

The main property that seems to delineate presuppositions from implicatures is the fact that they are *implied* in both positive and negative environments. It is common knowledge that utterances *a* and *b* both presuppose *c* in the following two examples.

(1.27) a. Peter regrets that he got a speeding ticket.
b. Peter does not regret that he got a speeding ticket.
c. Peter got a speeding ticket.

(1.28) a. The King of Buganda is bald.
b. The King of Buganda is not bald.
c. There exists a king of Buganda.

Soames [1989] proposes two classes of questions that are supposed to be answered by any theory of presupposition. Analogous questions should be addressed by a theory of implicature as well.

Foundational questions

1. What is presupposition — what does it mean to say X presupposes Y?
2. Why are there linguistically expressed presuppositions at all — what functions do presuppositions have in the representation and communication of information?
3. How are presuppositions of utterances affected by the semantic rules that determine the information encoded by a sentence relative to its context, and the pragmatic rules that specify the manner in which utterances increment sets of assumptions common among conversational participants?

   Descriptive questions

4. What presuppositions do various constructions give rise to?
5. Which constructions allow utterances to inherit the presuppositions of their constituents and which do not? This question is often called the *projection problem.*
6. What do utterances of arbitrary sentences presuppose?

Three different classes of answers have been given for the first question. The first is due to Frege [1892], who sees presupposition as a logical relation between two sentences. This category contains what are usually called semantic theories. Semantic theories are usually built on a definition that captures the fact that presuppositions are implied in both positive and negative environments, and that is similar to the one given below,

Definition 1.2.1 *Sentence S_1 semantically presupposes sentence S_2 if and only if $S_1 \models S_2$ and $\neg S_1 \models S_2$.*

This definition does not survive a logical examination because as Gazdar [1979, p. 90] has shown, the definition says that presuppositions *are always* true. Several variations on this definition have been developed using modal logic [Karttunen, 1971] or a heavy-parentheses notation [Katz and Langendoen, 1976]. A critique of these theories is given by Gazdar [1979, pp. 90–103]. Revivals of semantic theories are observed even later [Wilson and Sperber, 1979], where in order to circumvent the problems an ordered relation on the entailments is used. On the basis of this ordering, Sperber and Wilson distinguish between *focalized* and *peripheral* entailments. A critique of their proposal is given by Horton [1987, pp. 27–28].

The second class of answers is related mainly to Strawson [1950], who sees presupposition as a relation between a sentence and its use. In a Strawsonian world, the existence of the King of Buganda is a necessary precondition that should be satisfied before any truth value can be assigned to the sentence *The King of Buganda is bald.*

The third class of answers, the one that encompasses the pragmatic approaches, is the one that looks most successful. Pragmatic approaches offer a variety of definitions for presupposition that differ primarily in the role played by context [Karttunen, 1974], by the conversational participants [Stalnaker, 1973] or by the logic and pragmatic relation between these factors and presupposition [Gazdar, 1979, Horton, 1987]. We shall look at several of these approaches in detail in the following sections.

1.2.2 Karttunen and Peters

Karttunen and Peters's evolution of understanding in dealing with pragmatic inferences reflects a desire for simplification: in their 1979 paper for example, an important class of inferences is labelled not as presuppositions, as in their earlier work, but rather as implicatures. The inferences that are triggered by the use of word *even* are an example. We think this reflects the idea that presupposition and implicature do not differ too much in their intrinsic status. Essentially, they both constitute information added to the context beyond what is specifically uttered. Furthermore, they are both cancellable: sometimes felicitously, sometimes infelicitously.

Karttunen and Peters's theory addresses primarily the projection problem by introducing the *plugs, holes,* and *filter* mechanism. Plugs are constructs that do not allow the presuppositions of the underlying structure to become presuppositions of the matrix sentence. The verbs of saying (*say, ask, tell,* etc.) are categorized as plugs:

(**1.29**) Bill told me that the present King of France is bald.

(**1.30**) $\triangleright$ There is a king of France.

Holes are those constructs in which presuppositions always survive embedding. Factives (*know, regret*), aspectuals (*begin, stop*), implicatives (*manage, remember*) are the most common examples:

(**1.31**) John managed to open the door.

(**1.32**) $\triangleright$ John opened the door.

The filters are the constructs in which presuppositions act strangely: they sometimes survive and sometimes fail to survive. The filtering conditions provide the rules by which presuppositions are inherited. For example, a conditional having the form *if A then B* will presuppose P if B presupposes P unless there is some set X of assumed facts such that $X \cup \{A\}$ semantically entails P. According to this rule, (1.33) will presuppose (1.34), but (1.35) will not presuppose (1.36).

(**1.33**) If Mary came to the party, then John would regret that Sue came to the party.

(1.34) $\triangleright$ Sue came to the party.

(1.35) If Mary came to the party, then John would regret that she did.

(1.36) $\not\triangleright$ Mary came to the party.

A formalization of these rules is developed in Montaque's PTQ grammar.

The theory has been attacked by Gazdar [1979] and Soames [1979, 1982] for reasons related mainly to contradictory presuppositions such as that exhibited in *My teacher is a bachelor or a spinster*, and presuppositions cancelled by conversational implicatures, e.g. *It is possible that John has children and it is possible that his children are away.* Another problem is a methodological one: it is very difficult to construct a taxonomy of plugs, filters, and holes because the criteria needed to define these notions are ambiguous.

1.2.3 Gazdar

A major step taken by Gazdar [1979] is the shift from explaining presuppositions in terms of entailment towards explaining them in terms of consistency. For Gazdar, implicatures and presuppositions work together towards the enhancement of context. His theory is an elegant exploration of how pragmatic rules can explain the inheritance of presupposition. Gazdar's system is composed of three parts:

1. A set of presupposition-bearing functions, which, when applied to a sentence, generate all the presuppositions afferent to that sentence.

2. A trivial projection rule that projects all the presuppositions generated in part 1 as the potential presuppositions for the sentence.

3. A Gricean filter that cancels some of the potential presuppositions.

The critics of his theory were mostly concerned with the ad-hoc order in which the conversational context is enhanced, and the lack of grounds for this. However, they do not provide an insight into the logical properties that Gazdar's presupposition has. Mercer [1987] and Horton [1987] note that Gazdar's method does not allow presuppositions to be cancelled by knowledge added to the context by later utterances. This gives presuppositions a very interesting status, assigning them a dual life: they are cancellable in the first stage (the stage of their generation and their evaluation against the current context) but they become indefeasible knowledge in the second stage, after they have been accepted as presuppositions for the given utterance. Consider that one utters (1.37). The first sentence in the sequence presupposes (1.38). Gazdar's theory will appropriately reflect this. The second sentence in (1.37) comes to cancel the presupposition but Gazdar's theory is not able to provide an explanation for this cancellation.

(1.37) My cousin is not a bachelor. He is only five years old.

(1.38) $\not\triangleright$ ($\triangleright$) My cousin is an adult.

In these conditions, it becomes clear why the presuppositions are the last pieces of information that are added to the conversational context.

From our perspective, of maximal importance is the fact that Gazdar formalizes presuppositions, in positive environments, as logical implications [Gazdar, 1979, p. 140]. As we will show in section 1.3, this is a logical mistake.

1.2.4 Mercer

An orthogonal approach is taken by Mercer [1982, 1987, 1988b, 1988a, 1990, 1991]. He abandons the projection method for rules of inference in default logic. Mercer uses Gazdar's formalization of Grice's cooperative principle, one of the basic pillars of all pragmatic approaches, and he provides a proof-theoretic definition for presupposition in terms of default inferences. Our main objection to this approach is Mercer's use of natural disjunction as an exclusive disjunction, and the reduction of natural implication to logical equivalence. I present below our counter-arguments to the disjunctive treatment. For conditionals, the counter-arguments are similar.

Mercer [1987] argues that when a speaker utters a sentence in the form $A \vee B$, assuming the speaker respects Grice's conversational principles, and considering Gazdar's formalization of clausal implicatures, the corresponding default theory representing this utterance will be:

$$T = \{K_S(A \vee B), P_S A, P_S \neg A, P_S B, P_S \neg B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\} \tag{1.1}$$

where $\alpha_1, \ldots, \alpha_n$ represent the appropriate first-order statements, $\delta_1, \ldots, \delta_n$ represent the appropriate default rules, and K_S, P_S represent the common modal operators for necessity and possibility respectively. Using the S_4 modal logic system, Mercer argues that theory T yields two cases:

$$T_1 = \{A \wedge \neg B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\} \tag{1.2}$$

and

$$T_2 = \{\neg A \wedge B, \alpha_1, \ldots, \alpha_n, \delta_1, \ldots, \delta_n\} \tag{1.3}$$

since $A \wedge \neg B$ and $\neg A \wedge B$ *completely determine* the truth value of both A and B and since the statements $P_S(A \wedge \neg B)$ and $P_S(\neg A \wedge B)$ can be derived from the given theory T.

However, the following proof gives us $P_S(A \wedge B)$ as a candidate for the case analysis as well:

1.	$K_S(A \vee B) \rightarrow P_S A \vee K_S B$	(provable in S_4 modal logic [Chellas, 1980, p. 123])
2.	$K_S(A \vee B)$	(translation of speaker's utterance)
3.	$A \vee B \leftrightarrow (\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B)$	(PL)
4.	$K_S((A \vee B) \leftrightarrow (\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B))$	(3 + RN)
5.	$K_S(A \vee B) \leftrightarrow K_S((\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B))$	(4 + RE)
6.	$K_S((\neg A \wedge B) \vee (A \wedge \neg B) \vee (A \wedge B))$	(2 + 5 + MP)
7.	$P_S(\neg A \wedge B) \vee K_S((A \wedge \neg B) \vee (A \wedge B))$	(1 + 6 + MP)
8.	$P_S(\neg A \wedge B) \vee P_S(A \wedge \neg B) \vee K_S(A \wedge B)$	(1 + 7 + MP)
9.	$K_S(A \wedge B) \rightarrow P_S(A \wedge B)$	(T + PL)
10.	$P_S(\neg A \wedge B) \vee P_S(A \wedge \neg B) \vee P_S(A \wedge B)$	(8 + 9 + MP)

As shown, under a classical interpretation, the above theory yields three cases. Mercer does not include in his theory the scalar implicatures given by Gazdar [1979], which eventually eliminates the third case $P_S(A \wedge B)$. Therefore, his treatment of natural disjunction as an exclusive or cannot be explained on formal grounds. Mercer (personal communication) argued that he intended his "proof by cases" to be interpreted in a non-traditional way, in which "the cases are taken from a conjunctive statement, where the conjuncts are the disjuncts in the above proof". He argued that this non-standard notion is the one that must be used in nonmonotonic reasoning. If we restrain ourselves to a classical interpretation, the foundations of Mercer's theory collapse: his definitions do not reflect any longer one's intuitions.

1.2.5 Other approaches

Horton's theory [1987, 1988] has as its departure point the observation that many theories of presupposition make the following unrealistic assumptions:

- If sentence S presupposes proposition P, then P is true.
- If sentence S presupposes proposition P, then all agents involved share the prior belief that P is true.

Most of the intuitions provided by Horton sound acceptable, but when it comes to their formalization, insurmountable problems occur. The core of the system is built on two operators: *add-presupposition* and *retract-presupposition*, but they are never defined in terms of beliefs. Moreover, no hints are given concerning the conditions under which these operators are triggered.

A more recent approach to the projection problem is built from the observation that the data challenging Gazdar's solution introduced by Soames, Heine, and Fauconnier concern primarily hypothetical contexts and secondarily contexts of reported speech or thought. A theory based on these types of contexts is developed [Kay, 1992] but no references are made for a possible formalism in which such a theory can be embedded, nor for a computational model of it.

A different perspective is given by van der Sandt [1992] and Zeevat [1992] where presuppositions are understood as anaphoric expressions that have internal structure and semantic content. Because they have more semantic content than other anaphors, presuppositions are able to create an antecedent in the case that the discourse does not provide one. Van der Sandt provides a computational method for presupposition resolution in an enhanced discourse representation theory while Zeevat gives a declarative account for it using update semantics, but none of the methods is able to accommodate the cancellation of presupposition that is determined by information added later to the discourse.

There is only one attempt to embed the theories of presupposition within a working computational model. It was done by Weischedel [1979] and it claims that presuppositions are purely structural in nature, and can thus be generated during parsing. The claim is supported by an extension of an augmented transition network grammar with actions capable of generating presuppositions. Being so strongly bound to the lexical and syntactic structure for a particular utterance, the theory is not able to explain presupposition cancellations that occur late in the analyzed text. Actually none of these theories is able to account for the cancellation of the presupposition generated in (1.37).

1.3 A logical analysis of pragmatic inferences

The meaning, the implications, and the intention behind words are just one part of our day-to-day experience that makes the study of language difficult. We have already presented the main theories developed for handling implicatures and presuppositions. We review some of the examples from a logical perspective.

Consider the following sentence, which exhibits an entailment relation:

(**1.39**) All turtles are green.

This is the kind of statement that is representable in first-order logic. In accordance with a Tarskian semantics, the translation $(\forall x)(turtle(x) \rightarrow green(x))$ will allow one to infer from $turtle$(Wolfgang) that $green$(Wolfgang). By contraposition, anything that is not green cannot be a turtle:

(**1.40**) If something is not green, then it is not a turtle.

As we will show, many pragmatic inferences are not subject to this behavior, i.e., they do not obey the contrapositive rule, so they are not representable in first-order logic using material implication.

It has been already shown [Gazdar, 1979, p. 90] that a definition of presuppositions in terms of entailment is a logical mistake. To fix the problem, one would be tempted to treat presuppositions as defeasible information (see 1.2.4 for such an approach). Assume that presuppositions are defeasible in positive and negative environments as well. If this is the case, the behavior of the related presupposition (1.43) should be the same for each of the following utterances:

(**1.41**) John regrets that Mary came to the party.

(**1.42**) John does not regret that Mary came to the party.

(**1.43**) $\triangleright$ Mary came to the party.

This is not the case. Negating the presupposition is infelicitous for the first utterance, but is acceptable for the second one:

(**1.44**) $\star$ John regrets that Mary came to the party, but she did not come.

(**1.45**) John does not regret that Mary came to the party, because she did not come.

(**1.46**) $\not\triangleright$ Mary came to the party.

It is true that the surface form of a positive utterance is similar to its negation, but the inferences derived from these utterances have totally different properties, so they should be studied separately.

The same dichotomy occurs when one studies implicatures. Sometimes implicatures are felicitous to defeat, as in (1.47) or (1.48), but sometimes their cancellation is infelicitous as in (1.49).

(**1.47**) Ed has three exams this term. Actually he has four.

(**1.48**) The soup is warm, in fact hot.

(**1.49**) $\star$ Mary is unfaithful, but I don't believe this.

We exploit a fundamental difference between semantic and pragmatic information. On one hand, we consider that semantic information is information that pertains to our knowledge of the world. It may come in two flavours: indefeasible, as given in sentence (1.39), or defeasible, as given in (1.50).

(**1.50**) Typically, birds fly.

As a simplification, for the purpose of this thesis, we will deal only with indefeasible semantic knowledge, but the formalism and program we propose can be generalized to handle semantic defeasible information as well.

On the other hand, pragmatic information is information concerning our knowledge of language use. Presuppositions and implicatures are not explicitly uttered; therefore, *all* of them are defeasible. The examples we have given with respect to implicatures and presuppositions inspires us to consider more than one level of defeasibility for pragmatic inferences. The refinement we propose accounts for the different strength or commitment that seems to differentiate certain types of pragmatic inferences. When an implicature or a presupposition is infelicitous to defeat, we say that that implicature or presupposition is *infelicitously defeasible*. When an implicature or presupposition is felicitous to defeat, we say that that pragmatic inference is *felicitously defeasible*. The following table shows the defeasibility properties of implicatures and presuppositions in positive and negative environments:

Property	*Implicature*	*Presupposition*
Felicitously defeasible in positive sentences	yes/no	no
Felicitously defeasible in negative sentences	yes/no	yes

The difference between felicitously and infelicitously defeasible implicatures and presuppositions, which we have emphasized so far, determines us to attempt to build a taxonomy of pragmatic inferences that has a finer granularity. We still subscribe to the Gricean view that *all* conversational implicatures are defeasible, but we distinguish between the ones that are felicitous to cancel and those that are not. As shown, the taxonomy we propose does not overlap the previous one that divides pragmatic inferences into presuppositions and implicatures because both these classes can be felicitously or infelicitously defeated.

In summary, we consider that the knowledge of an agent can be divided as follows:

- Semantic knowledge
  - Indefeasible
  - Defeasible
- Pragmatic knowledge
  - Felicitously defeasible
  - Infelicitously defeasible

Our ultimate goal is to develop a computational method for determining pragmatic inferences and for signalling infelicities; this requires a classification of the pragmatic inferences in accordance not only with their source but also with their properties. A potential source is for example a factive such as the verb *regret* that normally presupposes its complement. After one identifies the environment in which a pragmatic inference is triggered, she should map it into a specific representation which should reflect the properties that the inference is expected to have.

If we formalize the infelicitously defeasible inferences as entailments, an inconsistency will occur. But there is no inconsistency when one utters (1.49). The fact that *I believe that Mary is unfaithful* is not uttered, it is only *implied*. Another reason that prevents one of formalizing infelicitously defeasible inferences as entailments is their violation of the contrapositive rule. For example, one would be tempted to say that (1.52) is a logical implication of (1.51) or that (1.54) is an implication of (1.53).

(**1.51**) John regrets that Mary came to the party.

(**1.52**) $\triangleright$ Mary came to the party.

(**1.53**) I failed CS100 again.

(**1.54**) $\triangleright$ I failed CS100 before.

This is not the case, because (1.55) does not imply in most of the cases either (1.56), or (1.57); and (1.58) does not imply either (1.59), or (1.60). The pairs (1.56), (1.57), and (1.59), (1.60) stand for the two kinds of natural language negation: internal and external.

(**1.55**) Mary did not come to the party.

(**1.56**) John does not regret that Mary came to the party.

(**1.57**) It is not the case that John regrets that Mary came to the party.

(**1.58**) I did not fail CS100 before.

(**1.59**) I did not fail CS100 again.

(**1.60**) It is not the case that I did not fail CS100 again.

If we consider again sentences that are infelicitous to utter such as (1.3), (1.12), (1.44), or (1.49), we notice that the notion of *pragmatic infelicity* can be associated with the cancellation of some Gricean inferences. For example, it is *infelicitous* to cancel an implicature derived using the maxim of quality (1.3), (1.49), the maxim of manner (1.12) or a presupposition triggered in a positive environment (1.44).

The taxonomy we propose does not contradict Gazdar's opinion [1979, p. 40]:

The implicature must not be a truth condition of the sentence involved.

1.4 A pragmatic analysis of default logics

We are interested in a mathematical formalism that is capable of capturing implicatures, presuppositions, and infelicities. Because pragmatic inferences are defeasible, it seems that default logics are appropriate tools for doing this. However, we have already emphasized that sometimes the cancellation of these pragmatic inferences is felicitous, while sometimes it is infelicitous. This dichotomy implies that if one wants to distinguish between these two types of behavior, she has to take into consideration more than one level of defeasible information. Moreover, to signal an infelicitous cancellation, one should have available at the same time not only the presupposed or implicated information but its negation as well.

The default theory proposed by Reiter [1980] is a formal object consisting of a pair (D, W) where D is a set of defaults and W a set of closed well-formed formulas. The defaults have the form

$$\frac{\alpha : \beta}{w}$$

where α is the *prerequisite* and w is the *consequent*. Intuitively, if α holds and if it is consistent to believe β, one would be entitled to derive w. For any default theory, one can construct a set of extensions, i.e., a set of acceptable beliefs that may hold, about the incompletely specified world W. In a simplified account, extensions are defined in terms of a fixed-point operator that maps an arbitrary set of formulas to the smallest deductively closed set E that contains W and satisfies the condition: for any default $\frac{\alpha:\beta}{\gamma} \in D$, if $\alpha \in E$ and $\neg\beta \notin E$ then $\gamma \in E$.

Mercer (section 1.2.4) tries to use this theory for formalizing presuppositions. Even if we assume that his approach has no problems, we are still unable to differentiate between felicitously defeasible and infelicitously defeasible inferences in such a framework. Reiter's default logic is built on consistency basis; that means that all the information that belongs to a given theory has the same strength. Therefore, there is no obvious encoding that can determine the difference between felicitously and infelicitously defeasible information. We cannot accommodate the two types of defeasibility even if we consider extensions of Reiter's logic such as those developed by Brewka [1994] or Delgrande [1994]. Their main interest is to formalize the notion of preference over extensions and priority over defaults, but their work is still inadequate for our purposes. Even if we assign a higher priority to a default that formalizes an infelicitously defeasible inference over a default that formalizes a felicitously defeasible inference, the extension we get keeps no trace of the defaults that were used for deriving the information within the extension. In the process of analyzing a sequence of utterances, we can observe how extensions change their content but we cannot specify at a given time what information is presupposed, what is implicated, and we cannot say if an infelicity has occured.

Nait Abdallah's [1989, 1991] ionic logic is build on the notion of *default ions.* A default ion, $(a,b)_\star$ is a syntactic object having a well-defined semantic behavior that is used to refine a partially defined theory; it should be read: if a is consistent with the current logical scope then infer b. Ions are used to construct partial models for theories. Ionic logic differs from Reiter's default logic in that instead of having a fixed-point extension construction, it exhibits Scott's notion of continuity in which the final solution is obtained as a limit of a continuous process that generates better and better approximations of the solution.

At the semantic level, ionic logic gives objects with two components: a *kernel* part that cannot be revised and a *belt* part that can be revised and changed. The belt part is supported by a set of justifications. Formally, a default interpretation is a triple $\langle i_0, J, i_1\rangle$ such that:

- i_0 (the kernel valuation) and i_1 (the belt valuation) are partial mappings from the set of propositional symbols to the interpretative structure;
- J is the justification set that supports the belt valuation.

The partial models are constructed using a generalization of the semantic tableau technique.

In ionic logic, all defeasible information is paired by a nonempty set of justifications. Therefore, one is able to signal when some defeasible information has been cancelled. Unfortunately, ionic logic deals with only one kind of defeasible information. Hence there is no possibility to accommodate within this theory the notion of felicitously and infelicitously defeasible inferences.

Kifer [Kifer and Lozinskii, 1992, Kifer and Subrahmanian, 1992] constructs his annotated logic in order to allow the reasoning in the presence of inconsistency. In classical logic, anything follows from an inconsistent theory; therefore, it is impossible to distinguish between $\{flies(tweety), \neg flies(tweety)\}$ and $\{flies(tweety), \neg flies(tweety), married(john, mary)\}$ even if intuition tells us that the second theory contains more information. Underlying annotated logic is a belief upper semi lattice with a unique upper bound for every pair of elements. A similar lattice for dealing with defaults is considered in Ginsberg [1988] and it looks like the one in figure 1.1.

The syntax of annotated logic differs from that of first-order logic in having the atomic formulas appended by an annotation that is drawn from the underlying belief semi lattice. Thus, well-formed formulas are $(\exists x)q(x,z) : f$ or $flies(tweety) : dt$. The semantics of annotated logic defines that an atomic formula $p(t_1,\ldots,t_n) : s$ is satisfied by a valuation v if and only if $s \leq I_P(p)(v(t_1),\ldots,v(t_n))$ where I_P associates to each predicate symbol p a function $I_P(p) : Domain \rightarrow Semilattice$.

It is obvious now that such a definition is able to deal with inconsistencies using an appropriate underlying semi lattice, but it still does not help us in differentiating between felicitously and infelicitously defeasible inferences. As soon as some stronger information is derived, in order to satisfy the new set, the $I_P(p)$ function should pick up a stronger value

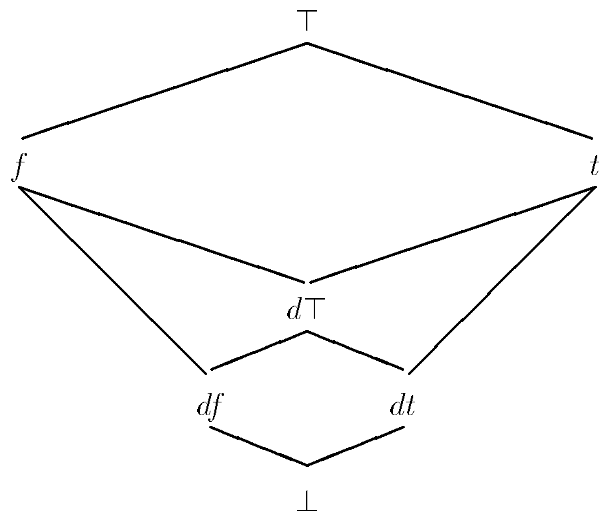

Figure 1.1: A bilattice for default reasoning

from the underlying bilattice. The new theory will be satisfied, but we will not be able to notice that some weaker information has been cancelled. The logic is not designed to deal with inconsistencies occuring between different levels of strength. A technical drawback is the fact that there are two types of negations and implications in annotated logic and there is no criterion to choose between one use or another.

Chapter 2

Stratified logic

A careful analysis of pragmatic inferences has shown that a simple dichotomy between defeasible and indefeasible inferences is not enough if one wants to deal with infelicities. A refined taxonomy built in terms of felicitously and infelicitously defeasible inferences has been proposed instead, but no candidate has been found in the default literature that fulfills our requirements and no obvious extension having the desired properties has been foreseen for any candidate. Either the formalisms studied handle only one type of defeasibility, or they cannot signal inconsistencies that occur between information that has been inferred with different strengths. Useful features have been identified however in Kifer's [Kifer and Subrahmanian, 1992, Kifer and Lozinskii, 1992] and Nait Abdallah's [1989, 1991] work. Our purpose is to find a way to put together the idea that logical formulas may have different strengths (Kifer), with the semantic tableau method that seems to be more attractive from a computational perspective (Nait Abdallah) than a fixed-point operator. The result is a new logical formalism called *stratified logic*.

2.1 Informal intuitions

The solution presented here is new. Instead of treating the notion of defeasibility on justification-based grounds, we conjecture that defeasible information should have a different status not only semantically, but also syntactically. The lattice in figure 2.1 underlies the semantics of stratified logic. The lattice depicts the three levels of strength that seem to account for the inferences that pertain to natural language semantics and pragmatics: indefeasible information belongs to the u layer, infelicitously defeasible information belongs to the i layer, and felicitously defeasible information belongs to the d layer. Each layer is partitioned according to its polarity in truth, $\top^u, \top^i, \top^d$, and falsity, $\bot^u, \bot^i, \bot^d$. The lattice shows a partial order that is defined over the different levels of truth. For example, something that is indefeasibly false, $\bot^u$, is stronger than something that is infelicitously defeasible true, $\top^i$, or felicitously defeasible false, $\bot^d$. Formally, we say that the u level is

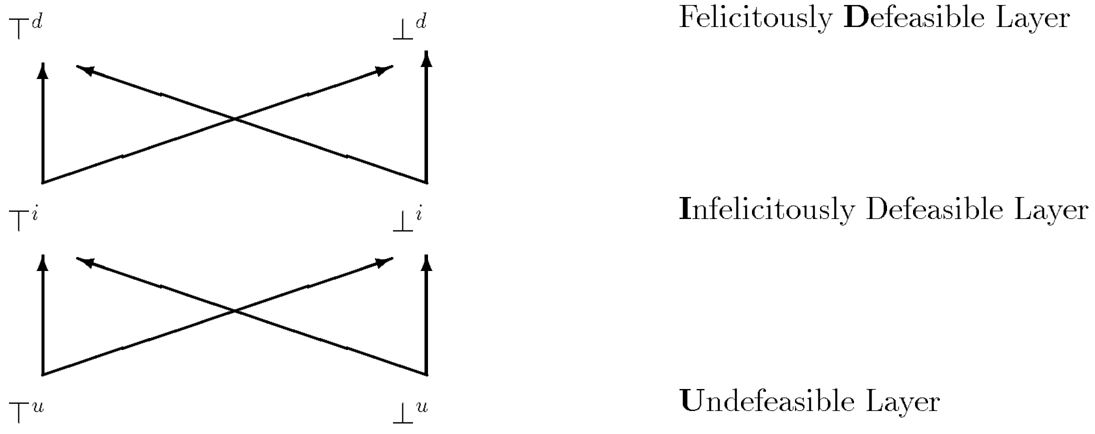

Figure 2.1: The lattice that underlies stratified logic

stronger than the i level, which is stronger than the d level: $u < i < d$. At the syntactic level we allow formulas to be labelled according to the same underlying lattice. This will give us formulas such as a^d, b^u, or $\neg c^i$ for the propositional case and such as $bird^u(tweety)$ or $\neg flies^d(tweety)$ for the first-order case.

We summarize the features that make stratified logic appropriate for handling pragmatic inferences and signalling infelicities:

- *Stratified logic has more than one level of truth.* This allows one to distinguish among the indefeasible information that characterizes the semantic inferences and the defeasible information that characterizes the pragmatic ones. Because there is more than one level of defeasibility, both infelicitously and felicitously defeasible inferences can be formalized.

- *In stratified logic formulas are labelled,* or are given a specific strength. Therefore, one will no longer need justifications for determining when a defeasible inference is acceptable. If some defeasible inference is overlapped by some stronger information we can easily account for this. On one hand we can update our sematic model so that it reflects the fact that some new stronger information has been inferred. On the other hand, there is no need for withdrawing the old weak information that is anyway overlapped by the stronger information, so our framework is in this sense monotonic. Because weak and strong information coexist at the same time, it is easy *to signal* when such an inconsistency between different levels of truth has occurred.

- *Stratified logic is computationally attractive.* Details about this issue are given in chapter 6.

The rest of the chapter contains a formal description of propositional and first-order stratified logic; therefore, for some people, it is inherently boring. Most of the definitions, lemmas, and theorems in this chapter are used only to facilitate the proofs that show that stratified tableaux are sound and complete. A reader who is not so enthusiastic about formal proofs can follow the rest of the thesis if she has a solid understanding of the way satisfaction and model-ordering is defined in our framework and if she is familiar with the tableau method.

2.2 The syntax and the semantics of stratified propositional logic

2.2.1 Syntax

For any set P of propositional letters, the stratified well-formed formulas *(s-wff)* are recursively defined as follows:

1. The constants $\top^u, \bot^u, \top^i, \bot^i, \top^d$, and $\bot^d$ are s-wffs.
2. Each propositional letter labelled with one of the symbols $\{u, i, d\}$ is a *s-wff*.
3. The set of formulas is closed under $\neg, \rightarrow, \wedge, \vee, \leftrightarrow$.

Examples of *s-wffs* are: $b^u \rightarrow f^d$, $b^u \wedge a^i \vee (a^d \rightarrow b^u)$.

The following are not *s-wffs*: $(a^i \rightarrow b^u)^i$, $a \rightarrow\rightarrow b$.

For the purpose of this thesis, as a simplification, we will omit the superscript u: whenever a formula α lacks a superscript, by default it will be read as α^u.

2.2.2 $\mathcal{S}$-propositional semantics

The satisfiability relation is extended to the three levels we have introduced. Hence, we will have *u-satisfiability*, $\models^u$, *i-satisfiability*, $\models^i$, and *d-satisfiability*, $\models^d$. An *extended valuation* or an *extended truth assignment* is a mapping from the set of propositional formulae into the partially ordered set $\{\top^u, \bot^u, \top^i, \bot^i, \top^d, \bot^d\}$ that constitutes the lattice in figure 2.1. We use the notion of *x-satisfiability* as a compact term for *u, i* or *d-satisfiability*. The strength of stratified logic relies in the way satisfaction is defined. An extended truth assignment will *x-satisfy* a set of formulas, if it uses only truth values weaker than or equal to x. This means that if one wants to *i-satisfy* a set of formulas, she will be allowed to choose for the truth assignment only values from the following set: $\{\top^i, \bot^i, \top^d, \bot^d\}$. If she wants to *u-satisfy* the same set of formulas, she is allowed to choose any value that is given in the underlying lattice.

For the satisfaction notion, we use the same simplification as the one we use for formulas, i.e., whenever a satisfiability relation, $\models$, lacks its superscript, it is read as $\models^u$.

Definition 2.2.1 *For any atomic formula a and any extended-valuation σ, the x-satisfiability relations are defined as follows:*

- $\sigma \models a$ *iff* $a^\sigma = \top$
- $\sigma \models a^i$ *iff* $a^\sigma \in \{\top, \bot, \top^i\}$
- $\sigma \models a^d$ *iff* $a^\sigma \in \{\top, \bot, \top^i, \bot^i, \top^d\}$
- $\sigma \models^i a$ *iff* $a^\sigma = \top^i$
- $\sigma \models^i a^i$ *iff* $a^\sigma = \top^i$
- $\sigma \models^i a^d$ *iff* $a^\sigma \in \{\top^i, \bot^i, \top^d\}$
- $\sigma \models^d a$ *iff* $a^\sigma = \top^d$
- $\sigma \models^d a^i$ *iff* $a^\sigma = \top^d$
- $\sigma \models^d a^d$ *iff* $a^\sigma = \top^d$

For any negation of an atomic formula $\neg a$ and any extended-valuation σ, the x-satisfiability relations are defined as follows:

- $\sigma \models \neg a$ *iff* $a^\sigma = \bot$
- $\sigma \models \neg a^i$ *iff* $a^\sigma \in \{\top, \bot, \bot^i\}$
- $\sigma \models \neg a^d$ *iff* $a^\sigma \in \{\top, \bot, \top^i, \bot^i, \bot^d\}$
- $\sigma \models^i \neg a$ *iff* $a^\sigma = \bot^i$
- $\sigma \models^i \neg a^i$ *iff* $a^\sigma = \bot^i$
- $\sigma \models^i \neg a^d$ *iff* $a^\sigma \in \{\top^i, \bot^i, \bot^d\}$
- $\sigma \models^d \neg a$ *iff* $a^\sigma = \bot^d$
- $\sigma \models^d \neg a^i$ *iff* $a^\sigma = \bot^d$
- $\sigma \models^d \neg a^d$ *iff* $a^\sigma = \bot^d$

Definition 2.2.2 *An extended truth valuation σ x-satisfies a formula α iff*

- *α is an atomic formula or the negation of an atomic formula and $\sigma \models^x \alpha$ as specified in definition 2.2.1;*
- $\sigma \models^x \alpha_1$ *where* $\alpha = \neg\neg\alpha_1$*;*

- $\sigma \models^x \neg\alpha_1$ *or* $\sigma \models^x \alpha_2$*, where* $\alpha = \alpha_1 \rightarrow \alpha_2$*;*
- $\sigma \models^x \alpha_1$ *and* $\sigma \models^x \neg\alpha_2$*, where* $\alpha = \neg(\alpha_1 \rightarrow \alpha_2)$*.*

Definition 2.2.2 can be easily extended to conjunction and disjunction as well, in the usual manner. For example, $\sigma \models^x \alpha$ where $\alpha = \alpha_1 \wedge \alpha_2$ iff $\sigma \models^x \alpha_1$ and $\sigma \models^x \alpha_2$. To ease our job, we will give our proofs only for negation and implication, but one should keep in mind that the same methodologies can be applied for other logical connectors as well. However, it is important to notice that our definitions imply neither that from $\sigma \models^x \alpha$ one may conclude $\sigma \not\models^x \neg\alpha$, nor that from $\sigma \not\models^x \alpha$ one may conclude $\sigma \models^x \neg\alpha$. For example $\{a/\top\} \models a^i$ and $\{a/\top\} \models \neg a^i$

Definition 2.2.3 *An extended truth valuation* σ *x-satisfies a set of formulas* Φ *iff* σ *x-satisfies each formula* $\varphi \in \Phi$

We are interested in interpreting a given theory with different degrees of *optimism*. This is exactly what the different levels of satisfaction provide us. The reader can notice that the way satisfaction is defined for the three levels of truth differs only with respect to the set of values that the corresponding truth assignment is allowed to range over. The u-level is the one that puts no constraints with respect to the values that a truth assignment can take from the underlying lattice. Therefore, there are more possibilities to *u-satisfy* a formula than to *d-satisfy* it. In this sense, the *u-satisfiability* relation is *more optimistic* than the other two. Note that a given theory becomes u-inconsistent only if it is inconsistent in the classical sense. We cannot say the same thing about the other two levels of satisfaction. For example the set $\{a, a^i, \neg a^d\}$ is *i-satisfiable* but is not *d-satisfiable*. The trichotomy among the satisfiability relations gives one a very powerful tool for analyzing a set of sentences. Depending on her purposes, one can interpret the same theory from an optimistic or pessimistic perspective. Thus, one is able to give the expected interpretation when some information is overlapped by a stronger one, but she is also able to signal that a cancellation has occurred. For example, the set $\{a, \neg a^i, \neg a^d\}$ is *u-satisfiable* but is not *i-* or *d-satisfiable*. Informally, one can still find a truth assignment to satisfy the initial set, but she is also aware that some infelicitously defeasible and felicitously defeasible information has been cancelled.

For the purpose of this thesis, an *extended propositional truth valuation* will be encoded as a list of pairs such that the first member in the pair represents a propositional variable and the second one an extended truth value, i.e., a member of the set $\{\top, \bot, \top^i, \bot^i, \top^d, \bot^d\}$. Here are some examples:

$\{a/\top\} \models \{a, \neg a^i, a^d\}$, but $\{a, \neg a^i, a^d\}$ is not *i-* or *d-satisfiable*;
$\{a/\top^i\} \models^i \{a, a^i\}$;
$\{a/\top^d\} \models^d \{a\}$;
$\{a/\top\} \not\models^i \{a^i\}$.

Definition 2.2.4 *A set of formulas Φ x-entails a formula α iff every extended truth assignment that x-satisfies Φ, x-satisfies α as well.*

Theorem 2.2.1 *If $\sigma \models^x \varphi$, then $\sigma \models^y \varphi$, where $y \leq x$. For example, if $\sigma \models^d \alpha$, then $\sigma \models^u \alpha$.*

Proof: For the atomic case, the theorem follows directly from the definition. For compound formulas, use induction on the degree of φ.

Theorem 2.2.2 *If $\sigma \not\models^x \varphi$, then $\sigma \not\models^y \varphi$, where $x \leq y$. For example, if $\sigma \not\models^u \alpha$, then $\sigma \not\models^i \alpha$.*

Proof: For the atomic case the theorem follows directly from the definition. For compound formulas, use induction on degree of φ.

2.2.3 Stratified propositional tableaux

Model construction in stratified logic is accomplished by a generalization of the "Beth tableau" technique. The method was introduced by Hintikka and Beth, and has its origins in the sequent calculus of Gentzen. A formal definition is given by Smullyan [1970]. We give a propositional calculus example to illustrate the method.

Suppose we wish to show that the sentence $[p \wedge (q \vee r)] \rightarrow [(p \wedge q) \vee (p \wedge r)]$ is valid. The tableau in Figure 2.2 does this. The tableau is constructed as follows. We see if we

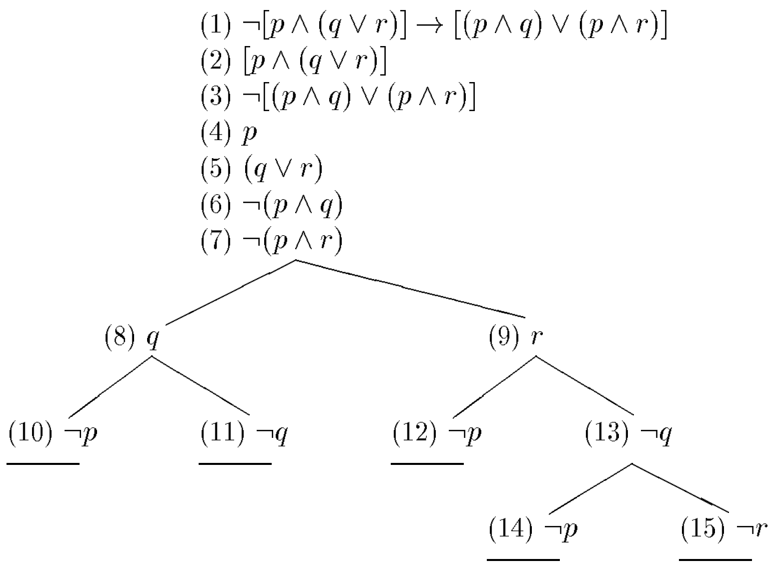

Figure 2.2: The Beth Tableau Technique — An Example

can derive a contradiction from the assumption that the given formula is false. So our first line consists of the negation of this formula. A formula of the form $x \rightarrow y$ can be false only if x is true and y is false. Thus (in the language of the tableau), x and $\neg y$ are *direct consequences* of the formula $\neg(x \rightarrow y)$. So we write lines (2) and (3) as direct consequences of line (1). Let us consider now line (2). Any formula of the form $x \wedge y$ can be true if only both x and y are true. So we derive lines (4) and (5) as direct consequences of line (2). Line (3) has the form $\neg(x \vee y)$. A disjunction to be false needs to have both disjuncts false. So lines (6) and (7) are added as direct consequences of line (3). Let us now turn to line (5) which has the form $(x \vee y)$. We cannot draw any direct conclusion about the truth value of x, nor about the truth value of y; all we can infer is that either x or y is true. So the tableau *branches* into two columns. So line (5) branches into two possibilities: line (8) and line (9). In the same manner, line (6) gives two possibilities : $\neg p$ and $\neg q$. Hence, the tableau now branches to four possibilities: (10), (11), (12), and (13). Looking now at the leftmost branch we shall see that (10) is a direct contradiction of (4), so we close this branch to signify that it leads to a contradiction. Similarly (11) contradicts (8), and (12) contradicts (4). So these branches are inconsistent too. Going back now to formula (7) — it is false when either p or r are false. Branches (10), (11) and (12) are already inconsistent so there is no point to add these new alternatives. The only consistent branch left is branch (13). Adding the new branches (14) and (15) leads us to inconsistency in both cases as (14) contradicts (4) and (15) contradicts (9). Thus all branches lead us to a contradiction, so line (1) is untenable. Thus $[p \wedge (q \vee r)] \rightarrow [(p \vee q) \wedge (p \vee r)]$ can never be false in any interpretation, so it is a tautology.

The tableau method can be used as a theorem prover. We know that a set of formulas $\Phi \models \alpha$ iff $\{\Phi, \neg\alpha\}$ is inconsistent. Therefore, to prove that α follows from Φ, it is enough to construct a closed tableau having as root the formulas $\Phi \cup \neg\alpha$.

In what follows, we will use the terminology related to classical propositional tableaux given by [Bell and Machover, 1986]. We generalize the classical propositional tableaux, such that if a given *stratified propositional tableau* T for a given theory Φ_0 has been obtained, we are allowed to extend it into a new one T' by any of the following three rules. In each case T' will have all the nodes of T, plus one or two new nodes.

- **Rule** $\neg\neg$: If among the formulas of a branch of T terminating at node Φ there is a formula $\neg\neg\alpha$, add a new node $\{\alpha\}$ as successor to Φ.

- **Rule** $\rightarrow$: If among the formulas of a branch of T terminating at node Φ there is a formula $\alpha \rightarrow \beta$, add two new nodes $\{\neg\alpha\}$ and $\{\beta\}$ as successors to Φ.

- **Rule** $\neg \rightarrow$: If among the formulas of a branch of T terminating at node Φ there is a formula $\neg(\alpha \rightarrow \beta)$, add a new node $\{\alpha, \neg\beta\}$ as successor to Φ.

A branch of a tableau is *x-closed* if there is a atomic formula α such that both α^y and $\neg\alpha^t$ are formulas of that branch, and $y, t \leq x$. In other words, a branch is *x-closed* if the set of formulas that belong to that branch is not *x-satisfiable*. The formulas α^y and $\neg\alpha^t$ are said to be *used* for closing the branch. A propositional tableau for Φ is called an *x-confutation* of Φ if all its branches are *x-closed*. To *x-confute* Φ is to construct an *x-confutation* of Φ.

For example, the tableau given in figure 2.3 has two branches. The left one is *i-closed* and the right one is *d-closed*. Thus, the whole tableau is *d-closed*. This follows from corollary 2.2.1.

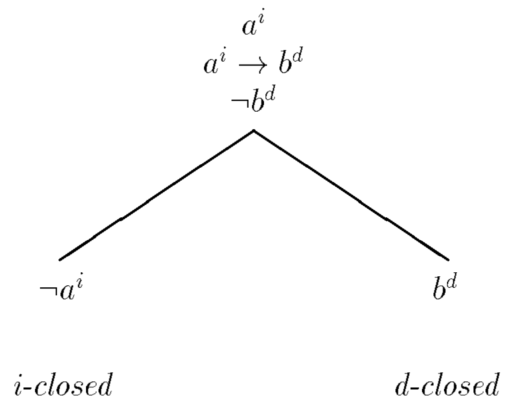

Figure 2.3: A *d-closed* stratified tableau

Corollary 2.2.1 *If a branch in a tableau is x-closed, that branch is also y-closed for any* $y \geq x$.

Proof: This is a direct consequence of theorem 2.2.2.

Theorem 2.2.3 Soundness of stratified propositional tableaux. *If a set of stratified propositional formulas* Φ *have a finite x-closed tableau, then* Φ *is x-unsatisfiable.*

One can reformulate the soundness theorem as follows: If an extended truth valuation σ *x-satisfies* all the formulas of a given branch in a tableau, and if that branch is extended into a new branch (or extended and split into two new branches) by one of the rules, then σ also *x-satisfies* the new branch (or at least one of the two branches).

Proof: Lemma 2.2.1 is a stronger form of the contrapositive of the theorem. Using it, the soundness follows immediately.

Lemma 2.2.1 *If* T *is a tableau for* Φ *and* σ *x-satisfies* Φ*, then for some branch of* T*,* σ *x-satisfies all formulas on that branch (hence, the branch is open).*

Proof: The proof is by induction on the height of the tableau T (Φ and σ are fixed).

Base case:

Assume that T consists of the root alone. The result is trivial.

Induction step:

- Assume that σ *x-satisfies* all formulas on a branch of height i that contains $\neg\neg\alpha$. In accordance with the tableau rule for negation, that branch can be extended with α into a branch of height $i+1$. Definition 2.2.2 assures one that this branch is also *x-satisfiable*.

- Assume that σ *x-satisfies* all formulas on a branch of height i that contains $\alpha_1 \to \alpha_2$. In accordance with the tableau rule for implication, this branch can be extended into two branches of height $i+1$, one containing $\neg\alpha_1$, the other containing α_2. Definition 2.2.2 assures one that at least one of these branches is also *x-satisfiable*.

- Assume that σ *x-satisfies* all formulas on a branch of height i that contains $\neg(\alpha_1 \to \alpha_2)$. In accordance with the corresponding tableau rule, this branch can be extended with two new formulas: α_1 and $\neg\alpha_2$. Definition 2.2.2 assures one that the extended branch is also *x-satisfiable*.

By induction, the lemma follows.

Theorem 2.2.4 Completeness of stratified propositional tableaux. *If Φ is x-unsatisfiable then Φ has a finite x-closed tableau.*

Proof: To prove this theorem we need some extra definitions:

Definition 2.2.5 *Let φ be a formula on a branch B of a tableau T. We say φ is* used up *on B iff one of the following holds:*

1. *φ is $\neg\neg\alpha$ and α is on branch B;*
2. *φ is $\neg(\alpha \to \beta)$ and $\alpha, \neg\beta$ are on B;*
3. *φ is $\alpha \to \beta$ and either α or β is on B;*
4. *φ is P^x or $\neg P^y$ for some atoms P, where $x, y \in \{u, i, d\}$.*

The branch B is used up *iff all formulas on B are* used up.
The tableau T is used up *or* exhausted *iff all its branches are* used up.

Completeness follows from lemmas 2.2.2 and 2.2.3.

Lemma 2.2.2 *Every finite[1] set of formulas Φ has a tableau which is used up.*

Proof: Consider an algorithm that starts with Φ as the single node of the tree and at each step extends one of the branches that are not used up in accordance with the tableau rules, i.e., applies these rules for a formula α that is not used up. Every formula generated

[1]A proof similar to the classical one can be given for a countable Φ as well.

on T is either of the form β or $\neg\beta$ where β is a sub-formula of α. Thus, each branch has at most $2k$ formulas where k is the number of sub-formulas in Φ. That means the algorithm terminates. Its result is an exhausted tableau.

Lemma 2.2.3 *If T is a tableau for Φ with a branch B that is used up and is not x-closed, then the set of formulas on B is x-satisfiable. Therefore, Φ is x-satisfiable.*

Proof: Each formula on B is used up, so for each atomic formula P, not both P^y and $\neg P^z$ occur on B where $y, z \leq x$. We define $P^\sigma = \top^m$ iff P^y is a label on the branch, $m = max(y, x)$ and for any z such that P^z or $\neg P^z$ occur on the branch $m \leq z$. We define $P^\sigma = \bot^m$ iff $\neg P^y$ is a label on the branch, $m = max(y, x)$ and for any z such that P^z or $\neg P^z$ occur on the branch $m \leq z$. Intuitively, we pick up the strongest labelled atomic formula, P^t or $\neg P^t$ and we assign to P the maximum value between t and x where x is the level for which we want to find an extended valuation. Assume that $x \leq t$. In this case, in accordance with definition 2.2.2, the truth assignment will satisfy P^t and any other weaker formula. If $x > t$, because there is no other formula of the form $\neg P^z$ with $t \leq z \leq x$, then the truth assignment will satisfy P^t and any other formula weaker than x.

The claim is that $\sigma \models^x \Phi$. We prove this by structural induction on Φ.

Base case:

If Φ is an atom or a negated atom, the result is trivial.

Induction step:

- Assume that $\Phi = \neg\neg\alpha$. Then α occurs on B. Following the definitions, any extended truth assignment that *x-satisfies* α also *x-satisfies* $\neg\neg\alpha$.

- Assume that $\Phi = \neg(\alpha \rightarrow \beta)$. Then both α and $\neg\beta$ occur on branch B, because it is used up. By induction hypothesis, the extended truth valuation *x-satisfies* both α and $\neg\beta$. In accordance with 2.2.2, the extended truth assignment *x-satisfies* $\neg(\alpha \rightarrow \beta)$.

- Assume that $\Phi = \alpha \rightarrow \beta$. Then either $\neg\alpha$ or β occurs on branch B, because it is used up. By induction hypothesis, the extended truth valuation *x-satisfies* either $\neg\alpha$ or β. In accordance with 2.2.2, the extended truth assignment *x-satisfies* $\alpha \rightarrow \beta$.

2.3 Stratified models and model ordering

As seen, propositional tableaux can be used as theorem provers, but there is another way one can look at them. Consider the following classical propositional theory: $\{a \vee b, \neg(a \rightarrow b), a \vee c\}$. If one constructs the tableau for this theory (see figure 2.4), she will obtain two open branches. Collecting the atomic formulas on each branch yields two model schemata for the given theory. For example, the branch that yields model schema m_0 contains atomic formulas $a, a, \neg b$, and a. Removing the duplicates, we obtain the model schema $m_0 = \{a, \neg b\}$. The branch ending in X contains both b and $\neg b$; therefore, it is closed, i.e., there is no way to construct a model for the formulas on that branch.

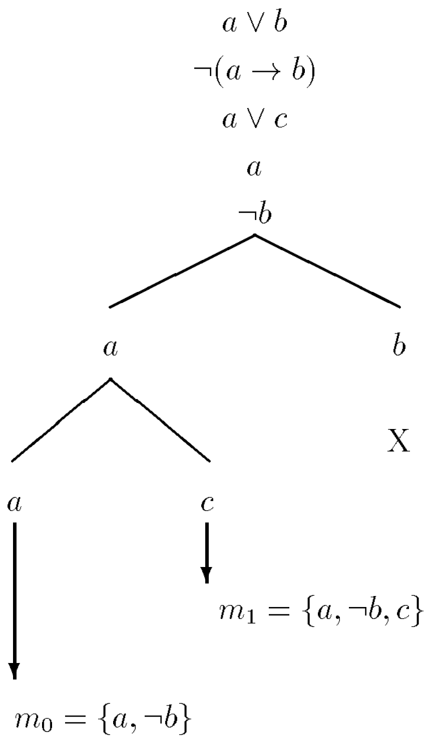

Figure 2.4: Semantic tableaux as means for building models

Observe that collecting the atomic formulas on each branch does not provide a model for the theory, but rather a model schema, because it may be the case that some of the atomic formulas that occur in that theory do not belong to the branch. In our example, model m_0 contains no reference to the atomic formula c. This means that the choice of the truth assignment for c has no impact on the satisfiability relation: both $\{a/\top, b/\bot, c/\top\}$ and $\{a/\top, b/\bot, c/\bot\}$ will satisfy the initial theory. Thus, the model schemata provide a compact representation for all the possible models of a theory.

The same idea will be applied to stratified semantic tableaux. Consider a classical example from the default reasoning literature, where the system knows that *Tweety is a bird* and *Birds typically fly* and would like to conclude that *Tweety flies*. As known, typicality is not representable in first-order logic, because a formula such as

$$(\forall x)(bird(x) \rightarrow flies(x))$$

yields incorrect results for penguins or ostriches for example, and a formula such as

$$(\forall x)(bird(x) \wedge \neg penguin(x) \wedge \neg ostrich(x) \wedge \ldots \rightarrow flies(x))$$

presupposes that one has to prove first that Tweety is not a penguin or an ostrich in order to prove that Tweety flies (see McCarthy [1980, 1986] for a detailed discussion).

2.3.1 Why does Tweety fly?

Consider the following example: *Tweety is a bird. Typically, birds fly. Penguins do not fly. Does Tweety fly?* An appropriate formalization in stratified propositional logic and its corresponding semantic tableau (figure 2.5) follows:

b	Tweety is a bird.
$b \rightarrow f^d$	Typically birds fly (this is defeasible information).
$p \rightarrow \neg f$	Penguins do not fly.

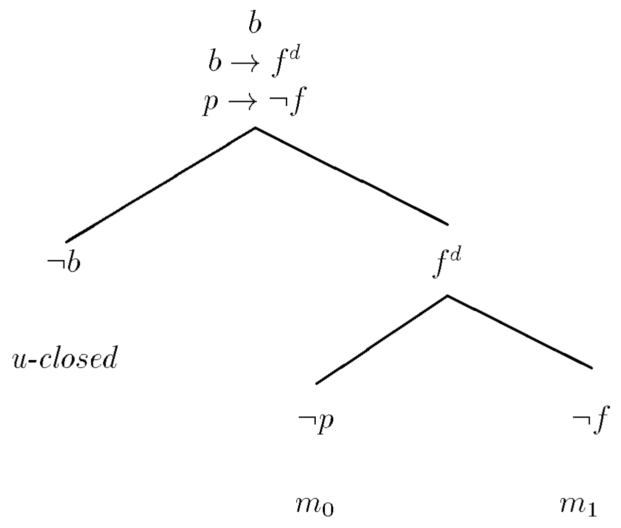

Figure 2.5: A propositional stratified tableau for theory *Tweety is a bird. Typically birds fly. Penguins do not fly.*

The semantic tableau has two open branches, which give two model skeletons or schemata:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	b $\neg p$		f^d
m_1	b $\neg f$		f^d

In model m_0, one can conjecture the fact that Tweety flies, but she cannot do this in model m_1. If we want an extended truth assignment to satisfy the skeleton m_1, we are constrained to assign f the truth value $\perp$. This is not the case with model skeleton m_0 which can be satisfied in a more "optimistic" way, assigning for example $\top^d$ to f. In this sense, from an optimistic perspective, m_0 is the preferred model for this theory. A formal definition for the model ordering relation is given in 2.4.7.

If we learn that Tweety is a penguin, the branch that gave the model m_0 will be *u-closed*, so that the only model for the theory can be constructed assigning $\perp$ to f. Hence, Tweety will no longer be thought to fly (figure 2.6).

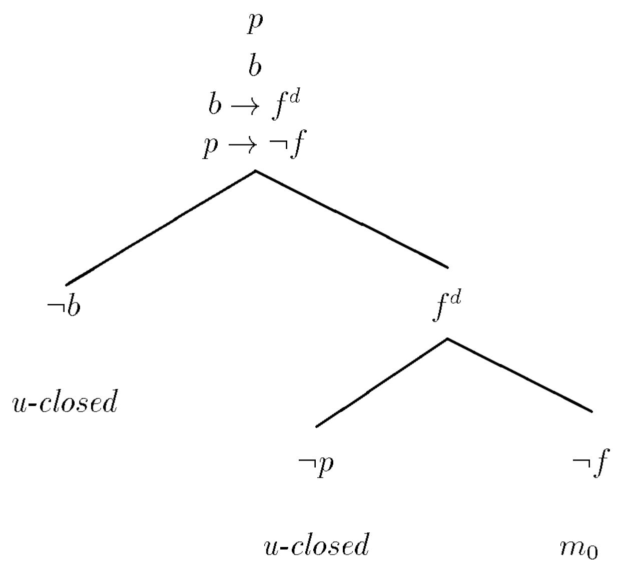

Figure 2.6: A propositional stratified tableau for theory: *Tweety is a bird and a penguin. Typically birds fly. Penguins do not fly.*

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	b		
	p		
	$\neg f$		f^d

2.4 Model ordering

We are interested in giving a formal account for the notion of preferred or *optimistic* model. We accomplish this by formalizing the following two intuitions.

1. Assume that one utters:

   (**2.1**) If the weather is good we can go to the beach tomorrow. We can swim, lie in the sun, and build sand castles. We can have a lot of fun.

There are two ways one can make the utterance true. The first one is to assign falsity to the antecedent, while the second one is to assume that the antecedent holds, therefore, the consequent holds too. If the antecedent is false we cannot say too much about the corresponding interpretation. However, if the antecedent is true, there are many other things that can be said within the corresponding interpretation. In other words, the second interpretation is more informative than the first one, so an *optimistic* agent will prefer it.

2. Assume that one knows that *Tweety is a bird* and that *Typically birds fly.* It may be the case that a rational agent finds only later that Tweety is a penguin, but as long as she doesn't have this information she is still inclined to believe that *Tweety flies.* In other words, an *optimistic* agent will prefer to trigger as many inferences as possible even though some of them may be retracted in the future.

Let us analyze the first example again, where we know nothing about Tweety being a penguin. As seen, the theory yields two model schemata for this case. There are two different ways one can analyze and compare these models. On one hand, notice that model m_0 is more informative than model m_1, i.e., it contains information about p, while m_1 does not. On the other hand notice that model m_0 puts no constraints on Tweety's flying (f). It assumes that Tweety flies, and this information is labelled as defeasible. The second model contains also the felicitously defeasible information that Tweety flies, but it also contains the indefeasible information that Tweety does not fly. This indefeasible information overlaps the defeasible inference and prevents a rational agent from finding an extented truth assignment that *d-satisfies* the initial theory: in other words, the second model is less optimistic. Formal definitions for the model ordering follow.

Definition 2.4.1 *An atomic formula a is said to be* affirmative[2] *in a model schema m, if a^x is a member of m and for any a^t or $\neg a^t$ occurring in the model schema, $t > x$.*

Definition 2.4.2 *An atomic formula a is said to be* negative *in a model schema m, if $\neg a^x$ is a member of m and for any a^t or $\neg a^t$ occurring in the model schema, $t > x$.*

Definition 2.4.3 *An atomic formula is* self-cancellable *in a model schema m, if both a^x and $\neg a^y$ occur in m, where $x, y \in \{u, i, d\}$.*

Definition 2.4.4 *A model schema m_1 is more* informative *than a model schema m_2 iff for any positive or atomic formula a^x in m_2, there is a positive or negative atomic formula a^y or $\neg a^y$ in m_1, where $x, y \in \{u, i, d\}$.*

[2]If $t = x$, no judgment can be made with respect to the polarity of the formula that is discussed. Finding a reasonable solution for this problem is the key issue in choosing among interactive defaults that have the same strength.

Definition 2.4.5 *An atomic formula a^x is* weaker *than a^y iff x and y are members of the set $\{u, i, d\}$ and $x \geq y$ $(u < i < d)$.*

Definition 2.4.6 *A model schema m_1 is* weaker *than a model schema m_2 if for every non-self-cancellable atomic formula a^x in m_2 there is a corresponding a^y formula in m_1 that is weaker and has the same polarity (they are both positive or negative).*

Definition 2.4.7 *A model schema m_1 is more* optimistic *than a model m_2 if it is more informative and weaker.*

Definition 2.4.8 *The set of* optimistic model schemata *for an exhausted tableau are given by the set of most optimistic and weakest models in the partial order defined by the above two relations.*

Intuitively, this corresponds to having a truth-valuation with as much defeasible information as possible. For example, for the atomic formula a^d, $\{a/\top^d\}$ is an optimistic valuation, while $\{a/\top\}$ and $\{a/\bot\}$ are not.

If we reconsider now the example involving Tweety, m_0 is more optimistic than m_1 and this corresponds to our intuitions.

2.5 First-order stratified logic

We now discuss the first-order extension of stratified logic. The reader is referred to [Bell and Machover, 1986] for a comprehensive study of classical first-order logic.

2.5.1 Syntax

In order to interpret stratified terms and formulas, it is necessary to fix a stratified structure or stratified interpretation $\mathcal{SL}$ consisting of the following elements:

- A non-empty class $\mathcal{D}$ called the *universe of discourse*, or the *domain.*
- A mapping that assigns to each function symbol f of $\mathcal{SL}$ an operation $f^{\mathcal{D}}$ on $\mathcal{D}$ such that if f is an n-ary function symbol, $f^{\mathcal{D}}$ is an n-ary operation on $\mathcal{D}$. The functions of arity 0 give the constants of the language. Let F be the set of all these mappings.
- A mapping that assigns to each extra-logical predicate symbol P of $\mathcal{L}$ a relation $P^{\mathcal{D}}$ on $\mathcal{D}$ such that if P is an n-ary predicate symbol, $P^{\mathcal{D}}$ is an n-ary relation on $\mathcal{D}$. We are interested in extending the stratification found in the propositional case to predicates; hence we consider that relations P can be assigned a strength (indefeasible, infelicitously defeasible, and felicitously defeasible relations) and a polarity

(positive and negative relations). Thus, the set of relations R will be given by the union $R^u \cup \overline{R^u} \cup R^i \cup \overline{R^i} \cup R^d \cup \overline{R^d}$ where R^u stands for positive indefeasible relations, $\overline{R^u}$ for negative indefeasible relations, R^i for positive infelicitously defeasible relations, $\overline{R^i}$ for negative infelicitously defeasible relations, R^d for positive felicitously defeasible relations, and $\overline{R^d}$ for negative felicitously defeasible relations.

It is important to notice that the stratification is embedded into the predicate level; hence, predicates are stronger or weaker. There is no strength ordering at the function or constant level. Another difference with respect to first-order logic is that the relations found in a stratified interpretation may have negative polarity. This is a consequence of the way we have defined the satisfiability relation.

The terms are recursively defined as follows:

1. Any constant or variable is a term.
2. Any expression $f(t_1, t_2, ..., t_n)$ is a term iff $f \in F$ and each t_i is a term.

The formulae of stratified first-order logic are recursively defined as follows:

1. Each member of the set $\{\top^u, \bot^u, \top^i, \bot^i, \top^d, \bot^d\}$ is a formula.
2. Each expression $p^x(t_1, t_2, ..., t_n)$ where $p \in R$, $x \in \{u, i, d\}$, and t_i are terms is an atomic formula.
3. Each combination of atomic formulae using the logical connectives $\neg, \rightarrow, \vee, \wedge$, and $\leftrightarrow$ is a formula.
4. If f is a formula and x is a variable then $(\forall x)f$ and $(\exists x)f$ are formulas.

For example, $(\forall x, y, z)(regret(x, come(y, z)) \rightarrow come^i(y, z))$ or $(\forall x)(\neg bachelor(x) \rightarrow adult^d(x))$ are well-formed formulas. The first one expresses that factive *regret* in a positive environment implies its complement. The implication is infelicitously defeasible in this case, because one cannot felicitously utter *John regrets that Mary came to the party, but she did not come.* The latter expresses that a *bachelor* referent in a negative environment implies that that referent is also an adult. The implication is felicitously defeasible, because one can felicitously utter *John is not a bachelor; he is five years old.*

2.5.2 Semantics of stratified first-order logic

Definition 2.5.1 *An* <u>$\mathcal{SL}$ *valuation*</u> σ *is a stratified structure* $\mathcal{SL}$ *together with an assignment of the value* $x^\sigma \in \mathcal{D}$ *to each variable* x.

Definition 2.5.2 *Given an* $\mathcal{SL}$ *valuation* σ *with universe* $\mathcal{D}$, *we define for each term* t, *the value of* t *under* σ (t^σ) *in such a way that* $t^\sigma \in \mathcal{D}$:

- *If x is a variable, then x^σ is already defined.*
- *If f is an n-ary function symbol in F and $t_1, t_2, \ldots, t_n$ are terms, then* $(f(t_1, t_2, \ldots, t_n))^\sigma = f^\sigma(t_1^\sigma, t_2^\sigma, \ldots, t_n^\sigma)$

Definition 2.5.3 *Assume σ is an $\mathcal{SL}$ valuation such that $t_i^\sigma = d_i \in \mathcal{D}$ and assume that $\mathcal{SL}$ maps n-ary predicates p to relations $R \subset \mathcal{D} \times \ldots \times \mathcal{D}$. For any atomic formula $p^x(t_1, t_2, \ldots, t_n)$, and any stratified valuation σ, where $x \in \{u, i, d\}$ and t_i are terms, the x-satisfiability relations are defined as follows:*

- $\sigma \models p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^u$
- $\sigma \models p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^u \cup \overline{R^u} \cup R^i$
- $\sigma \models p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^u \cup \overline{R^u} \cup R^i \cup \overline{R^i} \cup R^d$
- $\sigma \models^i p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^i$
- $\sigma \models^i p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^i$
- $\sigma \models^i p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^i \cup \overline{R^i} \cup R^d$
- $\sigma \models^d p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^d$
- $\sigma \models^d p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^d$
- $\sigma \models^d p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^d$

For any negation of an atomic formula $\neg p^x(t_1, t_2, \ldots, t_n)$, and any stratified valuation σ, where $x \in \{u, i, d\}$ and t_i are terms, the x-satisfiability relations are defined as follows:

- $\sigma \models \neg p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^u}$
- $\sigma \models \neg p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^u \cup \overline{R^u} \cup \overline{R^i}$
- $\sigma \models \neg p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^u \cup \overline{R^u} \cup R^i \cup \overline{R^i} \cup \overline{R^d}$
- $\sigma \models^i \neg p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^i}$
- $\sigma \models^i \neg p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^i}$
- $\sigma \models^i \neg p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in R^i \cup \overline{R^i} \cup \overline{R^d}$
- $\sigma \models^d \neg p(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^d}$
- $\sigma \models^d \neg p^i(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^d}$
- $\sigma \models^d \neg p^d(t_1, t_2, \ldots, t_n)$ *iff* $\langle d_1, d_2, \ldots, d_n\rangle \in \overline{R^d}$

For example, $\sigma \models p^d(t_1, t_2, \ldots, t_n)$ iff $\langle d_1, d_2, \ldots, d_n\rangle \in R^u \cup \overline{R^u} \cup R^i \cup \overline{R^i} \cup R^d$ should be read as: $\mathcal{SL}$ valuation σ *u-satisfies* the atomic formula $p^d(t_1, t_2, \ldots, t_n)$ if and only if the mapping that is defined for the predicate symbol p assigns to p one and only one relation from the union $R^u \cup \overline{R^u} \cup R^i \cup \overline{R^i} \cup R^d$.

For the purpose of this thesis, the predicate mappings will be encoded as lists of predicates of a given strength, optionally preceded by negation. If for example $\langle 1\rangle \in p^u$, $\langle 2\rangle \in p^u$, and $\langle 1\rangle \in \overline{q^i}$, we will represent it as $\{p(1), p(2), \neg q^i(1)\}$.

Let σ be a stratified valuation with universe $\mathcal{D}$ and let $u \in \mathcal{D}$. We define $\sigma(x/u)$ to be a stratified valuation which agrees with σ on every variable other than x as well as on every extra-logical symbol, while $x^{\sigma(x/u)} = u$. Thus, the stratified structure underlying $\sigma(x/u)$ is the same as that underlying σ.

Definition 2.5.4 *A stratified valuation σ x-satisfies a formula α iff*

- *α is an atomic formula or the negation of an atomic formula and $\sigma \models^x \alpha$, as specified in definition 2.5.3;*
- *$\sigma \models^x \alpha_1$ where $\alpha = \neg\neg\alpha_1$;*
- *$\sigma \models^x \neg\alpha_1$ or $\sigma \models^x \alpha_2$ where $\alpha = \alpha_1 \to \alpha_2$;*
- *$\sigma \models^x \alpha_1$ and $\sigma \models^x \neg\alpha_2$ where $\alpha = \neg(\alpha_1 \to \alpha_2)$;*
- *$\sigma(v/t) \models^x \alpha_1$ for all $t \in \mathcal{D}$, where $\alpha = \forall v \alpha_1$ and $\sigma(v/t)$ is the same as σ except $v^{\sigma(v/t)} = t$;*
- *$\sigma(v/t) \models^x \alpha_1$ for at least one $t \in \mathcal{D}$, where $\alpha = \exists v \alpha_1$ and $\sigma(v/t)$ is the same as σ except $v^{\sigma(v/t)} = t$;*
- *$\sigma(v/t) \models^x \neg\alpha_1$ for at least one $t \in \mathcal{D}$, where $\alpha = \neg\forall v \alpha_1$ and $\sigma(v/t)$ is the same as σ except $v^{\sigma(v/t)} = t$;*
- *$\sigma(v/t) \models^x \neg\alpha_1$ for all $t \in \mathcal{D}$, where $\alpha = \neg\exists v \alpha_1$ and $\sigma(v/t)$ is the same as σ except $v^{\sigma(v/t)} = t$.*

As in the propositional case, the other logical connectors are not taken into consideration in our proofs, because they are expressible in terms of negation and implication.

Definition 2.5.5 *For any stratified formula α and any term t, $\alpha(x/t)$ is the formula that results from replacing all free occurrences of x in α by the term t.*

Theorem 2.5.1 Substitution theorem for terms. *For any term s, $(s(x/t))^\sigma = s^{\sigma(x/t^\sigma)}$.*

Proof: The theorem follows immediately using induction on s and definition 2.5.2.

Definition 2.5.6 *A term t is free for x in α iff no free occurrence of x in α is within a sub-formula $\forall y\beta$, where y occurs in t.*

Theorem 2.5.2 *Let t be a term, and let σ and τ be stratified valuations which agree on all variables and function symbols occurring in t. Then $t^\sigma = t^\tau$.*

The proof is straightforward if one uses induction on the degree of the term t.

Theorem 2.5.3 *Let σ and τ be stratified valuations which have the same universe $\mathcal{D}$ and which agree on every free variable of α and on every extra-logical symbol. Then $\sigma \models^x \alpha$ iff $\tau \models^x \alpha$.*

The proof is by induction on the degree of α.

Theorem 2.5.4 Substitution theorem for formulas. *For any stratified valuation σ, formula α and term t free of v in α, $\sigma \models^x \alpha(v/t)$ iff $\sigma(v/t^\sigma) \models^x \alpha$.*

Proof: We use induction on the degree of α. The only case that may raise problems is $\alpha = \forall y\beta$:

Suppose that v is not free in α. Then $\alpha(v/t) = \alpha$. By theorem 2.5.3, $\sigma \models^x \alpha$ iff $\sigma(v/t^\sigma) \models^x \alpha$. Thus $\sigma \models^x \alpha(v/t)$ iff $\sigma \models^x \alpha$ iff $\sigma(v/t^\sigma) \models^x \alpha$.

Now suppose that v is free in α and t is free for v in β and y does not occur in t. Then we have

$$\sigma \models^x \alpha(v/t) \text{ iff } \sigma \models^x (\forall y[\beta(v/t)]) \tag{2.1}$$

Using definition 2.5.4 we have that

$$\sigma \models^x (\forall y[\beta(v/t)]) \text{ iff } \sigma(y/u) \models^x \beta(v/t) \text{ for all } u \text{ in } \mathcal{D} \tag{2.2}$$

Since the degree of β is smaller than the degree of α, using the induction hypothesis, we get:

$$\sigma(y/u) \models^x \beta(v/t) \text{ iff } \sigma(y/u)(v/t') \models^x \beta \text{ where } t' = t^{\sigma(y/u)} \tag{2.3}$$

We know that y does not occur in t. Hence by theorem 2.5.2

$$t' = t^{\sigma(y/u)} = t^\sigma \tag{2.4}$$

At the same time, v and y are different (otherwise v could not be free in α); hence

$$\sigma(y/u)(v/t^\sigma) = \sigma(v/t^\sigma)(y/u) \tag{2.5}$$

Hence we can rewrite 2.3 as

$$\sigma(y/u) \models^x \beta(v/t) \text{ iff } \sigma(v/t^\sigma)(y/u) \models^x \beta \tag{2.6}$$

Definition 2.5.4 yields:

$$\sigma(v/t^\sigma)(y/u) \models^x \beta \text{ for all } u \in \mathcal{D}, \text{ iff } \sigma(v/t^\sigma) \models^x \forall y\beta \tag{2.7}$$

Combining 2.1, 2.2, and 2.7 we get the required result.

2.5.3 Stratified first-order tableaux

We now generalize the semantic tableau technique to stratified logic. Preserving Smul-lyan's notation for conjunctive, disjunctive, universal, and existential types of formulas, the rules for constructing a stratified tableau are the following:

- *Conjunctive type*: If α appears in the tableau, both α_1 and α_2 are added:

α	α_1	α_2
$p \wedge q$	p	q
$\neg(p \vee q)$	$\neg p$	$\neg q$
$\neg(p \rightarrow q)$	p	$\neg q$
$\neg\neg p$	p	

- *Disjunctive type*: If β appears in the tableau, it is split into two branches, one headed by β_1 , the other by β_2:

β	β_1	β_2
$\neg(p \wedge q)$	$\neg p$	$\neg q$
$p \vee q$	p	q
$p \rightarrow q$	$\neg p$	q

- *Universal type*: An universal formula may be instantiated with an arbitrary constant. In systematically applying it, one chooses all of the constants that have already appeared in the tableau.

γ	$\gamma(t)$
$\forall x\alpha$	$\alpha(x/t)$
$\neg\exists x\alpha$	$\neg\alpha(x/t)$

- *Existential type*: The constant introduced by these rules must not have occurred previously in the tableau. It is essentially a skolemization operation, and it is only applied once to each existential quantifier.

δ	$\delta(c)$
$\neg\forall x\alpha$	$\neg\alpha(x/c)$
$\exists x\alpha$	$\alpha(x/c)$

A branch of a tableau is *x-closed* if there is an atomic formula $p(t_1, t_2, \ldots, t_n)$ such that both $p^y(t_1, t_2, \ldots, t_n)$ and $\neg p^t(t_1, t_2, \ldots, t_n)$ are formulas on that branch, and $y, t \leq x$.

Lemma 2.5.1 *A stratified valuation σ x-satisfies $\alpha(v/t)$ for any t in the universe $\mathcal{D}$ if it x-satisfies $\forall v\alpha$.*

Proof: Let σ be any valuation such that $\sigma \models^x \forall v\alpha$. By definition 2.5.4, $\sigma(v/u) \models^x \alpha$ for all u in $\mathcal{D}$. We take $u = t^\sigma$. That means, $\sigma(v/t^\sigma) \models^x \alpha$. Using the substitution theorem 2.5.4, we obtain:

$$\sigma(v/t^\sigma) \models^x \alpha \text{ iff } \sigma \models^x \alpha(v/t) \tag{2.8}$$

Hence $\sigma \models^x \alpha(v/t)$.

Lemma 2.5.2 *Suppose σ is a $\mathcal{SL}$ stratified valuation which x-satisfies a set of stratified formulas Φ. Let T be a finite tableau for Φ. Then there is a branch B of the tableau T and an extension σ' of σ to $\mathcal{SL}'$ such that σ' x-satisfies all formulas on B. Therefore, B is open.*

Proof: We use induction on T.

Base case:

If T is a single node, σ' is any extension of σ.

Induction step:

The arguments for $\neg\neg$, $\rightarrow$, and $\neg\rightarrow$ are the same those that were given for the propositional case.

Case 1 ($\forall$):

Assume that σ *x-satisfies* all formulas on a branch of height i that contains $\forall v\alpha$. Thus, in particular, it *x-satisfies* $\forall v\alpha$. By lemma 2.5.1, σ *x-satisfies* $\alpha(v/t)$ for any term t in the universe. Thus $\sigma' = \sigma$ already *x-satisfies* all formulas on the new branch.

Case 2 ($\neg\forall$):

Assume that σ *x-satisfies* all formulas on a branch of height i that contains $\neg\forall v\alpha$. Thus, in particular, it *x-satisfies* $\neg\forall v\alpha$. Assume the branch is expanded according to the corresponding rule, such that it contains now $\neg\alpha(v/c)$. We must modify σ to σ' so that σ'

x-satisfies all formulas on B, including the new one. Stratified valuation σ *x-satisfies* $\neg\forall v\alpha$, hence, by definition 2.5.4, $\sigma(v/u) \models^x \neg\alpha$ for some $u \in \mathcal{D}$. Let $\sigma' = \sigma(c/u)$. By substitution theorem 2.5.4, $\sigma(v/u) \models^x \neg\alpha$ iff $\sigma' \models^x \neg\alpha(v/c)$. Note that c did not occur on B earlier, so σ' continues to satisfy all formulas on branch B.

Cases $\exists$ and $\neg\exists$ can be treated in the same manner.

Theorem 2.5.5 Soundness of stratified first-order tableaux. *If a set of stratified first-order formulas Φ has a finite x-closed tableau, then Φ is x-unsatisfiable.*

Proof: The result follows as a contrapositive of lemma 2.5.2.

To prove completeness we need a definition and some lemmas:

Definition 2.5.7 *Let $\mathcal{L}$ be a language with at least one constant symbol and Ψ be a set of $\mathcal{L}$-sentences. We say that Ψ is an* extended Hintikka set with respect to level x, *provided for all $\varphi \in \Psi$*

1. *if φ is an atom of the form α^y then any negation $\neg\alpha^t$ with $y, t \leq x$ is not in Ψ;*
2. *if φ is a negated atomic formula of the form $\neg\alpha^y$, then any atomic formula α^t with $y, t \leq x$ is not in Ψ;*
3. *if φ has the form $\neg\neg\alpha$ then $\alpha \in \Psi$;*
4. *if φ has the form $\alpha \to \beta$ then either $\neg\alpha \in \Psi$ or $\beta \in \Psi$;*
5. *if φ has the form $\neg(\alpha \to \beta)$ then either $\alpha \in \Psi$ and $\neg\beta \in \Psi$;*
6. *if φ has the form $\forall x\alpha$ then $\alpha(x/t) \in \Psi$ for every ground term t;*
7. *if φ has the form $\neg\forall x\alpha$ then $\neg\alpha(x/t) \in \Psi$ for some ground terms t;*
8. *if φ has the form $\exists x\alpha$ then $\alpha(x/t) \in \Psi$ for some ground term t;*
9. *if φ has the form $\neg\exists x\alpha$ then $\neg\alpha(x/t) \in \Psi$ for all ground term t;*

Lemma 2.5.3 *Every extended Hintikka set with respect to level x is x-satisfiable.*

Proof: We construct an $\mathcal{SL}$ structure as follows:

1. let $\mathcal{D}$ be the set of all ground terms in the extended Hintikka set.
2. let $(f(t_1, t_2, \ldots, t_n))^{\mathcal{SL}} = f^{\mathcal{SL}}(t_1^{\mathcal{SL}}, t_2^{\mathcal{SL}} \ldots t_n^{\mathcal{SL}})$ for each n-ary function symbol f in $\mathcal{L}$
3. $\langle t_1, \ldots, t_n \rangle \in P^{x\mathcal{SL}}$ iff $P^x(t_1, \ldots, t_n) \in \Psi$.

4. $\langle t_1, \ldots, t_n \rangle \in \overline{P^x}^{\mathcal{SL}}$ iff $\neg P^x(t_1, \ldots, t_n) \in \Psi$.

One can prove using induction on the degree of φ that any formula $\varphi \in \Psi$ is *x-satisfiable* under $\mathcal{SL}$.

We extend now the notion of *used up* formula to the first-order case. The definitions for *used up branch* and *used up tableau* are similar with the ones found in the propositional case.

Definition 2.5.8 *Let φ be a first-order formula on a branch B of a tableau T. We say φ is* used up *on B iff one of the following holds:*

1. *φ is $\neg\neg\alpha$ and α is on branch B;*
2. *φ is $\neg(\alpha \rightarrow \beta)$ and $\alpha, \neg\beta$ are on B;*
3. *φ is $\alpha \rightarrow \beta$ and either α or β is on B;*
4. *φ is $(\forall x)\alpha$ and $\alpha(x/t)$ is on B for all constant symbols t occuring on formulas that belong to B.*
5. *φ is $(\exists x)\alpha$ and $\alpha(x/t)$ is on B for some constant symbol t occuring on formulas that belong to B.*
6. *φ is P^x or $\neg P^y$ for some atoms P, where $x, y \in \{u, i, d\}$.*

Lemma 2.5.4 *The set of sentences of any used up x-open branch is an extended Hintikka set with respect to level x.*

Lemma 2.5.5 *Every x-unsatisfiable countable or finite set of sentences Φ has a possibly infinite tableau in which every branch is used up.*

Proof: The proofs of the last two lemmas follow exactly the same steps as the ones for classical first-order logic found in [Bell and Machover, 1986].

One can notice that the last three definitions do not prohibit the existence of infinite branches. However, in our domain, we will use function symbols mainly to formalize sentences. As it is shown in chapter 6, it is counter intuitive to apply the universal instantiation rule over sentences as well. Therefore, out tableaux will be finite.

Theorem 2.5.6 Completeness of stratified first-order tableaux. *If Φ is an x-unsatisfiable countable or finite set of sentences, then Φ has a finite x-closed tableau.*

Proof: The theorem is a consequence of Lemma 2.5.3, 2.5.4, and 2.5.5.

2.6 Model ordering in first-order stratified logic

We generalize the notions concerning model ordering found in propositional stratified logic to first-order stratified logic.

Definition 2.6.1 *A* model schema *for an exhausted stratified tableau is represented by the set of all atomic formulas and negated atomic formulas that belong to an open branch.*

Definition 2.6.2 *An atomic formula $P(t_1, \ldots, t_n)$ is said to be* affirmative *in a model schema m, if $P^x(t_1, \ldots, t_n)$ is a member of m and for any $P^y(t_1, \ldots, t_n)$ or $\neg P^y(t_1, \ldots, t_n)$ occurring in m, $y > x$ $(u < i < d)$.*

Definition 2.6.3 *An atomic formula $P(t_1, \ldots, t_n)$ is said to be* negative *in a model schema m, if $\neg P^x(t_1, \ldots, t_n)$ is a member of m and for any $P^y(t_1, \ldots, t_n)$ or $\neg P^y(t_1, \ldots, t_n)$ occurring in m, $y > x$.*

Definition 2.6.4 *An atomic formula is* self-cancellable *in a model schema m, if both $P^x(t_1, \ldots, t_n)$ and $\neg P^y(t_1, \ldots, t_n)$ occur in m, where $x, y \in \{u, i, d\}$.*

Definition 2.6.5 *A model schema m_1 is more* informative *than a model schema m_2 iff for any positive or negative atomic formula φ in m_2, there is a positive or negative atomic formula φ in m_1.*

Definition 2.6.6 *An atomic formula $P^x(t_1, \ldots, t_n)$ is* weaker *than $P^y(t_1, \ldots, t_n)$ iff x and y are members of the set $\{u, i, d\}$ and $x \geq y$.*

Definition 2.6.7 *A model schema m_1 is* weaker *than a model schema m_2 if for every non-self-cancellable atomic formula φ^x in m_2 there is a corresponding φ^y formula in m_1 that is weaker and has the same polarity.*

Definition 2.6.8 *A model schema m_1 is more* optimistic *than a model schema m_2 if it is more informative and weaker. We write this as $m_1 < m_2$.*

Consider a theory described in stratified first-order logic that appropriately formalizes the classical default logic example involving Tweety:

$$\begin{array}{l} bird(tweety) \\ (\forall x)(bird(x) \rightarrow flies^d(x)) \\ (\forall x)(penguin(x) \rightarrow \neg flies(x)) \end{array}$$

If one applies the stratified tableau method, she will obtain two model schemata that *u-satisfy* the theory:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$bird(tweety)$ $\neg penguin(tweety)$		$flies^d(tweety)$
m_1	$bird(tweety)$ $\neg flies(tweety)$		$flies^d(tweety)$

Model schema m_0 is more optimistic than model schema m_1. It corresponds to an $\mathcal{SL}$ structure defined over a universe $\mathcal{D}$ that contains only one object, $tweety^{\mathcal{D}}$, and no function symbols. The relations defined on the universe are $\langle tweety^{\mathcal{D}} \rangle \in bird^u$, $\langle tweety^{\mathcal{D}} \rangle \in \overline{penguin^u}$, and $\langle tweety^{\mathcal{D}} \rangle \in flies^d$.

For the sake of compactness and clarity we represent model schemata as unions of relations partitioned according to their strength:

$$m_0 = \{bird(tweety), \neg penguin(tweety)\} \cup \emptyset^i \cup \{flies^d(tweety)\}$$

It remains to be seen how this approach to nonmonotonic reasoning can handle the sensitive problems found in the literature.

Chapter 3

Infelicity, presupposition, and lexical pragmatics

The lack of an appropriate framework capable of handling both felicitously and infelicitously defeasible information was the main reason that we could not formalize pragmatic inferences and signalling infelicities (see chapter 1). Chapter 2 was dedicated to the development of such a framework. Here, we argue that a bare translation of natural language utterances into stratified logic is still insufficient if one wants to account for the subtleties that are inherent to pragmatic inferences. The subtle feature that makes pragmatic inferences different from the semantic ones is the fact that, as Grice said [1975], they are carried by the saying of what is said, rather than by what is said. It is true that felicitously and infelicitously defeasible inferences can be associated with certain syntactic or lexical constructs, but they are triggered by *the saying* of those constructs. We devote this chapter to an incremental understanding of the notion of *lexical pragmatics*.

3.1 Felicitously versus infelicitously defeasible inferences

We have shown that the presuppositions carried by (1.41), *John regrets that Mary came to the party*, and (1.42), *John does not regret that Mary came to the party* do not have similar properties: the presupposition carried by sentence (1.41) cannot be felicitously defeated, but that related to sentence (1.42) can be. As we have already stated, it is a logical mistake to formalize the pragmatic inference in (1.41) as a logical implication because the contrapositive does not hold in all readings: *Mary did not come to the party* does not imply that *It is not the case that John regrets that Mary came to the party.* We now have stratified logic to capture these inferences. We delineate between felicitously and infelicitously defeasible inferences by assigning them different strengths. We still treat presuppositions as defeasible information, but those in positive environments are formalized as infelicitously defeasible inferences, while those in negative environments as felicitously defeasible inferences. Formally, we write:

$$\begin{aligned}&(\forall x,y,z)(regrets(x,\$come(y,z)) \to come^i(y,z))\\&(\forall x,y,z)(\neg regrets(x,\$come(y,z)) \to come^d(y,z))\end{aligned} \tag{3.1}$$

Obviously, in order to have well-formed formulas, proposition $come(x,y)$ should be a function in the antecedent and a predicate in the consequent of each implication. To avoid any ambiguity, we prefix function symbols with the \$ sign.

Consider again utterance 1.42. Its logical translation and the requisite pragmatic knowledge follows:

$$\begin{cases}\neg regrets(john,\$come(mary,party))\\(\forall x,y,z)(regrets(x,\$come(y,z)) \to come^i(y,z))\\(\forall x,y,z)(\neg regrets(x,\$come(y,z)) \to come^d(y,z))\end{cases} \tag{3.2}$$

This theory yields the stratified semantic tableau given in figure 3.1.

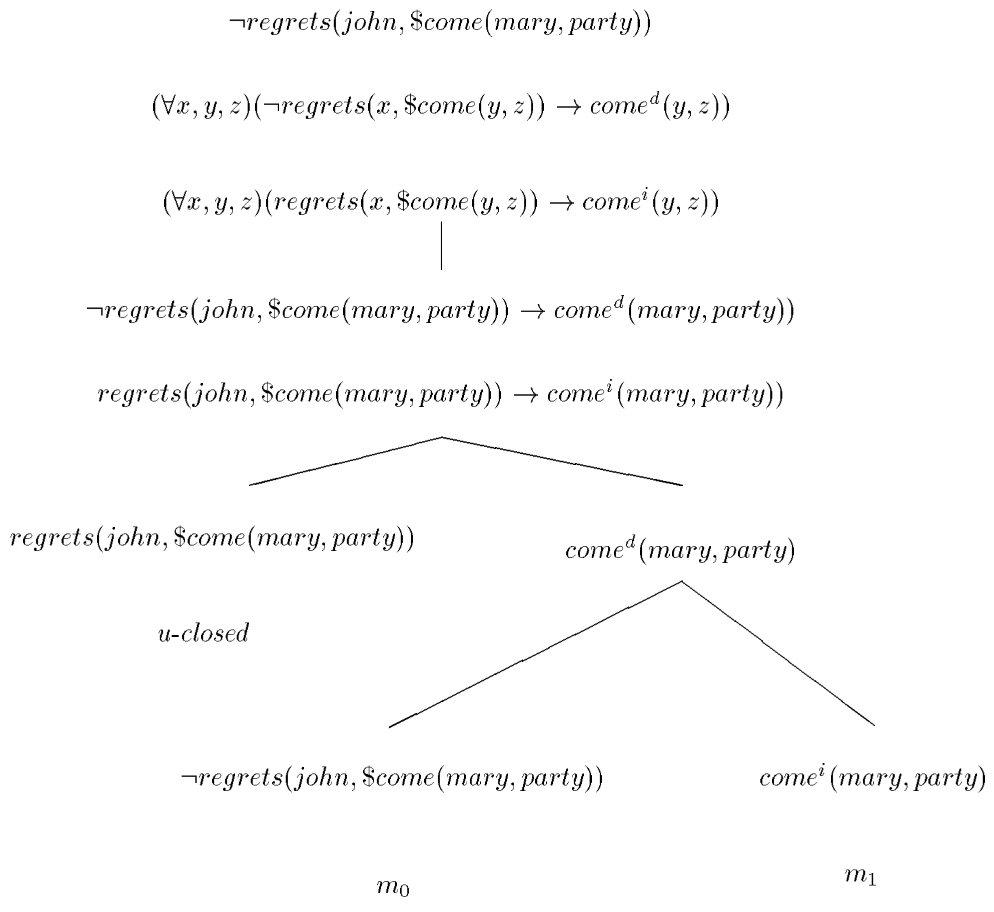
$\neg regrets(john,\$come(mary,party))$
$(\forall x,y,z)(\neg regrets(x,\$come(y,z)) \to come^d(y,z))$
$(\forall x,y,z)(regrets(x,\$come(y,z)) \to come^i(y,z))$
$\neg regrets(john,\$come(mary,party)) \to come^d(mary,party))$
$regrets(john,\$come(mary,party)) \to come^i(mary,party))$
$regrets(john,\$come(mary,party))$
$come^d(mary,party)$
u-closed
$\neg regrets(john,\$come(mary,party))$
$come^i(mary,party)$
m_0
m_1

Figure 3.1: Stratified tableau for *John does not regret that Mary came to the party*

The tableau yields two model schemata; in both of them, it is defeasibly inferred that *Mary came to the party.* The tableau in figure 3.1 is exhausted if one considers a sorted language to have been used and no instantiations over sentences have been made.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg regrets(john, \$come(mary, party)$		$come^d(mary, party)$
m_1	$\neg regrets(john, \$come(mary, party)$	$come^i(mary, party)$	$come^d(mary, party)$

As expected, the model-ordering relation $<$ (definition 2.6.8) establishes m_1 as the optimistic model for the theory because it contains as much information as m_0 and an extra default inference has been applied in it.

Reconsider now the utterance (1.45): *John does not regret that Mary came to the party because she didn't come.* An appropriate formalization in stratified logic yields again two model schemata:

$$\begin{cases} \neg regret(john, \$come(mary, party)) \wedge \neg come(mary, party) \\ (\forall x, y, z)(regrets(x, \$come(y, z)) \rightarrow come^i(y, z)) \\ (\forall x, y, z)(\neg regrets(x, \$come(y, z)) \rightarrow come^d(y, z)) \end{cases} \tag{3.3}$$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg regrets(john, \$come(mary, party))$ $\neg come(mary, party)$		$come^d(mary, party)$
m_1	$\neg regrets(john, \$come(mary, party))$ $\neg come(mary, party)$	$come^i(mary, party)$	$come^d(mary, party)$

For each of the two schemata, an $\mathcal{SL}$ structure can be found that *u-satisfies* the initial theory. However there is a fundamental difference between the two schemata: the first one is *i-satisfiable* while the second one is not. For example

$$\{\neg regrets^i(john, \$come(mary, party))\} \cup \{\neg come^i(mary, party)\} \cup \emptyset^d \models^i m_0 \tag{3.4}$$

but there is no $\mathcal{SL}$ structure such that

$$\mathcal{SL} \models^i m_1. \tag{3.5}$$

Rational agents tend to notice if something goes wrong. A model schema such as m_1 is a good example for this: any agent who notices that an infelicitously defeasible inference is cancelled, as it happens in m_1, will treat that model as infelicitous. We will make this idea precise in definition 3.2.2. This intuition gives us a good reason to discard the infelicitous model schemata, if possible. In this case, we are left with model m_0 where some felicitously defeasible information is cancelled. This corresponds entirely to our expectations: the initial utterance is felicitously described by only one model in which a presupposition has been cancelled. Formally, m_0 is *u-satisfiable* but not *d-satisfiable*. We make the following conjecture:

Conjecture 3.1.1 *An utterance is infelicitous if there is no stratified structure that i-satisfies its logical translation.*

In other words, if all the model schemata that characterize an utterance are not *i-satisfiable*, then that utterance is infelicitous. Obviously, this is not the case for theory 3.3. But reconsider utterance (1.44): *John regrets that Mary came to the party but she didn't come.* An appropriate formalization in stratified logic yields only one model schema that is *u-satisfiable*, but not *i-satisfiable*:

$$\left\{ \begin{array}{l} regret(john, \$come(mary, party)) \wedge \neg come(mary, party) \\ (\forall x, y, z)(regrets(x, \$come(y, z)) \rightarrow come^i(y, z)) \\ (\forall x, y, z)(\neg regrets(x, \$come(y, z)) \rightarrow come^d(y, z)) \end{array} \right. \tag{3.6}$$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$regrets(john, \$come(mary, party))$ $\neg come(mary, party)$	$come^i(mary, party)$	$come^d(mary, party)$

The theory we have obtained in stratified logic is still *u-consistent*. For example,

$$\{regret(john, \$come(mary, party)), \neg come(mary, party)\} \cup \emptyset^i \cup \emptyset^d \models 3.6 \tag{3.7}$$

We can add new utterances and interpret them, but the infelicity is signalled.

If one analyzes the above examples, she will notice that presuppositions can be associated with the felicitously defeasible information that belongs to a stratified model schema:

Conjecture 3.1.2 *The presuppositions and the implicatures of an utterance are given by the uncancelled defeasible information that belongs to an optimistic model schema of a theory described in terms of stratified logic.*

3.2 From lexical semantics to lexical pragmatics

3.2.1 Equating lexical semantics to lexical pragmatics — the first attempt

Consider the word *bachelor*. A semantic definition will specify that a bachelor is an unmarried male adult. However, a statement such as

(**3.1**) My cousin is not a bachelor.

is *usually* uttered with respect to objects that qualify as being bachelors, i.e., male adults. We say *usually* because is acceptable to repair a misuse of the word *bachelor*, (3.2) and (3.3), or to reinforce it, (3.4):

(**3.2**) My cousin is not a bachelor, he is five years old.

(**3.3**) My cousin is not a bachelor, she is a female.

(**3.4**) My cousin is not a bachelor, he is married.

It seems that *bachelor* is a word associated with male adults, and its salient property is being or not being married. Being unmarried is a salient property associated with bachelors, but not being a bachelor does not automatically imply that the entity having this property is married. Examples (3.2) and (3.3) show this. We use stratified logic to provide a tentative definition for the semantics and pragmatics of the word *bachelor*.

$$B_S = \begin{cases} (\forall x)(bachelor(x) \rightarrow \neg married(x) \land male(x) \land adult(x)) \\ (\forall x)(\neg bachelor(x) \rightarrow married^i(x)) \\ (\forall x)(\neg bachelor(x) \rightarrow adult^d(x)) \\ (\forall x)(\neg bachelor(x) \rightarrow male^d(x)) \end{cases} \tag{3.8}$$

At the first sight, this definition seems correct. One may argue that usually the term *bachelor* is used only for persons, so a correct formalization would be:

$$(\forall x)((person(x) \land \neg bachelor(x)) \rightarrow married^i(x)), \tag{3.9}$$

but for the moment we keep it simple in order to emphasize our point. If one knows that x is a bachelor, she may infer that he is an unmarried male adult. If x is not a bachelor, it is very likely that he is married — more likely than being a child or a female. In other words, being *married* is the salient feature of a non-bachelor; to reflect this, we assign it a stronger level in stratified logic. Let us see how this definition works. Reconsider utterance (3.1). A complete formalization of the utterance and the relevant commonsense knowledge is given by theory:

$$B_S \cup \neg bachelor(cousin) \tag{3.10}$$

If one applies the stratified semantic tableau, she will obtain one optimistic model schema for the theory:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg bachelor(cousin)$		
		$married^i(cousin)$	
			$male^d(cousin)$
			$adult^d(cousin)$

3.2.2 Equating lexical semantics to lexical pragmatics — the last attempt

The model we have obtained for theory 3.10 reflects one's intuitions: my cousin is not a bachelor because most likely he is a married male adult. However, it is unreasonable to assume that a real-world domain formalizes only facts about bachelors and cousins. Assume that the initial world contains also the fact that there is a linguist called *Chris* in the environment. For the same utterance, a complete formalization is given by theory:

$$B_S \cup \{\neg bachelor(cousin), linguist(Chris)\} \tag{3.11}$$

The theory provides two felicitous optimistic model schemata:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg bachelor(cousin)$		
		$married^i(cousin)$	
			$male^d(cousin)$
			$adult^d(cousin)$
	$linguist(Chris)$		
	$\neg bachelor(Chris)$		
		$married^i(Chris)$	
			$male^d(Chris)$
			$adult^d(Chris)$
m_1	$\neg bachelor(cousin)$		
		$married^i(cousin)$	
			$male^d(cousin)$
			$adult^d(cousin)$
	$linguist(Chris)$		
	$bachelor(Chris)$		
	$\neg married(Chris)$		
	$male(Chris)$		$male^d(Chris)$
	$adult(Chris)$		$adult^d(Chris)$

Up to a point, the results seem correct. In both model schemata, the cousin is not a bachelor and therefore, he may be a married male adult, and Chris is a linguist. As expected there are two ways one can extend the initial interpretation: in one of them Chris is not a bachelor, in the other Chris is a bachelor. But obviously, something goes wrong because both these extensions carry the extra defeasible information that Chris is a male adult. In other words, the defeasible information *Chris is a male adult* occurs in *all* felicitous optimistic model schemata of the theory. Hence, one would be inclined to assert that *Chris is a male adult* is a presupposition for the initial utterance as well. This blatantly contradicts our intuitions. The problem we have just emphasized pertains to what is called "ungrounded inference" in the literature of nonmonotonic reasoning. The solution we propose in the next section takes advantage of the fact that presuppositions are triggered by utterances. Therefore, the use of a presuppositional environment is the element that determines a default conclusion to be inferred or not.

3.2.3 Lexical semantics versus lexical pragmatics — drawing the boundaries

The problem with theory 3.11 is that the universal instantiation rule (see page 39) quantifies over all the objects in a domain. So any object, independent or not of the fact that it qualifies as being a bachelor, will trigger the associated inferences. To circumvent this problem one should pay attention to the heart of the issue: defeasible inferences are triggered by pragmatic maxims, so they should be fired only when a statement containing the corresponding syntactic or lexical construct is used in an utterance. Therefore, it seems appropriate to formalize pragmatic maxims as rules having the following form:

$$\underline{\text{if}}\ \textit{uttered(something)}\ \underline{\text{then}}\ \textit{infer(something-else)}. \qquad (3.12)$$

We solve the problem in three steps:

1. We delineate clearly what is uttered from what is not by introducing a meta-logical construct, $uttered(x)$, that takes one argument: the logical translation of an utterance. The metalogical symbol *uttered* is subject to the same logical rules as well-formed formulas. For example, if $uttered(a \wedge b)$, then it is appropriate to conclude $uttered(a)$ and $uttered(b)$. Hence, the utterance of (3.1) will be formalized as:
$$uttered(\neg bachelor(cousin)) \qquad (3.13)$$

2. We formalize pragmatic inferences as rules having the form given in 3.12.

3. We allow the pragmatic rules to apply only to utterances. To accomplish this, we introduce a new quantifier $\forall^{Ut}$. The semantics of $\forall^{Ut}$ enhances definition 2.5.4 as

follows:

Definition 3.2.1 *A stratified valuation σ x-satisfies a formula α with respect to an utterance $u = \alpha_1$ if and only if*

- $\sigma \models^x \alpha$ *where α is as the one specified in definition 2.5.4.*
- $\sigma(\vec{v}/\vec{t}) \models^x \alpha_1 \rightarrow \alpha_2$ *for all $\vec{t}$ such that $uttered(\alpha_1(\vec{t}))$ and $\alpha = (\forall^{Ut}\vec{v})(\alpha_1 \rightarrow \alpha_2)$*

The second part of definition 3.2.1 says that a pragmatic inference $(\forall^{Ut}\vec{v})(\alpha_1 \rightarrow \alpha_2)$ is instantiated only for those objects $\vec{t}$ that belong to an utterance having the form $\alpha_1(\vec{t})$. Hence, only if the antecedent of the pragmatic rule has been uttered can that rule be applied.

Using the new quantifier, the semantic and pragmatic connotation for the word *bachelor* is:

$$B_{SP} = \left\{ \begin{array}{l} (\forall x)(bachelor(x) \rightarrow \neg married(x) \wedge male(x) \wedge adult(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow married^i(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow adult^d(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow male^d(x)) \end{array} \right. \quad (3.14)$$

According to definition 3.2.1, the rules $(\forall^{Ut}x)(\neg bachelor(x) \rightarrow is\_like^{i,d}(x))$ are trigerred only if there is an object *someone* in the universe of discourse and an utterance containing $\neg bachelor(someone)$ has been issued. The above definition provides now the expected results even if the universe of discourse contains other objects. Theory 3.15 provides two optimistic model schemata that correctly reflect one's intuitions.

$$B_{SP} \cup \{linguist(Chris), uttered(\neg bachelor(cousin))\} \quad (3.15)$$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg bachelor(cousin)$ $linguist(Chris)$ $\neg bachelor(Chris)$	 $married^i(cousin)$	 $male^d(cousin)$ $adult^d(cousin)$
m_1	$\neg bachelor(cousin)$ $linguist(Chris)$ $\neg married(Chris)$ $male(Chris)$ $adult(Chris)$	 $married^i(cousin)$	 $male^d(cousin)$ $adult^d(cousin)$

In the first schema, the cousin is not a bachelor, but according to the pragmatic maxims one is inclined to believe that he is a married male adult; Chris is a linguist and not a bachelor. In the second model, the cousin is not a bachelor, but according to the pragmatic maxims one is inclined to believe that he is a married male adult; Chris is an unmarried male adult and he is a linguist too. Both schemata contain the felicitously defeasible information that the cousin is a male adult, and this corresponds to our intuition: they are presuppositions of the utterance.

As seen, the use of a certain presuppositional environment — in this case the word *bachelor* — warrants the application of the appropriate default rule. One may argue that one would like to apply the same default for inferences strongly support that a given object is a bachelor, even though this property has not been explicitely uttered. Defining the boundaries beyond which such a property should apply is not a trivial issue. Therefore, we will ignore it for the rest of this thesis.

The meta-logical symbol *uttered* and the quantifier $\forall^{Ut}$ are syntactic sugar used to keep a theory as clean as possible. It means that all properties concerning stratified logic are true for this extension as well. To clarify this, note that we can formalize the same problem by introducing a new predicate *uttered* and treating a given utterance as a conjunction between its logical translation and its uttered units. Thus, the utterance of (3.1) would be formalized as

$$\neg bachelor(cousin) \wedge uttered(not(bachelor(cousin))) \tag{3.16}$$

In the second conjunct, bachelor is a function. Instead of $\forall^{Ut}$ one can write the pragmatic rules as

$$(\forall x)(uttered(not(bachelor(x))) \rightarrow adult^{d}(x)) \tag{3.17}$$

We formalize now our intuitions:

Definition 3.2.2 *Let Φ be a theory described in terms of stratified first-order logic that appropriately formalizes the semantics of lexical items and the pragmatics of lexical and syntactic constructs. The semantics of lexical terms is formalized using the quantifier $\forall$, while the pragmatic maxims are captured using $\forall^{Ut}$. Let $uttered(u)$ be the logical translation of a given utterance or set of utterances. We say that utterance u is* <u>infelicitous</u> *if and only if there is no stratified structure $\mathcal{SL}$ that i-satisfies $\Phi \cup uttered(u)$.*

Definition 3.2.3 *Let Φ be a theory described in terms of stratified first-order logic that appropriately formalizes the semantics of lexical items and the pragmatics of lexical and syntactic constructs. The semantics of lexical terms is formalized using the quantifier $\forall$, while the pragmatic maxims are captured using $\forall^{Ut}$. Let $uttered(u)$ be the logical translation of a given utterance or set of utterances. We say that utterance u* <u>presupposes</u> *or* <u>implicates</u> *p if and only if p^{d} was derived using pragmatic maxims in at least one felicitous optimistic* *model of the theory $\Phi \cup uttered(u)$, and if p^d is not cancelled by any stronger information $(\neg p, \neg p^i, \neg p^d)$ in any felicitous optimistic model schema of the theory. Symmetrically, utterance u* <u>presupposes</u> *or* <u>implicates</u> *$\neg p$ if and only if $\neg p^d$ was derived using pragmatic maxims in at least one felicitous optimistic model of the theory $\Phi \cup uttered(u)$, and if $\neg p^d$ is not cancelled by any stronger information (p, p^i, p^d) in any felicitous optimistic model schema of the theory. In both cases, $\Phi \cup uttered(u)$ is u-consistent.*

3.3 Lexical pragmatics at work

3.3.1 Presupposition cancellation

Definitions 3.2.2 and 3.2.3 provide the expected results for the examples studied so far. They explain why *Mary came to the party* is a presupposition of the utterance *John does not regret that Mary came to the party* and why *My cousin is a male adult* is a presupposition of the utterance *My cousin is not a bachelor.* They also explain why *John regrets that Mary came to the party but she did not come* is an infelicitous utterance. We show now, how the same definitions explain the presupposition cancellation phenomenon. Reconsider utterance (3.2). If one evaluates its logical translation against a knowledge base that formalizes both the semantics and the pragmatics of the word *bachelor* as given in 3.14 , she will obtain one felicitous optimistic model schema.

$$B_{SP} \cup uttered(\neg bachelor(cousin) \wedge \neg adult(x)) \tag{3.18}$$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg bachelor(cousin)$		
		$married^i(cousin)$	
			$male^d(cousin)$
	$\neg adult(cousin)$		$adult^d(cousin)$

As expected, the felicitously defeasible information, $adult^d(cousin)$, is overwritten by some stronger information, $\neg adult(cousin)$. Therefore, according to definition 3.2.3, only *My cousin is a male* will be assigned the status of presupposition for the utterance. The same type of presupposition cancellation occurred in (1.45).

3.3.2 Implicatures, infelicities, and cancelability

Implicatures trigerred by the maxim of quality

We have seen that by uttering u, a speaker conversationally implicates (according to the maxim of quality) that she believes u and has adequate evidence for it. If we consider only one agent, in a second-order stratified model, this can be formalized as:

$$(\forall^{Ut} x)(\mathit{uttered}(x) \to \mathit{believe}(x)) \tag{3.19}$$

To stay within first-order logic, we instantiate the above formula for our domain of interest. Assume that (3.5) is uttered.

(**3.5**) The book is on the table.

An appropriate translation that reflects the Gricean implicature follows:

$$\begin{cases} \mathit{uttered}(\mathit{on}(\mathit{book}, \mathit{table})) \\ (\forall^{Ut} x, y)(\mathit{on}(x, y) \to \mathit{believe}^i(\$\mathit{on}(x, y))) \end{cases} \tag{3.20}$$

If one utters (3.6), theory 3.21 will have one model schema that is infelicitous.

(**3.6**) The book is on the table, but I do not believe it.

$$\begin{cases} \mathit{uttered}(\mathit{on}(\mathit{book}, \mathit{table}) \wedge \neg \mathit{believe}(\$\mathit{on}(\mathit{book}, \mathit{table}))) \\ (\forall^{Ut} x, y)(\mathit{on}(x, y) \to \mathit{believe}^i(\$\mathit{on}(x, y))) \end{cases} \tag{3.21}$$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\mathit{on}(\mathit{book}, \mathit{table})$		
	$\neg \mathit{believe}(\$\mathit{on}(\mathit{book}, \mathit{table}))$	$\mathit{believe}^i(\$\mathit{on}(\mathit{book}, \mathit{table}))$	

Scalar Implicatures

Reconsider (1.20), *John says that some of the boys went to the theatre*, and its implicatures (1.21), *Not many/most/all of the boys went to the theatre.* An appropriate formalization is given in 3.22, where the second formula captures the felicitously defeasible scalar implicatures and the third formula reflects the relevant semantic information for *all.*

$$\begin{cases} \mathit{uttered}(\mathit{went}(\$\mathit{some}(\mathit{boys}), \mathit{theatre})) \\ \mathit{went}(\$\mathit{some}(\mathit{boys}), \mathit{theatre}) \to (\neg \mathit{went}^d(\$\mathit{many}(\mathit{boys}), \mathit{theatre}) \wedge \\ \quad \neg \mathit{went}^d(\$\mathit{most}(\mathit{boys}), \mathit{theatre}) \wedge \neg \mathit{went}^d(\$\mathit{all}(\mathit{boys}), \mathit{theatre})) \\ \mathit{went}(\$\mathit{all}(\mathit{boys}), \mathit{theatre}) \to (\mathit{went}(\$\mathit{most}(\mathit{boys}), \mathit{theatre}) \wedge \\ \quad \mathit{went}(\$\mathit{many}(\mathit{boys}), \mathit{theatre}) \wedge \mathit{went}(\$\mathit{some}(\mathit{boys}), \mathit{theatre})) \end{cases} \tag{3.22}$$

The theory provides one felicitous optimistic model that reflects the expected pragmatic inferences.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\mathit{went}(\$\mathit{some}(\mathit{boys}), \mathit{theatre})$		
			$\neg \mathit{went}^d(\$\mathit{most}(\mathit{boys}), \mathit{theatre})$
			$\neg \mathit{went}^d(\$\mathit{many}(\mathit{boys}), \mathit{theatre})$
	$\neg \mathit{went}(\$\mathit{all}(\mathit{boys}), \mathit{theatre})$		$\neg \mathit{went}^d(\$\mathit{all}(\mathit{boys}), \mathit{theatre})$

Assume now, that after a moment of thought, the same person utters:

(**3.7**) Some of the boys went to the theatre. In fact all of them went to the theatre.

Adding the extra utterance to the initial theory 3.22, $\mathit{uttered}(\mathit{went}(\$\mathit{all}(\mathit{boys}), \mathit{theatre}))$, one would obtain one felicitous optimistic model schema in which the conventional implicatures have been cancelled:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\mathit{went}(\$\mathit{some}(\mathit{boys}), \mathit{theatre})$		
	$\mathit{went}(\$\mathit{most}(\mathit{boys}), \mathit{theatre})$		$\neg \mathit{went}^d(\$\mathit{most}(\mathit{boys}), \mathit{theatre})$
	$\mathit{went}(\$\mathit{many}(\mathit{boys}), \mathit{theatre})$		$\neg \mathit{went}^d(\$\mathit{many}(\mathit{boys}), \mathit{theatre})$
	$\mathit{went}(\$\mathit{all}(\mathit{boys}), \mathit{theatre})$		$\neg \mathit{went}^d(\$\mathit{all}(\mathit{boys}), \mathit{theatre})$

Natural language disjunction

It has been argued that there is a strong inclination to interpret the natural disjunction as an exclusive or. Here is our formal explanation: If one considers the linguistic scale $\langle \mathit{and}, \mathit{or} \rangle$, the intuition can be explained as being an instantiation of the *scalar implicature* phenomenon. One can formalize utterance (3.8) and its scalar implicature as we have done in 3.23.

(**3.8**) Mary or Susan is in France.

$$\left\{ \begin{array}{l} \mathit{uttered}(\mathit{is\_in}(Mary, France) \vee \mathit{is\_in}(Susan, France)) \\ (\mathit{is\_in}(Mary, France) \vee \mathit{is\_in}(Susan, France)) \rightarrow \\ \quad \neg(\mathit{is\_in}^d(Mary, France) \wedge \mathit{is\_in}^d(Susan, France)) \end{array} \right. \tag{3.23}$$

The stratified semantic tableau provides two felicitous optimistic models:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\mathit{is\_in}(Mary, France)$		$\neg \mathit{is\_in}^d(Susan, France)$
m_1	$\mathit{is\_in}(Susan, France)$		$\neg \mathit{is\_in}^d(Mary, France)$

The two schemata account for our inclination to treat the natural disjunction as an exclusive or. Notice that they are obtained on the basis of the Gricean Maxims. Scalar implicatures are felicitously defeasible: there is only one way in which one can *u-satisfy* utterance (3.9).

(**3.9**) Mary will wash the car or do the dishes. Actually, she will do both of them.

This is done by building an $\mathcal{SL}$ structure such as:

$$\{will\_wash(mary, car), will\_do(mary, dishes)\} \cup \emptyset^i \cup \emptyset^d. \tag{3.24}$$

Chapter 4

Solving the projection problem

The Achille's heel for any pragmatic theory is its vulnerability to the projection problem. A solution for the projection problem should explain which of the pragmatic inferences are inherited by a structure that uses the constructs that bear them, and why. Gazdar [1979] makes a clear delineation between the projection of implicatures and presuppositions. Other researchers [Karttunen and Peters, 1979, Soames, 1979, Soames, 1982] focus only on the projection of presuppositions. We are interested in a unified explanation for both phenomena. We believe that there is no fundamental difference between presuppositions and implicatures: they are both triggered by pragmatic rules and they are both felicitously or infelicitously defeasible. Moreover, we believe that a solution for the projection problem should not differ from a solution for individual utterances: it would be unnatural to consider that conversants apply different rules when they utter simple utterances than when they utter complex ones.

The solution we propose here adds nothing to the solution developed so far for simple utterances. As a matter of fact, the direct presupposition cancellation phenomenon is treated as a projection problem by many theories of presuppositions. We have already shown how direct cancellation, such as the one found in (1.45), (3.2), or (3.3), is handled by our formalism. We show now that the same definition 3.2.3 can explain how presuppositions are inherited in more complex cases as well.

4.1 Disjunctive utterances

Consider the following utterances and some of their associated presuppositions:

(**4.1**) Chris is not a bachelor or he regrets that Mary came to the party.

(**4.2**) Chris is a bachelor or a spinster.

(**4.3**) $\triangleright$ Chris is a (male) adult.

(**4.4**) Either John is away or his wife is away.

(**4.5**) Either John has no wife or his wife is away.

(**4.6**) $\triangleright$ (John has a wife).

As we have already seen, *Chris is not a bachelor* presupposes that *Chris is a male adult*; *Chris regrets that Mary came to the party* presupposes that *Mary came to the party.* There is no contradiction between these two presuppositions, so one would expect a conversant to infer both of them if she hears an utterance such as (4.1). However, when one examines utterance (4.2), she observes immediately that there is a contradiction between the presuppositions carried by the individual components. Being a bachelor presupposes that *Chris is a male*, while being a spinster presupposes that *Chris is a female.* Normally, we would expect a conversant to notice this contradiction and to drop each of these elementary presuppositions when she interprets (4.2). The same analysis can be done for examples (4.4) and (4.5). The first utterance inherits the elementary presupposition (4.6), while the second one does not.

4.1.1 *Or* — non-cancellation

We now study in detail how stratified logic and the model-ordering relation capture one's intuitions.

An appropriate formalization for utterance (4.1) and the necessary semantic and pragmatic knowledge is given in 4.1.

$$\left\{\begin{array}{l}
uttered(\neg bachelor(Chris) \vee regret(Chris, \$come(Mary, party))) \\
(\neg bachelor(Chris) \vee regret(Chris, \$come(Mary, party))) \rightarrow \\
\quad \neg(\neg bachelor^{d}(Chris) \wedge regret^{d}(Chris, \$come(Mary, party))) \\
\neg male(Mary) \\
(\forall x)(bachelor(x) \rightarrow male(x) \wedge adult(x) \wedge \neg married(x)) \\
(\forall^{Ut} x)(\neg bachelor(x) \rightarrow married^{i}(x)) \\
(\forall^{Ut} x)(\neg bachelor(x) \rightarrow adult^{d}(x)) \\
(\forall^{Ut} x)(\neg bachelor(x) \rightarrow male^{d}(x)) \\
(\forall^{Ut} x, y, z)(\neg regret(x, \$come(y, z)) \rightarrow come^{d}(y, z)) \\
(\forall^{Ut} x, y, z)(regret(x, \$come(y, z)) \rightarrow come^{i}(y, z))
\end{array}\right. \tag{4.1}$$

Besides the translation of the utterance, the initial theory contains a formalization of the felicitously defeasible implicature that natural disjunction is used as an exclusive *or*, the lexical semantics for the word *bachelor*, and the lexical pragmatics for *bachelor* and *regret.*

The stratified semantic tableau generates 12 model schemata. Among them, model m_8 and m_{10} are infelicitous. The ordering relation defined in 2.6.8 generates over the space of model schemata, the lattice given in figure 4.1. Here follow the model schemata and their ordering.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$regret(Chris, \$come(Mary, party))$ $\neg male(Mary)$ $\neg bachelor(Mary)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$	$come^i(Mary, party)$	$come^d(Mary, party)$
m_1	$regret(Chris, \$come(Mary, party))$ $\neg male(Mary)$ $\neg bachelor(Mary)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$	$come^i(Mary, party)$	
m_2	$regret(Chris, \$come(Mary, party))$ $\neg male(Mary)$ $\neg bachelor(Mary)$ $\neg bachelor(Chris)$	$come^i(Mary, party)$	$come^d(Mary, party)$
m_3	$regret(Chris, \$come(Mary, party))$ $\neg male(Mary)$ $\neg bachelor(Mary)$ $\neg bachelor(Chris)$	$come^i(Mary, party)$	
m_4	$regret(Chris, \$come(Mary, party))$ $\neg male(Mary)$ $\neg bachelor(Mary)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$	$come^i(Mary, party)$	$bachelor^d(Chris)$ $come^d(Mary, party)$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_5	$regret(Chris,$		
	$\$come(Mary, party))$		
			$bachelor^d(Chris)$
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
	$\neg married(Chris)$		
	$adult(Chris)$		
	$male(Chris)$		
		$come^i(Mary, party)$	
m_6	$regret(Chris,$		
	$\$come(Mary, party))$		
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
	$\neg bachelor(Chris)$		$bachelor^d(Chris)$
		$come^i(Mary, party)$	$come^d(Mary, party)$
m_7	$regret(Chris,$		
	$\$come(Mary, party))$		
			$bachelor^d(Chris)$
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
	$\neg bachelor(Chris)$		$bachelor^d(Chris)$
		$come^i(Mary, party)$	
m_8			$\neg regret^d(Chris,$
			$\$come(Mary, party))$
	$\neg bachelor(Chris)$		
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
*	$\neg married(Chris)$	$married^i(Chris)$	
	$adult(Chris)$		$adult^d(Chris)$
	$male(Chris)$		$male^d(Chris)$
m_9			$\neg regret^d(Chris,$
			$\$come(Mary, party))$
	$\neg bachelor(Chris)$		
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
		$married^i(Chris)$	
			$adult^d(Chris)$
			$male^d(Chris)$

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_{10}	$\neg bachelor(Chris)$		
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
$*$	$\neg married(Chris)$	$married^{i}(Chris)$	
	$adult(Chris)$		$adult^{d}(Chris)$
	$male(Chris)$		$male^{d}(Chris)$
m_{11}	$\neg bachelor(Chris)$		$bachelor^{d}(Chris)$
	$\neg male(Mary)$		
	$\neg bachelor(Mary)$		
		$married^{i}(Chris)$	
			$adult^{d}(Chris)$
			$male^{d}(Chris)$

$$\begin{tikzcd}
 & & & 4^{*} & \\
9^{*} & 2 \arrow[r, leftrightarrow] & 6 \arrow[ur] & 5 \arrow[u] & 0 \arrow[ul] \\
11 \arrow[u] & 3 \arrow[u] \arrow[ur] \arrow[r, leftrightarrow] & 7 \arrow[ul] \arrow[u] \arrow[ur] & & 1 \arrow[ul] \arrow[u]
\end{tikzcd}$$

Figure 4.1: The felicitous models and their ordering for utterance *Chris is not a bachelor or he regrets that Mary came to the party.* The models labelled with an asterisk sign (4 and 9) are the most optimistic ones.

The model-ordering relation gives m_4 and m_9 as optimistic models for the utterance. The presupposition candidates for the utterance are: $bachelor^{d}(Chris)$, $come^{d}(Mary, party)$, $\neg regret^{d}(Chris, \$come(Mary, party))$, $adult^{d}(Chris)$, and $male^{d}(Chris)$. Note that the optimistic models contain $\neg bachelor(Chris)$ and $regret(Chris, \$come(Mary, party))$, so according to definition 3.2.3 only $come^{d}(Mary, party)$, $adult^{d}(Chris)$, and $male^{d}(Chris)$ will be projected as presuppositions for the complex utterance (4.1). Therefore, according to our theory, *Mary came to the party; Chris is a male;* and *Chris is an adult* are labelled as presuppositions of utterance (4.1).

4.1.2 *Or* — cancellation

Consider now utterance (4.2). An appropriate translation in stratified logic and the relevant semantic and pragmatic knowledge are given in theory 4.2. Besides the translation of the utterance, the initial theory contains a formalization of the felicitously defeasible implicature that natural disjunction is used as an exclusive *or* and the lexical semantics and pragmatics for the word *bachelor* and *spinster*.

$$\left\{\begin{array}{l} uttered(bachelor(Chris) \vee spinster(Chris)) \\ (bachelor(Chris) \vee spinster(Chris) \rightarrow \\ \quad \neg(bachelor^d(Chris) \wedge spinster^d(Chris)) \\ (\forall x)(bachelor(x) \rightarrow male(x) \wedge adult(x) \wedge \neg married(x)) \\ (\forall x)(spinster(x) \rightarrow female(x) \wedge adult(x) \wedge \neg married(x)) \\ (\forall x)(male(x) \leftrightarrow \neg female(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow married^i(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow adult^d(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow male^d(x)) \\ (\forall^{Ut} x)(\neg spinster(x) \rightarrow married^i(x)) \\ (\forall^{Ut} x)(\neg spinster(x) \rightarrow adult^d(x)) \\ (\forall^{Ut} x)(\neg spinster(x) \rightarrow female^d(x)) \end{array}\right. \tag{4.2}$$

The corresponding stratified semantic tableau yields 16 models. Among them, four are infelicitous (m_0, m_4, m_8 and m_{12}).

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg bachelor(Chris)$		
	$adult(Chris)$		
	$female(Chris)$		
*	$\neg married(Chris)$	$married^i(Chris)$	
	$\neg male(Chris)$		
m_1	$spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg bachelor(Chris)$		
	$adult(Chris)$		
	$female(Chris)$		$female^d(Chris)$
	$\neg married(Chris)$		
	$\neg male(Chris)$		
m_2	$spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg bachelor(Chris)$		
	$adult(Chris)$		$adult^d(Chris)$
	$female(Chris)$		
	$\neg married(Chris)$		
	$\neg male(Chris)$		

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_3	$spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg bachelor(Chris)$		
	$adult(Chris)$		
	$female(Chris)$		
	$\neg married(Chris)$		
	$\neg male(Chris)$		
m_4	$spinster(Chris)$		
	$\neg bachelor(Chris)$		$\neg bachelor^d(Chris)$
	$adult(Chris)$		
	$female(Chris)$		
$*$	$\neg married(Chris)$	$married^i(Chris)$	
	$\neg male(Chris)$		
m_5	$spinster(Chris)$		
	$\neg bachelor(Chris)$		$\neg bachelor^d(Chris)$
	$adult(Chris)$		
	$female(Chris)$		$female^d(Chris)$
	$\neg married(Chris)$		
	$\neg male(Chris)$		
m_6	$spinster(Chris)$		
	$\neg bachelor(Chris)$		$\neg bachelor^d(Chris)$
	$adult(Chris)$		$adult^d(Chris)$
	$female(Chris)$		
	$\neg married(Chris)$		
	$\neg male(Chris)$		
m_7	$spinster(Chris)$		
	$\neg bachelor(Chris)$		$\neg bachelor^d(Chris)$
	$adult(Chris)$		
	$female(Chris)$		
	$\neg married(Chris)$		
	$\neg male(Chris)$		
m_8	$bachelor(Chris)$		
$*$	$\neg married(Chris)$	$married^i(Chris)$	
	$adult(Chris)$		
	$male(Chris)$		
	$\neg spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg female(Chris)$		
m_9	$bachelor(Chris)$		
	$\neg married(Chris)$		
	$adult(Chris)$		
	$male(Chris)$		$male^d(Chris)$
	$\neg spinster(Chris)$		$\neg spinster^d(Chris)$
	$\neg female(Chris)$		

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_{10}	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$		$adult^d(Chris)$ $\neg spinster^d(Chris)$
m_{11}	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$		$\neg spinster^d(Chris)$
m_{12} $*$	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$	$married^i(Chris)$	$\neg bachelor^d(Chris)$
m_{13}	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$		$\neg bachelor^d(Chris)$ $male^d(Chris)$
m_{14}	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$		$\neg bachelor^d(Chris)$ $adult^d(Chris)$
m_{15}	$bachelor(Chris)$ $\neg married(Chris)$ $adult(Chris)$ $male(Chris)$ $\neg spinster(Chris)$ $\neg female(Chris)$		$\neg bachelor^d(Chris)$

$$\begin{tikzcd}
 & 10^{*} & & 9^{*} & & & & 5^{*} & & 6^{*} & \\
14 \arrow[ur] & & 11 \arrow[ul] \arrow[ur] & & 13 \arrow[ul] & & 1 \arrow[ur] & & 7 \arrow[ul] \arrow[ur] & & 2 \arrow[ul] \\
 & & 15 \arrow[ull] \arrow[u] \arrow[urr] & & & & & & 3 \arrow[ull] \arrow[u] \arrow[urr] & &
\end{tikzcd}$$

Figure 4.2: The felicitous models and their ordering for utterance *Chris is a bachelor or a spinster.* The models labelled with an asterisk sign (5, 6, 9, and 10) are the most optimistic ones.

The model schemata exhibit the ordering given in figure 4.2. Among the presupposition candidates ($\neg bachelor^d(Chris)$, $female^d(Chris)$, $adult^d(Chris)$, $male^d(Chris)$, and $\neg spinster^d(Chris)$), only $adult^d(Chris)$ is not cancelled by some stronger information in the optimistic felicitous models. Therefore, *Chris is an adult* is the only presupposition for the utterance.

4.2 Conditional utterances

4.2.1 *If … then* — non-cancellation

An argument similar to that given for disjunctive utterances can be made for conditionals. Consider utterance (4.7). Its consequent presupposes (4.8). The antecedent contains no information that may cancel the presupposition. Hence, one's expectations would be that *Sue came to the party* is a presupposition of the whole conditional. An appropriate formalization in stratified logic is given in 4.3.

(**4.7**) If Mary came to the party then John will regret that Sue came to the party.

(**4.8**) $\rhd$ Sue came to the party.

$$\begin{cases} uttered(come(Mary, party) \rightarrow regret(John, \$come(Sue, party))) \\ (\forall^{Ut} x, y, z)(regret(x, \$come(y, z)) \rightarrow come^i(y, z)) \\ (\forall^{Ut} x, y, z)(\neg regret(x, \$come(y, z)) \rightarrow come^d(y, z)) \end{cases} \tag{4.3}$$

Theory 4.3 yields two felicitous optimistic model schemata. One of them contains a presupposition candidate, $come^d(Sue, party)$, that is not cancelled in the other model. Hence, *Sue came to the party* is projected as a presupposition for the conditional.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$regret(John, \$come(Sue, party))$	$come^i(Sue, party)$	$come^d(Sue, party)$
m_1	$\neg come(Mary, party)$		

4.2.2 *If ... then* — cancellation

Consider utterance (4.9). Its consequent presupposes (4.10), but the antecedent is no longer independent. If the antecedent holds, then the presupposition is overwritten; if it does not, the presupposition is cancelled. Therefore, one would not expect in this case the presupposition to be inherited by the whole conditional. An appropriate formalization in stratified logic is given in 4.4.

(**4.9**) If Mary came to the party then John will regret that she did.

(**4.10**) ($\triangleright$) $\not\triangleright$ Mary came to the party.

$$\begin{cases} uttered(come(Mary, party) \rightarrow regret(John, \$come(Mary, party))) \\ (\forall^{Ut} x, y, z)(regret(x, \$come(y, z)) \rightarrow come^{i}(y, z)) \\ (\forall^{Ut} x, y, z)(\neg regret(x, \$come(y, z)) \rightarrow come^{d}(y, z)) \end{cases} \tag{4.4}$$

The stratified tableau method determines two models. Model m_0 is more informative than model m_1 but it is not weaker (see definitions 2.6.6 and 2.6.7). This is the reason that model m_0 is *not* more optimistic than model m_1. One of the felicitous optimistic models, m_0, contains a presupposition candidate, $come^{d}(Mary, party)$, but the other one, m_1, cancels it, $\neg come(Mary, party)$. Therefore, the presupposition is not projected.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$regret(John, \$come(Mary, party))$	$come^{i}(Mary, party)$	$come^{d}(Mary, party)$
m_1	$\neg come(Mary, party)$		

4.3 Beyond complex utterances

Most theories of presupposition over-simplify the projection problem. They consider [Gazdar, 1979, Karttunen and Peters, 1979, Kay, 1992, Sandt, 1992, Soames, 1982, Wilson and Sperber, 1979, Zeevat, 1992] that it is enough if a theory can successfully explain how presuppositions are inherited in complex utterances such as disjunctions or conditionals. No inquiry is made into the status of presuppositions in sequence of utterances. As we have discussed in section 1.2.3, the most puzzling consequence is that these theories assign a dual life to presuppositions: in the initial stage, as members of a simple or complex utterance, they are defeasible. However, after that utterance is analyzed, there is no possibility left of cancelling that presupposition. But it is natural to have presuppositions that are inferred and cancelled along a sequence of utterances. Consider for example that Jane has two friends — John Smith and John McEnroe — and that her roomate Mary met only John Smith, a married fellow. Assume now that Jane has a conversation with Mary in which she mentions only the name John because she is not aware that Mary does not know that are two Johns. In this context, is natural for Mary to get confused and to come to wrong conclusions. For example, she may say that *John is not a bachelor*. She knows that John Smith is a married male, so this makes sense for her. At this point Jane realizes that Mary misunderstands her: all the time she was talking about her cousin, John McEnroe. The utterance in (4.11) is a possible answer that Jane may give to Mary in order to clarify the problem.

(4.11) a. John is a not a bachelor.
b. I regret that you have misunderstood me.
c. He is only five years old.
d. You realize he cannot date women.
e. It is not he who is not a bachelor!

The first utterance in the sequence presupposes (4.12).

(4.12) $\triangleright$ John is a male adult.

Utterance (4.11)b warns Mary that is very likely she misunderstood a previous utterance (4.13). The warning is conveyed by implicature — it is not entailed.

(4.13) $\triangleright$ The hearer misunderstood the speaker.

At this point, the hearer, Mary, starts to believe that one of her previous utterances has been elaborated on a false assumption, but she does not know which one. The third utterance (4.11)c comes to clarify the issue. It explicitly expresses that John is not an adult. Therefore, it cancels the early presupposition (4.12).

(4.14) $\not\triangleright$ John is an adult.

Note that there is a gap of one statement between the generation and the cancellation of this presupposition. In fact, it is more likely that this presupposition has been inferred sometime earlier in the dialog. The fourth utterance (4.11)d emphasizes the inappropriateness of presupposing that John is an adult. Again, the information (4.15) is conveyed using pragmatic mechanisms, as it was done in the second statement.

(4.15) $\triangleright$ John cannot date women.

The last statement clarifies the issue. The speaker, Jane, acknowledges that she sees now where the misunderstanding of the hearer comes from. It may be the case that someone else, for example John Smith, qualifies for not being a bachelor. This last piece of information (4.16) is conveyed using a cleft construct.

(4.16) $\triangleright$ There is another person who is not a bachelor.

Let us see how the above line of reasoning is reflected by our computational method. We consider that the initial knowledge base contains the relevant information for interpreting the sequence of utterances:

- the semantic information that bachelors are unmarried male adults;
- the pragmatic lexical information that those who are not bachelors are likely to be married male adults;
- first-order instances of the pragmatic lexical information that factive verbs such as *regret* and *realize* presuppose their complement;
- a first-order instance of the pragmatic syntactic information that cleft constructs presuppose that there exists another agent who has performed the corresponding action. More specifically, the last formula in the theory says that if someone uttered that x is not a bachelor using a cleft construct, then the universe of discourse should contain another object y that has the property of being a bachelor. The existence of this object is not entailed; therefore we formalize it as defeasible, $E!^d$.

The existential presupposition is formalized using the predicate $E!$. For a thorough analysis concerning existential presuppositions and their formalization in stratified logic, the reader is referred to chapter 5.

The initial theory containing the translation of the first utterance (4.11)a and the appropriate knowledge is given in theory 4.5.

$$\Phi_1 = \left\{ \begin{array}{l} uttered(\neg bachelor(John)) \\ (\forall x)(bachelor(x) \rightarrow male(x) \wedge adult(x) \wedge \neg married(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow married^i(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow adult^d(x)) \\ (\forall^{Ut} x)(\neg bachelor(x) \rightarrow male^d(x)) \\ (\forall^{Ut} x, y)(\neg regret(x, \$misunderstood(y, x)) \rightarrow misunderstood^d(y, x)) \\ (\forall^{Ut} x, y, z)(regret(x, \$misunderstood(y, x)) \rightarrow misunderstood^i(y, z)) \\ (\forall^{Ut} x, y)(\neg realize(x, \$cannot\_date\_women(y)) \rightarrow \neg date\_women^d(y)) \\ (\forall^{Ut} x, y)(realize(x, \$cannot\_date\_women(y)) \rightarrow \neg date\_women^i(y)) \\ (\forall^{Ut} x)(cleft(\$not\_bachelor(x)) \rightarrow (\exists y)(\neg bachelor(y) \wedge E!^d(y))) \end{array} \right. \tag{4.5}$$

The stratified semantic tableau yields one felicitous optimistic model schema for the theory, in which one can presuppose that John is a male adult.

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$\neg bachelor(John)$	$married^i(John)$	$adult^d(John)$ $male^d(John)$

When sentence (4.11)b is uttered, we have to examine a new theory that is obtained by adding $uttered(regret(speaker, \$misunderstood(hearer, speaker)))$ to theory 4.5. Therefore, the new theory is:

$$\Phi_2 = \Phi_1 \cup uttered(regret(speaker, \$misunderstood(hearer, speaker))) \tag{4.6}$$

Theory Φ_2 has 16 model schemata. Among them, eight are felicitous. The ordering relation $(<)$ leaves only four felicitous optimistic models; none of these models cancels any pragmatic information. Therefore, (4.12) and (4.13) are projected as presuppositions of the first two statements.

Assume now that sentence (4.11)c is uttered. The new theory, Φ_3, has only eight model schemata, all of them felicitous. Four model schemata are the most optimistic ones. Being an adult is a presupposition candidate for Φ_3 as well, but the information is now cancelled in each optimistic model. Therefore, only *John is a male* and *The hearer misunderstood the speaker* are projected as presuppositions of the first three sentences.

$$\Phi_3 = \Phi_2 \cup uttered(male(John) \wedge \neg adult(John)) \tag{4.7}$$

When the speaker utters sentence (4.11)d, the new theory to be analyzed is 4.8.

$$\Phi_4 = \Phi_3 \cup uttered(realize(hearer, \$cannot\_date\_women(John))) \tag{4.8}$$

Theory Φ_4 has 16 model schemata, all felicitous. Four model schemata are the most optimistic ones, and they presuppose that *John is a male*, *The hearer misunderstood the speaker*, and *John does not date women.*

When the speaker utters the last sentence (4.11)e, the theory to be evaluated is:

$$\Phi_5 = \Phi_4 \cup uttered(cleft(\neg bachelor(John))) \tag{4.9}$$

This has 24 felicitous model schemata and eight felicitous optimistic model schemata. There are eight optimistic possibilities of constructing stratified valuations for the sequence of utterances. They may differ in the way the world is carved up: in one world the speaker can be a bachelor, in the other he cannot. No restrictions have been specified. However, independent of the way we choose to solve this problem, in all eight optimistic models, the following presuppositions are never cancelled: *John is a male*; *The hearer misunderstood the speaker*; *John does not date women*; and *There is someone else who is not a bachelor.*

The results obtained here offer a computational explanation for Geis's work [Geis, 1982]. In his study of the language of TV advertising, Geis noticed that most of the information that bombards one during the commercial breaks is conveyed using pragmatic effects. Implicatures and presuppositions are the carriers of this information. Pragmatic inferences are defeasible, so according to Geis, a commercial that pragmatically conveys some information to the audience does not commit the originator of the commercial to the same extent as something specifically uttered. One can observe that for any of the examples discussed above, there is more than one interpretation. For example, utterance (4.2) is satisfiable regardless of whether Chris is a bachelor or a spinster. However, the pragmatic inference that *Chris is an adult* is not contradicted in any optimistic model. So even if it is defeasible information, there is a strong tendency to believe that *Chris is an adult.* This is exactly the reason that pragmatic inferences are psycholinguistically strong.

Chapter 5

A Special Case of Pragmatic Inference: The Existential Presupposition

It is common knowledge that a rational agent is inclined to presuppose the existence of definite references that occur in utterances. Hearing or uttering the examples below, a rational agent presupposes that the cheese, children, and car physically exist.

(**5.1**) The cheese I bought yesterday is very bad.

(**5.2**) I really don't know what to do with my children anymore.

(**5.3**) Sorry I couldn't make it; my car broke on my way.

However, day-to-day English provides an impressive number of cases when existential presuppositions are not inferred, or when they are defeated by some common sense knowledge (see [Hirst, 1991] for a comprehensive study). One can explicitly speak of nonexistence (5.4); events and actions that do not occur (5.5); existence at other times (5.6); or fictional and imaginary objects (5.7).

(**5.4**) No one got an A+ in this course.

(**5.5**) John's party is cancelled.

(**5.6**) Gödel was a brilliant mathematician.

(**5.7**) Sherlock Holmes is smarter than any other detective.

Note that the simple dichotomy found in most approaches to presupposition between existent and nonexistent objects is not enough for a full account of natural language expressiveness.

While trying to explain the whole phenomenon and to provide solutions for the projection problem, linguists have often omitted any explanation for the existential commitment of definite references or their explanation has been a superficial one. Similarly, philosophers who have studied existence and nonexistence have been more concerned with providing formal tools for manipulation of nonexistent objects than tools to capture our common sense commitment. This puts us in a difficult position. From a linguistic perspective, the literature provides a good set of theories able to more or less explain the commitment to the presupposed truth of factives and the like but not the existential commitment of definite references. From a philosophical perspective, we have quite a few theories which deal with existence and nonexistence, but they too offer no explanation for existential commitment.

We are interested in a theory that is able to handle all pragmatic phenomena in a unified manner. Therefore, we expect a solution for the existential presuppositions to be similar with a solution for factives and the like. This chapter reviews some of the most significant works that deal with nonexistence: we study their ability to reflect the presupposition that definite referents exist, and we criticize possible extensions of the formalisms that we study for reflecting the types of pragmatic inferences that we have presented in chapter 1.

5.1 On Frege-Russell-Quine nonexistence

Frege [1892] makes a sharp distinction between sense and reference — *nominatum*. According to Frege, constructions such as *the heavenly body which has the greatest distance from the earth* or *the series with the least convergence* [Frege, 1892, p. 87] certainly have some sense, but it is doubtful they have a *nominatum*.

> Therefore, the grasping of a particular sense does not with certainty warrant a corresponding nominatum [Frege, 1892, p. 87].

However, from a logical perspective, Frege rejects the words which do not refer: *He who does not acknowledge the nominatum cannot ascribe or deny a predicate to it.* For Frege, whenever something is asserted, the presupposition taken *for granted* is that the employed proper names, simple or compound, have nominata [Frege, 1892, p. 95]. Therefore, it is impossible to account for the cancellation of the pragmatic inferences. Moreover, sentences such as (5.4), (5.5), (5.6), or (5.7) are impossible to represent.

Russell was aware of the problems which occur when the denotation is absent in a universe where denoting phrases *express* a meaning and *denote* a denotation. We follow here Hirst's [1991] line of reasoning in presenting this mistake.

In a Russellian ontological space, an appropriate translation of sentence (5.8) is formula 5.1.

(**5.8**) Dragons exist.

$$(\exists x)(dragon(x)) \tag{5.1}$$

According to Russell, sentence (5.8) is false because there are no dragons:

$$(\forall x)(\neg dragon(x)) \tag{5.2}$$

If someone is committed to the truth of formula 5.2, then she is obviously committed to the falsity of formula 5.1.

Let us now try to assign properties to nonexistent objects. A sentence such as (5.9) is vacuously true if there are no dragons (formula 5.3), but a sentence about a specific dragon (5.10) is false (formula 5.4).

(**5.9**) Dragons like children.

$$(\forall x)(dragon(x) \rightarrow likes\_children(x)) \tag{5.3}$$

(**5.10**) My dragon likes cake.

$$(\exists x)(my\_dragon(x) \wedge likes\_cake(x)) \tag{5.4}$$

Therefore, any statement that contains references to fictional characters is false. Even though our common sense assigns the value *true* to a sentence such as (5.11), according to the above formalism, the logical translation 5.5 is *false*.

(**5.11**) Sherlock Holmes is smart.

$$(\exists x)(is\_Holmes(x) \wedge smart(x)) \tag{5.5}$$

If we take now the negation of (5.10), i.e., (5.12), we can see that negation does not change the truth value of the corresponding translation. The appropriate logical form 5.6 remains *false* because we are still committed to the nonexistence of dragons.

(**5.12**) My dragon does not like cake.

$$(\exists x)(my\_dragon(x) \wedge \neg likes\_cake(x)) \tag{5.6}$$

To circumvent these problems, Russell [1905, p. 111] renounces taking presuppositions *for granted*. Instead, he explains the commitment to the *existence* of definite referents by the use of paraphrase. Thus (5.13) is paraphrased as (5.14).

(**5.13**) The author of Waverley was a man.

(**5.14**) One and only one entity is the author of Waverley and that one is a man.

$$(\exists x)(author\_of\_waverley(x) \wedge (\forall y)(author\_of\_waverley(y) \rightarrow y = x) \wedge man(x)) \tag{5.7}$$

If we paraphrase a sentence such as (1.28), *The King of Buganda is (not) bald*, we will have to choose between a representation that contains an internal negation and one that contains an external one according to whether we know or we do not know that there exists a king of Buganda. Unfortunately, there is no criterion for determining the right translation. Quine [1949, p. 27] shows how Russell's method can be extended to include proper names, but this is done in a framework where

> To say that *something* does not exist, or that is something which *is not* is clearly a contradiction in terms; hence $(x)(x\ .exists)$ must be true [Quine, 1947, p. 150].

Hence, Quine's solution cannot accommodate the cancellation of presuppositions.

Strawson's [1950] attack in the early fifties was only the beginning of the denial of the Russellian approach to presuppositions. Today, the paraphrase theory is no longer widely accepted.

5.2 Meinongian approaches

Meinong's [1904] philosophical enterprise aimed to be a theory of nonexistent objects based on mental acts. For Meinong, every mental act is directed towards an object, but it is not necessary that the objects of thought exist. The mental acts include knowing, believing, judging, presupposing, thinking, etc. In a Meinong universe, the existence of objects is no longer a tautology, as it was for Quine: one can affirm or can deny the existence of the objects, and more than that, one is free to assign properties to them whether or not the objects exist. A conversational perspective of the philosophical principles that underlie Meinong's work will be discussed in section 5.4.1. These principles constitute the foundation for several logical formalisms. We review Parsons's and Hirst's work and we analyze their suitability to reflect the existential commitment that characterizes the use of definite referents.

5.2.1 Parsons's Theory

Parsons's ontology consists of objects built on two different kinds of properties and one mode of predication. *Nuclear* properties include the ordinary ones: being blue, being a mountain, or being kicked by Socrates. *Extranuclear* properties include existence, being possible, being thought about by X. Informally speaking, the objects are constructed on infinite sets of properties, and they enjoy the following features:

1. No two objects (real or unreal) have exactly the same nuclear properties.
2. For any set of nuclear properties, some objects have all the properties in that set and no other nuclear properties.

Nonexistent objects are correlated with the set of properties which does not correspond to an existing object.

Parsons avoids Russell's paraphrase of the definite description by using the symbol ι. For Parsons, $(\iota x)\Phi$ refers to the unique object, that satisfies Φ if there is such an object. Otherwise, it does not refer to anything at all. For a sentence such as (5.15), Parsons argues [1980, p. 114] that translation 5.8 will not do it, because this translation is not committed to the existence of the man in the doorway. So he proposes that the translation should be 5.9, where $E!$ stands for the existential predicate.

(**5.15**) The man in the doorway is clever.

$$(\iota x)((man(x) \wedge in\_the\_doorway(x)) \wedge clever(x)) \tag{5.8}$$

$$(\iota x)((E!(x) \wedge man(x) \wedge in\_the\_doorway(x)) \wedge clever(x)) \tag{5.9}$$

A proposition $(\iota x)\Phi\alpha$ is true, if $(\iota x)\Phi$ refers to an object, and the object referred to has the property that α stands for. If $(\iota x)\Phi$ fails to refer, then $(\iota x)\Phi\alpha$ is automatically false. The problem with this is that it embeds the existential commitment in the logical translation not as something which is *implied* or presupposed, but as something *said* or specified. This is not the case with linguistic presuppositions. Therefore, the first translation is too weak, i.e., unable to capture our commitment, and the second one too strong, i.e., the commitment becomes part of the translation.

The direct embedding of the existential presupposition in the logical form reinforces the natural language negation ambiguity in sentences which deal with nonexistent entities. Hence, sentence (5.16), in the case there is no king of Buganda, can be both true and false depending on the translation we choose:

(**5.16**) The king of Buganda does not exist.

$$\neg(\iota x)(E!(x) \wedge king\_of\_buganda(x))[\lambda x_n E!(x_n)] \tag{5.10}$$

$$(\iota x)(E!(x) \wedge king\_of\_buganda(x))[\lambda x_n \neg E!(x_n)] \tag{5.11}$$

5.2.2 Hirst's Theory

Hirst [1991] presents an extensive study of the abilities of knowledge representation formalisms to capture nonexistent entities, as they appear in natural language. After a short review of the main philosophical trends, he shows that Russell-Quine's ontology, intensional approaches, free logics, and possible-world formalisms are not appropriate for the study of nonexistence. According to Hirst, a natural language ontology should reflect at least eight types of existence: physical existence in the present real world, physical existence in a past world, abstract necessary existence, existence outside a world but with causal interaction with that world, abstract contingent existence in the real world, existence as a concept, unactualized existence and existence in fiction. On the basis of this rich ontology he provides a generalization for Parsons's idea of nuclearity, and he formalizes it using Hobbs's transparent predicates [Hobbs, 1985]. Hirst is not interested in explaining the commitment to physical existence, so the ontology does not provide any preference for the inclusion of a given object in one category or another.

5.3 Other approaches

5.3.1 Lejewski's unrestricted interpretation

We were able to track attacks on Quine's slogan, *everything exists*, back to early works of Lejewski and Hintikka.

Lejewski [1954] builds his arguments starting from two puzzling examples which seem to disprove the validity of the rules of universal instantiation and existential generalization applied to reasoning with empty noun expressions. In a universe à la Quine, using the universal instantiation rule and the existential predicate $E!$, from $(\forall x)(E!(x))$ we infer *Pegasus exists*. From the sentence *Pegasus does not exist*, we infer by existential generalization that $(\exists x)(\neg E!(x))$. However, these inferences are objectionable to our intuitions. In other words the logical laws:

$$(\forall x)(F(x)) \rightarrow F(y) \qquad \text{Universal Instantiation — UI} \qquad (5.12)$$

and

$$F(y) \rightarrow (\exists x)F(x) \qquad \text{Existential Generalization — EG} \qquad (5.13)$$

do not hold for every interpretation of F and every substitution for the free variable.

Seeking solutions to this problem, Lejewski turns to the interpretations of the quantifiers. Let us consider first a universe with two objects, a and b. In this universe, following Quine, we can express the existential quantification $(\exists x)(F(x))$ as logical disjunction $F(a) \vee F(b)$, and the universal quantification as logical conjunction $F(a) \wedge F(b)$. In this fixed universe, introducing noun-expressions can be regarded as a linguistic matter in the sense that it does not affect our assumed universe, which continues to consist only of a and b. However, we can use these objects. For example we can say that d satisfies the predicate D, or that the object c does not exist. In this universe, $(\forall x)(E!(x))$ no longer implies that c *exists*. The universal instantiation and existential generalization are valid rules but are restricted to reasoning only with a and b. But this is not the only possible interpretation of a quantifier. We can read them differently, such as:

$F(a) \vee F(b) \vee F(c) \vee F(d)$ for $(\exists x)(F(x))$ and

$F(a) \wedge F(b) \wedge F(c) \wedge F(d)$ for $(\forall x)(F(x))$.

Thus $(\exists x)(\neg E!(x))$ is true because c does not exist and $(\forall x)(E!(x))$ is false. Under such an interpretation the rules of Universal Instantiation and Existential Generalization are valid without restrictions. Actually, Lejewski emphasizes a dichotomy between two possible ways of interpreting the quantifiers: a *restricted interpretation* under which every component of an expression contains a noun-expression which has a *nominatum*, and an *unrestricted interpretation* under which we can only say that the objects are meaningful noun-expressions. This gives two different theories of quantification.

One way to understand expressions is in light of the restricted interpretation:

$$(\forall x)(F(x)) \rightarrow F(y) \tag{5.14}$$

$$F(y) \rightarrow (\exists x)(F(x)) \tag{5.15}$$

$$(\forall x)(F(x)) \leftrightarrow \neg(\exists x)(\neg F(x)) \tag{5.16}$$

In light of the unrestricted interpretation, we can express the same thing as:

$$(\forall x)(E!(x) \rightarrow F(x)) \rightarrow F(y) \tag{5.17}$$

$$F(y) \rightarrow (\exists x)(E!(x) \wedge F(x)) \tag{5.18}$$

$$(\forall x)(E!(x) \rightarrow F(x)) \leftrightarrow \neg(\exists x)(E!(x) \wedge \neg F(x)) \tag{5.19}$$

It is obvious now that under the *unrestricted interpretation*, the two inferences about Pegasus cannot be used as counterexamples to disprove the validity of the rules of universal instantiation and existential generalization.

This theory has a wonderful advantage. It no longer merges the idea of quantification with the notion of existence, but from a presuppositional point of view, it does not provide an explanation for the existential commitment. A translation of a sentence such as (1.28), *The king of Buganda is bald*, under the unrestricted interpretation is neutral with respect to the king's existence 5.20.

$$(\exists x)(King\_of\_Buganda(x) \wedge bald(x)) \tag{5.20}$$

Lejewski's universe is flat, it does not reflect our commitment to existence, and this is not what we would like to have.

5.3.2 Hintikka's existential presuppositions

Another detractor of Russell-Quine's theory is Hintikka [1970, 1959]:

> Of course, we are often likely to drop a name altogether if it turns out to be empty. But this is because there is usually very little to be said about what there is not, and not because an empty name is a logical misnomer [Hintikka, 1959, p. 127].

Hintikka's belief is that the meaningfulness and the logical status of names and of other singular terms is not affected by the failure to refer. Hintikka states clearly that there are existential presuppositions embodied in the usual systems of quantification theory [Hintikka, 1959, p. 130]. To avoid this embedding, he explicitly introduces Quine's slogan, *to exist is to be a value of a variable*, in the Existential Generalization rule. Thus, he replaces

$$f(a/x) \rightarrow (\exists x)f \tag{5.21}$$

with

$$(\exists x)(x = a) \wedge f(a/x) \rightarrow (\exists x)f \tag{5.22}$$

All the other axioms are kept unmodified. The premise $(\exists x)(x = a)$ is all that is needed to make a behave like a name with a reference. The solution is somehow similar to the one offered by Lejewski, if one reads the condition $(\exists x)(x = a)$ as a predicate and interprets quantifiers under their unrestricted interpretation. But unfortunately, as we have already mentioned, this universe is flat also. We are free to talk about Pegasus and dragons, but we still cannot explain our commitment to the existence of definite referents.

5.3.3 Hobbs's ontological promiscuity

Quine [1949, p. 47] specifies that

> We seem to have a continuum of possible ontologies here, ranging from a radical realism at the one extreme, where even a left-hand parenthesis or the dot of an "i" has some weird abstract entity as designatum, to a complete nihilism at the other extreme.

On the nihilistic extreme we find Hobbs's approach, presented in [Hobbs, 1985] and later developed and implemented in [Hobbs *et al.*, 1993]. Hobbs is mainly interested in solving three classical problems: opaque adverbials, the distinction between *de re* and *de dicto* belief reports, and the problem of identity in intensional contexts. Hobbs's approach is constructed on Lejewski and Hintikka's idea that the set of things we can talk about, therefore including nonexistent things also, and the set of things we quantify over, are equal. His language does not contain modalities, intensions, nor even negation. His approach does not assume the existence of any object unless this is explicitly specified or logically derivable. For example, sentence (5.17) is represented in formula 5.23.

(**5.17**) Ross worships Zeus.

$$worship'(E, Ross, Zeus) \wedge Exist(E) \tag{5.23}$$

The first conjunct says that E is the event of worshiping Zeus by Ross, and the second says that E exists in the real world. Hobbs assigns a *transparency* property to the predicates. For $worship'$, this property entails the existence of its second argument in the physical world:

$$\forall E \forall x \forall y((worship'(E, x, y) \wedge Exist(E)) \rightarrow Exist(x)) \tag{5.24}$$

Apparently, the commitment to Ross's existence is solved. *Worship* is a transparent verb regarding its second argument, so in accordance to this, Ross is existent. Hirst [1991] studies the implications induced by the introduction of the concept of *anti-transparent* predicates into the system. Nevertheless, this allows one to derive nonexistence but it does not give any insights into the presuppositional phenomenon. As one can utter (5.17), she can also utter (5.18).

(**5.18**) The king of Buganda worships Zeus.

For the latter sentence two things can happen assuming we know nothing about Buganda:

- If we translate it as $worship'(E, king\_of\_Buganda, Zeus)$, we cannot fire the transparency axiom related to predicate $worship'$, so there is no way to explain our commitment to the existence of Buganda's king.

- If we translate it as $worship'(E, king\_of\_Buganda, Zeus) \wedge Exist(E)$ we can infer via the transparency axiom that the king of Buganda exists. Once we do that, we are no longer able to cancel this inference, although we may find later that Buganda has no king.

The choice of the appropriate translation is not a simple one, so from a presuppositional perspective, Hobbs's theory will not do it either.

5.3.4 Atlas's aboutness and topicality

Atlas argues that the existential puzzles arise neither from the mistake of taking existence as a first-order predicate, nor from equating naming with meaning but due to a conceptual mistake, namely, a mistake as to what existential statements are *about*. According to Atlas, *aboutness* is a fundamental topic in linguistics and philosophical logic, one that if it is overlooked creates important problems. For example, even if the singular term *the queen of England* is used both in (5.19) and (5.20), only the former presupposes that the singular term is used to refer to someone, or in other words only the first statement is *about* the queen of England. "If the queen of England were not, one should not be talking $\underline{a}$bout anyone when he uses the term $\underline{t}$he queen of England" [Atlas, 1988, p. 377]. The latter sentence (5.20) does not presuppose that there is a queen of England because one would not fail to talking *about* anything.

(**5.19**) The queen of England raises race horses.

(**5.20**) It is the queen of England who raises race horses.

According to Atlas's theory, a statement $A(t)$ presupposes that t *exists* only if t is a topic noun phrase. Atlas establishes a set of criteria for noun phrase topicality [1988, p. 385], but he gives no hint of how this theory could be extended to deal with factives or verbs of judging, and defining the notions of aboutness and topicality for them is not trivial. Even if we did manage to do this, such presuppositions can never be cancelled. Either they are generated or they are not. This leads us to believe that sentences such as (1.45), *John does not regret that Mary came to the party because she didn't come*, cannot be appropriately interpreted in this manner. A good reason for circumventing the notion of topicality is given by Lasersohn [1993].

5.3.5 Lasersohn's context dependency

We interpret Lasersohn's theory [1993] as a natural extension of Labov's work on the boundaries of words and their meaning [Labov, 1973]. William Labov noticed that the distinction between *cup* and *bowl* is affected more by whether the interpreter is situated in a *coffee* or in a *mashed potatoes* context than by factors such as the ratio of height to diameter of the artifact. The same thing happens with sentences containing non-referring definite descriptions. Some of them are clearly false (5.21), while others are ambiguous (5.22).

(**5.21**) The King of France is sitting in that chair.

(**5.22**) The King of France is bald.

To explain this dependency on context, Lasersohn uses a semantics constructed in terms of Data Sets [Velman, 1981]. An ordering relation, $\leq$, is defined on the data sets: $D \leq D'$ if D' is a consistent superset of D. Sentence (5.21) seems to be false because *even if we suspend our knowledge that there is no king of France, there is no way of consistently extending our information to include the proposition that the King of France is sitting in the chair* [Lasersohn, 1993, p. 116]. This is a subtle observation and our contribution will subsume it. The only inconveniences we notice are the ones related to the use of the Russellian paraphrase in capturing the existential commitment, and the fact that Data Sets are not computationally attractive.

5.4 Nonexistence and existential presuppositions — a solution

5.4.1 Methodological principles

The approach to nonexistent objects and presuppositions that we are going to present is constructed on the basis of a modified set of Meinongian principles about nonexistence. They are embedded in a stratified logic framework in which quantifiers are taken under

Lejewski's unrestricted interpretation. The ontology is enhanced with the eight types of existence listed by Hirst [1991], though in this thesis, we will deal only with physical existence, represented as $E!$, unactualized existence, represented as $UE!$, existence outside the world but with causal interaction with that world, $EOW!$, and existence in fiction, $F!$.

Following Rapaport's style [1985], we propose a set of methodological principles based on Meinong [1904] that are meant to capture the ability of an intelligent agent to deal with existence and nonexistence rather from a conversational perspective than from a rational one.

MC1. Every uttered sentence is *directed* towards an *object*, because every uttered sentence can be seen as a materialization of a mental act.

MC2. All uttered sentences exist (technically, *have being*). However, this does not imply the existence of their referents, which are *ausserseiend* (beyond being and non-being).

MC3. It is not self-contradictory to deny, nor tautologous to affirm, the existence of a referent.

MC4. Every referent and every uttered sentence has properties.

MC5. The principles MC2 and MC4 are not inconsistent.

Corollary: Even referents of an uttered sentence that do not exist have properties.

MC6. (a) Every set of properties (Sosein) corresponds to the utterance of a sentence.
(b) Every object of thought can be uttered.

MC7. Some referents of an utterance are incomplete (undetermined with respect to some properties).

In accordance with Grice [1975], we need two additional principles:

GC1. The speaker is committed to the truth of the sentences he utters.

GC2. Using and deriving presuppositions requires, from both speaker and listener, a sort of "optimism".

Principle GC1 is formalized by the translation of the uttered sentences into classical logic formulas in which quantifiers are read under their unrestricted interpretation. Principle GC2 is formalized by the rules containing defeasible information that exist in the knowledge base of the speaker and the hearer, and the notion of optimism in the model-ordering relation. Note that a non-optimistic interpretation of utterances will never be able to account for any of the pragmatic inferences, because they are not explicitly uttered.

5.4.2 Formalizing existential presuppositions

We have already seen how in stratified first-order logic one can express the pragmatic information associated with the word *bachelor* (see formulas 3.14).

That uttering definite references imply the existence of their referents constitutes another example of pragmatic inference. We can capture this either by adding a new formula 5.25 to our knowledge base, and by embedding syntactic terms into the logical form, as Hobbs did [1985], or by representing this defeasible commitment explicitly in the translation of each utterance containing a definite reference or proper noun.

$$(\forall^{Ut}x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \tag{5.25}$$

Both approaches exhibit the same semantic behavior, and due to the model-ordering relation they explain our commitment to a referent's existence (in the case that we do not know otherwise). Because $\mathsf{definite\_reference(x)}$ is syntactic information, we depict it using a different font, but the reader should understand that x is bound by the same quantifier as x is.

As a last step, we abandon the Fregean reading of the quantifiers and we adopt Lejewski's unrestricted interpretation [1954]. This means that $\exists$ and $\forall$ do not mix quantification with ontological commitment: $(\exists x)object(x)$ does not entail the existence of x, so the set of things we can talk about equals the set of things we can quantify over.

5.4.3 What the approach can do with existent and nonexistent objects

Assume that someone utters sentence (1.28) *The king of Buganda is (not) bald.* If we know nothing about Buganda and its king, the complete theory of this utterance and the available knowledge in stratified logic is this:

$$\left\{ \begin{array}{l} uttered((\exists x)(king\_of\_buganda(x) \wedge \mathsf{definite\_reference(x)} \wedge (\neg)bald(x))) \\ (\forall^{Ut}x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \end{array} \right. \tag{5.26}$$

This theory has one optimistic model 5.27 that reflects one's commitment to the king's existence. The king's existence has the status of felicitously defeasible information; it is derived using knowledge of language use and, according to definition 3.2.3, is a presupposition of the utterance.

$$m = \{king\_of\_buganda(\xi_0), (\neg)bald(\xi_0)\} \cup \emptyset^i \cup \{E!^d(\xi_0)\} \tag{5.27}$$

Knowledge about the political system of France can inhibit the inference regarding the existence of its king in a sentence such as (5.23).

(**5.23**) The king of France is (not) bald.

Assume that we know there is no king of France ($\neg E!$). A complete formalization follows:

$$\left\{\begin{array}{l} uttered((\exists x)(king\_of\_france(x) \wedge \mathsf{definite\_reference(x)} \wedge (\neg)bald(x))) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \\ (\forall x)(king\_of\_france(x) \rightarrow \neg E!(x)) \end{array}\right. \tag{5.28}$$

For this theory, we obtain only one model schemata:

Schema #	*Indefeasible*	*Infelicitously defeasible*	*Felicitously defeasible*
m_0	$king\_of\_france(\xi_o)$ $(\neg)bald(\xi_0)$ $\neg E!(\xi_0)$		$E!^d(\xi_0)$

One can notice that the existential presupposition is now cancelled by some background knowledge. The only way one can satisfy the initial theory is if she has a stratified structure where $\neg E!(\xi_0)$. Thus, the theory yields one model:

$$m = \{king\_of\_france(\xi_0), (\neg)bald(\xi_0), \neg E!(\xi_0)\} \cup \emptyset^i \cup \emptyset^d \tag{5.29}$$

Asserting existence or nonexistence affects *defeasible inferences* due to knowledge of language use and restricts some of the models. If someone utters (5.24) and we know nothing about Buganda, the translation 5.30 gives one model 5.31.

(**5.24**) The king of Buganda exists.

$$\left\{\begin{array}{l} uttered((\exists x)(king\_of\_buganda(x) \wedge \mathsf{definite\_reference(x)} \wedge E!(x))) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \end{array}\right. \tag{5.30}$$

$$m = \{king\_of\_buganda(\xi_0), E!(\xi_0)\} \cup \emptyset^i \cup \emptyset^d \tag{5.31}$$

If we *know* that the king of Buganda does not exist, or in other words we evaluate the above sentence against a knowledge base that contains

$$(\forall x)(king\_of\_buganda(x) \rightarrow \neg E!(x)), \tag{5.32}$$

there is no model for this theory, so the utterance is interpreted as false. It is noteworthy that the inconsistency appears due to specific knowledge about the king's physical existence and not because of a quantification convention as in classical first-order logic. On the other hand, the negation (5.16), *The king of Buganda does not exist*, is consistent with the knowledge base and provides model 5.33.

$$m = \{king\_of\_buganda(\xi_0), \neg E!(\xi_0)\} \cup \emptyset^i \cup \emptyset^d \tag{5.33}$$

Our approach subsumes the work done by Lasersohn [1993] because sentence (5.21), *The King of France is sitting on that chair*, is false regardless of whether the king of France exists or not, if we know there is no one on the chair. The corresponding theory is *u-inconsistent*, so it has no stratified models.

$$\left\{\begin{array}{l} uttered(sitting(king\_of\_france, that\_chair) \wedge \\ \quad \mathsf{definite\_reference(king\_of\_france)} \wedge \mathsf{definite\_reference(that\_chair)}) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^{d}(x)) \\ (\forall x)(\neg sitting(x, that\_chair)) \end{array}\right. \tag{5.34}$$

One can see now that the proposed method is general in the sense that it captures all presuppositional environments. Reconsider for example utterance (1.42), *John does not regret that Mary came to the party.* A complete formalization in stratified logic follows:

$$\left\{\begin{array}{l} uttered(\neg regret(john, \$come(mary, party)) \wedge \mathsf{definite\_reference(john)} \wedge \\ \quad \mathsf{definite\_reference(mary)} \wedge \mathsf{definite\_reference(party)}) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^{d}(x)) \\ (\forall^{Ut} x, y, z)(\neg regret(x, \$come(y, z)) \rightarrow come^{d}(y, z)) \\ (\forall^{Ut} x, y, z)(regret(x, \$come(y, z)) \rightarrow come^{i}(y, z)) \end{array}\right. \tag{5.35}$$

The optimistic model computed by our program is this:

$$\begin{array}{l} m = \{\neg regret(john, \$come(mary, party))\} \cup \emptyset^{i} \cup \\ \qquad \{E!^{d}(john), E!^{d}(mary), E!^{d}(party), come^{d}(mary, party)\} \end{array} \tag{5.36}$$

This model reflects our intuitions that Mary came to the party and that all the definite references exist.

If one utters now sentence (5.25), the new model computed by our program will reflect the fact that a presupposition has been cancelled, even though this cancellation occurred later in the discourse. Thus, the new optimistic model will be 5.37.

(**5.25**) Of course he doesn't. Mary did not come to the party.

$$\begin{array}{l} m = \{\neg regret(john, \$come(mary, party)), \neg come(mary, party)\} \cup \emptyset^{i} \cup \\ \qquad \{E!^{d}(john), E!^{d}(mary), E!^{d}(party)\} \end{array} \tag{5.37}$$

Our approach correctly handles references to unactualized objects such as averted strikes. An appropriate formalization for sentence (5.26) is theory 5.38.

(**5.26**) The strike was averted.

$$\left\{\begin{array}{l} uttered((\exists x)(strike(x) \wedge \mathsf{definite\_reference(x)} \wedge averted(x))) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \\ (\forall x)(averted(x) \rightarrow UE!(x)) \\ (\forall x)(UE!(x) \rightarrow \neg E!(x)) \end{array}\right. \tag{5.38}$$

This gives one optimistic model:

$$m = \{strike(\xi_0), averted(\xi_0), \neg E!(\xi_0), UE!(\xi_0)\} \cup \emptyset^i \cup \{E!^d(\xi_0)\} \tag{5.39}$$

5.5 A comparison with Parsons's and Hobbs's work

5.5.1 On Parsons's evidence for his theory of nonexistence

Parsons argues that is impossible to distinguish between the shape of the logical form of two sentences like these, in which one subject is fictional and the other is real:

(**5.27**) Sherlock Holmes is more famous than any other detective.

(**5.28**) Pelé is more famous than any other soccer player.

In our approach, similar syntactic translations give different semantic models when interpreted against different knowledge bases. A complete theory for sentence (5.27) is this:

$$\left\{\begin{array}{l} uttered((\exists x)(sherlock\_holmes(x) \wedge \mathsf{definite\_reference(x)} \\ \quad \wedge(\forall y)((detective(y) \wedge (x \neq y)) \rightarrow \\ \quad\quad more\_famous(x, y)))) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \\ (\forall x)(sherlock\_holmes(x) \rightarrow F!(x)) \\ (\forall x)(F!(x) \rightarrow \neg E!(x)) \end{array}\right. \tag{5.40}$$

This theory gives only one model:

$$\begin{aligned} m = \{&sherlock\_holmes(\xi_0), \neg E!(\xi_0), F!(\xi_0), \\ &detective(y), more\_famous(\xi_0, y)\} \cup \emptyset^i \cup \emptyset^d \end{aligned} \tag{5.41}$$

This corresponds to an object ξ_0 that does not exist in the real world but exists as a fiction, has the property of being Sherlock Holmes, and for any other object y, real or fictional that has the property of being a detective, the object ξ_0 is more famous than object y. Of course, in this model, it is impossible to commit ourselves to Holmes's physical existence, but is possible to talk about him.

The theory for the second sentence (5.28) is this:

$$\left\{\begin{array}{l} uttered((\exists x)(pele(x) \wedge \mathsf{definite\_reference(x)} \wedge \\ \quad (\forall y)((soccer\_player(y) \wedge (x \neq y)) \rightarrow \\ \quad more\_famous(x,y)))) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \end{array}\right. \tag{5.42}$$

This theory exhibits one optimistic model:

$$\begin{array}{l} m = \{pele(\xi_0), soccer\_player(y), \\ \qquad more\_famous(\xi_0, y)\} \cup \emptyset^i \cup \{E^d!(\xi_0)\} \end{array} \tag{5.43}$$

Model m states that the object ξ_0, being Pelé, exists in a defeasible sense and this is the existential presupposition of the initial utterance.

As seen, it is needless to mention the existence of specific objects in the knowledge base. The model-ordering relation rejects anyhow models that are not optimistic. In this way, the commitment to Pelé's existence is preserved, and appears as a presupposition of the utterance. Parsons's theory provides different logical forms for the above sentences, but fails to avoid the commitment to nonexistent objects.

5.5.2 A comparison with Hobbs's work

We have mentioned that Hobbs's transparency pertains to relations and not to objects. In our approach, a sentence such as (5.17), *Ross worships Zeus*, can be satisfied by a set of semantic models that correspond to each possible combination of the existence and nonexistence of Ross and Zeus.

$$\left\{\begin{array}{l} uttered((\exists x)(\exists y)(ross(x) \wedge zeus(y) \wedge worship(x,y) \\ \quad \wedge \mathsf{definite\_reference(x)} \wedge \mathsf{definite\_reference(y)})) \\ (\forall^{Ut} x)(\mathsf{definite\_reference(x)} \rightarrow E!^d(x)) \end{array}\right. \tag{5.44}$$

Among them, only one is optimistic: the one that explains the commitment to both Ross's and Zeus's existence.

$$m = \{ross(\xi_0), zeus(\xi_1), worship(\xi_0, \xi_1)\} \cup \emptyset^i \cup \{E!^d(\xi_0), E!^d(\xi_1)\} \tag{5.45}$$

But let us assume we know that there is no entity in the real world that enjoys the property of being Zeus, but rather one who exists outside the real world as a god ($EOW!$).

$$\left\{\begin{array}{l} uttered((\exists x)(\exists y)(ross(x) \wedge zeus(y) \wedge worship(x,y) \\ \quad \wedge \textsf{definite\_reference(x)} \wedge \textsf{definite\_reference(y)})) \\ (\forall^{Ut} x)(\textsf{definite\_reference(x)} \rightarrow E!^{d}(x)) \\ (\forall x)(zeus(x) \rightarrow EOW!(x)) \\ (\forall x)(EOW!(x) \rightarrow \neg E!(x)) \end{array}\right. \tag{5.46}$$

This theory is no longer satisfiable by a model in which Zeus exists as a physical entity. However, the optimistic model explains our commitment to Ross's existence.

$$\begin{aligned} m = \{&ross(\xi_0), zeus(\xi_1), \neg E!(\xi_1), EOW!(\xi_1), worship(\xi_0,\xi_1)\} \\ &\cup \emptyset^{i} \; \cup \; \{E!^{d}(\xi_0)\} \end{aligned} \tag{5.47}$$

Chapter 6

Implementation issues

Stratified logic has been implemented in Common Lisp and it makes extensive use of two packages: *Screamer* [Siskind, 1991], a set of nondeterministic macros, and *Iterate* [Amsterdam, 1990], an iteration macro similar to *loop*. The interested reader can find our program in annex A. Details on nondeterministic programming and Screamer are given by [Siskind and McAllester, 1993a, Siskind and McAllester, 1993b].

6.1 Representation

Constants, variables, functions, and predicates are represented as Lisp symbols. Stratified predicates are prefixed by *U-*, *I-*, or *D-*: for example, $adult^d(x)$ is represented as *(D-adult x)*. Terms and atomic formulas such as $come(Mary, party)$ are represented as Lisp s-expressions: *(come Mary party)*. Utterances are represented as Lisp s-expressions as well, as in *(uttered (bachelor cousin))*. Compound formulas are represented as follows:

Formula	Lisp representation
$\Phi \wedge \Psi$	(and Φ Ψ)
$\Phi \vee \Psi$	(or Φ Ψ)
$\neg\Phi$	(not Φ)
$\Phi \rightarrow \Psi$	(implies Φ Ψ)
$\Phi \leftrightarrow \Psi$	(iff Φ Ψ)
$\exists x \Phi$	(exists x Φ)
$\forall x \Phi$	(forall x Φ)
$uttered\,\Phi$	(uttered Φ)
$\forall^{Ut}\vec{x}(\Phi \rightarrow \Psi)$	(forallutt ($\vec{x}$) (implies Φ Ψ))

Any symbol that appears as the first element in a list except *and, or, not, implies, iff, exists, forall, uttered*, and *forallutt* is interpreted as a function symbol or predicate. The predicates are the symbols that occur as first elements of top level list, or as first elements of a list embedded in a logical connective or utterance. All the other first symbols are function symbols.

The function symbols that are prefixed by the \$ sign are used to restrict the universal quantification only to the objects that constitute the arguments of that function. Consider the following two formulas:

$$\begin{cases} Iregret(was\_bad(movie)) \\ (\forall x)(Iregret(was\_bad(x)) \rightarrow was\_bad^{i}(x)) \end{cases} \tag{6.1}$$

The first one is the logical translation of utterance (6.1).

(**6.1**) I regret that the movie was bad.

The second one captures the pragmatic inference that a factive, *regret*, implies infelicitously defeasibly its complement. If we are to apply exhaustively the universal instantiation rule, we will have to do it for two logical objects: *movie* and *was_bad(movie)*. This will yield meaningless formulas such as

$$Iregret(was\_bad(was\_bad(movie))) \rightarrow was\_bad^{i}(was\_bad(movie)) \tag{6.2}$$

To avoid meaningless formulas, we prefix the function symbols that formalizes linguistic sentences with the \$ sign. The convention we use removes \$ sign functions from the set of objects over which an universal quantifier instantiates. Therefore, a correct formalization for utterance (6.1) and the adequate pragmatic knowledge should be as follows:

$$\begin{cases} Iregret(\$was\_bad(movie)) \\ (\forall x)(Iregret(\$was\_bad(x)) \rightarrow was\_bad^{i}(x)) \end{cases} \tag{6.3}$$

Under the convention given above, the universal instantiation rule is applied for only one object: *movie*.

6.2 The algorithms

Our algorithm (see figure 6.1) takes as input a set of first-order stratified formulas Φ that represents an adequate knowledge-base and the translation of an utterance or set of utterances $uttered(U)$. The output of the program is a set of felicitous optimistic model schemata that correspond to an optimistic interpretation of the given utterances. The program also outputs the set of presuppositions that are carried by the uttered sentences; if the utterances are infelicitous it reports that.

```
Input: $\Phi \cup uttered(U)$

Find the set $\mathcal{MS}$ of all model schemata for the theory $\Phi \cup uttered(U)$.
if $\mathcal{MS} = \emptyset$ then return $U$ is false.
    else Find the set $\mathcal{FMS}$ of felicitous model schemata.
        if $\mathcal{FMS} = \emptyset$ then return $U$ is infelicitous.
            else Find the set $\mathcal{FOMS}$ of felicitous
                     optimistic schemata.
                 Collect the presuppositions.
                 Report the results.
        endif
endif
```

Figure 6.1: The Main Algorithm

6.2.1 Finding the model schemata

Finding the model schemata is the most difficult task. The solution generalizes Siskind's nondeterministic approach for constructing first-order tableaux [1991]. The implementation makes use of the following data structures:

Γ is the queue of formulas that remain to be processed.

M is a list of true and negated stratified formulas that constitute the model schema constructed for the initial set Γ.

Δ is a list of pairs $(formula, const)$ that monitors the constants on which a universal formula has been already instantiated.

I is a list of pairs $(utterance, (\vec{cont}))$ that monitors the constants on which a pragmatic rule should be applied.

We also use the following functions:

$push\_formula(\alpha, \Gamma)$ pushes formula α on the queue Γ;

$pop\_formula(\Gamma)$ pops the first formula from the queue Γ;

$ground\_terms()$ returns all the ground terms that occur at a specific moment on a branch;

$uttered\_terms(\alpha)$ returns all the ground terms that occur at a specific moment on an utterance about α.

To find all the model schemata we construct an exhausted tableau for the initial theory. Each exhausted open branch yields a model schema by collecting the positive and negative atomic formulas on it.

```
$\Gamma := \Phi \cup uttered(U)$;
$M := \emptyset$;
while $\neg empty(\Gamma)$do
        $\alpha := pop\_formula(\Gamma)$;
        case $\alpha$
                $\alpha_1 \wedge \alpha_2$: $push\_formula(\alpha_1, \Gamma)$ and $push\_formula(\alpha_2, \Gamma)$;
                $\alpha_1 \vee \alpha_2$: $push\_formula(\alpha_1, \Gamma)$ or $push\_formula(\alpha_2, \Gamma)$;
                $\alpha_1 \rightarrow \alpha_2$: $push\_formula(\neg\alpha_1 \vee \alpha_2, \Gamma)$;
                $\vdots$
                $\alpha_1^i, \neg\alpha_1^i, \alpha_1^d, or \neg\alpha_1^d$, and $\alpha_1$ is atomic: $M := M \cup \alpha$;
                $\alpha^u$ is atomic: if $\neg\alpha^u \in M$ then Fail;
                                                    else $M := M \cup \alpha^u$;
                $\neg\alpha^u$ is atomic: if $\alpha^u \in M$ then Fail;
                                                    else $M := M \cup \{\neg\alpha^u\}$;
                $(\exists x)\alpha_1$: $push\_formula(\alpha_1(x/gensym()), \Gamma)$;
                $(\forall x)\alpha_1$: $T := ground\_terms()$ ;
                          for $t \in T$ do $push\_formula(\alpha_1(x/t), \Gamma)$;
                          $push\_formula((\forall x)\alpha_1, \Gamma)$;
                          endfor
                $(\forall^{Ut} x)\alpha_1$: $T := uttered\_terms()$ ;
                            for $t \in T$ do $push\_formula(\alpha_1(x/t), \Gamma)$;
                            $push\_formula((\forall^{Ut} x)\alpha_1, \Gamma)$;
                            endfor
        endcase
endwhile
```

Figure 6.2: The Algorithm for Building a Model Schema

Algorithm 6.2 terminates when there are no formulas left on Γ. Infinite loops due to universal instantiation are avoided by keeping track in Δ of the constants each universal formula has been instantiated. When an attempt is made to instantiate a universally quantified formula on the same constants, that formula is removed from Γ. Alternatives in constructing the tableau are explored using *Screamer*'s nondeterministic construct *either* that allows one to keep only one branch in memory at a time. To collect all the model schemata, we have used the *all-values* macro.

6.2.2 Finding the felicitous model schemata

If all models for an utterance are infelicitous, i.e., they are not *i-satisfiable* (see definition 3.2.2), the utterance is infelicitous. The felicitous model schemata are obtained by preserving from the set $\mathcal{MS}$ only the model schemata that are felicitous, i.e., the ones in which no infelicitously defeasible information is cancelled. The algorithm is $O(n)$, where n is the number of model schemata.

6.2.3 Finding the optimistic model schemata

The algorithm is $O(n^2)$ and it compares the list of models $\mathcal{FMS}$ two by two and eliminates the ones that are not optimistic with respect to definition 2.6.8.

6.2.4 Collecting the presuppositions

We have seen that presuppositions are associated with the defeasible inferences that are triggered by pragmatic maxims. Therefore, in order to collect the presuppositions for a given set of utterances, all defeasible inferences are considered as presupposition candidates. The ones that are not cancelled in any optimistic model are projected as presuppositions for the initial set of utterances.

Chapter 7

Conclusions

7.1 Contributions

This thesis has provided a principled approach to the notion of *infelicity* and shown how pragmatic infelicities, i.e., those that appear when *infelicitously defeasible pragmatic inferences* are cancelled, can be signalled. We exploited a fundamental difference between the semantic content of an utterance and the inferences that can be drawn from this content, and the pragmatic content and its afferent inferences. First-order logic and the notion of entailment seem appropriate for characterizing the first class because semantic information is undefeasible. Since implicatures and presuppositions are not explicitly uttered, one can argue that *all* pragmatic inferences are defeasible. First-order logic cannot accommodate the notion of nonmonotonic reasoning; hence, in order to account for the pragmatic phenomena a formalism able to handle defeasible information is needed.

The crucial observation we made is that a taxonomy that has a finer granularity is needed if one wants to consider more subtle aspects of natural language interaction. Hence, we noted that pragmatic information may be inferred with different degrees of strength or commitment: some pragmatic inferences are felicitous to defeat, while others, are infelicitously defeasible. Figure 7.1 shows the properties that pragmatic inferences have with respect to their cancelability. The reader should interpret the taxonomy we propose only as a proof that there are differences between the degree of commitment that characterizes different types of pragmatic inferences. No claims are made that the classification we give is definitive or exhaustive. Future research could reveal that a finer granularity is needed. However, figure 7.1 constitutes a strong argument for considering a taxonomy of natural language inferences with at least three layers:

- The *undefeasible layer* characterizes the semantic information and all the inferences that derive from it.
- The *infelicitously defeasible layer* characterizes the pragmatic inferences that are

infelicitous to defeat.

- The *felicitously defeasible layer* characterizes the pragmatic inferences that are felicitous to defeat.

Pragmatic inferences	*The strength of pragmatic inferences with respect to their cancelability*	*Example*
Implicatures derived using the maxim of quality	Infelicitously defeasible	Sentence (1.3)
Implicatures derived using the maxim of quantity	Felicitously defeasible	Sentence (1.6)
Implicatures derived using the maxims of relevance	Felicitously defeasible	Sentence (1.9)
Implicatures derived using the maxims of manner (order)	Infelicitously defeasible	Sentence (1.12)
Particularized implicatures	Felicitously defeasible	Sentence (1.16)
Floating implicatures	Felicitously defeasible	Sentence (1.19)
Scalar implicatures	Felicitously defeasible	Sentence (1.22)
Clausal implicatures	Felicitously defeasible	Sentence (1.26)
Presuppositions in affirmative sentences	Infelicitously defeasible	Sentence (1.44)
Presuppositions in negative sentences	Felicitously defeasible	Sentence (1.45)

Figure 7.1: The strength of pragmatic inferences with respect to their cancelability

In order to deal with these three layers, we have devised a new logical framework, called *stratified logic*. Stratified logic allows one to account not only for the nonmonotonic aspects that are specific to pragmatic reasoning, but also for the determination of infelicitous utterances. The framework yields a formal definition for the notion of *infelicitous utterance* and *presupposition*, and a tractable algorithm for computing interpretations for utterances, for determining their associated presuppositions, and for signalling infelicities. The proposed method is general, i.e., it handles simple and complex utterances, sequences of utterances, and it gives a uniform explanation for the presuppositions carried by definite references and the ones determined by other presupposition environments. The solution for the existential presuppositions is not subject to the classical logic puzzles.

We have contrived our framework having in mind that the result ought to be computationally tractable. A generalization of the Beth tableau technique provides the theoretical foundation for our implementation. Stratified logic is implemented in Common Lisp and it makes extensive use of the nondeterministic facilities of the Screamer system [Siskind, 1991]. The result is a program that takes as input a set of stratified formulas that constitute the necessary semantic and pragmatic knowledge and the logical translation of an utterance or set of utterances and that computes a set of *optimistic* interpretations for the given utterances. The program computes for each set of utterances the associated presuppositions, and signals when an infelicitous sentence has been uttered.

7.2 Future work

There are a number of important directions in which one can expand this research:

Consider agents with different knowledge bases: There are no restrictions that prevent our system from considering that the participants in a conversation do not share the same knowledge. If one does so, we expect that she will be able to distinguish between the pragmatic inferences that pertain to different participants in a conversation; to understand better the way they interact; to see how semantic and pragmatic information is transferred between them; and to determine where semantic or pragmatic misunderstandings occur.

Extend the notion of *infelicity* to syntax, semantics, and speech acts: We believe that a principled account can be given for the other types of infelicities using the same ideas we have presented in this thesis. For example, in order to signal a syntactic infelicity, such as the one that occurs when the noun in a sentence does not have the same number as the corresponding verb, one will have to loosen the syntactic rules. The same methodology that we used when we split the classical notion of truth into undefeasible, infelicitously defeasible, and felicitously defeasible truth should be applicable at this level as well. The number agreement rule will be assigned an *infelicitously defeasible* strength, so that the syntactic analysis will proceed, but the infelicity is signalled.

Formalizing infelicities in speech acts seems to be a more complex issue because one will have to accommodate the different levels of truth within a theory of action. In a universe that changes over time it is not even clear *when* an infelicity occurs. Assume for example that John utters sentence (7.1) during a conversation he has with Robert.

(7.1) I promise I'll play badminton with you tomorrow. See you at 8 o'clock at the gym.

If John is not able to get up in time, his promise will be infelicitous, but it is not very clear when did his utterance become infelicitous. At the moment he uttered it? At the moment John did not manage to get up? At the moment Robert realized that John is not going to show up? Or at the moment when they met the next day and Robert reminded John that he had waited for him.

Study the relation between stratified logic and other knowledge representation formalisms: So far, we have shown only that stratified logic is an appropriate tool for modelling the pragmatic inferences and for signalling infelicities. It would be interesting to see exactly what the place is of stratified logic with respect to the other frameworks that are concerned with nonmonotonic reasoning, what the problems are that stratified logic is able to solve, and what its drawbacks are.

Appendix A

Computing pragmatic inferences and infelicities

```
;;; Stratified Semantic Tableaux


(in-package :screamer-user)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Handyman's corner - general
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;; explode a symbol into a list of characters

(defun explode(symbol)
  (map 'list
       #'(lambda(x) (intern (string x)))
       (symbol-name symbol)))


;;; implode a list of characters into a symbol

(defun implode(lista)
        (read-from-string
                (coerce (map 'list
                        #'(lambda(x) (coerce x 'character))
                        lista) 'string)))


;;; collapse a list three times
;;; ((((a b)) (((((a) c d))))) -> (a b (a) c d)


(defun collapse-list-thrice (l)
  (let ((rez nil))
    (mapc
     #'(lambda(x)
         (mapc
          #'(lambda(y)
              (mapc
               #'(lambda(z) (push z rez))
              y))
          x))
     l)
    rez))


(defun collapse-list-twice (l)
  (let ((rez nil))
         (mapc
          #'(lambda(y)
              (mapc
               #'(lambda(z) (push z rez))
               y))
          l)
    rez))

;;; flatten a list
;;; ((A) (((b) x))) -> (A b x)

(defun flatten(l)
  (cond ((null l) nil)
        ((atom (first l)) (cons (first l) (flatten (rest l))))
        (t (append (flatten (first l)) (flatten (rest l))))))

;;; remove special symbols from a list ($)

(defun remove-special (l)
  (remove-if #'(lambda(a) (if (eq (first (explode a)) '$) t nil))
             l))

;;;

(defun is-list-in (l1 l2)
  (cond ((null l1) t)
        (t (and
            (member (car l1) l2)
            (is-list-in (rest l1) l2)))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Handyman's corner - functions specific to stratified logic
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defun atomic-formula-type (phi)
  (when (equal (first phi) 'not) (setq phi (second phi)))
  (let* ((tip1 (first (explode (first phi))))
         (tip2 (second (explode (first phi))))
         (tip (implode (list tip1 tip2))))
    (case tip
          (I- 'I)
          (D- 'D)
          (otherwise 'U))))


(defun remove-predicate-type(pred)
  (let ((pred-list (explode pred)))
    (if (and
           (or (equal (car pred-list) 'U)
              (equal (car pred-list) 'I)
              (equal (car pred-list) 'D))
          (equal (second pred-list) '-))
         (implode (rest (rest pred-list)))
         pred)))


(defun change-strength-atomic-formula (phi strength)
  (let* ((f (if (equal (first phi) 'not) (second phi) phi))
         (negated? (if (equal (first phi) 'not) t nil))
         ; (tip (atomic-formula-type phi))
         (cleaned-pred (remove-predicate-type (first f))))
    (unless (equal strength 'U)
            (setq cleaned-pred
                  (implode (append (list strength '-)
                                   (explode cleaned-pred)))))
    (if negated?
        `(not ,(cons cleaned-pred (rest f)))
        (cons cleaned-pred (rest f)))))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Stratified semantic tableaux - or a variation
;;; on a Siskindian theme
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defvar *gamma* nil)         ; the queue of formulas that remain
                             ; to be processed

(defvar *model* nil)         ; a list of true and negated atomic formulas
                             ; constituting the model constructed for the
                             ; initial set *gamma*

(defvar *my-gensym-counter* -1) ; the index of the last symbol created by gensym

(defvar *delta* nil)         ; used during the universal instantiation

(defvar *instantiation* nil) ; contains an association list that
                             ; memorizes the utterances and their
                             ; associated constants

(defun empty? ()             ; returns T if the queue of formulas is empty
  (null *gamma*))


(defun pop-formula()
  (when (empty?) (break "Attempt to pop an empty queue"))
  (let ((phi  (first *gamma*)))
    (local (setf *gamma* (rest *gamma*)))
    phi))


(defun push-end-formula(phi)
  (if (empty?)
      (local (setf *gamma* (list phi)))
      (local (setf *gamma* (append *gamma* (list phi))))))
```

```
(defun push-formula(phi)
  (local (setf *gamma* (cons phi *gamma*))))


(defun negate(phi)
  (if (and (listp phi) (= (length phi) 2) (eq (first phi) 'not))
      (second phi)
      `(not ,phi)))


(defun add-to-model(phi &optional (utterance? nil))
  (when (and
         (member (negate phi) *model* :test #'equal)
         (equal (atomic-formula-type phi) 'U))
        (fail)) ; fails only if a U-contradiction occurs
    (unless (member phi *model* :test #'equal)
                 (local (setf *model* (cons phi *model*))))
    (let ((temp nil)
          (phi (if (eq (first phi) 'not) (second phi) phi)))
     (when utterance?
          (if (setf temp (assoc (first phi) *instantiation*))
              (local (setf (second temp)
                           (cons (remove-special (flatten (rest phi)))
                                 (second temp))))
              (local (setf *instantiation*
                         (cons (list (first phi)
                                     (list (remove-special
                                            (flatten (rest phi)))))
                               *instantiation*))))))
    *instantiation*)




(defun empty-model () '())

(defun subst-term (s v f)  ; substitutes term s for all free occurences
                           ; of v in the formula f.    f(v->s)
 (case (first f)
   (and `(and ,@(iterate (for g in (rest f)) (collect (subst-term s v g)))))
   (or `(or ,@(iterate (for g in (rest f)) (collect (subst-term s v g)))))
   (not `(not ,(subst-term s v (second f))))
   (implies `(implies ,(subst s v (second f)) ,(subst s v (third f))))
   (iff `(iff ,(subst s v (second f)) ,(subst s v (third f))))
   (exists (if (eq v (second f))
               f
               `(exists ,(second f) ,(subst-term s v (third f)))))
   (forall (if (eq v (second f))
               f
               `(forall ,(second f) ,(subst-term s v (third f)))))
   (otherwise (subst s v f :test #'eq))))




(defun ground-term? (s variables)
 (or (and (symbolp s)
          (not (member s variables :test #'equal)))
     (and (consp s)
          (every #'(lambda (u) (ground-term? u variables)) (rest s)))))



(defun a-subterm-of (s)
 (either
  s
  (progn (unless (consp s) (fail)) (a-subterm-of (a-member-of (rest s))))))



(defun a-ground-subterm-of (f &optional variables)
 (if (member f variables :test #'eq) (fail))
 (if (symbolp f)
     f
     (case (first f)
       (uttered (a-ground-subterm-of (second f) variables))
       (and (a-ground-subterm-of (a-member-of (rest f)) variables))
       (or (a-ground-subterm-of (a-member-of (rest f)) variables))
       (not (a-ground-subterm-of (second f) variables))
       (implies (a-ground-subterm-of (either (second f) (third f)) variables))
       (iff (a-ground-subterm-of (either (second f) (third f)) variables))
       (exists (a-ground-subterm-of (third f) (cons (second f) variables)))
       (forall (a-ground-subterm-of (third f) (cons (second f) variables)))
       (forallutt (a-ground-subterm-of (third f) (append (second f) variables)))
       (otherwise (let ((s (a-subterm-of (a-member-of (rest f)))))
                    (unless (ground-term? s variables) (fail))
                    s)))))


(defun a-ground-term()
  (a-ground-subterm-of (either (a-member-of *gamma*) (a-member-of *model*))))


(defun ground-terms()
  (remove-duplicates (all-values (a-ground-term))))


(defun make-constant-symbol()
  (local (setf *my-gensym-counter* (+ 1 *my-gensym-counter*)))
  (implode (coerce (format nil "C~D" *my-gensym-counter*) 'list)))


(defun loop-over(count)
;  (format t "~% count ~D " count)
;  (format t " ~% --- ~A ~% ---- ~A ~% ---- ~A ~% ----  ~% --- ~D ~%~%~% "
;             *gamma* *model* *delta* *instantiation*  *my-gensym-counter*)
;  (BREAK)
    (unless (or (zerop count) (empty?))
      (let* ((phi (pop-formula)))
        (case (first phi)
           (uttered ; it is an utterance
             (setf phi (second phi))
             (case (first phi)
               (not
                (case (first (second phi))
                   (not (push-formula (list 'uttered (second (second phi)))))
                   (and (push-formula (list 'uttered
                         `(or ,@(iterate
                          (for g in (rest (second phi)))
                          (collect `(not ,g)))))))
    (or (push-formula (list 'uttered
                          `(and ,@(iterate
                           (for g in (rest (second phi)))
                           (collect `(not ,g)))))))
    (implies (push-formula (list 'uttered
                             `(and ,(second (second phi))
                              (not ,(third (second phi)))))))
    (iff (push-formula (list 'uttered
                          `(iff ,(second (second phi))
                             (not ,(third (second phi)))))))
    (exists (push-formula (list 'uttered
                             `(forall ,(second (second phi))
                               (not ,(third (second phi)))))))
    (forall (push-formula (list 'uttered
                             `(exists ,(second (second phi))
                               (not ,(third (second phi)))))))
    (otherwise (add-to-model phi t)))  ; ! no rule for
                                 ; (not (forallutt form))
 (loop-over (length *gamma*)))
(and (mapc #'(lambda(x) (push-formula (list 'uttered x)))
               (rest phi))
     (loop-over (length *gamma*)))
(or  (push-formula (list 'uttered (a-member-of (rest phi))))
     (loop-over (length *gamma*)))
(implies (push-formula (list 'uttered
                        `(or (not ,(second phi)) ,(third phi))))
         (loop-over (length *gamma*)))
(iff (push-formula (list 'uttered
                     `(and (implies ,(second phi) ,(third phi))
                     (implies ,(third phi) ,(second phi)))))
 (loop-over (length *gamma*)))
(exists (push-formula (list 'uttered
                      (subst (make-constant-symbol) ; add this!
                         (second phi)
                         (third phi))))
    (loop-over (length *gamma*)))
(forall (let ((again? nil)
              (gt (ground-terms)))
          (cond ((null gt)
                 (push-formula (list 'uttered
                                  (subst (make-constant-symbol)
                                    (second phi)
                                    (third phi))))
                 (global (setf again? t)))
            (t
               (mapc
                 #'(lambda(x)
                    (unless (or
                             ; is not yet expanded
                             (member (list phi x) *delta*
                                    :test #'equal)
                             ; is not a special function
                             (and
                              (listp x)
                              (equal `$
                                  (first (explode (first x))))))
                          (local (setf *delta* (cons (list phi x)
                                                     *delta*)))
                          (push-formula (list 'uttered
                                              (subst x
                                                 (second phi)
                                                 (third phi))))
                          (global (setf again? t))))
                     gt)))
          (push-end-formula (list 'uttered phi))
          (if again?
              (loop-over (length *gamma*))
              (loop-over (- count 1)))))
    (otherwise
     (add-to-model phi t)
     (loop-over (length *gamma*)))))
(not
  (case (first (second phi))
    (not (push-formula (second (second phi))))
    (and (push-formula `(or ,@(iterate
                              (for g in (rest (second phi)))
                              (collect `(not ,g))))))
    (or (push-formula `(and ,@(iterate
                               (for g in (rest (second phi)))
                               (collect `(not ,g))))))
    (implies (push-formula `(and ,(second (second phi))
                                  (not ,(third (second phi))))))
    (iff (push-formula `(iff ,(second (second phi))
                             (not ,(third (second phi))))))
    (exists (push-formula `(forall ,(second (second phi))
                                   (not ,(third (second phi))))))
    (forall (push-formula `(exists ,(second (second phi))
                                   (not ,(third (second phi))))))
    (otherwise (add-to-model phi)))  ; ! no rule for
                                     ; (not (forallutt form))
  (loop-over (length *gamma*)))
(and (mapc #'push-formula (rest phi))
     (loop-over (length *gamma*)))
(or  (push-formula (a-member-of (rest phi)))
     (loop-over (length *gamma*)))
(implies (push-formula `(or (not ,(second phi)) ,(third phi)))
         (loop-over (length *gamma*)))
(iff (push-formula `(and (implies ,(second phi) ,(third phi))
                         (implies ,(third phi) ,(second phi))))
     (loop-over (length *gamma*)))
(exists (push-formula (subst (make-constant-symbol) ; add this!
                             (second phi)
                             (third phi)))
        (loop-over (length *gamma*)))
(forall (let ((again2? nil)
              (gt (ground-terms)))
          (cond ((null gt)
                 (push-formula (subst (make-constant-symbol)
                                      (second phi)
                                      (third phi)))
                 (global (setf again2? t)))
                (t
                 (mapc
                   #'(lambda(x)
                      (unless (or
                                            ; is not yet expanded
                                            (member (list phi x) *delta*
                                                   :test #'equal)
                                            ; is not a special function
                                            (and
                                             (listp x)
                                             (equal `$
                                                 (first (explode (first x))))))
                                     (local (setf *delta* (cons (list phi x)
                                                                *delta*)))
                                     (push-formula (subst x
                                                          (second phi)
                                                          (third phi)))
                                     (global (setf again2? t))))
                                gt)))
                     (push-end-formula phi)
                     (if again2?
                         (loop-over (length *gamma*))
                         (loop-over (- count 1)))))
           (forallutt (let* ((again3? nil)
                             (param (second phi))
                             (antecedent (second (third phi)))
                             (ant (if (equal (first antecedent) 'not)
                                      (first (second antecedent))
                                      (first antecedent)))
                             (gtc (assoc ant  *instantiation*))
                             (gt (if gtc (second gtc)  nil)))
                        (unless (null gt)
                         (unless (member (list phi gt) *delta* :test #'equal)
                            (local (setf *delta* (cons (list phi gt) *delta*)))
                            (let ((newphi phi))
                              (mapc
                               #'(lambda(xx)
                                   (mapc
                                    #'(lambda (p val)
                                        (setf newphi (subst val p newphi)))
                                    param
                                    xx)
                                   (push-formula (third newphi))
                                   (setf newphi phi)
                                   (global (setf again3? t)))
                               gt))
                            (global (setf again3? t))))
                        (push-end-formula phi)
                        (if again3?
                            (loop-over (length *gamma*))
                            (loop-over (- count 1)))))
           (otherwise (add-to-model phi)
                      (loop-over (length *gamma*)))))))



(defun a-model-of (sigma)
  (global (setf *gamma* nil)
          (setf *model* nil)
          (setf *delta* nil)
          (setf *instantiation* nil)
          (setf *my-gensym-counter* -1))
  (mapc #'push-end-formula sigma)
  (loop-over (length *gamma*))
  *model*)


(defun entails? (sigma phi)
  (all-values (a-model-of (cons (negate phi) sigma))))

(defun find-models (sigma)
  (remove-duplicates (all-values (a-model-of sigma)) :test #'equal))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Model specific functions - the basics
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



(defun arrange-model (formulas)
  (let ((models '()))
    (mapc
     #'(lambda(x)
         (let* ((Ufo (change-strength-atomic-formula x 'U))
                (negd? (if (equal (first x) 'not) 'yes 'no))
                (fftype (atomic-formula-type x))
                (pUfo (if (equal (first x) 'not) (second Ufo) Ufo)))
           (unless (member pUfo models :test #'equal  :key #'first)
               (push (list pUfo nil nil nil nil nil nil) models))
           (case fftype
             (U
              (case negd?
                (yes
                 (setf (third (first (member pUfo models
                                             :test #'equal :key #'first))) t))
                (otherwise
                 (setf (second (first (member pUfo models
                                             :test #'equal :key #'first))) t))))
             (I
              (case negd?
                (yes
                 (setf (fifth (first (member pUfo models
                                             :test #'equal :key #'first))) t))
                (otherwise
                 (setf (fourth (first (member pUfo models
                                             :test #'equal :key #'first))) t))))
             (D
              (case negd?
                (yes
                       (setf (seventh (first (member pUfo models
                                             :test #'equal :key #'first))) t))
                (otherwise
                 (setf (sixth (first (member pUfo models
                                             :test #'equal :key #'first))) t))))
             (otherwise
              (break "Unknown predicate type")))))
     formulas)
    models))
```

```
(defun get-all-models2 (formulas) ; the main function
  (let ((all-models nil)
        (felicitous-models nil)
        (felicitous-minimal-models nil))
    (mapc #'(lambda(x) (push (arrange-model x) all-models))
          (find-models formulas))
    (print 'MODELS)
    (print-models all-models)
    (if (null all-models)
        (format t "~% The utterance is false")
      (progn
        (print-model-ordering all-models #'smaller-content-3)
        (setf felicitous-models
              (remove-if #'null
                         (mapcar
                          #'(lambda(x) (is-the-model-felicitous? x))
                          all-models)))
        (if (null felicitous-models)
            (format t "~% The utterance is infelicitous")
          (progn
            (print 'FELICITOUS-MODELS)
            (print-models felicitous-models)
            (print-model-ordering felicitous-models #'smaller-content-3)
            (setf felicitous-minimal-models
                  (keep-minimal-models felicitous-models #'smaller-content-3))
          (print 'FELICITOUS-OPTIMISTIC-MODELS)
          (print-models felicitous-minimal-models)
          (print 'PRESUPPOSITIONS)
          (mapc #'(lambda(x) (print x))
                (collect-presuppositions felicitous-minimal-models))))))
    (terpri)
    (terpri)
    t
    ))


(defun print-one-model(model)
  (mapc #'(lambda(x)
            (format t "~% ~A" (first x))
            (when (second x) (format t "~41T*"))
            (when (third x) (format t "~47T*"))
            (when (fourth x) (format t "~56T*"))
            (when (fifth x) (format t "~62T*"))
            (when (sixth x) (format t "~71T*"))
            (when (seventh x) (format t "~77T*")))
        model))

(defun print-models (models)
  (let ((no 0))
    (mapc #'(lambda(x)
              (format t "~% Model number ~D" no)
              (unless (is-the-model-felicitous? x)
                    (format t " is infelicitous."))
              (format t "~% Formulas ~40TUndefeasible ~55TInfelicitous ~70TDefeasible")
              (format t "~% ~40TAff ~46TNeg ~55TAff ~61TNeg ~70TAff ~76TNeg")
              (incf no)
              (print-one-model x)
0              (format t "~%~%"))
```

```
          models)))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Model ordering functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defun print-model-ordering (models ordering-function)
  (let ((counter1 0))
    (mapc
     #'(lambda(m1)
         (do* ((go-over models (rest go-over))
               (counter2 0 (+ counter2 1))
               (m2 (when (listp go-over) (first go-over))
                   (when (listp go-over) (first go-over))))
              ((null go-over))
              (unless                       ; I do not compare the same models
               (or (null m2)
                   (equal (member m1 models)
                          (member m2 models)))
               (when (funcall ordering-function m1 m2)
                     (format t "~% m~D > m~D" counter1 counter2))))
         (setf counter1 (+ counter1 1)))
   models)))




;;; O(n^2) algorithm
(defun keep-minimal-models (models ordering-function)
  (mapc
   #'(lambda(m1)
       (block pupu
       (do* ((go-over models (rest go-over))
             (m2 (when (listp go-over) (first go-over))
                 (when (listp go-over) (first go-over))))
            ((null go-over))
            (unless                       ; I do not compare the same models
             (or (null m2)
                 (equal (member m1 models)
                        (member m2 models)))
             (when (funcall ordering-function m2 m1)
                   (setf (first (member m1 models :test #'equal)) nil)
                   (return-from pupu))))))
;            (print models)))
   models)
  (remove-if #'null models))



(defun smaller-content-3(m1 m2)
  (block pupu
    (mapc
     #'(lambda(x)
         (let ((y (if (member (first x) m1 :test #'equal :key #'first)
                      (first (member (first x) m1 :test #'equal :key #'first))
```

```
                    nil))
                (aff2? nil)
                (aff1? nil)
                (where1? nil)
                (where2? nil)
                (bool 'yes)
                (cancel1? nil)
                (cancel2? nil)
                (where 1))
             (when (null y) (return-from pupu nil)) ; x does not belong to m1
             (mapc #'(lambda(val2 val1)
                          (when (and val1 (null aff1?))
                                (setf aff1? bool))
                          (when (and val1 (not (equal aff1? bool)))
                                (setf cancel1? t))
                          (when val1
                                (push where where1?))
                          (when (and val2 (null aff2?))
                                (setf aff2? bool))
                          (when (and val2 (not (equal aff2? bool)))
                                (setf cancel2? t))
                          (when val2
                                (push where where2?))
                          (setf bool (if (eq bool 'yes) 'no 'yes))
                          (setf where (+ where 1)))
                   (rest x) (rest y))
             (unless (and (not cancel1?) cancel2?)
                     (when (not (eq aff1? aff2?)) (return-from pupu nil))
                     (when (and
                            (not (equal where1? where2?))
                            (is-list-in where1? where2?))
                           (return-from pupu nil))
                     (when (< (apply #'max where1?) (apply #'max where2?))
                           (return-from pupu nil)))))
     m2)
    t))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Analyzing the results
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


(defun is-the-model-felicitous? (model)
  (block pupu
         (mapc
          #'(lambda(x)
              (when (or
                     (and (second x) (fifth x))
                     (and (third x) (fourth x))
                     (and (fourth x) (fifth x)))
                    (return-from pupu nil)))
          model)
         model))


(defun collect-facts (models)
 (remove-duplicates
  (remove-if-not #'consp
                (collapse-list-thrice models)) :test #'equal))

(defun collect-presup-candidates (models)
 (remove-duplicates
  (let ((rez nil))
    (mapc
     #'(lambda(x)
         (when (or (sixth x) (seventh x))
               (push (first x) rez)))
     (collapse-list-twice models))
    rez)
  :test #'equal))


(defun collect-presuppositions (models)
  (let ((presuppositions nil))
    (mapc
     #'(lambda(fact)
         (block pupu
                (let ((affirmative? nil)) ; yes, no, or nil
                  (mapc
                   #'(lambda(model)
                       (let ((form (first (member fact model
                                                 :test #'equal :key #'first))))
                        (when form
                         (when (and ; contradiction
                                (or (second form) (fourth form) (sixth form))
                                (or (third form) (fifth form) (seventh form)))
                               (return-from pupu))
                         (when (and ; contradiction
                                (or (second form) (fourth form) (sixth form))
                                (eq affirmative? 'no))
                               (return-from pupu))
                         (when (and ; contradiction
                                (or (third form) (fifth form) (seventh form))
                                (eq affirmative? 'yes))
                               (return-from pupu))
                         (when (and ; set the polarity of the formula
                                (or (second form) (fourth form) (sixth form))
                                (or (null affirmative?)
                                    (eq affirmative? 'yes)))
                               (setf affirmative? 'yes))
                         (when (and ; set the polarity of the formula
                                (or (third form) (fifth form) (seventh form))
                                (or (null affirmative?)
                                    (eq affirmative? 'no)))
                               (setf affirmative? 'no)))))
                   models)
                  (if affirmative?
                      (push fact presuppositions)
                    (push `(not ,fact) presuppositions)))))
     (collect-presup-candidates models))
    presuppositions))
```

Appendix B

Stratified logic at work — a set of examples

This appendix provides a comprehensive set of examples that shows how stratified logic works. Each example contains

- the utterance or utterances that are analyzed;
- their logical translation in stratified logic and the relevant semantic and pragmatic knowledge;
- the result of the program having as input the corresponding theory.

B.1 Simple presuppositions

B.1.1 My cousin is not a bachelor

```
(defun bac1 ()
  (get-all-models2
   '((uttered (not (bachelor cousin)))
     (forall x (implies (bachelor x) (and (male x) (adult x) (not (married x)))))
     (forall x (implies (not (bachelor x)) (I-married x)))
     (forallutt (x) (implies (not (bachelor x)) (D-male x)))
     (forallutt (x) (implies (not (bachelor x)) (D-adult x))))))
```

```
SCREAMER-USER(28): (bac1)

MODELS
 Model number 0 is infelicitous.
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                           *
 (MARRIED COUSIN)                            *         *
 (MALE COUSIN)                          *                             *
 (ADULT COUSIN)                         *                             *


 Model number 1
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                           *
 (MARRIED COUSIN)                                       *
 (MALE COUSIN)                                                          *
 (ADULT COUSIN)                                                         *


FELICITOUS-MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                             *
 (MARRIED COUSIN)                                       *
 (MALE COUSIN)                                                          *
 (ADULT COUSIN)                                                         *


FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                             *
 (MARRIED COUSIN)                                       *
 (MALE COUSIN)                                                          *
 (ADULT COUSIN)                                                         *


PRESUPPOSITIONS
(ADULT COUSIN)
(MALE COUSIN)
```

B.1.2 John does not regret that Mary came to the party

```
(defun jm1()
  (get-all-models2
   '((uttered (not (regret john ($come-party mary))))
     (forallutt (x y) (implies (not (regret x ($come-party y)))
                               (D-come-party y)))
     (forallutt (x y) (implies (regret x ($come-party y))
                               (I-come-party y))))))

SCREAMER-USER(32): (jm1)

MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))              *
 (COME-PARTY MARY)                                      *              *


 Model number 1
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))              *
 (COME-PARTY MARY)                                                     *


FELICITOUS-MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))              *
 (COME-PARTY MARY)                                      *              *


 Model number 1
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                                                    *


FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                                     *               *


PRESUPPOSITIONS
(COME-PARTY MARY)
```

B.2 Presupposition cancellation

B.2.1 John does not regret that Mary came to the party because she didn't come

```
(defun jm3()
  (get-all-models2
   '((uttered (and (not (come-party mary))
                   (not (regret john ($come-party mary)))))
     (forallutt (x y) (implies (not (regret x ($come-party y)))
                               (D-come-party y)))
     (forallutt (x y) (implies (regret x ($come-party y))
                               (I-come-party y))))))

SCREAMER-USER(34): (jm3)

MODELS
 Model number 0 is infelicitous.
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                            *        *              *


 Model number 1
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                            *                       *


FELICITOUS-MODELS
 Model number 0
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                            *                       *


FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))             *
 (COME-PARTY MARY)                            *                       *
```

B.3 Signalling infelicities

B.3.1 John regrets that Mary came to the party, but she didn't come

```
(defun jm4()
  (get-all-models2
   '((uttered (and (not (come-party mary))
                   (regret john ($come-party mary))))
     (forallutt (x y) (implies (not (regret x ($come-party y)))
                               (D-come-party y)))
     (forallutt (x y) (implies (regret x ($come-party y))
                               (I-come-party y))))))

SCREAMER-USER(35): (jm4)

MODELS
 Model number 0 is infelicitous.
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))       *
 (COME-PARTY MARY)                            *        *                *


 Model number 1 is infelicitous.
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY MARY))       *
 (COME-PARTY MARY)                            *        *

 The utterance is infelicitous
```

B.4 Solving the projection problem

From now on we will reproduce only the felicitous optimistic models and the associated presuppositions for each utterance or set of utterances.

B.4.1 My cousin is a bachelor or a spinster *or—cancellation*

```
(defun bac2 ()
  (get-all-models2
   '((uttered (or (bachelor cousin) (spinster cousin)))
     (implies (or (bachelor cousin) (spinster cousin))
              (or (not (D-bachelor cousin)) (not (D-spinster cousin))))
     (forall x (implies (bachelor x) (and (male x) (adult x) (not (married x)))))
     (forallutt (x) (implies (not (bachelor x)) (I-married x)))
     (forallutt (x) (implies (not (bachelor x)) (D-male x)))
     (forallutt (x) (implies (not (bachelor x)) (D-adult x)))
     (forall x (implies (spinster x) (and (female x) (adult x) (not (married x)))))
     (forallutt (x) (implies (not (spinster x)) (I-married x)))
     (forallutt (x) (implies (not (spinster x)) (D-female x)))
     (forallutt (x) (implies (not (spinster x)) (D-adult x)))
     (forall x (iff (male x) (not (female x)))))))

SCREAMER-USER(29): (bac2)

FELICITOUS-OPTIMISTIC-MODELS
```

```
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (SPINSTER COUSIN)                       *
 (BACHELOR COUSIN)                             *                             *
 (MARRIED COUSIN)                              *
 (ADULT COUSIN)                          *
 (FEMALE COUSIN)                         *                             *
 (MALE COUSIN)                                 *


 Model number 1
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (SPINSTER COUSIN)                       *
 (BACHELOR COUSIN)                             *                             *
 (MARRIED COUSIN)                              *
 (FEMALE COUSIN)                         *
 (ADULT COUSIN)                          *                             *
 (MALE COUSIN)                                 *


 Model number 2
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                       *
 (MARRIED COUSIN)                              *
 (ADULT COUSIN)                          *
 (MALE COUSIN)                           *                             *
 (SPINSTER COUSIN)                             *                             *
 (FEMALE COUSIN)                               *


 Model number 3
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR COUSIN)                       *
 (MARRIED COUSIN)                              *
 (MALE COUSIN)                           *
 (ADULT COUSIN)                          *                             *
 (SPINSTER COUSIN)                             *                             *
 (FEMALE COUSIN)                               *


PRESUPPOSITIONS
(ADULT COUSIN)
```

B.4.2 If Mary came to the party then John will regret that Sue came to the party — *if ... then ... no—cancellation*

```
(defun jms1()
  (get-all-models2
   '((uttered (implies (come-party mary)
                       (regret john ($come-party sue))))
     (forallutt (x y) (implies (not (regret x ($come-party y)))
                               (D-come-party y)))
     (forallutt (x y) (implies (regret x ($come-party y))
                               (I-come-party y))))))

SCREAMER-USER(36): (jms1)

FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (REGRET JOHN ($COME-PARTY SUE))         *
 (COME-PARTY SUE)                                       *              *


 Model number 1
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (COME-PARTY MARY)                             *


PRESUPPOSITIONS
(COME-PARTY SUE)
```

B.5 Existential presuppositions

B.5.1 Ross worships Zeus (we know nothing about Zeus)

```
(defun ross1()
  (get-all-models
   '((exists x
            (exists y
                    (and (ross x)
                         (definite-reference x)
                         (zeus y)
                         (definite-reference y)
                         (worships x y))))
     (forall x (implies (definite-reference x) (D-exists! x))))))

SCREAMER-USER(24): (ross1)

FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (WORSHIPS C0 C1)                        *
 (DEFINITE-REFERENCE C1)                 *
 (ZEUS C1)                               *
 (DEFINITE-REFERENCE C0)                 *
 (ROSS C0)                               *
 (EXISTS! C1)                                                          *
 (EXISTS! C0)                                                          *


PRESUPPOSITIONS
(EXISTS! C0)
(EXISTS! C1)
```

B.5.2 Ross worships Zeus (we know *everything* about Zeus)

```
(defun ross2()
  (get-all-models
   '((exists x
            (exists y
                    (and (ross x)
                         (definite-reference x)
                         (zeus y)
                         (definite-reference y)
                         (worships x y))))
     (forall x (iff (ross x) (not (zeus x))))
     (forall x (implies (zeus x) (not (exists! x))))
     (forall x (implies (zeus x) (eow! x)))
     (forall x (implies (definite-reference x) (D-exists! x))))))

FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (WORSHIPS C0 C1)                       *
 (DEFINITE-REFERENCE C1)                *
 (ZEUS C1)                              *
 (DEFINITE-REFERENCE C0)                *
 (ROSS C0)                              *
 (ROSS C1)                                    *
 (ZEUS C0)                                    *
 (EOW! C1)                              *
 (EOW! C0)                              *
 (EXISTS! C1)                                 *                              *
 (EXISTS! C0)                                                                *

PRESUPPOSITIONS
(EXISTS! C0)
```

B.6 Sequence of utterances

B.6.1 John is a not a bachelor. I regret that you have misunderstood me. He is only five years old. You realize he cannot date women. It is not him who is not a bachelor!

```
(defun story5()
  (get-all-models2
   '((uttered (not (bachelor john)))
     (uttered (regret speaker ($misunderstood hearer speaker)))
     (uttered (and (male john) (not (adult john))))
     (uttered (realize hearer ($cannot-date-women john)))
     (uttered (cleft ($not-bachelor john)))
     (forall x (implies (bachelor x) (and (male x) (adult x) (not (married x)))))
     (forallutt (x) (implies (not (bachelor x)) (I-married x)))
     (forallutt (x) (implies (not (bachelor x)) (D-male x)))
     (forallutt (x) (implies (not (bachelor x)) (D-adult x)))
     (forallutt (x y) (implies (not (regret x ($misunderstood y x)))
                               (D-misunderstood y x)))
     (forallutt (x y) (implies (regret x ($misunderstood y x))
                               (I-misunderstood y x)))
     (forallutt (x y) (implies (not (realize x ($cannot-date-women y)))
                               (not (D-date-women y))))
     (forallutt (x y) (implies (realize x ($cannot-date-women y))
                               (not (I-date-women y))))
     (forallutt (x) (implies (cleft ($not-bachelor x))
                               (exists y (and (not (bachelor y))
                                              (D-exists! y))))))))

FELICITOUS-OPTIMISTIC-MODELS
 Model number 0
 Formulas                              Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR JOHN)                              *
 (REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
 (REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
 (CLEFT ($NOT-BACHELOR JOHN))           *
 (MARRIED SPEAKER)                            *
 (ADULT SPEAKER)                        *
```

```
(MALE SPEAKER)                          *
(MARRIED HEARER)                              *
(ADULT HEARER)                          *
(MALE HEARER)                           *
(MARRIED JOHN)                                       *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                                           *


Model number 1
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
(MARRIED SPEAKER)                             *
(ADULT SPEAKER)                         *
(MALE SPEAKER)                          *
(MARRIED HEARER)                              *
(ADULT HEARER)                          *
(MALE HEARER)                           *
(MARRIED JOHN)                                         *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                            *
(EXISTS! CO)                                                          *
(BACHELOR CO)                                 *
(MARRIED CO)                                  *
(ADULT CO)                              *
(MALE CO)                               *


Model number 2
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
(MARRIED SPEAKER)                             *
(ADULT SPEAKER)                         *
(MALE SPEAKER)                          *
(BACHELOR HEARER)                             *
(MARRIED JOHN)                                         *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                                           *


Model number 3
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
(MARRIED SPEAKER)                             *
```

```
(ADULT SPEAKER)                         *
(MALE SPEAKER)                          *
(BACHELOR HEARER)                             *
(MARRIED JOHN)                                         *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                           *
(EXISTS! CO)                                                          *
(BACHELOR CO)                                 *
(MARRIED CO)                                  *
(ADULT CO)                              *
(MALE CO)                               *

Model number 4
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
(BACHELOR SPEAKER)                            *
(MARRIED HEARER)                              *
(ADULT HEARER)                          *
(MALE HEARER)                           *
(MARRIED JOHN)                                         *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                                           *


Model number 5
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
(BACHELOR SPEAKER)                            *
(MARRIED HEARER)                              *
(ADULT HEARER)                          *
(MALE HEARER)                           *
(MARRIED JOHN)                                         *
(MALE JOHN)                             *                             *
(ADULT JOHN)                                  *                       *
(MISUNDERSTOOD HEARER SPEAKER)                         *              *
(DATE-WOMEN JOHN)                                            *
(EXISTS! CO)                                                          *
(BACHELOR CO)                                 *
(MARRIED CO)                                  *
(ADULT CO)                              *
(MALE CO)                               *


Model number 6
Formulas                               Undefeasible   Infelicitous   Defeasible
                                       Aff   Neg      Aff   Neg      Aff   Neg
(BACHELOR JOHN)                               *
(REGRET SPEAKER
      ($MISUNDERSTOOD HEARER SPEAKER))  *
(REALIZE HEARER
      ($CANNOT-DATE-WOMEN JOHN))        *
(CLEFT ($NOT-BACHELOR JOHN))            *
```

```
 (BACHELOR SPEAKER)                            *
 (BACHELOR HEARER)                             *
 (MARRIED JOHN)                                         *
 (MALE JOHN)                             *                             *
 (ADULT JOHN)                                  *                       *
 (MISUNDERSTOOD HEARER SPEAKER)                         *              *
 (DATE-WOMEN JOHN)                                                           *


 Model number 7
 Formulas                               Undefeasible   Infelicitous   Defeasible
                                        Aff   Neg      Aff   Neg      Aff   Neg
 (BACHELOR JOHN)                               *
 (REGRET SPEAKER
       ($MISUNDERSTOOD HEARER SPEAKER))  *
 (REALIZE HEARER
       ($CANNOT-DATE-WOMEN JOHN))        *
 (CLEFT ($NOT-BACHELOR JOHN))            *
 (BACHELOR SPEAKER)                            *
 (BACHELOR HEARER)                             *
 (MARRIED JOHN)                                         *
 (MALE JOHN)                             *                             *
 (ADULT JOHN)                                  *                       *
 (MISUNDERSTOOD HEARER SPEAKER)                         *              *
 (DATE-WOMEN JOHN)                                            *
 (EXISTS! CO)                                                          *
 (BACHELOR CO)                                 *
 (MARRIED CO)                                  *
 (ADULT CO)                              *
 (MALE CO)                               *


PRESUPPOSITIONS
(EXISTS! CO)
(MISUNDERSTOOD HEARER SPEAKER)
(MALE JOHN)
(NOT (DATE-WOMEN JOHN))
```

Bibliography

[Amsterdam, 1990] J. Amsterdam. The Iterate Manual. Technical Report A.I. Memo No. 1236, M.I.T. Artificial Intelligence Laboratory, October 1990.

[Atlas, 1988] J.D. Atlas. What are negative existence statements about? *Linguistics and Philosophy*, 11:373–394, 1988.

[Austin, 1962] J.L. Austin. *How to Do Things with Words.* Harvard University Press, 1962.

[Bell and Machover, 1986] J.L. Bell and M. Machover. *A Course in Mathematical Logic.* Elsevier Science Publishers B.V., 1986.

[Brewka, 1994] G. Brewka. Reasoning about priorities in default logic. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 940–945, 1994.

[Chellas, 1980] B.F. Chellas. *Modal Logic: an Introduction.* Cambridge University Press, 1980.

[Delgrande, 1994] J.P. Delgrande. A preference-based approach to default reasoning: Preliminary report. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 902–908, 1994.

[Frege, 1892] G. Frege. Über sinn und bedeutung. *Z. Philos. Philos. Kritik*, 100:373–394, 1892. reprinted as: On Sense and Nominatum, In Feigl H. and Sellars W., editors, *Readings in Philosophical Analysis*, pages 85–102, Appleton-Century-Croft, New York, 1947.

[Gazdar, 1979] G.J.M. Gazdar. *Pragmatics: Implicature, Presupposition, and Logical Form.* Academic Press, 1979.

[Geis, 1982] M.L. Geis. *The Language of Television Advertising.* Academic Press, 1982.

[Ginsberg, 1988] M.L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.

[Gordon and Lakoff, 1975] D. Gordon and G. Lakoff. Conversational postulates. In Cole P. and Morgan J.L., editors, *Syntax and Semantics, Speech Acts*, volume 3, pages 83–106. Academic Press, 1975.

[Green, 1989] G.M. Green. *Pragmatics and Natural Language Understanding.* Lawrence Erlbaum Associates, Inc., Publishers, 1989.

[Green, 1990] N. Green. Normal state implicature. In *Proceedings 28th Annual Meeting of the Association for Computational Linguistics*, pages 89–96, 1990.

[Green, 1992] N. Green. Conversational implicatures in indirect replies. In *Proceedings 30th Annual Meeting of the Association for Computational Linguistics*, pages 64–71, 1992.

[Grice, 1975] H.P. Grice. Logic and conversation. In Cole P. and Morgan J.L., editors, *Syntax and Semantics, Speech Acts*, volume 3, pages 41–58. Academic Press, 1975.

[Grice, 1978] H.P. Grice. Further notes on logic and conversation. In Cole P., editor, *Syntax and Semantics, Pragmatics*, volume 9, pages 113–127. Academic Press, 1978.

[Hintikka, 1959] K.J.J. Hintikka. Existential presuppositions and existential commitments. *Journal of Philosophy*, 56:125–137, 1959.

[Hintikka, 1970] K.J.J. Hintikka. Existential presuppositions and uniqueness presuppositions. In Lambert K., editor, *Philosophical Problems in Logic*, pages 20–55. Humanities Press, New York, 1970.

[Hirschberg, 1985] J.B. Hirschberg. A theory of scalar implicature. Technical Report MS-CIS-85-56, Department of Computer and Information Science, University of Pennsylvania, 1985.

[Hirst, 1991] G. Hirst. Existence assumptions in knowledge representation. *Artificial Intelligence*, 49:199–242, 1991.

[Hobbs *et al.*, 1993] J.R. Hobbs, M. Stickel, D. Appelt, and P. Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[Hobbs, 1985] J.R. Hobbs. Ontological promiscuity. In *Proceedings 23rd Annual Meeting of the Association for Computational Linguistics*, pages 61–69, 1985.

[Horn, 1972] L.R. Horn. *On the Semantic Properties of Logical Operators in English.* PhD thesis, University of California, Los Angeles, 1972.

[Horton, 1987] D.L. Horton. Incorporating agents' beliefs in a model of presupposition. Master's thesis, Dept. of Computer Science, University of Toronto, 1987. Tech. Report CSRI-201, Computer Systems Research Institute, University of Toronto.

[Horton and Hirst, 1988] D. Horton and G. Hirst. Presuppositions as beliefs. In *Proceedings of the International Conference on Computational Linguistics, COLING*, pages 255–260, 1988.

[Kaplan, 1982] J. Kaplan. Cooperative responses from a portable natural language database query system. In Brady M. and Berwick R.C., editors, *Computational Models of Discourse*, pages 167–208. The MIT Press, 1982.

[Karttunen and Peters, 1979] L. Karttunen and S. Peters. Conventional implicature. In Oh C.K. and Dinneen D.A, editors, *Syntax and Semantics, Presupposition*, volume 11, pages 1–56. Academic Press, 1979.

[Karttunen, 1971] L. Karttunen. Implicative verbs. *Language*, 47:340–358, 1971.

[Karttunen, 1974] L. Karttunen. Presupposition and linguistic context. *Theoretical Linguistics*, 1:3–44, 1974.

[Katz and Langendoen, 1976] J.J. Katz and D.T. Langendoen. Pragmatics and presupposition. *Language*, 52:1–17, 1976.

[Kay, 1992] P. Kay. The inheritance of presuppositions. *Linguistics & Philosophy*, 15:333–379, 1992.

[Kifer and Lozinskii, 1992] M. Kifer and E.L. Lozinskii. A logic for reasoning with inconsistency. *Journal of Automated Reasoning*, 9(2):179–215, November 1992.

[Kifer and Subrahmanian, 1992] M. Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12(4):335–368, April 1992.

[Labov, 1973] W. Labov. The boundaries of words and their meaning. In Bailey C.J. and Shuy R., editors, *New Ways of Analyzing Variation in English*, pages 340–373. Georgetown University Press, Washington DC, 1973.

[Lasersohn, 1993] P. Lasersohn. Existence presuppositions and background knowledge. *Journal of Semantics*, 10:113–122, 1993.

[Lejewski, 1954] C. Lejewski. Logic and existence. *British Journal for the Philosophy of Science*, 5:104–119, 1954.

[Levinson, 1983] S.C. Levinson. *Pragmatics*. Cambridge University Press, 1983.

[McCarthy, 1980] J. McCarthy. Circumscription — a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–40, 1980.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[Meinong, 1904] A. Meinong. Über gegenstandstheorie. In Meinong A., editor, *Untersuchungen zur Gegenstandstheorie und Psychologie*. Barth, Leipzig, 1904. reprinted in: The theory of objects, Chisholm R.M. editor, *Realism and the Background of Phenomenology*, pages 76–117. Free Press, Glencoe, IL, 1960.

[Mercer and Reiter, 1982] R.E. Mercer and R. Reiter. The representation of presuppositions using defaults. In *Proceedings of the Fourth Biennal Conference of the Canadian Society for the Computational Studies of Intelligence (CSCSI/SCEIO)*, pages 103–107, 1982.

[Mercer, 1987] R.E. Mercer. *A Default Logic Approach to the Derivation of Natural Language Presuppositions.* PhD thesis, Department of Computer Science, University of British Columbia, 1987.

[Mercer, 1988a] R.E. Mercer. Solving some persistent presupposition problems. In *Proceedings of the International Conference on Computational Linguistics, COLING*, pages 420–425, 1988.

[Mercer, 1988b] R.E. Mercer. Using default logic to derive natural language presuppositions. In *Proceedings of the Seventh Biennal Conference of the Canadian Society for the Computational Studies of Intelligence (CSCSI/SCEIO)*, pages 14–21, 1988.

[Mercer, 1990] R.E. Mercer. Deriving natural language presuppositions from complex conditionals. In *Proceedings of the Eighth Biennal Conference of the Canadian Society for the Computational Studies of Intelligence (CSCSI/SCEIO)*, pages 114–120, 1990.

[Mercer, 1991] R.E. Mercer. Presuppositions and default reasoning: A study in lexical pragmatics. In Pustejovski J. and Bergler S., editors, *ACL SIG Workshop on Lexical Semantics and Knowledge Representation*, pages 224–237, 1991.

[Nait Abdallah, 1989] A. Nait Abdallah. An extended framework for default reasoning. In Csirik J., Demetrovics J., and Gecseg F., editors, *International Conference on Fundamentals of Computation Theory*, pages 339–348. Springer LNCS 380, 1989.

[Nait Abdallah, 1991] A. Nait Abdallah. Kernel knowledge versus belt knowledge in default reasoning: a logical approach. In Dehne F., Fiala F., and Koczkodaj W., editors, *International Conference on Computing and Information*, pages 675–686. Springer LNCS 497, 1991.

[Parsons, 1980] T. Parsons. *Nonexistent Objects.* Yale University Press, New Haven, CT, 1980.

[Quine, 1947] W.V.O. Quine. *Mathematical Logic.* Cambridge, 1947.

[Quine, 1949] W.V.O. Quine. Designation and existence. In Feigl H. and Sellars W., editors, *Readings in Philosophical Analysis*, pages 44–51. Appleton-Century-Croft, New York, 1949.

[Rapaport, 1985] W.J. Rapaport. Meinongian semantics for propositional semantic networks. In *Proceedings 23rd Annual Meeting of the Association for Computational Linguistics*, pages 43–48, 1985.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1990] E. Reiter. The computational complexity of avoiding conversational implicatures. In *Proceedings 28th Annual Meeting of the Association for Computational Linguistics*, pages 97–104, 1990.

[Russell, 1905] B. Russell. On denoting. *Mind n.s.*, 14:479–493, 1905. reprinted in: Feigl H. and Sellars W. editors, *Readings in Philosophical Analysis*, pages 103–115. Appleton-Century-Croft, New York, 1949.

[Sandt, 1992] R.A. van der Sandt. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377, 1992.

[Siskind and McAllester, 1993a] J.M. Siskind and D.A. McAllester. Nondeterministic Lisp as a substrate for constraint logic programming. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 133–138, 1993.

[Siskind and McAllester, 1993b] J.M. Siskind and D.A. McAllester. Screamer: A portable efficient implementation of nondeterministic Common Lisp. Technical Report IRCS-93-03, University of Pennsylvania, Institute for Research in Cognitive Science, July 1 1993.

[Siskind, 1991] J.M. Siskind. Screaming yellow zonkers. Technical report, M.I.T. Artificial Intelligence Laboratory, September 1991.

[Smullyan, 1970] R.M. Smullyan. *First-Order Logic.* Springer-Verlag, New York, 1970.

[Soames, 1979] S. Soames. A projection problem for speaker presuppositions. *Linguistic Inquiry*, 10(4):623–666, Fall 1979.

[Soames, 1982] S. Soames. How presuppositions are inherited: A solution to the projection problem. *Linguistic Inquiry*, 13(3):483–545, Summer 1982.

[Soames, 1989] S. Soames. Presupposition. In Gabbay D. and Guenthner F., editors, *Handbook of Philosophical Logic — Topics in the Philosophy of Language*, pages 553–617. D. Reidel Publishing Company, 1989.

[Stalnaker, 1973] R.C. Stalnaker. Presuppositions. *Journal of Philosophical Logic*, 2:447–457, 1973.

[Strawson, 1950] P.F. Strawson. On referring. *Mind*, 59:320–344, 1950.

[Velman, 1981] F. Velman. Data semantics. In Groenendijk J.A.G. et. al., editor, *Formal Methods in the Study of Language*. Mathematisch Centrum, Amsterdam, 1981.

[Walker, 1992] M.A. Walker. Redundancy in collaborative dialogue. In *Proceedings of the International Conference on Computational Linguistics, COLING*, pages 345–351, 1992.

[Weischedel, 1979] R.M. Weischedel. A new semantic computation while parsing: Presupposition and entailment. In Oh C.K. and Dinneen D.A, editors, *Syntax and Semantics, Presupposition*, volume 11, pages 155–182. Academic Press, 1979.

[Wilson and Sperber, 1979] D. Wilson and D. Sperber. Ordered entailments: An alternative to presuppositional theories. In Oh C.K. and Dinneen D.A, editors, *Syntax and Semantics, Presupposition*, volume 11, pages 299–324. Academic Press, 1979.

[Zeevat, 1992] H. Zeevat. Presupposition and accommodation in update semantics. *Journal of Semantics*, 9:379–412, 1992.