Contexts : A Formalization and Some Applications

Ramanathan V. Guha

February 10, 1995

Contents

Chapter 1

Introduction

Most research on symbolic Artificial Intelligence (AI) has assumed a certain model of an AI system. We begin by outlining the model and analyzing certain fundamental assumptions underlying this model. We then discuss some problems these assumptions lead to and as we go along, outline a solution for these problems. The solution, the mechanism of contexts, is developed in detail in the later chapters.

1.1 The Symbolic Model of AI

Though many different architectures have been proposed for (symbolic) AI systems, most of them share the following theme [Advice Taker] [Symbol System Hypothesis]. There is a repository of knowledge, the Knowledge Base (KB), and a set of procedures which operate on this to produce the intelligent behavior. Inputs to the system are added to the KB. They are translated by an appropriate front end into the language of the KB before being added to the KB.

In the currently dominant school of thought, this KB uses a declarative encoding (an encoding with a denotational semantics) and the procedures carry out deductions. The system has some domain of competence and the overall goal of AI is to build a system whose domain of competence is comparable to that of humans. The KB primarily contains the systems knowledge about its domain of competence. Occasionally the KB might also have some meta-knowledge about how to use this knowledge.

In this thesis, we will be concerned primarily with the knowledge base part of this model. The knowledge base constituent of most realizations of this model (especially those with a declarative KB) share certain distinctive features by virtue of the assumptions made by the model. Let us begin by discussing the more significant of these features. After discussing these features, we will examine the problems they lead to.

1.2 Aspects of the Symbolic Model

The KB consists of a set of expressions (sentences) in a certain vocabulary. Each sentence conveys some truth about the domain. The meaningfulness and truth of each sentence is independent of the presence or absence of other sentences. To use Quine's terminology

[Word and Object], these are assumed to be *eternal* or *universal* sentences, i.e., all foreseeable dependencies (of the sentence) have been made explicit in the sentence.

1.2.1 Communicative Expressions vs KB Expressions

It should be noted that sentences in the KB of this model are very different from natural language (NL) utterances. NL utterances are far from being *universal*. They usually make a number of assumptions and depend very heavily on situational factors to convey their intended meaning. These situational factors include not just the previous utterances, but also broader factors such as the goals of the discoursants, the socio-cultural setting of the discourse, etc.

1.2.2 Exceptions

Since a sentence does not depend on any other sentence for its meaningfulness or truth, it is not allowed to make any assumptions. This is certainly the case for systems based on first order logic. This is indeed a very desirable feature for a system being used primarily to guarantee the rigor of arguments. However, it is not so desirable in a system intended to arrive at conclusions about the common sense world and leads to bottlenecks such as the qualification problem [Philosophical problems of AI].

Most general (i.e., quantified) statements we can make about the world have exceptions. To use these statements to derive conclusions we have to assume that we are not dealing with an exception. The inability to make assumptions therefore translates into an inability to deal with exceptions. The extension of logic to deal with defaults ([Circumscription] [Default Logic]) was intended to repair this. Sentences in this extension of first order logic might make an assumption that the objects involved are not exceptions and other sentences in the KB provide a (partial) specification of these exceptions. Of course, for this scheme to work, the exceptions should be specifiable in the vocabulary chosen to encode the KB.

Since defaults are primarily intended to enable the system to cope with exceptions, the assumptions are expected to be satisfied most of the time. Therefore, even with the introduction of defaults, it is still the case that sentences are not expected to depend crucially on external factors for their meaningfulness or truth. This point will become more apparent when we discuss the problems with this framework.

1.2.3 Homogeneity of the KB

Another interesting aspect of most knowledge bases is their homogeneity. This manifests itself in a number of ways.

- While many KBs structure the domain (e.g., in taxonomic hierarchies) they rarely structure the knowledge they have about the domain. E.g., The same vocabulary is used uniformly throughout the KB.

- Typically, the KB contains a single theory of the world. To allow meaningful inference, this theory must be kept consistent.

- The single theory approach also implies that the theory should be kept independent of particular problems, i.e., the representation should not be tailored specifically towards solving certain problems.

It should be noted however that there have been proposals such as Minsky's [Society of Mind] which propose a slightly different strategy, one which is far more heterogenous than the model described above. However, little or no work has been done in developing this model to a point where a technical assessment of it is possible.

1.3 Problems with this model

While this model is an excellent one to begin with, it does have its share of problems. In fact, much of the work in AI over the last thirty years has been directed towards realizing this model and towards refining it. Our position, unlike that of [Brian Smith] [Rumelhart], is that this model is imminently plausible as a model of AI and that the best line of attack is to identify and solve these problems (as opposed to abandoning this model in favor of a situated or connectionist approach.)

We now consider some of the problems that this model has and illustrate these with examples. As we go along, we also try to develop a rough solution for these problems. The later chapters will be devoted to providing the details of this proposed solution.

Most of these problems mentioned here are from experiences gathered in trying to build the Cyc knowledge base. The Cyc knowledge base [Guha and Lenat], which is being built at MCC, is intended to be a large common sense KB covering most "common sense" domains. It currently contains over 1.5 million sentences and covers a very wide range of phenomena. Being the first attempt to build the knowledge base that has been assumed by most work in AI, it exposes certain problems which have to be dealt with before this approach can succeed.

1.3.1 The Vocabulary

One of the first things we need to do in representing any domain is to pick a vocabulary for encoding the KB's knowledge of that domain. The vocabulary should allow expression of the phenomena we expect to find. The choice of the vocabulary can therefore exclude certain phenomena from consideration by the KB. If certain phenomena are excluded, this constitutes an assumption that these phenomena are not important/do not exist. The decision to exclude these could either be by design or simply accidental. It is almost inevitable that certain parts of the domain will be overlooked in the representation process.

Consider representing a theory of commercial transactions. More specifically, assume we are interested in representing the concept of buying and selling. We decide that to refer to the event of X buying Y from Z, we use the term (Buy X Y Z). We realize that this is insufficient since it cannot distinguish between two separate events involving X buying Y from Z (i.e., if Z sells Y to X, Z then buys Y back from X and sells it to him again, both the first and third transaction will be (Buy X Y Z) and we cannot distinguish between them.)

To uniquely identify the sale, we add an extra argument, to represent the time at which the sale took place. So, to refer to X buying Y from Z at time T1, we use the term (Buy

X Y Z T1). At first glance, this seems adequate in that it allows us to accurately refer to any particular buying event. However, even this excludes certain phenomena. Consider the following admittedly farfetched situation. Z has engaged two agents (computer programs) to sell Y. Both of these simultaneously make deals with X's agents (again, computer programs) to sell Y. Here we have two sales of Y to X at the same time and our vocabulary does not allow us to distinguish between them.

We decide that the way out of this problem is to reify the action. Rather than writing (Buy X Y Z), we write Buying(Buying001) $\wedge$ buyer(Buying001 X) $\wedge$ object(Buying001 Y) $\wedge$ seller(Buying001 Z). If two simultaneous deals are made between Z and X to buy Y, these will be different buying actions (Buying001 and Buying002). If the price negotiated for Buying001 is p, we write price(Buying001) = p.

Notice that though this new scheme based on the reification of event is more expressive, it is also significantly more cumbersome.

Unfortunately, it is still not the case that we have taken all possible (buying related) phenomena into account. Remember that we write price(Buying001) = P to state that the price paid was P. Now, someone points out that this is ambiguous. In certain purchases, there might actually be many different prices involved (and it might not be possible to agglomerate them into a single term). For example, in addition to a sum of money, the buyer might also have a certain obligation towards the seller (e.g., in the case of one country selling another weapons) and it is not really possible to lump the obligation and money exchanged into a single term. By using a single concept of price, our vocabulary precludes us from considering these phenomena. So it is still the case that not all buying related phenomena are expressible.

We could of course further refine our vocabulary to cover these phenomena. But in the process we also make the vocabulary more awkward to use. Requiring a tradeoff between the part of the domain which can be covered and the usability of the vocabulary seems undesirable.

It also seems that with a little searching, we can always uncover inexpressible phenomenon. At some point, we have to finalize the vocabulary and carry on with the representation.

The obvious objection to this argument is that after a certain point, the excluded phenomena will be increasingly strange and unlikely to be encountered and therefore it is acceptable to ignore them. For example, the phenomenon of multiple simultaneous deals between the seller and buyer is indeed a little out of the ordinary.

But, despite its strangeness, we (humans) can quite easily conceptualize this phenomenon. We might be somewhat at a loss to predict what exactly will happen, but we can certainly understand it. There is a certain "upward compatibility" of our vocabularies which our programs should have.

The first step towards designing these upwardly compatible vocabularies is to capture the fact that the choice of the vocabulary incorporates certain assumptions.

1.3.2 The Theory

Similarly, the theory that we encode using out chosen vocabulary also makes assumptions. We might have axioms which state that in a purchase, the buyer usually needs the object bought, the seller needs the money, after the sale the buyer owns the object, etc. Though these rules are very basic, they still make many assumptions. They assume a certain amount of rationality on the part of the buyer and seller. If the buyer was engaging in the event for no reason, or for inflicting loss on himself, almost none of the axioms in the theory would be applicable to him. It is not as though the individual rules in the theory are defaults and this particular buyer is an exception : He is simply beyond the scope of the theory.

Both of these, limitations of the vocabulary and assumptions made by the theory, are problems we face in building a knowledge base. The standard approach is to develop a very rich vocabulary (which can express all the phenomena we can encounter), and to ferret out and make explicit any assumption our theory might make. While this is a good practice, it is not adequate as the only solution to these problems. Let us consider some of its shortcomings.

- It is almost inevitable that there will be some limitations in the vocabulary and that there will be some assumptions behind the theory. When these limitations are eventually discovered, extending the KB to deal with these phenomena could require a reworking of relevant parts of KB. This is undesirable in that we would like a more graceful way of extending the KB to deal with hitherto unexpected phenomena.

- A vocabulary (and theory) which attempts to have a very broad scope is very likely to be cumbersome. E.g., in the case of buying, (Buying X Y Z) is much more compact than Buying(Buying001) $\wedge$ buyer(Buying001 X) $\wedge$ object(Buying001 Y) $\wedge$ seller(Buying001 Z). This is undesirable both from the perspective of writing the axioms and from the perspective of inference.

Even a naive theory of buying and selling, one which could not deal with the international intrige associated with arms sales and which made assumptions about the participants being psychologically normal is certainly very useful. In fact, this theory is all we need for most purposes. At the same time, we humans can understand and cope with the more abnormal situations and the KB should similarly be capable of understanding these.

1.3.3 A solution

The problem is that the naive theory by itself does not contain all the information associated with it. The information associated with a naive theory of commercial transactions includes not just the axioms describing buying and selling, but also information about the assumptions made by the theory, when these assumptions are reasonable, when this theory is applicable, etc. These are 'external' to the axioms constituting the theory itself.

If, in the statement of the theory, we can make it clear that there is something (possibly as yet unstatable) left out, this theory could be included in the KB without ruling out the possibility of later extending the KB. We now consider a syntax for this.

Consider the axiom "after the sale, the buyer owns the object". For convenience let us refer to this as A1. Let us name this theory the 'NaiveMoneyMt'. To say that A1 is an axiom of NaiveMoneyMt, we write,

ist(NaiveMoneyMt A1)

ist stands for 'is true in'. The first argument to ist denotes a *Context.*

What is a Context

Contexts [1] are objects in the domain, i.e., we can make statements "about" contexts. They are rich objects [Philosophical problems in AI] in that a context cannot be completely described. The contextual effects on an expression are often so rich that they cannot be captured completely in the logic. This is what leads us to incorporate contexts as objects in our ontology.

In the formula ist(NaiveMoneyMt A1), the context denoted by the symbol NaiveMoneyMt is supposed to capture everything that is not in A1 that is required to make A1 a meaningful statement representing what it is intended to state. This includes assumptions made by A1 about the domain, assumptions about the applicability of A1, how A1 might relate to statements in other contexts, etc.

The idea is that A1 itself need not be a completely decontextualized or universal statement. It might depend on some contextual aspects that have not yet been specified, and these aspects are to be captured by the context argument. Indeed, it might not be possible to ever completely list all of these context dependencies. At any time, we might have only a partial description of the context and this is why contexts are assumed to be rich objects. In other words, the context object can be thought of as the reification of context dependencies of the sentences associated with the context.

Another way of looking at the context argument is as follows. Imagine a robot which in the course of its duties has to deal with a certain new domain. In order to do this, it examines the domain and writes a set of sentences representing this domain. Let us view this action of the robot as a function that computes the representation. The domain itself is of course an argument to this function. In addition, a number of other factors such as the resource constraints on the robot, the parts of teh domain that escaped the attention of the robot, the duty the robot was performing that lead it to examine the domain, etc. are also influential and hence are arguments to this function. When the robot writes ist(<context> <the theory>), the second argument to *ist* accounts for the domain argument and the context argument accounts for that fact that there were these other factors influencing the representation. In a sense, we are in a position similar to that of the robot (except that we are writing sentences for the computer, not for ourselves) and there are similar factors influencing our representation. The context argument in the sentences we write are meant to capture the effect of these factors.

Using Contexts

Of course, if all we could do with this new syntax was to partition the KB into separate theories, that would not be very useful. Though different theories in the KB might have different context dependencies, there has to be some relation between them. We would like to be able to provide at least partial descriptions of these contexts and reason with them to combine information from different contexts, ensure that a theory is not used inappropriately, extend a theory to cover new phenomena, etc.

To get a feel for the things we would like our systems to do with our new syntax, consider some examples of statements we would like to be able to make.

[1] This concept of contexts was first introduced by McCarthy in [Advice Taker] [Generality In AI].

(a) We want to state that Fred (who happens to be very abnormal) is outside the scope of NaiveMoneyMt. To state this, we need the concept of the *scope* of a theory. In a later chapter, we will provide a more detailed description of what the scope of a context means. For now, let us write

$\neg$presentIn(NaiveMoneyMt Fred)

to state Fred is outside the scope of the NaiveMoneyMt and

presentIn(NaiveMoneyMt Joe)

to state that Joe is within the scope of NaiveMoneyMt.

If we have a theory of some phenomenon, we would like to maximize its scope, i.e., unless the system knows otherwise, it should assume that a given object is within its scope. This policy can be enforced by maximizing the extent of the predicate *presentIn*. There are certain situations in which we would like to minimize the extent of *presentIn* and these will be covered later.

(b) Assume we have a category of theories called Naive theories. We want to state that all of them include an explicit assumption that humans are rational. We write this as,

(forall c NaiveTheory(ci) $\Rightarrow$ ist(ci (forall x Human(x) $\Rightarrow$ Rational(x))))

Here, we are making a statement not just about a particular theory, but about a whole class of theories. Based on some property of the theory (i.e., that it is naive), we are concluding that a certain assertion holds in it.

(c) We might later introduce a more sophisticated theory of transactions, GeneralTransactionTheory (GTT), which makes a distinction between different kinds of costs associated with an object. However, we want to use at least some of the axioms in NaiveMoneyMt in GTT and so want to specify the relation between these two theories. We want to state that if the cost of an object X is A in NaiveMoneyMt, the "list cost" of X is A in GTT. We write this as,

($\forall$ x ist(NaiveMoneyMt cost(x) = A) $\Leftrightarrow$ ist(GTT cost(x List) = A))

There are a few interesting points to note about this example. Firstly, note that the these theories use the symbol cost in different ways. In GTT it is a binary function and in NaiveMoneyMt it is a unary function.

Also note that based on a statement made about an object (the binding for x) in one context, something may be derived about that object in another context. The two contexts use different vocabularies and make different attributions of an object, but these attributions are about the same object.

There might be differences in the set of objects included within the scope of different theories of a domain, but to a significant extent, different theories of a domain are about the same objects.

The formula in GTT is a little more explicit than the one in NaiveMoneyMt. We say that it has been *Partially Decontextualized* or *Relatively Decontextualized*. This process is also called *Lifting* (this concept was first introduced in [jmc ?].)

The formula in NaiveMoneyMt makes some contextual assumptions. The assumptions behind GTT are different from those of NaiveMoneyMt. To compensate for this, the encoding of the proposition 'the list price of an object is A' has to be changed when going from one context to another. This process of relative decontextualization is a substitute for complete decontextualization that is assumed by the traditional model.

(d) We might have more than one theory associated with monetary transactions and these might make mutually inconsistent assumptions. Assume one context (C_1) takes the list price of an object as its cost while another (C_2) includes the sales tax in the cost. So it is possible to derive both ist(C_1 cost(g) = A) and ist(C_2 cost(g) = B). In the previous example, the two contexts involved used different vocabularies. Therefore, different formulas were used to state the same proposition about cost. Here, the two contexts use the same vocabularies, but make different assertions about the cost of the object both refer to as g.

Though the two sentences (cost(g) = A) and (cost(g) = B) are mutually inconsistent, the KB as a whole should not become inconsistent. We should only be required to maintain local consistency within each contexts and not necessarily across contexts.

(e) Given a problem for which the NaiveMoneyMt was adequate, we would like to be able to somehow focus attention on this theory and solve the problem using just this theory. Similarly, we should be able to shift focus, enlarge focus, etc.

If the KB contains formulas such as ist(C_1 P_1), ist(C_1 P_2), ...,ist(C_1 P_n), the system can *enter* C_1 to *be in* C_1. When this is done, the only axioms used by the system are P_1, P_2, ... After problem solving is completed and the system has derived q from P_1, P_2, ..., P_n, it can *exit* C_1 to get ist(C_1 q) where q was the conclusion.

If the system was asked the query Q or given an input P when it was *in* the context C_1, this would be equivalent to exiting C_1 and asking it ist(C_1 Q) or asserting ist(C_1 P) respectively.

Entering (and Exiting) serve two purposes. One is to provide focus in the problem solving behavior. The other is to provide a context for the interaction with the system.

(f) Given a problem, we might not have a context with just the right theory for solving this problem. For example, if the system were asked to determine the least expensive way of getting from Austin to Palo Alto, it might require some axioms from the NaiveMoneyMt and some axioms from the TransportationMt. We should be able to *create* a new context, *lift* the relevant axioms from these two theories into this context, *enter* this context and solve the problem. Note that these two theories might make different assumptions and use different vocabularies. The lifting process needs to perform the requisite relative decontextualization.

Traditionally, research on problem solving has assumed that the representation (theory) is a given and has been set up before the problem solving starts. The goal of the *create, lift, enter, problem solve, exit* sequence is to bring the choice of the theory used to solve a problem within the scope of the problem solving.

1.3.4 Continuing with the problems

The above examples are meant to provide a flavor for some of the things we would like to be able to do with contexts. The examples given above use context as some sort of "theory maintenance and use tool". Let us now consider a very different kind of problem with the model described at the beginning of this chapter and see how the context mechanism might be used to tackle it.

1.3.5 Communication

Part of what it means to be intelligent is to be able to communicate in natural language. The model described above posits a "front end" which takes natural language utterances and outputs sentences for inclusion in the knowledge base. These sentences of course must satisfy the requirement imposed on the other sentences in the KB (by the traditional model), i.e., they must be completely decontextualized.

The operation of the front end (the NL program) is guided mostly by linguistic information. In cases where purely linguistic information is inadequate to disambiguate the meaning of an utterance, the NL program might call upon the knowledge base. This use of the knowledge base (in disambiguation) usually takes the form of the NL program generating the possible set of interpretations of an utterance and asking the knowledge base to either eliminate some or rank them in order of relative likelihood.

The NL program might keep some state to help it resolve anaphora, ellipsis and other reference problems. This state usually consists of utterances previously made and the objects these utterances referred to. It is normally expected that this state is adequate to resolve the context dependency of natural language utterances. At the end of this process, the NL program is expected to have *the* completely decontextualized translation which is then added to the KB.

There is an (implicit) assumption made that the context dependency of natural language utterances is completely accountable by linguistic knowledge. It is also assumed that the utterance can be completely decontextualized and that the only context dependencies of an utterance are on the utterances preceding it.

However, as the following examples show, it seems that there are certain context dependencies that don't obey this assumption. NL utterances often have many non-linguistic assumptions and it is difficult for the front end to uncover these.

Rather than expecting the NL front end to completely decontextualize utterances, we might demand less of it. When the NL front end produces a sentence which is a translation of an utterance, that sentence is added to a context corresponding to that discourse. Loosely speaking, this context is supposed to account for the context dependencies remaining in the translation. Let us consider a few examples of this use of contexts.

(a) Consider the literature which accompanies a model kit. Some of the material on the cover of the kit assumes the person is considering buying the product. Some of the instructions inside the box assume the person has bought the model and would like to assemble it. They also assume the person would like the assembly to be capable of all the kit was advertised to be.

These assumptions are certainly non-linguistic and very hard for the NL front end (which relies largely on linguistic knowledge) to detect. In the least, the translation can and should should acknowledge the fact that assumptions have been made.

Let us call the translation of the utterance the NL program can produce the *literal translation* of the utterance. If the literal translation of a sentence in the instructions is F, the formula ist(C_1 F) is added to the KB. Here C_1 is the context in which the user of the kit interprets these instructions.

It is important to note that we cannot simply add F to the KB. There might be other instructions accompanying the model which specify what should be done in case of certain problems (such as a child swallowing the adhesive provided with the model). Call the literal translation of these other instructions F1. These instructions make assumptions which are radically different from those made by the assembly instructions. For example, in the context of an emergency such as a child swallowing the adhesive, completing the model successfully becomes irrelevant. If we add F1 and F2 to the KB, the system might arrive at unintended conclusions.

In addition to sentences such as ist(C_1 F1), the system should also be given information about C_1, e.g., that the sentences (such as F1) in it are from instructions for assembling a model kit, the kind of kit involved, the reader's intent, etc. Based on this information about C_1, the context dependencies of F1 can be made explicit asynchronously, later on demand. It is also possible that later utterances can provide information that enable us to deduce these context dependencies. By bringing the context dependency within the realm of the KB, all the knowledge and tools available in the KB can be used to reveal and exploit these dependencies.

(b) Consider a section of a travel advisory telling people what to expect, how to dress, etc. in the northeast of the United States during winter. Somewhere at the beginning, there might be the phrase 'during winter'. This sets up the context for the remainder of the advisory. Each following statement is assumed to inherit the temporal qualification 'during winter'. This 'assumption' made by the body of the text, might be treated in a manner similar to the treatment given to the unstated assumptions.

There might be other similar but unstated temporal qualifications. For example, there might be a statement such as, "the snow ploughs run between 3 and 4 in the morning (in Boston)". This statement (like much of the rest of the section) assumes the implicit temporal qualification of holding true at around the time period the advisory was published.

A statement such as "It is snowy is Boston" (call it's literal translation L), could be added to the KB as ist(C_2 L) where C_2 is the context of the advisory. The effect of the preamble "during winter" is to temporally qualify all the statements occurring in C_2.

(c) The first example dealt with an assumption which was not mentioned anywhere in the text. The second dealt with an assumption that was mentioned early in a section and had a scope that extended over the entire section. In the first example, the scope of the context (into which translations were added) was a significant part of the instruction manual. In the second, it was the relevant section of the travel advisory. As the scope of context is made finer, the class of contextual phenomena which can be retained in the translation increases. Let us next consider a slightly finer context - at the level of a single sentence.

Consider a single sentence such as 'the lady owns the bag'. A very straightforward translation of this could be, owns((The Lady) (The Bag)). Of course, the scope of the context must be kept sufficiently small so that we don't have another occurrence of '(The Lady)' in the same context referring to some other lady.

At first sight, this seems to be just a gimmick with the syntax, i.e., there doesn't appear to be any obvious advantage in retaining definite references such as 'the lady' in the translation. However, allowing the translation to retain such references opens the door for the following interesting and useful strategies.

(i) The problem of resolving the definite referents can be seen as an inference problem. The problem of resolving the referent of (The Lady) is that of deducing the x such that (The Lady) = x is true. We might be able to use later utterances, unstated assumptions, domain knowledge, etc. to determine x. This sort of reformulation allows us to view natural language understanding as a constraint resolution problem.

The aim of this reformulation is not to use deduction as the computational framework, i.e., the aim is not to try and move from lisp to prolog. The aim is to be able to integrate information from very diverse sources to determine the denotation of (The Lady). The distinction between these two aims is made most clear when the language used by the KB is undecidable. In such a case, our aims are still satisfied - we can talk about the referent that the KB *entails* for (The Lady), though a general deduction machine might not be able to determine this.

(ii) Consider a statement such as 'He is an engineer' translated as Engineer(He). Now consider the question 'has he gone to school?'. To answer this question, we do not need to resolve the referent of 'He'. There are many inferences that can be carried out without completely decontextualizing the given utterance. Since the utterance is usually far more compact than its decontextualized translation, inference using the contextual translation is usually much simpler.

In this thesis we will discuss only the second of these uses. The first use is the subject of current work which will be discussed briefly in chapter four.

We carry the idea of allowing translations of utterances that retain context dependencies one step further. In principle, once we have the mechanism for accepting contextual sentences, general theories such as the NaiveMoneyMt are also able to exploit these techniques for making the theory more compact. This simplifies both the task of writing the theory and performing inference with these theories.

(d) In a discourse, it is assumed that the sentences uttered depend on the context. Neither this, nor the actual context is made explicit in the utterances. Associated with an utterance is a *current context* and the utterance is meant to be interpreted in this context.

This concept can be extended to cover interactions with computer systems. As has been stated earlier, even non-natural language sentences might exploit contexts. Any interaction (asking questions, making assertions, receiving answers, answering the systems questions) has a context associated with it.

Earlier, when discussing inference, we introduced the concept of the system *being in a context.* The context a system is *in* is also referred to as the *current* context. All interactions with the system take place in this context. We also mentioned the concept of entering a context. When the system enters a context, it is *in* that context. Conversely, the system might *exit* that context and then it will be *in* the outer context. This outer context in turn might have another context outer to it, and so on.

1.4 One Problem or Many?

We have described a diverse range of problems with the traditional model of AI. We also introduced the concept of contexts as a solution to these problems. Given the fairly broad range of phenomenon covered by these problems, the natural question which arises is whether these are very different problems requiring very different solutions. Implicit in the brief sketch of the solution was the hypothesis that a single solution is adequate for all these problems. At this stage, it is important to clarify this hypothesis.

Though these are certainly very different kinds of problems, they share a common origin, i.e., the intended universal and decontextualized nature of expressions used in representation. This suggests that at least at a logical level, a single machinery, one that allows for this requirement on representations to be relaxed, might be shared by the solutions to these different problems.

The logical machinery is just part of the solution. The other part of the solution is in the form of axioms describing and relating contexts. The kind of context related axioms required could vary significantly from one problem to another.

The goal of this thesis is to present a[2] logical machinery required for contexts and provide techniques for using this machinery to tackle the above problems.

1.5 Outline of thesis

From the above discussion, it seems that some of the problems with the traditional model of AI might be effectively addressed by the introduction of contexts. There are of course many issues to be dealt with before contexts can be said to actually solve these problems. Broadly, there are two categories of issues we need to address.

The sentences given above, with their use of 'ist', clearly extend known knowledge representation formalisms. We need an extension of traditional logical formalisms to deal with contexts. Chapter 2 provides a model theory and proof theory for contexts. The reader not interested in logical formalisms may skip this section.

After the discussion of the model theory and proof theory, there are some sample sentences and proofs involving these sentences in chapter 2. Following these is a discussion and

[2] not *the*

presentation of a general framework for lifting. Even the reader who has decided to skip the first part of this chapter might want to go through this section.

Merely providing a syntax and semantics for a new logic does not solve any problem. We need to have a better understanding of what contexts can be used for and how they are to be used. This problem, which is the main focus of this thesis, will be dealt with in detail in chapter three. The discussion of the uses of contexts will be in the form of a number of examples from the use of contexts in the Cyc system. This chapter contains much of the information in this thesis which a person interested in using contexts would find useful.

Along with this chapter, the reader who is more interested in experimental validation issues might want to look at appendix A, which contains a script of a Cyc based application that uses contexts.

Chapter four discusses some work underway on the use of contexts in natural language understanding. Finally, we compare contexts with related work in AI.

Chapter 2

The Logic Of Contexts and Lifting

In this chapter we provide the formalization of the logic of contexts. In the first half of this chapter present the model theory and a proof theory of the logic. To begin with, using a set of examples, we draw up a desiderata of the properties the logic should have. We then present a syntax and model theory that satisfies this desiderata. After that, a proof theory for this logic is described. The reader not interested in logic may skip the sections on the model theory and proof theory (sections 2.1.2 and 2.1.3.)

The second part of this chapter deals with the problem of lifting. We first formalize the concept of default coreference and discuss its implications for lifting. Then an analysis of lifting from a model theory perspective is carried out.

2.1 The Logic Of Contexts

2.1.1 The Desired Logical Properties of Contexts

To help us get started with the list of desired properties, let us assume that we have objects called contexts. Formulas of first order logic will be formulas of our logic too. We use the symbol "ist" to explicitly relate a formula to a context in which it holds. The following is a list of the desired logical properties of contexts and the symbol ist.

(1) Contexts are first class objects and ist formulas are first class formulas. I.e., ist(c F), ($\forall$ ci ist(ci F)), $\neg$ist(c F), ist(C_1 F) $\wedge$ ist(C_2 F), ist(C_1 ist(C_2 F)) are all formulas.

(2) Different contexts could use different languages, i.e., the language that may be used by a formula F in a context C could depend on C. Whether an expression is a meaningful formula could depend on the context in which it occurs. i.e. F might be meaningful in a context C_1 but not in a context C_2. The need for this is illustrated by the following examples.

   (a) Consider the statement 'the king of France is bald' [translated as bald(King-of-France)]. This sentence is meaningless in the context of this discussion, but is meaningful in the context of a play on the French Revolution.

(b) The statement 'Lincoln is 16 years old' [age(Lincoln) = 16] is meaningless in the context of this discussion, but is meaningful in the context of the beginning of a play on Lincoln.

(c) Sixteenth century mechanics employed a single term "vis viva" for the quantity of motion of an object. This concept was later refined into momentum and energy. The expression "vis viva" is meaningless in modern mechanics but is was meaningfully used in sixteenth century mechanics.

The important point to note is that given ist(C F), whether F is meaningful depends not the syntactic properties of C but on what it denotes. Traditionally, in predicate logic, the class of meaningful expressions is defined before any mention of denotation (or any other semantic concept) is made. I.e., Grammatically is equated with meaningfulness.

However, if the meaningfulness of a formula is to be dependent on the context in which it occurs, syntax and semantics are intertwined. Before we can determine whether F is meaningful in ist(c F), we need to know the denotation of c i.e., in ist(COTD bald(King-of-France)), bald(King-of-France) [where COTD denotes the context of this discussion] is meaningless not because of some syntactic property of the symbol COTD, but because of what it denotes.

So, it seems that if we are going to allow a formula to be meaningful in one context but not in another, we need to distinguish between meaningfulness and grammaticality. i.e. a formula can be grammatical without being meaningful (all meaningful formulas are grammatical.)

Each context defines its own language (in the sense of first order logic) We will use a "mutual recursion" between syntax and semantics to capture he notion of a meaningful sentence.

(3) First order logic (FOL) (with additions for nonmonotonicity) is enough for solving many problems and we clearly shouldn't have to reinvent solutions for these. If one is dealing with a set of first order formulas not mentioning ist, all the devices (semantics + proof theory) of FOL should be applicable.

We should be able to 'embed' first order theories in framework of contexts. What does it mean to 'embed' a first order theory in a framework of contexts? Semantically, if $\gamma \models_{FOL} \alpha$, then $\gamma \models_{LOC} \alpha$ should be true irrespective of which context γ is in ($\models\models_{FOL}$ is first order entailment and $\models_{LOC}$ is entailment in the logic we are developing). Proof theoretically, deriving α from γ (using FOL proof theory) in a context C should be equivalent to deriving ist(C α) from ist(C γ). The reason for imposing this requirement is to allow us to use existing FOL based representations and problem solvers with contexts.

This requirement of being able to embed first order theories in a framework of contexts leads to the concept of the theory associated with a context, i.e., the deductive closure (under the rules of FOL) of the set of formulas α such that ist(c α).

(4) A symbol might denote different things in different contexts. i.e., *ist* should be referentially opaque. Though there are many motivations for this (related to the need for

modalities), let us consider an example of a certain behavior we would our systems to be able to demonstrate.

Consider a statement such as 'he is an engineer'. We can answer a number of questions such as 'is he a baby?' without knowing the referent of 'he'. For our system to do this, the sentence 'he is an engineer' needs to be represented. For this, the word 'He' needs to be admitted as a term. Our programs should be able to do this without insisting that 'He' always be used to denote the same person throughout the database.

A symbol might be used as a predicate in one context, individual constant in another, and function in yet another.

(5) Nested contexts/ Nesting of ist

Consider 2 contexts
C_1: 16th century mechanics
C_2: 19th century mechanics

Letvv be an abbreviation for vis viva, - used by C_1
mom be an abbreviation for momentum - used by C_2
en be an abbreviation for kinetic energy - used by C_2

C_1 only has the crude notion of quantity of motion referred to as vv, while C_2 distinguishes between momentum and kinetic energy. If two object m1 and m2 have the same mass, and m1 has greater vv than m2 (in C_1), then we can conclude that m1 has both greater mom and en than m2 (in C_2). This relation is expressed by the following axiom.

$$\begin{aligned}&\mathrm{ist}(\mathrm{C}_1\ (\mathrm{mass}(\mathrm{m1}) = \mathrm{mass}(\mathrm{m2}) \wedge (\mathrm{vv}(\mathrm{x}) > \mathrm{vv}(\mathrm{y}))))\\ &\qquad \Rightarrow \mathrm{ist}(\mathrm{C}_2\ ((\mathrm{mom}(\mathrm{x}) > \mathrm{mom}(\mathrm{y})) \wedge (\mathrm{en}(\mathrm{x}) > \mathrm{en}(\mathrm{y}))))\end{aligned}$$

This axiom, let's call it F, is meta to both these theories, i.e., is not in either theory, but is a statement relating the two theories. When writing F, we left the context it is true (c0) in implicit and we could potentially want to make this explicit. So we could have the meta-meta-level statement

$$\begin{aligned}&\mathrm{ist}(\mathrm{c0}\ \mathrm{ist}(\mathrm{C}_1\ (\mathrm{mass}(\mathrm{m1}) = \mathrm{mass}(\mathrm{m2}) \wedge (\mathrm{vv}(\mathrm{x}) > \mathrm{vv}(\mathrm{y}))))\\ &\qquad \Rightarrow \mathrm{ist}(\mathrm{C}_2\ ((\mathrm{mom}(\mathrm{x}) > \mathrm{mom}(\mathrm{y})) \wedge (\mathrm{en}(\mathrm{x}) > \mathrm{en}(\mathrm{y}))))\end{aligned}$$

This itself is (implicitly) in a context outer to c0 we could potentially want to make this explicit. So we should be able to nest ist formulas. i.e.,

$$\mathrm{ist}(\mathrm{c}_i\ \mathrm{ist}(\mathrm{c}_j\ \alpha)),$$
$$\mathrm{ist}(\mathrm{c}_k\ \mathrm{ist}(\mathrm{c}_l \ldots \mathrm{ist}(\mathrm{c}_j\ \alpha) \ldots))$$

are all formulas.

(6) Being in a context

Since the formulas of a formal system have to be finite, the context associated with a formula has to be left implicit at some level in this meta-chain. This implicit context is the 'outermost context' and the system itself is 'in' this context.

Now consider a system whose outermost context is c0 with formulas such as ist(C_1 F1), ist(C_2 F2), ...We should be able to 'focus' the system on an inner context C_1. Being focused in this manner means acting as if the only axioms in the system are α, where ist(C_1 α) is true is in the outer context. The focusing is done by 'entering C_1'. The system would then be 'in' the context denoted by C_1. The inverse action is 'exiting'.

What does it mean for the program to be 'in' context C_1?

(a) In terms of interactions - being in the context that a program refers to as C_1 and stating α is equivalent to being in the outer context and saying ist(C_1 α)

(b) In terms of problem solving - the only axioms available to the problem solver are those axioms α such that ist(c α) is true in the outer context.

(c) Another motivation is to use the context a system is 'in' as the essential indexical [Perry]. [1]

Now we describe the syntax and the semantics of our logic.

2.1.2 The Syntax and Semantics of Contexts

(1) We have a set of object called contexts

(2) Grammaticality definition

(a) We have the 'Total Vocabulary' - a set of constant symbols

(b) The logical parameters - $\forall$, $\neg$, OR, ist, = and a set of variable symbols (such as x, y, z, ...)

(c) Grammatically well formed formulas

(i) Every symbol is a term

(ii) (<constant-symbol> <term> <term> ...) is a term

(iii) (<constant-symbol> <term> <term> ...) is a formula

(iv) $\neg$ <formula> is a formula

(v) (<formula> $\wedge$ <formula>) is a formula

(vi) ($\forall$ <variable-symbol> <formula>) is a formula

(vii) (<formula> $\vee$ <formula>) is a formula

(viii) (= <term> <term>) is a formula

[1] Briefly, the conjecture made by Perry is as follows. Though natural languages support many different indexicals (Now, I, Here ...), one can attempt to replace these with definite descriptions, thereby reducing the number of different indexicals we have to deal with. However, Perry makes the point that it is not possible to altogether do away with indexicals. We need at least one, and this he calls the essential indexical.

(ix) (¡formula¿ ⇒ ¡formula¿) is a formula

(x) ist(<term> <formula> is a formula

(3) Language

A language is a subset of the total vocabulary which does the following

- each constant symbol (in this subset) is designated as one of
  * predicate constant
  * function constant or
  * individual constant
- each predicate and function symbol (in this subset) is assigned a fixed arity

(4) An interpretation/model U of a language

- has a domain D of objects
- has an assingment function I that assigns to each
  * individual constant of the language an element of D
  * each function symbol (of the language) of arity n a set of n-tuples on D
  * each predicate symbol (of the language) of arity n a set of n-tuples on D

[Note : This definition of a model is the same as that of FOL]

Given a language, we have the class of models of the language.

(6) Context Structure

A context structure CS is a function that assigns to each context C

- A Language L(C)
- A set of structures CM(C)

Intuitively, a context structure defines a context's way of describing the world. L defines the language/ vocabulary, and CM defines the theory of the world according to the context.

(7) A formula F is meaningful in a context C under a context structure CS iff every symbol not within an ist is part of the language L(C) and is used appropriately, i.e.

- individual symbols are used as terms, predicate symbols used as predicates and function symbols used as functions
- every function and predicate symbol is used with the arity assigned to it by L(C).

Example. If we have the context '16th century physics context' and a CS function such that L('16th century physics context') does not include the symbol momentum as a function symbol, in this context, under this context structure, the formula (= momentum(A) B) is not meaningful.

However, there could be a different CS function under which this formula would be meaningful in this context.

(8) A substitution function S is a mapping from symbols to objects.

Given a substitution function S and a model U, we get an assignment S_U of symbols and terms to things such that

- if x is a variable symbol, $S_U(x) = S(x)$ where x element of D
- if x is a constant symbol, predicate symbol or function symbol, $S_U(x) = I(x)$
- if x is of the form (function term1 term2 ...), $S_U(x)$ is the value of the function S_U(function) at [S_U(term1), S_U(term2) ...].

where I is the assignment function of U (the function that assigns to each symbol an individual or a set of tuples.)

(8) **Satisfaction**

Given a formula F in a context C, the tuple <U, CS, S> is said to satisfy F in C iff

- U is an element of CM(C)
- F is meaningful in C under CS
- The truth recursion rules of first order logic
  * If F is of the form (predicate a1 a2 ...), then F is satisfied if the tuple [S_U(a1), S_U(a2) ...] is an element of S_U(predicate).
  * If F is of the form $\neg$G, then F is satisfied iff G is not satisfied in C by <U, CS, S>.
  * If F is of the form P $\vee$ Q, then F is satisfied iff either P is satisfied or Q is satisfied in C by <U, CS, S>.
  * If F is of the form P $\wedge$ Q, then F is satisfied iff either P is satisfied and Q is satisfied in C by <U, CS, S>.
  * If F is of the form P $\Rightarrow$ Q, then F is satisfied iff either P is not satisfied or Q is satisfied in C by <U, CS, S>.
  * If F is of the form (= a1 a2), then F is satisfied iff S_U(a1) is equal to S_U(a2)
  * If F is of the form ($\forall$ x P), then F is satisfied iff for every element o of D, P is satisfied in C by the tuple <U, CS, S_o> where S_o is identical to S except at x where $S_o(x)=o$
- Satisfaction rules for ist formulas

  If F is of the form ist(ck P), then F is satisfied in C by <U, CS, S> iff

  Let the object denoted by ck, i.e. S_U(ck) be Ck.
  * Ck is a context, and
  * For every U_j in CM(Ck),
    · to every variable occurring free in P, S assigns an element of Dj, and
    · P is satisfied in Ck by <U_j, CS, S>

10. Entailment

    The formula F in C entails the formula G in C iff every <U, CS, S> tuple satisfying F in C also satisfies G in C.

11. Defined predicates/functions

    – presentIn(c A) $\Leftrightarrow$ ($\forall$ z (= A z) $\Rightarrow$ ist(c ($\exists$ y z = y)))
      E.g., presentIn(ConanDoyleContext SherlockHolmes)

      It should be noted that the above axiom cannot be rewritten to
      presentIn(c A) $\Leftrightarrow$ ist(c ($\exists$ y A = y))).
      The right hand side of this axiom (if meaningful) is a tautology and presentIn would not be very useful. For one, the object denoted by A in the outer context could be called something else (if it has a name at all) in c. Also, the term A might not be in the vocabulary of c. This rewriting can be done only in the case where the outer and inner contexts have the same domain (i.e., all contexts have the same domain) and all ground terms denote the same objects in all contexts.

    – corefer(C_1 A C_2 B) $\Leftrightarrow$ ($\forall$ z ist(C_1 (= A z)) $\Leftrightarrow$ ist(C_2 (= B z)))

      E.g., C_1 = Context of Lincoln's speech at Gettysberg
      C_2 = Context of this talk

      We would then have,
      corefer(C_1 Now C_2 TheYear1865).

    – value(c A) = z $\Leftrightarrow$ ($\forall$ x (= z A) $\Rightarrow$ ist(c (= B x)))
      The example axiom for corefer could be rewritten as,
      value(C_1 Now) = value(C_2 TheYear1865)

12. The function CM defines a set of (directed) graphs of contexts. We could restrict our attention to CM functions where these graphs are directed acyclic graphs. Axiomatically, this is equivalent to having

    ($\forall$ ci $\neg$presentIn(ci ci))
    hold in all contexts. This is to avoid the possibility of a context being able to refer to itself.

13. Nonmonotonicity

    The above logic is monotonic and we will need nonmonotonicity. Ideally, the issues related to nonmonotonic logic and hence the nonmonotonic mechanism used should be orthogonal to contexts. We will use circumscription (pointwise, prioritized). All predicates starting with 'ab' will be assumed to be minimized (the circumscription policy will be specified for each separately.) (Note : The implementation in Cyc uses Argumentation Based Default Reasoning.)

15. Discussion

    (a) Capturing the notion of meaningfulness :

* Meaningless statements are neither true nor false. If F is meaningless in all consistent interpretations of c, we have $\neg$ist(c F) $\wedge$ $\neg$ist(c $\neg$F).
  So, though age(Lincoln)=16 might be meaningless in the ContextOfThisDiscussion, ist(ContextOfThisDiscussion age(Lincoln)=16) is not meaningless. It is just false.
* Propagation of meaninglessness : Even if F is meaningless (in all interpretations of c), ist(c F) is meaningful. i.e., meaninglessness does not propagate much.
  If all the parts of a formula are used appropriately, it cannot be meaningless, i.e. contexts are Rectangular (in the sense of [Barwise].)
  The treatment of meaninglessness is very limited. So for example, the formula bald(King-of(France)) could be meaningful just because King-of, bald and France, are all part of the vocabulary of the context.

(b) The 'domains' of different contexts could be different. i.e., The Barkan formula

ist(c ($\forall$ x p(x))) $\Leftrightarrow$ ($\forall$ x ist(c p(x)))

does not hold.
For the rest of this thesis , I will assume that the domain of of the outer context is larger than the domain of the inner contexts.
This gives us the inference rules k and l.

(c) Some Non-Axioms

* ist(c p $\vee$ q) $\Rightarrow$ ist(c p) $\vee$ ist(c q)
* $\neg$ist(c p) $\Rightarrow$ ist(c $\neg$p)

However, if the theory associated with c was known to be complete, i.e., for every formula p, either p or its negation was in the theory of c, then the above two formulas become axioms. Also, reversing the direction of implication makes all of these true.

(d) Nesting of contexts
For the rest of this note, I will assume only one level of nesting. The implementation has only one level of nesting.
Note the distinction between one context being meta or outer to another versus one context being more general than another.
Going back to a previous example, consider the 2 contexts
C_1: 16th century mechanics
C_2: 19th century mechanics

Let vv be vis viva, - used by C_1
mom be momentum used by C_2
en be kinetic energy used by C_2

C_1 only has the crude notion of quantity of motion, while C_2 distinguishes between momentum and kinetic energy. The following axiom (mentioned earlier) gives a partial relation between these two contexts.

$$\begin{aligned}&\mathrm{ist}(C_1\ (\mathrm{mass}(m1) = \mathrm{mass}(m2) \wedge (\mathrm{vv}(x) > \mathrm{vv}(y))))\\ &\qquad\qquad\qquad \Rightarrow \mathrm{ist}(C_2\ ((\mathrm{mom}(x) > \mathrm{mom}(y)) \wedge (\mathrm{en}(x) > \mathrm{en}(y))))\end{aligned}$$

This axiom is in c0. c0 is 'outer' or 'meta to' both C_1 and C_2, but C_2 is more 'general' than C_1. The generality relation between contexts is not defined by the logic, but the meta relation is.

2.1.3 Proof theory

The concept of a proof in first order logic is modified to include the concepts of Enter and Exit.

A proof of a formula P is a finite sequence of statements (ending in P) that satisfy the following constraints.

1. Each line in the proof has a context sequence associated with it. A context sequence is a list of contexts. E.g., [C_1 C_2, ..., C_n]. The sequence may be the null sequence, i.e., [] and may be omitted in this case.

2. Each line in the proof is

   (i) either a formula or

   (ii) *Enter ci* (where ci is a context) or

   (iii) *Exit* (only if the context sequence associated with that statement is non-null).

3. The context sequence associated with a line is the same as that of the preceding line if there is one, except,

   (i) If there is no preceding line, it is [].

   (ii) If the preceding line is *Enter ci* and the context sequence associated with it is [cj ck ...], the context sequence associated with the line is [ci cj ck ...].

   (iii) If the preceding line is *Exit* and the context sequence associated with it is [cj ck cl ...], the context sequence associated with the line is [ck cl ...].

4. Each formula F in the proof must satisfy at least one of the following constraints. Let the context sequence associated with the line is [C_1 C_2 c3 ... cn].

   (i) The formula ist(cn ist(cn-1 ...ist(C_2 ist(C_1 F)))) must be an axiom.

   (ii) The formula ist(C_1 ist(C_2 ...ist(ck F))) is a preceding statement in the proof and the context sequence associated with that line is [ck+1 ... cn].

(iii) The context sequence associated with F is [] and it is obtained by applying a rule of inference to one or more axioms or preceding statements in the proff whose context sequence is [].

(iv) The formula F is obtained by applying a rule of inference to one or more preceding statements in the proof whose context sequence is also [C_1 C_2 c3 ...cn].

If we constrain the proof to not contain Enter or Exit, the above definition reduces to the definition of a proof in first order logic.

The inference rules are as follows. Though all these rules are sound I don't know whether the logic is complete The following is not a minimal set of inference rules.

(a) If a set of formulas does not involve ist, then all the rules of first order proof theory may be used (i.e. we can 'embed' a first order system in contexts.)

(b) Modus ponens, modus tolens and the other rules of propositional logic.

(c) ist(Ci α) LC ist(Ci γ)

———————

ist(Ci α LC γ)

where LC may be one of $\wedge$, $\vee$ or $\Rightarrow$.

(d) ist(Ci $\alpha \Rightarrow \gamma$)
ist(Ci α)

—————

ist(Ci γ)

Modus Ponens from outside. This is a short form for entering Ci, applying modus ponens and exiting.

(e) ist(Ci $\alpha \wedge \gamma$)

———————

ist(Ci α) $\wedge$ ist(Ci γ)

(f) ist(ci α)

—————-

$\neg$ist(ci $\neg\alpha$))

(g) Skolemization.

($\exists$ x α(x))
A is a new symbol

—————

α(A)

(h) Universal instantiation (UI). Since ist is referentially opaque, the first order version of universal instantiation does not hold.

The appropriately modified version of UI is –

$$\frac{\begin{array}{l}(\forall\ x\ \alpha(x)\ LC\ \alpha 1(x)\ LC\ \alpha 2(x)\ \ldots)\\ A = value(C_1\ A1) = value(C_2\ A2)\ \ldots\end{array}}{\alpha(A)\ LC\ \alpha 1(A1)\ LC\ \alpha 2(A2)\ \ldots}$$

where $\alpha(x)$ does not mention ist and $\alpha 1$ only mentions the context C_1, $\alpha 2$ only mentions C_2 ...

Any of the LCs can be replaced by any logical connective.

(i) $$\frac{ist(c\ (\forall\ x\ \gamma(x)))}{(\forall\ x\ presentIn(c\ x) \Rightarrow ist(c\ \gamma(x)))}$$

(j) $$\frac{ist(c\ p \Rightarrow q)}{ist(c\ p) \Rightarrow ist(c\ q)}$$

(k) If the domain of the outer context is larger than the domain of the inner contexts (an assumption we will make), we have

$$\frac{(\forall\ x\ presentIn(c\ x) \Rightarrow ist(c\ \gamma(x)))}{ist(c\ (\forall\ x\ \gamma(x)))}$$

(l) The following rule also holds only with the above assumption

$$\frac{(\forall\ x\ presentIn(c\ x) \Rightarrow ist(c\ \gamma(x)) \Rightarrow ist(c\ Phi(x)))}{ist(c\ \gamma(x) \Rightarrow Phi(x))}$$

2.2 Example Axioms

To give us a feel for the logic, we now examine some sample axioms involving multiple contexts and show some proofs using these axioms. This will then be used to motivate the discussion on lifting.

The following are some sample uses of lifting axioms. These are not intended to be a discussion of issues involved in the use of contexts in general, but are meant only as examples of axioms in this logic. These lifting axioms deal with some of the standard syntactically characterizable differences between contexts. C_1 and C_2 are two contexts.

(1) Dropping an argument to a predicate.

In C_1 : the predicate Tall is unary, so that we say Tall(Joe)
In C_2 : the predicate Tall is binary and we make the sample population explicit.

$(\forall\ x\ \mathrm{ist}(C_1\ \mathrm{Tall}(x)) \Leftrightarrow \mathrm{ist}(C_2\ \mathrm{Tall}(x\ \mathrm{Person})))$

2. Dropping an argument to a function.

In C_1 : the function cost (denoting the absolute dollar value of an object) is unary.
In C_2 : the function cost takes 2 arguments - one for the object and other for the person who associates this value with that object.

The context C_1 may be used only if everyone associates the same value for that object.

$(\forall\ x, y\ \neg(\text{ab-cost}\ C_1\ C_2\ x) \Rightarrow \mathrm{ist}(C_1\ \mathrm{cost}(x){=}y) \Leftrightarrow \mathrm{ist}(C_2\ (\forall\ z\ \mathrm{cost}(x\ z){=}y))$

3. Making an assumption about the domain of a context.

C_1 : Contains information about fixtures in a house. Assumes that all fixtures all installed in houses. (So we could say that all light sockets are connected to electrical wires)
C_2 : Does not make this assumption.

$(\forall\ x\ \mathrm{ist}(C_2\ \mathrm{Fixture}(x) \wedge \neg\mathrm{installed}(x)) \Leftrightarrow \neg\mathrm{presentIn}(x\ C_1))$

Now if we have

$\mathrm{ist}(C_1\ (\forall\ x\ \mathrm{LightSocket}(x) \Rightarrow (\exists\ y\ \mathrm{connectedTo}(x\ y) \wedge \mathrm{Wire}(y))))$

We then have,

$$\begin{array}{l}\mathrm{ist}(C_2\ (\forall\ x\ (\mathrm{Fixture}(x) \Rightarrow \mathrm{installed}(x)) \Rightarrow \\ \qquad (\mathrm{LightSocket}(x) \Rightarrow (\exists\ y\ \mathrm{connectedTo}(x\ y) \wedge \mathrm{Wire}(y)))))\end{array}$$

The derivation of this is given in a following proof.

4. Fixing time. The term fix-time(ci ti) denotes a context that is a snapshot of the context ci at the time ti.

   $\neg$(ab-time ci ti) $\Rightarrow$ ist(fix-time(ci ti) p) $\Leftrightarrow$ ist(ci holds(p ti))

   [not an axiom but a schema]

5. Indexicals.

   C_1 contains the formula holds(P Now). We want to lift the formula to another context C_2 which might not associate the same temporal extent with Now as C_1 does. If the time instant T1 is associated (in C_2) with Now in C_1, we write

   corefer(C_1 Now C_2 T1)

   If we also have

   ($\forall$ x ist(C_1 holds(p x)) $\Rightarrow$ ist(C_2 holds(p x)))

   This together with the corefer, implies

   ist(C_2 holds(p T1))

2.3 Sample Proofs

Now let us consider a couple of simple proofs involving one of the above lifting axioms and the inference rules given earlier. Note that the null context sequence [] is omitted in the following proofs.

a. We use the lifting axiom given in the first of the above examples.

   C_1 : the predicate Tall is unary, so that we say Tall(Joe)
   C_2 : the predicate Tall is binary and we make the sample population explicit.

   (A1) ($\forall$ x ist(C_1 Tall(x)) $\Rightarrow$ ist(C_2 Tall(x Person)))

   We are also given

   (A2) ist(C_2 ($\forall$ x Tall(x Person) $\Rightarrow$ (> height(x) (Feet 6))))

   (A3) ist(C_1 Tall(Fred))

   (A4) ist(C_2 ($\forall$ q l p (> q l) $\wedge$ (> l p) $\Rightarrow$ (> q p)))

   (A4a) ist(C_2 (> n m) $\Rightarrow$ (> (Feet n) (Feet m)))

   The Query is,

   (Q1) ist(C_2 (> height(Fred) (Feet 5)))

   The proof seems quite obvious to start with. Apply the universal instantiation (UI) rule with (A1) substituting x with Fred. However, this does not work out. The information given is actually inadequate and it does not follow that Fred is taller than 5 ft in C_2.

This is because the symbol 'Fred' could potentially refer to different people in C_1 and C_2. So we need the additional axiom,

(A5) value(C_1 Fred) = value(C_2 Fred)

Apply UI with A1 and A4 to get

(D1) ist(C_1 Tall(Fred)) $\Rightarrow$ ist(C_2 Tall(Fred Person))

Use Modus Ponens with (D1) and (A3) to get

(D2) ist(C_2 Tall(Fred Person))

Enter C_2 ;; enter the context C_2

(A2) ($\forall$ x Tall(x Person) $\Rightarrow$ (> height(x) (Feet 6))) [C_2]

Use UI to get

(D3) Tall(Fred Person) $\Rightarrow$ (> height(Fred) (Feet 6)) [C_2]

Use Modus ponens on (D3) and (A2) to get

(> height(Fred) (Feet 6)) [C_2]

(A4) ($\forall$ q l p (> q l) $\wedge$ (> l p) $\Rightarrow$ (> q p)) [C_2]

Use UI to get

(D4) (> height(Fred) (Feet 6)) $\wedge$ (> (Feet 6) (Feet 5))
$\Rightarrow$ (> height(Fred) (Feet 5)) [C_2]

Applying modus ponens a couple of times, get

(D5) (> height(Fred) (Feet 5)) [C_2]

Exit ;; implicitly C_2.

From D5, and nothing that we have exited C_2,

(D6) ist(C_2 (> height(Fred) (Feet 5)))

Which is what we set about to prove.

b. We demonstrate another proof involving making an assumption about the domain of a context.

C_1 : Contains information about fixtures in a house. Assumes that all fixtures all installed in houses. (So we could say that all light sockets are connected to electrical wires)
C_2 : Does not make this assumption.

(A1) ($\forall$ x ist(C_2 (Fixture(x) $\wedge$ (not installed(x)))) $\Leftrightarrow$ $\neg$presentIn(C_1 x)) i.e., this is a necessary and sufficient condition on the domain of C_1.

(A2) ist(C_1 ($\forall$ x $\bar{\text{L}}$ightSocket(x) $\Rightarrow$ ($\exists$ y connectedTo(x y) $\wedge$ Wire(y))))

Let us denote the formula (LightSocket(x) $\Rightarrow$ ($\exists$ y connectedTo(x y) $\wedge$ Wire(y)) as P(x)
and the formula (Fixture(x) $\wedge$ $\neg$installed(x)) as Q(x).
So we have

(A1) ($\forall$ x ist(C_2 Q(x)) $\Rightarrow$ $\neg$ presentIn(C_1 x))

and

(A2) ist(C_1 ($\forall$ x P(x)))

Let us assume further that if P is true of an object in C_1, it is true of that object in C_2. This can be written as,

(A3) ist(C_1 P(x)) $\wedge$ presentIn(C_2 x) $\Rightarrow$ ist(C_2 P(x))

What follows is an outline of a proof (note that it is not the whole proof - only an outline.)

Using inference rule (i) with (A2) we get

(D1) ($\forall$ x presentIn(C_1 x) $\Rightarrow$ ist(ci P(x)))

Using Rule T (of first order logic), with (A3), we get

(D2) ($\forall$ x presentIn(C_2 x) $\Rightarrow$ (ist(C_1 P(x)) $\Rightarrow$ ist(C_2 P(x))))

Using the transitivity of $\Rightarrow$, with (A1),

(D4) ($\forall$ x presentIn(C_2 x) $\Rightarrow$ (ist(C_2 q(x)) $\Rightarrow$ ist(C_2 P(x))))

Apply Inference rule l to D4 we get

(D5) ist(C_2 ($\forall$ x q(x) $\Rightarrow$ P(x)))

2.4 Lifting

In the two proofs shown above, we used formulas in one context to conclude formulas in another context. This involved performing a *relative decontextualization* of the formula, i.e. the differences between the origin and target context had to be taken into account to obtain a formula with the same truth conditions in both the contexts. This procedure is called *lifting* and the axioms which enable us to do this are called *lifting rules.*

As the examples in the next chapter will show, it is very important to be able to combine information from different contexts. Since this process is done using these lifting rules, it is important have a good understanding of these rules.

Let us look more closely at the two proofs given above and examine them from the perspective of lifting. One of the striking aspects of the first proof is that we had to supply an axiom such as (A5). Similarly, in the second proof, we had to supply the axiom (A3). Both these axioms have the property that they posit that some property of an expression carries over from one context to another. In the first case, A5 states that the denotation of "Fred" is the same in both contexts and in the second case, A3 states that the the truth value of the formula p(x) (for specific x) is same in both contexts. It seems that the number of axioms required of this sort is going to be very large. We need some general lifting axioms which will obviate the need to write separate axioms of this sort.

Our goal is to provide a general framework for writing lifting rules both to simplify writing lifting rules and to lay out the general approach to be taken for writing these rules.

2.4.1 What should a lifting rule do?

To begin with let, us define what a lifting axiom is supposed to specify. Given a pair of contexts (C_1 and C_2) and a formula F1 in C_1, we would like to determine what an equivalent formula in C_2 would be, i.e. we are interested in computing a formula F2 which in C_2 would 'state the same thing' in C_2 as F1 states in C_1. For the present, this concept of 'states the same thing' will be simplified to a biconditional. Later we will provide a model theoretic definition of this concept. Let us consider a simple example of the concept of 'states the same thing as'.

Example : Consider the following situation. Fred is standing in front of Chris. There is a flower pot to the left of Chris. Fred says "I like that flower pot to your left". Let this statement be F1 and the context in which this is uttered by C_1. Chris then moves so that the flower pot is to his right and tells Fred that he did not hear what Fred just said and asks him to say it again. Fred wants to convey the same message in this new context (C_2) but cannot use the same sentence (F1). The sentence which states the same thing in this new context is "I like that flower pot to your right". Call this second sentence F2. F1 states in C_1 exactly what F2 states in C_2. Given F1, C_1 and C_2, the process of obtaining F2 is called lifting and the rules we use to obtain F2 are called lifting rules.

2.4.2 Some intuitions behind lifting

Let us now examine the intuitions that we wish to capture in the general lifting rules and/ or lifting framework.

Although there will be differences between contexts, in general we expect that there will be a significant similarity and overlap between contexts; at least between contexts we are trying to use together. This overlap will manifest itself in a significant number of the terms corefering across the two contexts. Also, many of the formulas might get lifted without much modification. In this case, it would be desirable not to have to write separate lifting rules (and coreference statements) to enable the lifting (and coreference). In general, if a lifting is not going to involve any change to the formula under consideration, we should not have to write an explicit lifting rule to capture this.

Consider a lifting rule such as (A1) that specifies the change to ground atomic formulas involving the predicate Tall. Using this rule, if we know Tall(Fred) in C_1, we can conclude Tall(Fred, Person) in C_2. However, this rule is not very useful if all we knew was

(Tall(Fred) $\vee$ Tall(Bill))

Intuitively, this formula should get lifted as
(Tall(Fred, Person) $\vee$ Tall(Bill, Person)).

This is just a special case of a more general requirement we would like to impose - i.e., that of *compositionality.* Rather than specifying lifting rules for each formula separately, we should be able to specify lifting rules for individual predicates (and functions) and formulas involving these predicates (and functions) should get lifted automatically based on this specification.

There are differences in the vocabulary (both in the words that are used and in the intended denotations of these words) between contexts. The lifting rules will specify how these vocabularies can be mapped between each other. Using this specification, formulas (involving these words) should be lifted. So, in the above example, using (A1), if any formula in C_1 involves Tall(x), in the formula lifted into C_2, Tall(x) should be replaced with Tall(x, Person).

Based on these two requirements, we use the following framework for lifting. There are two components in this framework.

(a.) The Default Coreference Rule (DCR). Informally, this rule states that as a default, the meaning of a symbol does not change from one context to another. The rule has two parts, one for terms and the other for predicate and function symbols.

(i.) Term DCR : Given a (term) symbol S, if we pick any two contexts ci and cj, this symbol denotes the same object in both contexts.

(DCR-T) ($\forall$ ci ci $\neg$ab-dtc(S ci cj)) $\Rightarrow$ (value(ci S) = value(cj S)))

This rule is a dual of the unique names assumption (UNA) in the following sense. The UNA states that given two different symbols, as a default, they denote different things. DCR-T states that given a symbol and two different contexts, as a default, the symbol denotes the same thing in both contexts. The DCR shares a problem with the UNA in that it is very hard to completely axiomatize. Since the S in the above formula is an argument to the function 'value', which makes a statement not about objects but about terms, we cannot quantify over this argument. Therefore, we assume an axiom such as the above for each symbol. In practice, as with the UNA, the problem solver is modified to deal with this, i.e., we don't really have to state all these axioms.

(ii.) Function/ Predicate DCR : DCR-T, if applied to predicates (and functions) would imply that as a default, the set of tuples denoted by a given predicate symbol in two contexts are the same. Consider a predicate symbol such as Tall. If we applied DCR-T to this, we could conclude that (as a default), the set of tall people in C_a was the same as the set of tall people in C_b. Based on this, if we knew that Fred was a member of this set in C_a, he is a member of this set in C_b.

However, if we found even one exception, i.e., some person who is a member of the set in one context but not in the other context, the default about the sets being equal breaks down and we cannot make any predictions about any of the other members of the set. This is certainly undesirable. Rather than stating that as a default the two sets are equal, we need to state that they have a maximal intersection, i.e. as a default, if an object is an element of the set in one context, it is an element of the set in the other context, irrespective of whether another element is a member of both sets.

(DCR-P) $(\forall$ ci cj P X $\neg$ab-dcp(P ci cj X) $\Rightarrow$ (ist (ci P(X)) $\Leftrightarrow$ ist(cj P(X))))

(DCR-F) $(\forall$ ci cj F X $\neg$ab-dcf(F ci cj X) $\Rightarrow$ (ist (ci F(X) = y) $\Leftrightarrow$ ist(cj F(X) = y)))

X is a vector of variables of length equal to the arity of P (or F) in ci.

These two axioms take care of the lifting of atomic formulas and conjunctions and negations of atomic formulas. These axioms might also be used to lift quantified formulas. If the domains of two contexts is the same, then formulas of the form ($\forall$ x P(x)) can also be lifted from one context to another without any modification.

(b.) We now turn our attention to non-atomic formulas. If C_1 contains (P(a) $\vee$ P(b)), the above lifting formulas[2] are insufficient to lift this from C_1 to C_2.

We now present some axiom schemas to take care of lifting non-atomic formulas. The intuition behind these axioms is as follows. We have two contexts C_1 and C_2 and pairs of expressions $((e_1C_1\ e_1\ C_2)(e_2C_1\ e_2C_2)\ \ldots)$ such that eiC_1 states the same thing in as eiC_2 states in C_2. The formula F is composed from (e_1C_1, e_2C_1, etc.) using a composition function L. i.e. $F = L(e_1C_1, e_2C_1, \ldots)$. Let $G = L(e_1C_2, e_2C_2, \ldots)$. Then F should state the same proposition in C_1 as G states in C_2.

(CR) ($\forall$ ci cj costate(ci α_1 cj α_2) $\wedge$ costate(ci γ_1 cj γ_2) $\Rightarrow$
costate(ci (α_1 LC γ_1) cj (α_2 LC γ_2)))

where LC could be either one of $\vee$ or $\Rightarrow$. costate is like a 'macro' and the following schema specifies costate. costate(ci Aplha cj γ) is equivalent to
ist(ci α) $\Leftrightarrow$ ist(cj γ).

The above axiom allows us to lift (P(a) $\vee$ P(b)) from C_1 to C_2 using the DCR.

[2]There are arguably some cases where we don't want the above lifting axiom to allow us to perform this lifting, but we will not delve with those here.

It should be noted that the above axioms provide only a framework and in some sense, allow for very "promiscuous" lifting across contexts. In the next chapter we will cover several restrictions where these lifting axioms are blocked.

We now show an example of using this framework for lifting and then discuss the semantic basis for this framework.

2.5 Example of a proof involving lifting

Given

(A1) $\mathrm{ist}(C_1\ (\forall\ x\ \mathrm{Plays}(x\ \mathrm{BasketBall}) \Rightarrow \mathrm{Tall}(x)))$

Given no other information, we have from the DCP and DCT,

(D1) $\mathrm{corefers}(C_1\ \mathrm{BasketBall}\ C_2\ \mathrm{BasketBall})$

(D2) $\neg\text{ab-dcp}(C_1\ C_2\ \mathrm{Plays}\ [x\ \mathrm{BasketBall}])$
$$\Rightarrow (\mathrm{ist}(C_1\ \mathrm{Plays}(x\ \mathrm{BasketBall})) \Leftrightarrow \mathrm{ist}(C_2\ \mathrm{Plays}(x\ \mathrm{BasketBall})))$$

(D3) $\neg\ \text{ab-dcp}(C_1\ C_2\ \mathrm{Tall}\ x) \Rightarrow (\mathrm{ist}(C_1\ \mathrm{Tall}(x)) \Leftrightarrow \mathrm{ist}(C_2\ \mathrm{Tall}(x)))$

Let us assume that we don't have any exceptions, i.e., all occurrences of ab-dcp are negative. This allows us (using (DCR)) to conclude

(D4) $\mathrm{ist}(C_2\ (\forall\ x\ \mathrm{Plays}(x\ \mathrm{BasketBall}) \Rightarrow \mathrm{Tall}(x)))$

That this conclusion follows can be seen more intuitively by the following argument. For any particular A, we can conclude that

$(\mathrm{ist}(C_1\ \mathrm{Tall}(A)) \Leftrightarrow \mathrm{ist}(C_2\ \mathrm{Tall}(A)))$
and
$(\mathrm{ist}(C_1\ \mathrm{Plays}(A\ \mathrm{BasketBall})) \Leftrightarrow \mathrm{ist}(C_2\ \mathrm{Plays}(A\ \mathrm{BasketBall})))$.
Based on this, we can conclude
$\mathrm{ist}(C_2\ \mathrm{Plays}(A\ \mathrm{BasketBall}) \Rightarrow \mathrm{Tall}(A))$
Since we can do this for any A, it follows that this is true for all objects. Hence we have,

$\mathrm{ist}(C_2\ (\forall\ x\ \mathrm{Plays}(x\ \mathrm{BasketBall}) \Rightarrow \mathrm{Tall}(x)))$

Now we decide that C_2 needs a more sophisticated vocabulary. More specifically, we decide that we distinguish between professional basketball and backyard basketball. To say that Fred plays Professional basketball, we write Plays(Fred BasketBall Professional). C_1 implicitly assumed that we were talking about Professional participation.
So we have

(A2) ($\forall$ x ist(C_1 Plays(x BasketBall)) $\Leftrightarrow$ ist(C_2 Plays(x BasketBall Professional)))

This axiom replaces (D2) in the earlier proof. We conclude that

(D6) ist(C_2 ($\forall$ x Plays(x BasketBall Professional) $\Rightarrow$ Tall(x)))

The most important aspect of this example is that based on (A2) which made no mention of Tall or (A1), the appropriate modification to (A1) can be derived.

In this example, we assumed that there were no exceptions, i.e. that the instances of ab-dcp were all negative. Of course, this is an unreasonable assumption to make. Some conditions under which ab-dcp(C_1 C_2 p x) will be true include,

a. One or more elements of x are not presentIn C_2.

b. p is not in the vocabulary of C_2.

c. The arity of p, or the argument types of p are different in C_2.

2.6 Model Theoretic Account of Lifting

The discussion of lifting has until now been very "proof theoretic" in flavor. I.e., we have only discussed what should be done during lifting. Given that lifting is very central to contexts, it would be desirable to also have a model theoretic account of lifting. We now present such an analysis.

In the introduction we said that both sentences used in the KB for storing information and those used for communication depend on context. We then defined the process of 'lifting' or 'relative decontextualization' as that of changing a formula in one context to obtain another formula which states in a new context, what the original formula stated in the first context. Up until this point, the concept of 'states the same thing as' was used informally. We now provide a formalization of this concept and show how the lifting rules discussed earlier map into this analysis.

We motivate our analysis using communicative sentences (i.e., utterances). Though most utterances are in natural language, there is no reason why communications cannot use a more formal language. For our purposes, we will assume that utterances are formulas in first order logic. The analysis we present can be extended to formulas in KBs as well.

2.6.1 Context Dependence of an Utterance

We first begin by analyzing the concept of context dependence of an utterance. Since there is a theory associated with each context, instead of saying that the meaning of an utterance depends on the context, let us say that there is a background theory associated with each utterance. We would like to characterize how exactly this background (B) theory affects the meaning of the utterance formula (U).

One view of a theory is as set of sentences that can be derived from the given axioms. Another view of a theory (a logical theory, i.e., a set of sentences closed under entailment), is that it specifies a set of possibilities. If we have no axioms (i.e. know nothing), anything is possible. As we get more axioms, the set of possibilities shrinks and if we have enough axioms to get a complete theory, there is exactly one possibility (this is very rarely achieved). The set of possibilities corresponding to a theory is the set of models that satisfy it, i.e., are consistent with it. The addition of an axiom, if the axiom has any new information, serves to eliminate some of these possibilities. The information conveyed by the new axiom/ formula U with respect to the background theory B, is the set of models eliminated from B by the addition of U to B.

The background theory B restricts what can be conveyed by the formula U. If B is already sufficiently restricted so that no model is eliminated by U, no information is conveyed by U (with respect to B). In this case, no new conclusion can be derived because of the addition of the formula. Indeed, the formula was already part of the theory.

To illustrate this, consider the following situation. We have a room full of people, listening to a talk. The talk is about the people in the room. So, the theory associated with this talk constrains the set of people to those in the room. I.e., the only people in all the models of this theory are those people who are in that room. The speaker makes a statement "Everyone is wearing a tie" which is translated as ($\forall$ x human(x) $\Rightarrow$ hasClothes(x Tie)). This statement does not intend to state that all humans (on the planet) are wearing ties. Since the background theory has already restricted the set of humans to be those in the room, all it conveys is that all the humans in the room are wearing ties. To elaborate this further, assume that the background theory has ($\forall$ x human(x) $\Rightarrow$ inRoom(x)) to constrain the set of people to be those in the room. The statement "everyone is wearing a tie" could also have been stated as "everyone in the room is wearing a tie",
i.e., ($\forall$ x human(x) $\wedge$ inRoom(x) $\Rightarrow$ hasClothing(x Tie)).
This statement, would be exactly as the statement made earlier and in fact can be derived from that one. Given the background theory, the two statements are the same. Without the theory, they are very different. Thus the meaning conveyed by the statement depends on the background theory it assumes.

2.6.2 Relative Decontextualization of an Utterance

Now let us consider the problem of relative decontextualization. Going back to the example above, assume that one of the people in the room is called outside and asked what the speaker just said. When this person (who is called outside - call him P1) tells the questioner what the speaker said, the context, i.e., background theory associated with the answer, is different from the one the speaker used. Assume that in this new theory, the objects are the questioner, the speaker, the person called outside (P1), and a secretary who is nearby. With this as a background theory, P1 cannot claim that the speaker said that everyone was wearing a tie. That would imply that the speaker claimed that the secretary was wearing a tie, which he did not. An appropriate report of what the speaker said would be "everyone in the room was wearing a tie". The questioner is not interested in anyone else in the room (other than the speaker and P1) and the only objects in this theory also in the theory of the speaker's utterance are the speaker and P1. So the only possibilities that the questioners theory has which are eliminated by the speakers utterance are the ones which claim that the speaker and P1 are not wearing ties. So the speakers statement can be reported relative to this other background theory as, "the speaker and P1 are wearing ties."

More generally, the problem of relative decontextualization can be defined as follows. We have a theory T_1 relative to which a formula F_1 is uttered. This formula eliminates T_{11} (subset of T_1) from T_1. If we consider a different background theory T_2, we a the corresponding formula F_2 to eliminate certain possibilities from T_2. How do the possibilities eliminated by F2 from T_2 correspond to T_{11} in the case that F_2 is the relative decontextualization of F_1?

In order to specify this correspondence, we have to assume that the relation and function tuples of the models in T_{11} can be mapped onto the relation and function tuples of the models in T_2. This mapping is called the *correspondence function (CF)*. So CF(R U_1 U_2) is the relation in U_2 that corresponds to the relation R in U_1. We allow for CF to be a partial function so that not every relation and function in U_1 has a corresponding relation or function in U_2. Intuitively, given two models U_1 and U_2, and a relation which we refer to as "likes", CF will map from the relation corresponding to the concept of likes in model to the concept of likes in the other model.

We now define the concept of the restriction of a model by a domain. The restriction of a model U by a domain D defines a new model UD, as follows.

a. The domain of UD is the intersection of U and D.

b. For each relation (or function) RU in U, UD contains a relation (or function) tuple RUD such a tuple appears in RUD iff it appears in RU and all the objects in the tuple are in D.

c. The interpretation function's range is of course restricted to the domain of UD. The symbols that were assigned denotations outside the domain of UD by U are not assigned any denotation by UD.

A model UT_2 is eliminated from T_2 iff there is a model UT_{11} in T_{11} such that,

a. Let D be the intersection of the domains of UT_{11} and UT_2. D should be non-null.

b. Let UTD_{11} and UTD_2 be the restrictions of UT_{11} and UTD_2 by D respectively. For each relation (and function) R in UTD_{11}, CF(R UTD_{11} UTD_2) is equal to R and R should be non-null.

The formula F_2 which has the effect of removing all UT_2 satisfying the above conditions is the relative decontextualization of F_1 (in T_1) with respect to T_2. In the case that F_1 does not really add any new information, there will be no such UT_2. In the case that F_1 is contradictory with the background information in T_2, every model in T_2 will be eliminated. Note that it is possible that not all the information conveyed by F_1 in T_1 is captured by F2. In the example on people in the room wearing ties, the statement reported by P1 "the speaker and I are wearing ties" does not capture everything that the speaker said. This is because the narrow scope of the background theory used to report what the speaker said.

It is also possible that there is no decontextualization of the utterance. In the above example, consider the situation where the speaker's statement had been "Someone is wearing a tie". In this case, there is statement that states the same thing relative to the theory used by P1. In this case, no model will be eliminated from the theory used by P1 for reporting what the speaker said.

2.6.3 Extending the Analysis to KBs

This analysis (of relative decontextualization of utterances) extends to formulas in the KB too. Consider a context C_1 in the KB. There are a number of assumptions associated with the context and these assumptions form a theory. Even before we state any axiom of the form ist(C_1 F), there is a background theory associated with the context. E.g., if we assume that all the people in C_1 satisfy the constraint that they are rational, law abiding, etc., this defines the background theory with respect to which the axioms we add to this context should be interpreted. So the assumptions made by a context define a theory which serves as the background theory for the axioms in that context. The rest of the analysis is identical to the analysis for utterances.

2.6.4 Mapping the Semantic Analysis to Lifting Rules

The above analysis has been completely at the model theory level. It gives us no clues as to how one might go about generating the formula F_2. Let us now relate the framework for lifting described earlier to this semantic analysis. Informally, the lifting framework defines the proof theory for the semantic definition of relative decontextualization given above. We now make this relationship more precise.

The semantic analysis used the function CF to map relations between models. We assumed that we are given this mapping function F. There is the obvious problem that we simply don't have any such function available. There is the additional problem that given a relation R in one context, there might not be any one relation corresponding to this in another context but might correspond to a relation with a different arity or some complex constraint between relations in the other context.

E.g., C_1 has the unary relation Tall. C_2 has the binary relation Tall such that whenever Tall(x) is true in C_1, Tall(x Person) and Person(x) is true in the other context. In such a case, before the semantic analysis can be applied, we need to "normalize" the models of C_1 and have F map between relations (and functions) of these normalized models to relations and functions of the models of C_2.

Let us now see the role played by the different elements of the approach used for lifting in "implementing" the semantic definition of relative decontextualization. The default coreference rule and other lifting rules for defining the relationship between the vocabularies of the two context specify the F function. They also take care of decontextualization in the case where the formula F_1 is a ground atomic formula. The compositionality rule CR using the lifting for atomic formulas takes care of decontextualization of non-atomic ground formulas. The inference rule (i) - (l) take care of the restriction of models (based on differences in the domain) and take care of the lifting of the quantifiers.

2.7 Discussion

There are many other issues related to lifting and most of these are dealt in the next chapter. However, one of these issues belongs to this chapter and is taken up now. This is the issue of specifying the function F (which in some sense defines the costatement relation between the vocabularies) using biconditionals.

Costatement as biconditional : As we pointed out earlier and as the examples showed, to say that the formula A in C_1 states the same thing as the formula B in C_2 states, we write ist(C_1 A) $\Leftrightarrow$ ist(C_2 B), i.e., we equate costatement with a biconditional. This is inadequate for two reasons.

i. Costatement is much stronger than biconditionality, i.e. biconditionality does not capture the intuitive import of costatement.
   E.g., In C_1, F is (> 10 no-of-planets(SolarSystem)). Assume that in C_2, the formula that states the same thing is is also F. However, in C_2, we know that no-of-planets(SolarSystem) is 9. Therefore we have,
   ist(C_1 (> 10 no-of-planets(SolarSystem))) $\Leftrightarrow$ ist(C_2 (> 10 9)). Now, (> 10 9) certainly does not state the same thing as F in C_1.

   The problem of distinguishing costatement from biconditionals is similar to that of distinguishing definitions from biconditionals. There are numerous murky philosophical issues involved here and I plan to avoid these altogether.

   Rather than trying to capture the full import of costatement, we will try and make do with biconditionals in my treatment.

ii. The biconditionals might imply more than we want out of costatement. E.g., In C_1, F is (= 9 no-of-planets(SolarSystem)). C_2 corresponds to astronomy in the 17th century and so the no-of-planets is equal to 7. So the formula
   ist(C_1 (= 9 no-of-planets(SolarSystem))) $\Leftrightarrow$ ist(C_1 (= 9 no-of-planets(SolarSystem)))
   is not true. However, the 'proposition' stated by F (i.e., the content of F) can be stated in C_2 and is the formula F. There is no way for us to state this without also stating that F is true in C_2.

Pragmatically, the problem is that we might not know much about the theory associated with C_2 when writing the lifting rules. The lifting rules are supposed to map vocabularies, and needing to know much about the theories associated with the contexts involved significantly complicates matters. The crux of the problem here is that by equating costatement with biconditionals, the distinction between costatement and truth conditions is lost.

We cheat and use defaults as a way of getting out of this problem. The biconditionals are written as defaults. In the case that we don't want to infer the truth of the relevant formula in one of the contexts, the default is assumed to be violated.

What to lift (assuming the lifting can be done) is stated using default biconditionals. When the lifting may or may not be done is stated by specifying when the default does not hold.

Chapter 3

The Applications of Contexts

The focus of this chapter is on the different uses of the context mechanism. After listing some of these uses, we examine them in detail by means of examples chosen from the Cyc implementation of contexts.

Though there is a single logical mechanism underlying contexts, contexts may be used for many different purposes. Implementing the logical machinery is only a small fraction of the effort involved in building a context based system. The bulk of the effort lies in writing the rules for describing and interelating contexts. While the logical machinery itself is neutral to the use made of contexts, the structure and content of these rules is heavily dependent on the kind of use.

We therefore begin by listing these different kinds of uses. Later in the chapter, we will, with the aid of detailed examples, provide a more thorough description of each of these kinds of uses.

3.1 Different Uses of Contexts

The different uses described below are not meant to be mutually exclusive and some are actually specializations of others. Most contexts in a system will involve more than one of the following uses.

(1) As a means of referring to a group of related assertions (closed under entailment) about which something can be said. Such a group of assertions might form

- **(a)** a theory of some topic, e.g., a theory of mechanics, a theory of the weather in winter, a theory of what to look for when buying cars, etc. Contexts used in this sense are called "Microtheories" (Mts) and usually have the suffix *Mt*. Different microtheories might make different assumptions and simplifications about the world with contexts providing a mechanism for recording and reasoning using these assumptions.
- **(b)** a representation (of some situation) that is tailored for the problem it was set up to solve, e.g., a model of a christmas tree as a perfect cone, used for determining whether it will fit in a given box; a model of an object as a point mass for determining its trajectory, etc. Contexts specific to a particular problem solving

task are called Problem Solving Contexts (PSC). A problem solving task might involve answering a single question or a number of related questions.

(c) a "very slightly decontextualized" representation of the utterances made in a discourse, e.g., a discourse representation that retains anaphoric and indefinite references. In such a context a phrase such as "the person" might not be translated to the represent the actual individual referred to, but may be represented by using a term such as (The Person). Such contexts are called Discourse Contexts.

Some of the reasons for bundling together a set of assertions into a named object include the following.

- Assumptions that are reasonable only for limited domains may be made. These in turn allow simplification of the statement of the theory (of the limited domain).
- By collecting a small relevant subset of the knowledge base into a bundle, they provide a mechanism which the problem solver may use to focus on the relevant information.
- By naming theories, they allow for statements about a theory (regarding its utility, reliability, precision, etc.) to be made.
- By keeping different theories distinct, the problem of maintaining consistency is transformed from that of maintaining global consistency to maintaining local consistency, which is arguably a simpler problem.

(2) As a mechanism for combining different theories. If the assertions in one context were not automatically available (as appropriate) in other contexts, the system might as well be a set of disconnected knowledge bases.

Though different assertions might be made in different contexts, when solving a problem, the system needs to pull together information from these different contexts. The rules that specify how this "pulling together" is to be done are called *lifting rules* and the process of using these rules is called "lifting".

The Context of a Query

A knowledge base such as Cyc includes a number of contexts, each making its own set of assumptions. Queries posed to the KB are also likely to make their own set of assumptions. When querying the KB, the question should be posed in a Problem Solving Context that captures the assumptions made by the question. For doing so, we might need to create a new Problem Solving Context and ask the query in this new context.

The process of asking a query in a certain context is done by *entering* that context and then asking the query as would normally be done in a system without contexts. More generally, the system is always *in* some context and this context is called the *current context.* The interactions with the system (i.e., the assertions we make and questions we ask) presuppose the assumptions made by the context the system is in at that time.

When answering a query in some context, the system might need to use information from other contexts. For example, during the course of a certain interaction with a user, Cyc may be provided a description of a situation involving a waitress and a troublesome customer in a particular restaurant. This description is in a context (a Problem Solving Context) which has been created for this interaction. Cyc is then asked a question about how the waitress is likely to behave with this particular customer. At this point, though there are other contexts in Cyc which contain assertions about the behavior of people in different situations, this new context does not contain any of those assertions. In order to answer the question, information must be lifted from those other contexts into this current Problem Solving Context.

The assumptions made by the Problem Solving Context could be different from the assumptions made by the contexts from which the information is lifted. These differences need to be "factored in" when the assertion is used in the Problem Solving Context. The determination of which contexts from which to lift and the "factoring in" of these differences in assumptions is taken care of by the lifting rules. The following short example illustrates the concept of "factoring in".

Example:

Cyc contains a microtheory describing the behavior of individuals while at work. It assumes that the individuals follow the code of conduct of the workplace. Based on this assumption, the theory states that in office settings, people don't usually make loud noises or create scenes in any other fashion, even if provoked.

We are then given a situation where a small child is brought into the office by her parent. The system cannot reasonably assume that the child follows (or even knows) the appropriate code of conduct. The system should therefore *not* predict that the child will remain quiet even if provoked. On the other hand, the system should be able to predict that the parent is not likely to create a scene. So while the system needs to use the rule about behavior in the workplace to determine the behavior of the parent, it should "factor in" the assumption made about the code of conduct being followed, infer that small children don't usually follow such codes and therefore should not apply this rule for the child. The lifting rules should appropriately modify assertions so they apply only to parts of a situation that satisfy the assumptions. In this case, the rule should be modified so that only those aware of the code of conduct don't make loud noises, etc.

Suppose that later in the development of the system, we realize that the microtheory mentioned above makes other assumptions as well. E.g., it assumes that the people involved are rational, that if a person is in an office he has some reasons for being there and these reasons are in keeping with the goals of the organization, etc. Going back to the example about the parent bringing the child to the office, given these additional assumptions, the system should be able to infer that the parent is rational, has a reason for being in the office, etc.

Contexts provide a mechanism for stating and using these lifting rules.

(3) For limiting the scope of a theory. Common sense theories are almost never universally applicable. The objects and situations within the purview of a theory (i.e., the scope of the universal quantifier), are determined by the assumptions made by the theory. In the example given in the previous paragraph, the microtheory about the behavior of people in work places assumes the code of conduct being followed. This restricts the scope of the theory to exclude children, mentally retarded adults, etc. This assumption simplifies the theory and could either have been made by design or could have been made unintentionally but then later noticed and made explicit. In either case, the context mechanism can be used to restrict the application of this theory to individuals satisfying this assumption.

It should be noted that at any stage of building and using a knowledge base, a significant number of assumptions made by the microtheories in that knowledge base will not have been made explicit. But if these assumptions have been made by almost all the contexts in the system, it might not be very important to make them explicit.

Example:

Another assumption made by the aforementioned theory is that the ability to make loud noises is generally under the conscious control of individuals. This assumption (not yet explicitly stated in Cyc) is satisfied in almost every situation the system might encounter. So though this assumption is made by the theory, it is an assumption both made by the theory and satisfied in situations where this theory is used. Therefore it does not matter that the assumption has not yet been made explicit. What matters is that when the scope of Cyc is increased to a point where it knows about phenomenon such as individuals spontaneously making loud noises, it can be told that this assumption has been made all along.

(4) Multiple models of a situation. Many situations are able to be modeled using different representations with each making a different tradeoff between efficiency, simplicity and accuracy. Though there are many examples of this in engineering domains (e.g., the Navier Stokes equation for fluid flow versus the simpler equations for fluid flow), let us pick a more mundane example to illustrate the ubiquity of this phenomenon.

Example:

Consider a commonplace task such as determining the path to be followed in walking out of a room. We would like to minimize the total effort (and therefore the distance covered). A very simple model would be to walk straight towards the door avoiding large obstacles with the obstacles being modeled as cuboids. However, one can progressively get more and more sophisticated by considering the exact shape of the obstacles, the number of turns involved, ..., all the way up to and including the wind resistance from the currents set up by the exhausts in the room. There are conceivably situations where each of these models is optimal. If one doesn't really care to distinguish between paths with a difference of only one foot between them, or worry about turns, the most naive theory might be enough. However, if one is transporting a heavy load that could spill, one might care more about the number of turns and the distance traveled.

Unlike the example of contexts given in use 2, where there is a clear notion of the predictions of the theory being false when applied to small children, none of these theories is *a priori* true or false. We need to replace the concept of a theory being true with the concept of a theory being adequate for the problem we are trying to solve.

It should also be noted that these theories are often mutually contradictory and therefore might lead to contradictory answers. When there is a contradiction between theories, one of the theories will usually be more accurate than the other. Sometimes, both theories are equally "correct" and a choice must be made as to which theory to use in which instance. The canonical example of this is the choice that has to be made between the particle and wave theories of matter. Though it is reasonable to require internal consistency within each of the theories, it is not reasonable to require consistency across theories.

Contexts keep these theories apart. They also enable us to represent the concept of a theory being suitable for some task but not necessarily all tasks.

Given a knowledge base with many of these theories, the system must determine which single theory or which combination of theories to use to solve a given problem. The context mechanism is used to keep the different theories separate and the selection of the appropriate theory is specified using lifting rules. The problem of putting together a theory that is adequate for solving a given problem is often called the "model formation" problem.

(5) Natural language and representation. Though computer representations are certainly very different from natural languages, there are many influences and constraints from natural language utterances which carry over to computer representations.

Natural language utterances are generally very compact. They derive this property in part by exploiting the context in which they are made. The context of an utterance is set up by a very wide variety of parameters. These parameters range from very coarse grained factors such as the cultural settings and the socioeconomic backgrounds of the conversants, to medium grained factors such as the goals of the conversants, to fine grained factors such as the immediately preceding utterance or even preceding gestures. Many of these factors are coarser grained and unrelated to the contextual phenomena (such as anaphora, indexicals, etc.) that are studied by linguists. For example, instructions on where to look for a particular star in the sky (such as those that can be heard on a radio program or found in the Scientific American magazine) assumes a terrestrial perspective. An account of an automobile accident given by a witness is assumed to be from the perspective of that witness. What a geologist means by the term "fast" is certainly different from what a computer scientist means by the same term.

A few of the effects of natural language utterances we will attempt to take into account, and constraints we will attempt to satisfy, include the following.

(a) For a program such as Cyc, we expect to eventually have a natural language front end. The examples of coarse grained contextual effects in the natural language utterances given above are clearly outside the scope of linguistic theories and require domain knowledge to be understood. The system might not *a priori* know all the contextual aspects when approaching a text or a conversation, but instead it may have to infer these as the discourse proceeds.

What we have here is a chicken and egg situation. The system needs to use the information in the knowledge base together with a new assertion to determine the context dependence of that new assertion. To effectively combine the information in the knowledge base with the new assertion, the assertion must already be in the knowledge base. But the system needs to factor out the context dependence of the assertion before it can be added into the knowledge base! The only way out of this cycle is to add the mechanism required to allow us to add the assertion to the knowledge base without factoring out all the contextual effects.

So, there are at least two requirements imposed by the desire to have a natural language front end to a knowledge based system.

(i) The front end should be able make a set of assertions that have some as yet unidentified contextual aspects left in them without these assertions being inappropriately used together with other assertions already in the KB.

(ii) The system should be able to deduce at least the more significant of these contextual assumptions so that the assertions obtained from the natural language front end could then be used together with the assertions already in the knowledge base. Some of this deduction would be proactive and some would take place when there was a need to use these utterances together with utterances made in some other context.

When processing a text (or holding a conversation), the system maintains a discourse context. The natural language front end performs whatever disambiguation is able to be performed based on purely linguistic information and then asserts the resulting translations into this discourse context. The system then deduces whatever contextual effects are apparent. After this point, these discourse contexts are just like microtheories or problem solving contexts with further decontextualization being performed on demand. This strategy is explained in greater detail in the next chapter.

The knowledge base should both be able to represent and reason about the coarse grained contextual aspects of natural language. It should also be able to accept translations of natural language utterances which have these contextual dependencies left in them.

(b) For most topics (especially those not closely linked to perception), writing a theory (of that topic) in terms of formulas is usually preceded by writing the theory in english. This often manifests itself in some of the aspects of natural language remaining in the resulting representation.

Example:

Two theories might both involve the concept of "seller". The english word "seller" is highly polysemous, i.e., has many very closely related but distinct meanings: the person with whom a buyer interacts and might ask questions, the shop the where the buying takes place, the organization owning the shop where the buying takes place, etc. The concept "seller" may be represented in the knowledge base using a single predicate with the polysemous nature of the english word "seller" reflected as different theories using the predicate with these different english meanings as the intended denotation. The situation is especially bad with words that assume a very wide range (or continuum) of meanings. E.g., The term "USA" could be used to refer to any of the following: the United States government, the executive branch of the United States government, the judiciary branch of the United States government, the economy of the United States, the geographical region encompassing the United States, the defense forces of the United States, a delegation from the United States, ...

The mechanism of contexts is used for insulating these different meanings from each other and for translating from one to the other.

(c) If we can capture some of the earlier mentioned coarse grained context dependencies using the context mechanism, we might be able to make computer representations comparable to natural language utterances in compactness. With this in mind, we will occasionally set up natural language utterances as targets to emulate in our formal representations.

3.2 Lifting Rules

One of the primary reasons for introducing contexts is to simplify the construction of the knowledge base. The arguments presented earlier and the examples presented later make the case that the statement of domain theories gets simplified by the introduction of contexts. However, the price to be paid for the use of contexts is the need to write lifting rules. Given the number of axioms in each context and the total number of contexts, if the system required a separate lifting rule for each axiom, origin context, and target context tuple, much of the system would just consist of lifting axioms. This is clearly unacceptable.

One of our goals is to provide some very general lifting rules which hopefully will take care of the majority of the lifting. It should also be possible to implement these general lifting rules efficiently. We now present two heuristics/defaults which provide such a general basis. A more detailed discussion of this can be found in chapter 2.

The intuition behind these defaults can be explained by the following analogy. Consider a physicist and a geologist talking to each other about some problem. There will be differences in way each looks at the problem. These differences will be reflected in the vocabulary and language they use. So, statements made by each will have to be interpreted in a slightly different way. When the physicist says that some process is fast, he means something different than when the geologist says that a process is fast. However, despite the differences in their languages, there is more in common between the two languages than is different between them. They refer to the same concept when they say "time" or "earth" or "rock". A person writing the translation rules between these two languages should be able to exploit the fact that there is a great deal in common between these two people.

Our situation is similar. Though there are differences between contexts, there is even more in common between them. This is what makes it feasible to write the translation/lifting rules. The following two defaults make this concept of "there is a great deal in common" more precise and shows exactly how this may be exploited.

Default coreference. It is possible for a term to denote different things in different contexts. However, most terms are used to denote the same object across contexts. This default is called the "Default Coreference Rule" (DCR). Though special classes of terms such as indexicals, definite pronouns, etc. violate this assumption, most terms do follow this default.

The Term DCR

To state that a term A denotes in context c_1 the same thing B denotes in c_2 we write corefers(c_1 A c_2 B). Using corefers, the DCR can be written as

(DCR-T) $(\forall\ c_i\ c_j\ \neg\text{ab-dct}(S\ c_i\ c_j)) \Rightarrow (\text{value}(c_i\ S) = \text{value}(c_j\ S)))$

for each term S.

Predicate/ Function DCR

Similarly, if the system knows that P(a) is true in c_1, as a default we would like it to infer that P(a) is true in c_2. This is stated by the following rule.

(DCR-P) $(\forall\ c_i\ c_j\ P\ X\ \neg\text{ab-dcp}(P\ c_i\ c_j\ X) \Rightarrow (\text{ist}\ (c_i\ P(X)) \Leftrightarrow \text{ist}(c_j\ P(X))))$

ab-dcp(c_1 c_2 a p) is true if the default is violated for c_1, c_2, a and p.

(DCR-F) $(\forall\ c_i\ c_j\ F\ X\ y\ \neg\text{ab-dcf}(F\ c_i\ c_j\ X) \Rightarrow (\text{ist}\ (c_i\ F(X) = y) \Leftrightarrow \text{ist}(c_j\ F(X) = y)))$

Implications of the DCR

Because of the DCR (for terms and predicates), any ground atomic formula can be a default lifted from one context to another. So if the system has p(a) in one context, this can be (as a default) lifted without any modification to any other context. If two contexts have the same domains, $(\forall\ x\ p(x))$ can be lifted from one context to another without any modification.

However, we do not want to liberally lift assertions from one microtheory to another. Therefore, we restrict this lifting policy to lift assertions from microtheories (such as the AutoMt and HumanKinshipMt) to Problem Solving Contexts (such as CSPSC).

Given a formula p(a) in a microtheory c_1, the default lifting of this formula (i.e., lifting without any modification) to a problem solving context c_2 might be blocked for a number of reasons. A couple of these include:

(a) The representation of p in c_2 could be different. For example, it might have more arguments or the sort constraints on it could be different.

(b) The object denoted by the term 'a' in c_1. is not in the domain of c_2.

Other reasons will be considered in the examples that follow.

Compositional lifting In systems which use a declarative representation, the meaning of a symbol structure is constructed out of the meaning of its components, i.e., the meaning is constructed compositionally (as opposed to holistically). We would like a similar property for lifting. Consider the following situation. We have two formulas pc_1 and qc_1 in the context c_1. We also have pc_2 and qc_2 which state the same thing in

c_2 as pc_1 and qc_1 state in c_1 respectively. We want a formula F constructed from pc_1 and pc_2 to state the same thing in c_1 as a formula G (constructed identically from qc_1 and qc_2) in c_2.

So if bodyStyle(Car001 Notchback) states the same thing in c_1 as bodyStyle(Car001 NB) states in c_2, and bodyStyle(Car001 Convertible) in c_1 states the same thing as bodyStyle(Car001 CV) in c_2, we want (bodyStyle(Car001 Notchback) $\vee$ bodyStyle(Car001 Convertible)) to state the same in c_1 as (bodyStyle(Car001 NB) $\vee$ bodyStyle(Car001 CV)) states in c_2. The following schema captures this constraint.

$$\text{(CR) } (\forall \text{ ci cj costate(ci } \alpha_1 \text{ cj } \alpha_2) \wedge \text{costate(ci } \gamma_1 \text{ cj } \gamma_2) \Rightarrow$$
$$\text{costate(ci } (\alpha_1 \text{ LC } \gamma_1) \text{ cj } (\alpha_2 \text{ LC } \gamma_2)))$$

where LC could be either one of $\vee$ or $\Rightarrow$. costate is like a 'macro' and the following schema specifies costate. costate(ci Aplha cj γ) is equivalent to
ist(ci α) $\Leftrightarrow$ ist(cj γ).

With this additional rule, the default lifting applies not just to atomic formulas but to abitrary formulas. For more details, see the discussion and examples in the previous chapter.

These two defaults take care of the situation where the lifting does not involve any modification to the assertion. Other lifting rules which modify assertions as they are lifted will override these two defaults.

We are now ready to consider the examples illustrating these uses in some detail. These are all examples of the use of contexts in the Cyc KB.

3.3 Problem solving with contexts

Queries are addressed to the system in a Problem Solving Context. There are two different strategies which may be adopted in answering the query. The system might also use a combination of these two strategies.

Lift and Solve The system lifts assertions from other contexts into the Problem Solving Context and uses a conventional problem solver once inside the Problem Solving Context.

Shift and Solve The system switches into an already existing context, solves the problem in that context (using a conventional problem solver) and lifts the answer back into the context in which the query was posed.

The term "conventional problem solver" refers to problem solvers for a first order language (possibly with extensions for default reasoning), i.e., a problem solver which does not have mechanisms for dealing with contexts. Once inside a context, contexts do not affect problem solving and so one of the goals is to be able to use existing problem solvers in conjunction with contexts.

In the current implementation, the first of the above two strategies is more widely used and we restrict our attention to that strategy in this section. In one of the examples we will illustrate the use of the second strategy. We now discuss the first strategy.

3.3.1 Lift and Solve

To solve the query, the system lifts assertions from other contexts into the Problem Solving Context (PSC) in which the question was raised. The lifting process involves running the lifting rules. The lifting rules specify which assertions hold in the Problem Solving Context, i.e., the assertions that may be used for answering the question.

One approach would be to run the lifting rules to completion, allowing them to "build" a theory in the PSC, and then invoke a traditional problem solver (e.g., problem solver for first order logic). This has the advantage modularizing the context related problem solving steps (lifting) from the other problem solving steps. It also allows one to continue using traditional problem solvers for inferencing within a context.

However, this approach is infeasible in that it might take much too long for the lifting rules to run to completion. In the ideal case, the system should lift only those assertions and exactly those assertions which will be required to answer the query. Unfortunately, we have no (efficient and guaranteed) means of determining this set of axioms.

Even traditional problem solving systems usually employ different indexing mechanisms to access the most relevant axioms and the system can try to exploit this.

Using the conventional problem solver for help in lifting

Consider a simple model of a deductive problem solver. We distinguish between the following two components.

The Access Module which is responsible for indexing and accessing expressions from the KB.

The Inferencing Module which is responsible for using the expressions given to it by the access module to derive conclusions.

In the case of a backward chaining problem solver, given a query Q, the inferencing module asks the access module for assertions in the knowledge base which match Q. If none are found, the access module is then asked for expressions matching ($\alpha \Rightarrow$ Q). After substituting the right bindings for variables, the inferencing module sets up Alpha as its goal. This process ends when all the subgoals are found in the knowledge base.

A similar cycle of operation works for almost all deductive problem solvers. At any time, the inferencing module is manipulating only a few sentences and the choice of these sentences is based on the query asked and these expressions are obtained through the access module. This model holds true even for very complex problem solvers such as those in Cyc.

The queries that the inferencing module asks of the access module during the inferencing process define a loose set of constraints on which assertions are potentially relevant. Furthermore, the inferencing module asks for these relevant assertions only as it needs them. The queries the inferencing module makes of the access module depend heavily on the success or failure of the earlier steps in the inferencing.

Our approach is to use only the inferencing module of traditional problem solvers and replace the access module (which is usually dependent on the idiosyncrasies of the data structures used), with the mechanism for doing the lifting. I.e., instead of asking the access module for expressions, the inference module asks the lifting module for expressions. The lifting module determines what is to be lifted, and hands the inference module the lifted expressions. Of course, the inference module need not know where these expressions came from.

From the perspective of the inferencing module, it is dealing with a KB without contexts. Except of course, the number and complexity of assertions the inferencing module handles for each problem could be radically different. This scheme allows us to exploit the information the inferencing module can provide to guide the lifting. The lifting module, as one would expect, is significantly more complicated than the access module it replaces (actually the access module is incorporated into the inferencing module).[1]

Assume that the system is considering importing an assertion P from a context c_1 into a context c_2. The following decisions must be made.

(1) Determining whether an assertion P should be imported at all. There are two orthogonal factors determining this.

Relevance : Is the information in c_1 relevant at all to the goals of c_2? For example, one of the microtheories in Cyc is a fairly technical description of the structure of plants and flowers. At least *a priori* it seems the information in this microtheory is completely irrelevant to the problem we shall consider later of determining which car a person should buy. So for reasons of efficiency, even though this microtheory might contain assertions matching the pattern the inferencing module is searching for, it might be appropriate for the lifting module not to return the assertions from this microtheory.

The inferencing module is trying to determine the part types of Car001 in the context c_2. So it asks for rules matching the form (Q $\Rightarrow$ partType(Car001 x)). The AutoMt of course contains rules about the part types of cars. The BotanyMt also contains rules with right hand sides of the form partType(x ObjectType) where ObjectType is a constant. Notice that partType(Car001 x) unifies with partType(x ObjectType) and therefore the access module should in principle lift the rules from the BotanyMt too. However, we *a priori* know that Botany is irrelevant to determining the parts of a car and we would like to ignore this rule.

Unfortunately, there are very few cases in which entire microtheories may be deemed completely irrelevant. Furthermore, having the lifting rules restrict the importing of assertions from these less relevant microtheories in order to speed up problem solving seems inappropriate. The lifting axioms specify what is true in a context. The issue we are concerned with here is one of focus, i.e., we want the system to focus attention on what is more likely to yield answers first - without making any statement about the truth of the potentially less relevant statements.

[1] There is an analogy between this scheme and virtual memory. The current context is the physical/real memory, the other contexts are the virtual memory, the lifting module is the program for detecting page faults and swapping pages, and the user program is the inferencing module. Just as the user program might think that everything is in the physical memory, the inferencing module thinks that everything has actually been lifted into the current context right at the start.

One solution to both these problems is to abandon the binary relevant/irrelevant distinction and use the concept of more and less relevant instead. E.g., the inferencing module is searching for assertions matching a certain pattern and there are assertions in both the AutoMt and BotanyMt which match this. The assertions from the AutoMt seem to be *a priori* more relevant to selecting a car than those in the BotanyMt. The lifting module should therefore order the results it hands back based on this relative relevance information. This relative relevance statement can be made as follows:

moreRelevantMt(C_2 AutoMt BotanyMt)

In this way, the second problem is avoided (since we are not mixing truth and relevance) and the first is avoided since the doesn't make binary relevant/irrelevant distinctions. If the more/less relevant information we provide is incorrect, the system will still be able to derive the answers, it will simply take more time.

The moreRelevantMt statements are derived using heuristics such as - "if the operation of X depends on Y but does not depend on Z, the theory of Y is more relevant to the selection of the X than the theory of Z."

Appropriateness is the theory in c_1 accurate enough for the goals of c_2? For example, if the goal of c_2 were to determine which car the user should buy and the theory in c_1 was so coarse that it could not really distinguish between a Yugo and a BMW, (all that matters in c_1 is which means of transportation one uses), the predictions made by c_1 would not be adequate given the goals of c_2. Note that the predictions might be consistent with the other assertions in c_2, but they would still be inadequate for the goals.

This sort of appropriateness, where we are concerned with the accuracy of the prediction is dealt with in some of the examples presented later in this chapter.

Another parameter in determining the appropriateness is whether lifting the assertion will lead to an inconsistency in the target context. If it will lead to an inconsistency in the target context, it should not be lifted. Determining whether adding an assertion to the target context will make the target context inconsistent could (in theory) take forever. In practice the system uses a few heuristics which seem capable of detecting most cases where there would be a contradiction. These primarily involve determining whether the arity and sort constraints (in the origin context) on predicates and functions in the expression being lifted are satisfied by the assertions (involving that predicate or function) in the target context.

A policy of "risky" lifting is followed. I.e., the system does only cursory checking for consistency before lifting. However, if an inconsistency is discovered later during problem solving, the system backtracks and tries to verify that the lifting done by it was appropriate.

(2) The changes that should be made to P when lifting it from c_1 to c_2. The previous chapter contains a detailed discussion of lifting. Here, we provide a brief summary of the material presented in that chapter.

Given the formula P in c_1, the system wants to obtain a formula in c_2 that states the same thing as P in c_1. I.e., though the encoding of the assertion might be different across the two contexts (because of the differences in the assumptions made by the two contexts), the content of the assertion should not be changed. In other words, the proposition denoted by the formula P in c_1 should be the same as the proposition added to c_2. It is the job of the lifting rules to specify this change in the encoding of a proposition as it is moved from one context to another.

The general pattern of these lifting rules will be

$$\mathrm{ist}(c_i\ P) \wedge R(c_i,c_j,P,Q) \Rightarrow \mathrm{ist}(c_j\ Q)$$

where P is a formula, $R(c_i,c_j,P,Q)$ is a set of constraints that specify that some relation should hold between c_i, c_j, P, and Q, and Q is the formula that is added to the target context c_j.

One complication that arises from this approach is as follows. We would like to separate out rules specifying costatement, i.e. rules which specify that the formula P in context c_1 states the same thing as the formula Q in context c_2, from rules which specify when it is appropriate to add Q in c_2 based on P being true in c_1. So Q might very well be a valid formula in c_2 and we might want to know that Q states in c_2 what P states in c_1 without being forced to accept that Q is true in c_2 because P is true in c_1.

The rule schema given above simply asserts that Q is true in c_2 based on P being true in c_1. What we need is a hook into this rule that could be used to block the lifting under certain conditions. To enable this, these lifting rules will be defaults and one of the conjuncts in R will be of the form $\neg ab(c_i, c_j, \ldots)$ where ab is a predicate that is minimized (i.e., $ab(c_i, c_j, \ldots)$ may be assumed to be false). Unless required, I will omit these abs and assume that the rule is disjoined with such an assumption literal.

(3) Ambiguity/conflicts in lifting. There may be a situation where the system can either lift P from c_1 or G from c_3 (into c_2) but not both. There are three categories of cases where this might occur.

- **(a)** P (actually, the form of P appropriately modified for c_2) together with the assertions in c_2 makes predictions that contradict the predictions made by G (again, really the modified form of G) together with the assertions in c_2.
- **(b)** There is more than one lifting rule that may be used to lift P from c_1 to c_2 and the rules modify P differently and in a mutually inconsistent fashion. For example, P might involve a slot (i.e., binary predicate) such as seller. The first lifting rule might retain this as a slot while the other makes seller a ternary predicate. Since each predicate/function has a fixed arity in each context, one of the two (but not both) of the lifting rules should be used.
- **(c)** The lifted form of P uses a predicate (or function) with a certain arity while the lifted form of G uses the same predicate (or function) with a different arity. As in (b), since each predicate/function has a fixed arity in each context, one of the two but not both of P and G can be lifted in c_2.

The approach taken towards such ambiguity is as follows. If using one of the candidates yields a result but the other does not, the system prefers the one that yields the answer. If both yield the same answer, it does not really matter which one the system uses. If they yield different answers, nothing much can really be done. Theoretically, in such cases, the disjunction of the answers given by the alternatives should be asserted. However, in the implementation, such ambiguities are seen as gaps in the knowledge base (i.e., missing preference rules) and the user is queried about how to proceed.

3.4 Preliminaries to the examples

The Cyc KB, though it incorporates extensive inference facilities, is not geared towards any particular application. As a result, the information found in Cyc (and therefore the use of contexts in Cyc) is distributed over a very large number of domains. To obtain a slightly more focused presentation and to illustrate context related issues that arise in problem solving, many of the examples chosen are from a "car selection" application that has been built on top of Cyc. To help with the presentation of the example, we first briefly describe the purpose and structure of this application and then proceed to discuss the examples.

3.4.1 The Car Selection Program (CSP)

The Car Selection Program (CSP) accepts information about the user and makes suggestions on which kind of new car the person should buy. The information can be entered as formulas or by using a graphical interface to the application. Since almost any aspect of a person's life could potentially affect the car that is suitable for that person, this application is very suitable for illustrating the use of contexts for common sense representation and reasoning. This program attempts to do the following:

(a) Accept information about the user's occupation, budget, residence, family, hobbies, etc. The background information in Cyc is used to ensure the meaningfulness of the information provided. For example, if the user claims that he is 17 years old and has four kids, the program will question his input. This use of Cyc is in keeping with one of its stated aims, namely, to reduce the possibility of the "Garbage in garbage out" phenomenon.

(b) Based on the information the user provides about himself, make suggestions about features he might want in the car. These suggestions are deduced using the theory in Cyc concerning cars. The user may accept or reject these suggestions.

While knowledge of some of the features available on cars available at the present time (such as emmission control packages) is quite technical, an understanding of when which features (anti-lock brakes, fog lights, power windows, back seats, etc.) are useful still requires a good bit of common sense. For instance, one needs fog lights if one will be driving in fog often, a back seat if one expects to have more than one other person as passenger. A person often has to provide transportation to family members and so if one has a large family, a car with a back seat would be preferred. In addition to suggestions based on purely utilitarian grounds, factors such as the image the user wants to convey while using the car and recreational uses of the car are also taken into account.

(c) Using the list of features the user desires as well as his budget, a short list of the possible cars is generated based on the information available in a set of databases associated with the application.

The structure of the program and relevant parts of Cyc are as follows. The application itself consists of just the interface including functions for keeping the list of possible cars updated. The domain related reasoning is carried out by Cyc.

The information about automobiles is in the AutoMt, a microtheory about automobiles. Actually, much of it is stored in external, commercially available databases such as the Kelly Blue Book database, but for the purpose of the following examples, we can pretend that this information is explicitly there in Cyc. Given the wide scope of information that may be provided by the user, almost any of the microtheories in Cyc could be used in the course of an auto selection session. The wide range of microtheories touched upon by the examples that follow illustrate this.

At the beginning of each application session, a new problem solving context, called Car Selection Problem Solving Context (CSPSC) is created.[2] The system then enters this context and the rest of the session is spent in this context. The system makes a number of assumptions about this context, e.g., that the discussion is taking place in the late 20th century, it is taking place in the United States, the user wants to use the car for its intended purpose, etc. Later in some of the examples, we will consider in detail the representation of these assumptions. The inferencing about the user and about the car are performed in this context (CSPSC).

At the outset, the system creates Cyc terms (aka units) corresponding to the user and the car. Of course, at the beginning, the system doesn't know much about the car, and it might even turn out to be a hypothetical car. However, in CSPSC the system will assume that the car indeed exists and the problem is one of inferring enough of its properties so that its make and model can be constrained.

3.5 Examples of Uses of Contexts

The examples that follow are not intended to be a description of this application or form a script of a session using the application. The reader interested in a script containing many of the following examples should refer to Appendix A. This application is used merely to provide continuity and to focus the examples. Each of the examples has been chosen to cover a different application (though there are a couple of repetitions).

Short Overview of Examples

The following is a very brief description of the application illustrated by each of the examples.

[2]Actually, CSPSC is a class of contexts and a new one of these is created for each session with the user. However, for this discussion, we will just refer to the problem solving context used for the car selection as CSPSC.

(1-5) A sequence of increasingly sophisticated assumptions about time, dealing mostly with leaving different assumptions and constraints about time implicit.

(6) Combining a system with a particularly abnormal use of a certain term with a larger system and extensions of the scheme used in examples 1-5 for non-temporal modalities.

(7) Hypothetical reasoning.

(8-10) Representing the scope of a theory, implicit assumptions about the scope of the theory, dealing with assumptions made by a theory that cannot be stated within that theory.

(11) Use of terms similar to definite references (in natural language) for representation.

(12-15) Different ways of dealing with the multiple models of phenomenon.

(16-17) Coping with different theories using a term to denote slightly different things.

(18) Capturing perspectives with contexts.

(19) Granularity issues in question answering

(20-21) Database integration and the different standard kinds of inconsistencies that occur between databases.

3.6 Example 1: Fixing time implicitly to "Now"

Much of the information about the buyer is information that is true at the time of the purchase. The system is interested in where the user lives now, where he works now, etc. and not in where he was living or working ten years ago. There is a distinct bias towards "now" in our temporal reasoning and the first few examples are aimed at capturing this.

3.6.1 The bias towards Now

In a typical session, the user might make a sequence of statements such as the following. (See the appendix for details on the interface using which statements are made to the system. T_1 is the current time. In the following examples, the constant term "User" is used to refer to the person for whom the selection is being made.)[3]

(S1)	My budget is 20K	[holds(T_1 (= budget(User) 20k))]
(S2)	I am married	[hold(T_1 maritalStatus(User Married))]
(S3)	I live in Seattle	[holds(T_1 residesIn(User Seattle))]
⋮		

[3]A note on the convention used for labeling formulas. S1,S2, ...are statements made by the user, A1, A2, ...are axioms, and D1, D2, ...are conclusions drawn by the system.

The feature of this sequence of inputs that we are interested in, is the regular structure of the formulas, i.e., all of them are of the form holds(T_1 p) for different p. Since the temporal qualification of all these assertions is the same, we (and the system) might be able to leave it implicit.

At first sight, all this seems to be is a potential default which the interface can use. That is, instead of telling the interface holds(T_1 (= budget(User) 20k)), one simply tells the interface (= budget(User) 20k) and the interface adds the appropriate temporal qualification before asserting this into the knowledge base. However, we will now show that even the knowledge base can exploit this regularity to simplify inferencing.

As the above interaction shows, one of the pieces of information the user might provide is his marital status. If the person is married, the system can ask him more questions about his family, the requirements imposed by his family on selection of the car, who else in his family will be driving the car, etc.

3.6.2 Exploiting the bias towards *Now*

The HumanKinshipMt is a microtheory that contains a number of rules about human relations and familial organizations. In this example, we take a few axioms from this theory and study them with respect to simplifications in representation and inferencing from the standpoint of time.

(A1) If a person is married (at a time t_i), he/she has a spouse (at time t_i).
holds(t_i maritalStatus(x Married)) $\Rightarrow$ ($\exists$ y holds(t_i spouse(x y)))

(A2) A person usually lives (at time t_i) with his/her spouse (at time t_i).
holds(t_i spouse(x y)) $\rightarrow$ holds(t_i livesWith(x y))

(A3) A person's spouse (at time t_i) is of the opposite gender (at time t_i).
holds(t_i spouse(x y)) $\Rightarrow$ holds(t_i (= gender(x) oppositeGender(gender(x))))

[Note: I use the symbol "$\rightarrow$" to denote default implication. P(x) $\rightarrow$ Q(x) is a short form for $(P(x) \wedge \neg ab_i(x) \Rightarrow Q(x))$ where i is the number of the axiom.]

These rules, just like the input assertions given earlier, have the property that all the literals (such as spouse (x y) and livesWith(x y)) share the same temporal qualification. The temporal argument is the same throughout and serves purely to ensure that the system doesn't conclude something about the user having a spouse 10 years ago based on the fact that he is married today. If we take any time instant, the rule "a person usually lives with his/her spouse" still holds. The temporal qualification could simply be dropped (as the HumanKinshipMt does) and the rules rewritten as follows.

3.6.3 Simplified version

(A1a) If a person is married, he/she has a spouse.
maritalStatus(x Married) $\Rightarrow$ ($\exists$ y spouse(x y))

(A2a) A person usually lives with his/her spouse.
spouse(x y) $\rightarrow$ livesWith(x y)

(A3a) A person's spouse is of the opposite gender as the person.
spouse(x y) $\Rightarrow$ (= gender(x) oppositeGender(gender(x)))

When the user makes a statement such as "I am married", the system will try to determine who the user lives with and whether the users family will likely to be co-users of the car. Let us trace the inference path that would be taken to derive that the user lives with his/her spouse (using these simpler rules).

(S2)	maritalStatus(User Married)	Asserted by user
(D1)	spouse(User G001)	G001 is a skolemized term (from S1 and A1a)
(D1)	livesWith(User G001)	(from D1 and A2a)

3.6.4 Lifting to make time explicit again

This is of course nothing new. There have been many systems built without any notion of time or temporal representation. Clearly, there is a lot of common sense knowledge involving action, change, etc., that require temporal representations. The price of the increased complexity because of the introduction of time must be paid for representing action and change.

However, the introduction of time for dealing with one phenomenon should not complicate the representation of other phenomena that don't need an explicit concept of time. As representations become more expressive they also become more cumbersome and inefficient, i.e., inferencing with them becomes more time consuming. The expressiveness of the temporal (or other) representation used by a theory should not be dictated by the requirements of some other theory in the system.

3.6.5 Structuring the Representation based on Time

The description of any system (family, mechanical devices, geography, etc.) can be broken down into two parts. One includes the static constraints on the different parts of the system, i.e., the relations and constraints between the parts that can be found to hold in any "snapshot" of that system. For example, in a car, the seats are inside the body, the steering wheel is near the drivers seat, the seats have seat belts, etc. The other includes the actions/events in which the system might take part and change the state of the system. The first part of the description, being static (in the sense of describing relations that hold at any instant of time), can be done without involving any explicit notion of time.

3.6.6 Complications

However, the need to use portions of the static part of the description together with portions of the second part of the description (describing events and changes to the system) introduces some complexity. Let us go back to the HumanKinshipMt and study this in detail.

After a person gets married, their marital status changes from Single (or Divorced) to Married. If the system is told that a person got married a short while back, it should be able to deduce that the person now lives with someone. This inference involves using a rule about actions and change (the action of getting married changing the marital status of a person) and a static rule (about living with one's spouse). We have

(A4) If a person is the performer in a marriage, after that he is married.

$$\text{allInstanceOf(e Marriage)} \wedge \text{performer(e x)} \wedge \text{occursAt(e } t_i) \wedge \text{after}(t_i\ t_j) \Rightarrow \text{holds}(t_j \text{ maritalStatus(x Married))}$$

[allInstanceOf(x Marriage) is the Cyc representation of isa(x Marriage) / Marriage(x)]
after(t_i t_j) is true if t_j is the time instant immediately following t_i.

Given

(S4) allInstanceOf(G002 Marriage) $\wedge$ performer(G002 User) $\wedge$ occursAt(G002 t_1) $\wedge$ after(t_1 t_2)

i.e., G002 is the User's marriage and occurs at time t_1 and time t_2 is after t_1. From this we want the system to conclude that at time t_2, the User is living with someone, i.e.,

(D2) ($\exists$ y holds(t_2 livesWith(User y)))

Unfortunately, if the system uses (A2a) instead of (A2), it cannot derive this. This is because (A4) concludes holds(t_2 maritalStatus(User Married)) and the left hand side of (A2a) is maritalStatus(User Married) (without the holds). To conclude that the User is living with someone, the system needs (A2) whose left hand side includes holds(ti maritalStatus(User Married)).

We would like to obtain the simplicity of (A2a) and still be able to derive (D2). In order to derive (D2), the system has to make the temporal scope of (A2a) explicit to obtain (A2), i.e., the implicit temporal scope of (A2a) needs to be made explicit to get (A2). Why don't we just enter (A2) into the system in the first place? The difference between entering (A2a) and having it lifted to (A2) and entering (A2) in the first place is that (i) this lifting is now done automatically, and (ii) the lifting is done only when required and in many cases (A2a) itself is adequate and used. We now consider some of the lifting axioms required for this.

3.6.7 Temporal qualifications of contexts

Different contexts might use completely different temporal representations. The simplest temporal representation is to completely ignore time. Though a context might internally ignore time, from without that context, it might be possible to assign a blanket temporal scope to all the axioms in that context. There are different kinds of blanket temporal scopes we can associate with a context.

(a) The scope could be a constant as in the case with the information asserted about the user, i.e., the implicit temporal scope is "Now" (T_1). That is, if the user states P without explicitly specifying when P is true, it is assumed that he means P is true now.

(b) The scope could be unconstrained, i.e., the assertions in the context are true at any time instant. This is equivalent to putting a ($\forall\ t_i$ holds(t_i ...)) around the whole context. This is the scope associated with the HumanKinshipMt. If the user states P without specifying when P is true, it is assumed he means that P is always true.

(c) Between (a) and (b)above there is a wide range of possibilities. For example, the assertions in the context might be true only during some parts of the year, only in the twentieth century, only when the person is alive, etc.

In a later example we will consider an example of (c). Here we just consider (a) and (b). The KB has some contexts (such as the CSPSC) that ignore time by assuming implicit temporal scope as in (a). Some other contexts such as the HumanKinshipMt ignore time by assuming implicit temporal scope as in (b).

3.6.8 Lifting to make time explicit

Our goal here is to represent the assumptions (related to time) made by these contexts. The purpose of doing this is to enable the system to make the temporal scope of the assertions in these contexts explicit when they are lifted out into contexts which do not make the same temporal assumptions, i.e., to be able to go from (2a) to (2) as required. Let us first examine the behavior we want out of the system.
Given

(S5)	In HumanKinshipMt	spouse(x y) $\rightarrow$ livesWith(x y)
(S6)	In PSC_2	allInstanceOf(G002 Marriage), performer(G002 User), occursAt(G002 t_1), after(t_1 t_2)
(S7)	In CSPSC	(= gender(User) Male)

HumanKinshipMt and CSPSC ignore time. CSPSC implicitly temporally scopes assertions to be true at time T_1. PSC_2, a different problem solving context, has an explicit model of time.

We want,

(D3)	In PSC_2	holds(t_i spouse (x y)) $\rightarrow$ holds(t_i livesWith(x y)) (or holds(t_i spouse (x y) $\rightarrow$ livesWith(x y)))
(D4)	In PSC_2	holds(T_1 (= gender(User) Male))

The lifting rules (and assertions about CSPSC, PSC_2 and HumanKinshipMt) which do this are

(A5) timeFormalismUsedBy(CSPSC ImplicitFixedTime)
CSPSC implicitly fixes time to a constant

(A6) timeFormalismUsedBy(HumanKinshipMt ImplicitUniversalTime)
HumanKinshipMt implicitly quantifies time universally

(A7) timeFormalismUsedBy(PSC_2 DiscreteTime)
PSC_2 uses an explicit discrete time formalism.

(A8) timeFormalismUsedBy(c_i ImplicitFixedTime) $\vee$ timeFormalismUsedBy (c_i ImplicitUniversalTi
$\Rightarrow$ timeFormalismUsedBy(c_i ImplicitTime)
If the timeFormalismUsedBy is ImplicitFixedTime or ImplicitUniversalTime, it is also ImplicitTime.

(A9) timeFormalismUsedBy(c_i ImplicitFixedTime) $\wedge$
$\neg$(timeFormalismUsedBy (c_j ImplicitTime))
$\wedge$ ist(c_i p) $\wedge$ (= timeOf(c_i) t_i)
$\Rightarrow$ ist(c_j holds(t_i p))
If the time formalism use is ImplicitFixedTime and the time is fixed at t_i, wrap a holds(t_i ...) around the assertion being lifted.

(A10) timeFormalismUsedBy (c_i ImplicitUniversalTime) $\wedge$
$\neg$(timeFormalismUsedBy (c_j ImplicitTime)) $\wedge$ ist(c_i p)
$\Rightarrow$ ist(c_j ($\forall$ t_i holds(t_i p)))
If the time formalism used is ImplicitUniversalTime, wrap a holds(t_i ...) around the assertion being lifted and quantify t_i universally.

With these rules, the system can derive (D2) in PSC_2 and still leave time implicit in CSPSC.

3.7 Example 2: Structuring Theories using temporal properties

This example is isomorphic to the previous one and also illustrates the use of implicit time. The StaticAutoMt, which contains the static description of cars, the users relationship to the car, etc., uses an implicit universal model of time.

Though the information about actual models of cars, their prices, and features are in the database, some general facts which hold about cars are included in the StaticAutoMt and the AutoMt.

As mentioned earlier, the description of any system can be broken down into two parts - one with the static axioms (the axioms that do not need an explicit concept of time and are true at every time instant) and those that describe the actions and changes that might occur to that system and need an explicit model of time. The lifting axioms mentioned earlier provide for the lifting of the axioms from the static part of the description to the dynamic part of the description. However, those axioms were defaults and we need something stronger. If some axiom is true in the static part, it should always lift into the dynamic part. The function staticMt may be used to relate the static and dynamic mts that together constitute the description. We have the following assertions.

(A11) staticMt(AutoMt) = StaticAutoMt.

(A12) ist(staticMt(x) p) $\Leftrightarrow$ ist(x ($\forall$ t_i holds(t_i p)))
The assertions in the static part of the theory are lifted into the non-static part by wrapping a holds(t_i ...) around each assertion and quantifying the t_i universally.

One of the assertions in the StaticAutoMt is that the body style of LuxuryCoupes is always Notchback. This is stated in the StaticAutoMt as

(A13) allInstanceOf(x LuxuryCoupe) $\rightarrow$ bodyStyle(x Notchback) All luxury coupes have a Notchback body style

This is lifted into the AutoMt as,

(D4) ist(AutoMt ($\forall$ t_i holds(t_i allInstanceOf(x LuxuryCoupe)) $\rightarrow$
holds(t_i bodyStyle(x Notchback))))

Note that the lifted axiom is

ist(AutoMt ($\forall$ t_i holds(t_i allInstanceOf(x LuxuryCoupe)) $\rightarrow$
holds(t_i bodyStyle(x Notchback))))

and not

($\forall$ t_i ist(AutoMt holds(t_i allInstanceOf(x LuxuryCoupe)) $\rightarrow$
holds(t_i bodyStyle(x Notchback))))

The later would be wrong because the AutoMt's scope might be restricted in other ways. For example, the statements in the AutoMt hold only in the late twentieth century and so the time points covered by the $\forall$ within and without the scope of the ist could be radically different.

It should be pointed out that the Cyc system today does not use the scheme (based on the function StaticMt) outlined in this example. As the later examples will show, there are other schemes which have turned out to be preferable to this. However, for knowledge bases and applications with a narrower scope than Cyc, this scheme is perfectly ok.

3.8 Example 3: Making the implicitness of time a default

In the previous two examples, the assumption about the temporal qualification was rigid. We now consider a more flexible framework.

Though it is true that most of the statements made by the user will be concerned with the present time, occasionally, there may be statements he wishes to assert which are not about the present. For example he may be planning to move, or he may drive to Lake Tahoe every summer in his car, and these might have to be taken into consideration in selecting the car. We need to enrich the vocabulary of implicit time to enable these statements to be made. However, we should still be able to retain the compactness of statements made about the present.

3.8.1 Analysis of the origin of the simplification

In the scheme outlined in Examples 1 and 2, the problem solving context ignored time, i.e., the temporal qualification was left implicit. Stated alternately, the assumption was made that time was equal to the present time. Every assumption can be violated (that is why it is called an assumption). What we have here is a case of the assumption that time (associated with every assertion) equals the present time being violated. If the Problem Solving Context had an explicit model of time to begin with, this assumption could have been states as,

(A14) ist(CSPSC ($\forall t_i$ holds(t_i p) $\rightarrow$ ($= t_i\ T_1$)))

For any assertion, the default is that the assertion is true at time T_1.

However, we didn't stop at this. This assumption was exploited to simplify the representation to eliminate time (and the predicate "holds") from the vocabulary altogether. In the simplified vocabulary, it is no longer possible to state this assumption, which is why we had to capture this assumption as a lifting rule.

3.8.2 Contexts vs. Defaults

At this point, one of the distinctions between the context mechanism and the mechanism of default reasoning should be clear. While both allow an assertion to make assumptions, in default reasoning the assumption does not affect the vocabulary. It is presumed that exceptions to the assumption are statable. If we encounter exceptions that are not even statable, the mechanism of defaults cannot help us. While the system needs the mechanism of defaults to derive conclusions from default rules, we have to use contexts in order to exploit the assumptions for both simplifying the representation and for rescuing us when we encounter unstatable assumptions. When this happens, the assumption is stated in a more general context and the scope and applicability of the earlier context is restricted to situations where the assumption is satisfied.

3.8.3 First solution

Following this strategy, the simplest way out of the current problem would be to use another Problem Solving Context for statements that require an explicit concept of time. Let us call this context CSPSC and the context that left time implicit at T_1 CSPSC-T_1. The relation between CSPSC and CSPSC-T_1 is the following.

(A15) fixTime(CSPSC T_1) = CSPSC-T_1
(A16) ist(fixTime(c T_1) p) $\Leftrightarrow$ ist(c holds(T_1 p))
(A17) timeof(fixTime(x t_i) t_i)
The above two axioms are the definition of fixTime.

The function fixTime takes two arguments - a context c and a time point t_i and returns a new context which uses implicit fixed time at t_i. It contains all the assertions that are true in c which have a temporal scope of t_i. fixTime(c T_1) is a "projection" or "filter" of c with the time fixed at T_1.

Statements made about times other than the present or which require an explicit notion of time are made in CSPSC and those about the present time are made in CSPSC-T_1. Similarly, queries which require an explicit model of time are solved in CSPSC and those that do not are handled in CSPSC-T_1. In addition to CSPSC, there might be other such contexts derived from CSPSC using fixTime.

3.8.4 Refinement

However, it is both inconvenient and unreasonable to expect the user to direct his statements to the appropriate context. Therefore, the system will allow assertions with no explicit temporal scope to be made in CSPSC as well and let the temporal scope of these assertions default to the present, T_1. We have the schema

(A18) ist(CSPSC p $\rightarrow$ holds(T_1 p))
i.e., implicit time defaults to T_1.

To state that a context c_i has a default time t_i associated with it, we assert defaultTime(c_i t_i).

(A19) defaultTime(c_i t_i) $\wedge$ ist(c_i p) $\rightarrow$ ist(c_i holds(t_i p))

Now consider how some statements made to the program would be represented. All the assertions are made in CSPSC.

(S8) maritalStatus(user Married)
I am married.
(S9) residesIn(user Seattle)
I live in Seattle.
(S10) holds(1985 residesIn(user PaloAlto))
In 1985, I lived in Palo Alto.
(S11) ($\forall$ t_i allInstanceOf(t_i Summer) $\rightarrow$
($\exists$ e allInstanceOf(e Traveling)
$\wedge$ performer(e user) $\wedge$ destination(e PaloAlto)))
Every summer I go to Palo Alto.

Note that some statements have time implicit and some have time explicit.

3.8.5 Other temporal constraints

The phrase "Every summer" in (S11) was translated as ($\forall$ t_i allInstanceOf(t_i Summer) $\rightarrow$ ...). The formula is a literal translation of the english sentence and says that the traveling event takes place every summer. It is certainly not true that the user has been traveling to

Palo Alto every summer, e.g., he certainly couldn't have traveled to Palo Alto before he was born. So even in the case where time is explicitly specified, the specification is usually not complete and assumptions about time must be invoked to complete the specification. In this case, the system can reasonably assume that the temporal scope of the CSPSC is restricted to a few years in the vicinity of the present. In a later example we will consider the details of how this assumption may be stated.[4]

Now we consider the problem solving the behavior of the system when dealing with contexts involving default implicit time. As soon as an assertion is made in CSPSC, the system does the following.

- If it does not involve an explicit temporal qualification, the assertion is stored in CSPSC-T_1 (or more generally in the fix-time projection of the current context corresponding to the default time of the current context).

- If the assertion does involve explicit temporal qualification, and is of the form holds(T_i p), the system stores p in fixTime(CSPSC T_i).

This is equivalent to running the two lifting rules (A16) and (A19) in the forward direction. When a query is asked in CSPSC, if the query is of the form holds(T_i p), the system addresses the query p to fixTime(CSPSC t_i) and lifts the answer back to CSPSC.

3.8.6 Problems with this approach

It should be noted that the answer that is lifted back to CSPSC is only a default. There might be an interaction between the conclusions of the static and non-static parts of a theory and the predictions of these might contradict each other (when both sets of assertions involve defaults). As the following example shows, the conclusion from the context fixTime(CSPSC T_i) is based purely on the static rules and sometimes, the derivations of the static and non-static rules could contradict each other.

Assume the system is told that Car001 is a convertible. One of the static rules about convertibles is that they have cloth roofs. So in CSPSC the system contains the following (remember that CSPSC-T_1 is equal to fixTime(CSPSC T_1):
In CSPSC :

(A20) allInstanceOf(Car001 Convertible)
(A21) allInstanceOf(x Convertible) $\rightarrow$ ($\exists$ e allInstanceOf(e ClothCarRoof) $\wedge$ parts(x e))
Every convertible has a cloth roof.

Note that this is a default and has an implicit "ab" in it. Making the implicit ab explicit, the system concludes:

[4]Consider a pair of statements such as maritalStatus(user Married) and holds(T_1 maritalStatus(user Married)). Though both of these assertions use "maritalStatus", formally, the first uses it as a predicate and the second uses it as a function. Note that within a context, we are just using first order logic (with additions for defaults). So the unique readability theorem of first order logic holds and there is no ambiguity about the sense in which each occurrence of maritalStatus is used.

$$\text{allInstanceOf(x Convertible)} \wedge \neg\text{ab-A21(x)} \Rightarrow$$
$$(\exists \text{ e allInstanceOf(e ClothCarRoof)} \wedge \text{parts(x e))}$$

When we ask (in CSPSC)

(D5) $\quad$ holds(t_1 ($\exists$ e allInstanceOf(e ClothCarRoof) $\wedge$ parts(Car001 e))),

the query (allInstanceOf(e ClothCarRoof) and parts(Car001 e)) is addressed to CSPSC-T_1. The system enters CSPSC-T_1, derives this formula which it then lifts back into CSPSC. This is the desired behavior.

Now consider the following situation. The system is told that at time t_0, Car001 was in an accident which removed its cloth roof. So, we have an event (the accident) which causes a change in Car001. This information about the accident is recorded in CSPSC. The system can forward propagate information about the effects of the event from CSPSC to its projections (such as CSPSC-T_1) but this might prove expensive. If we know that queries to the system are going to be asked in CSPSC, the system can do the following.

When applying the default that convertibles have cloth roofs, the system assumed that Car001 was not abnormal, i.e., that ab_i(x) was not true. The problem in this situation is that ab_i is indeed true, though this cannot be derived purely from the formulas the system has already lifted into CSPSC-T_1. Given that we have not lifted all the information that might be true at time t_1 in CSPSC into CSPSC-T_1, the system has to exercise more caution in lifting conclusions back from CSPSC-T_1 to CSPSC. Therefore, the discharging of assumptions (such as ab_i(Car001)) will be postponed until later and performed in the more general context. So the conclusion we lift back into CSPSC-T_1 from CSPSC is not ($\exists$ e allInstanceOf(e ClothCarRoof)and parts(Car001 e)) but

(D6) $\quad$ $\neg$ab-A21(Car001) $\Rightarrow$ ($\exists$ e allInstanceOf(e ClothCarRoof) $\wedge$ parts(Car001 e))

Upon lifting, this becomes (in CSPSC)

(D7) $\quad$ holds(t_1 $\neg$ab-A21(Car001)) $\Rightarrow$
($\exists$ e holds(t_i allInstanceOf(e ClothCarRoof) $\wedge$ parts(Car001 e)))

Based on the information about the accident, the system has (in CSPSC)

(D8) $\quad$ $\neg$($\exists$ e holds(t_1 allInstanceOf(e ClothCarRoof) $\wedge$ parts(Car001 e)))

These two together imply holds(t_1 ab-A21(Car001)) and the system is blocked from drawing the wrong conclusion that Car001 still has its cloth roof. In general, when deriving conclusions in one context and lifting it to another, the assumptions made within the origin context may not be locally discharged, but must be accumulated and discharged later (in the target context).

3.8.7 Discussion on Problem Solving

The problem solving strategy explained here is different from the one explained in Example 1. In that earlier scheme, the axioms were lifted into the current context and inferencing was done within the current context. Here, instead of lifting all the required axioms into the current context and then performing inference on them, the query itself is handed to another context where the inferencing takes place. If the inferencing is successful, i.e., an answer is derived, this answer is then lifted back to the context where the question was posed.

The implementation of this scheme is significantly trickier than that of the scheme of Example 1. It is easiest to implement in a hybrid reasoning system, i.e., a system which has many problem solvers and uses many different reasoning techniques.

The strategy of switching contexts and solving the query is implemented in Cyc as a special problem solving method in itself. This problem solving method is different from other problem solving methods in that it cannot exist by itself: after the switch, it depends on the other problem solving methods to actually perform the inferencing. At some point, the system (usually the central "controller" which decides on the reasoning method to use), must decide whether to switch contexts. As of now, Cyc can perform this context switching only for a few kinds of queries (including the one described in this example) and these cases are hardwired into the code. It should be noted that even if the decision to switch contexts is the wrong one, the quality of the answer does not suffer. The system might take longer to derive the answer, but the final answer should be the same.

3.9 Example 4: Quantifying time universally as a default

The implicit time in the previous example defaulted to a constant. The other possibility is to assume that the temporal qualification defaults to universal quantification. Going back to some of the assertions in the HumanKinshipMt,

In HumanKinshipMt

(A2a) $\text{maritalStatus(x Married)} \Rightarrow (\exists\ \text{y spouse(x y)})$

(A4) $\text{allInstanceOf(e Marriage)} \land \text{performer(e y)} \land \text{occursAt}(\text{e } t_i) \land \text{after}(t_i\ t_j) \Rightarrow \text{holds}(t_j \text{ maritalStatus(y Married)})$

(A2a) is implicitly universally quantified and is equivalent to,

(A2) $(\forall\ t_i\ \text{holds}(t_i \text{ maritalStatus(x Married)}) \Rightarrow (\exists\ \text{y holds}(t_i \text{ spouse(x y)})))$

The contexts that contain general theories (and therefore explicitly mention very few spatio-temporal individuals), use a default universal quantification whereas the Problem Solving Contexts (into which general assertions are rarely made) might default the implicit time to a constant.

The important difference between this scheme and the one based on the function StaticMt is that here, the implicit time is a default and violations of this default are able to be stated. Hence it is no longer necessary to use two different contexts - one for the static part and one for the dynamic part of the theory. Both of these parts of the theory can be stated in the same context. This is the scheme currently employed by Cyc.

3.9.1 Lifting And Problem Solving

Lifting and problem solving proceed as in Example 3 with an interesting addition. Consider lifting a rule such as (A2a) from a context which uses DefaultUniversalTime to a context which uses ImplicitFixedTime. Because of the limited expressiveness of the later context, it cannot capture the universal quantification over time that is implicit in the assertion in the former context. Therefore the assertion cannot be lifted as it is. However, the system can reason in the former context to obtain a conclusion of the general assertion, which can then be lifted.

In HumanKinshipMt

(A2a) maritalStatus(x Married) $\Rightarrow$ ($\exists$ y spouse(x y))

Since this is a default universal quantification,

(A2) ($\forall$ t_i holds(t_i maritalStatus(x Married) $\Rightarrow$ ($\exists$ y spouse(x y))))

from which

(D9) holds(t_1 maritalStatus(x Married) $\Rightarrow$ ($\exists$ y spouse(x y)))

which can be lifted into a context such as CSPSC-T_1 to give

(D10) In CSPSC-T_1 : maritalStatus(x Married) $\Rightarrow$ ($\exists$ y spouse(x y))

It is interesting to note that the eventual formula the system obtains in CSPSC-T_1 is exactly the same as the formula it began with in the HumanKinshipMt. However, it should be noted that the two occurrences of this formula mean different things!

There is an alternate interpretation for assertions such as (A2) in the HumanKinshipMt. Rather than associating temporal information with assertions and leaving this temporal information implicit, an alternative would be to associate temporal information with objects. Associating temporal information with objects provides a more expressive language and leads to the formalism of "SubAbstractions" which is discussed in detail in [cyc book]. Since the problem solving and lifting behavior is the same under both interpretations, we shall not go into the details of the SubAbstraction formalism here.

3.10 Example 5: Intermediate implicit temporal constraints

In the previous examples, we considered the cases where time was left implicit and defaulted to either a constant or to a universal quantification. Now, in the last two of the examples regarding time, we consider intermediates between these two defaults. The general form of the lifting rule for making the implicit time explicit has been the following.

$$\text{(A22)}\quad \mathrm{ist}(\mathrm{c}_1\ \mathrm{p}) \wedge \mathrm{r}(\mathrm{c}_1, \mathrm{c}_2, \mathrm{p}) \Rightarrow \\ (\mathrm{ist}(\mathrm{c}_2\ (\forall\ t_i\ \mathrm{Q}(t_i) \Rightarrow \mathrm{holds}(t_i\ \mathrm{p}))))$$

In the case of the implicit time defaulting to a constant (say T_1), Q is $(= t_i\ T_1)$ and in the case of time defaulting to the universal quantifier, Q is the constant True. In between these two extremes, is a wide range of possibilities. Many of these are quite useful and raise interesting issues.

3.10.1 Weather and the Car

One of the factors affecting which features one wants in a car is the climate of the area in which one lives. For example, if the user lives in an area where it snows or rains a lot, he might want to consider antilock brakes and avoid convertibles. If it is very foggy, he might want fog lights. In order to make suggestions based on the weather in a given area of the country, the system needs to have information about the kinds of weather in different places during the different seasons as well as the requirements imposed by each of these types of weather on the car. Let us therefore consider the representation of the weather during winter - first in english and then in CycL.

3.10.2 English vs. Logical Formulas

English:

During winter, it is snowy in most of the North and North East (here we are only considering the United States). In the North West, it is rainy and foggy. In the San Fransico Bay area, there is usually no snow or rain ...

CycL:

$$\text{(A23)}\quad \mathrm{during}(t_i\ I_i) \wedge \mathrm{allInstanceOf}(I_i\ \mathrm{Winter}) \rightarrow \\ \mathrm{holds}(t_i\ \mathrm{weather}(\mathrm{NorthEast}(\mathrm{USA})\ \mathrm{Snowy}))$$

$$\text{(A24)}\quad \mathrm{during}(t_i\ I_i) \wedge \mathrm{allInstanceOf}(I_i\ \mathrm{Winter}) \rightarrow \\ \mathrm{holds}(t_i\ \mathrm{weather}(\mathrm{NorthWest}(\mathrm{USA})\ \mathrm{Rainy}))$$

$$\text{(A25)}\quad \mathrm{during}(t_i\ I_i) \wedge \mathrm{allInstanceOf}(I_i\ \mathrm{Winter}) \rightarrow \\ \mathrm{holds}(t_i\ \mathrm{weather}(\mathrm{NorthWest}(\mathrm{USA})\ \mathrm{Foggy}))$$

$$\text{(A26)}\quad \mathrm{during}(t_i\ I_i) \wedge \mathrm{allInstanceOf}(I_i\ \mathrm{Winter}) \rightarrow \\ \neg\mathrm{holds}(t_i\ \mathrm{weather}(\mathrm{BayArea}\ \mathrm{Rainy}))$$

$$\text{(A27)}\quad \mathrm{during}(t_i\ I_i) \wedge \mathrm{allInstanceOf}(I_i\ \mathrm{Winter}) \rightarrow$$

$$\neg \text{holds}(t_i \text{ weather(BayArea Snowy)})$$

$\vdots$

The most striking characteristic of the above axioms is that they share the same left hand side. All of them have the structure

<the time is during winter> $\rightarrow$ <some attribute about the weather>

It is also interesting to note the difference between the english description and the CycL description. Unlike the CycL description, the english description does not repeat the phrase "during winter". The first time it is made, it has the effect of setting up the context for the remaining statements. The english description exploited the structure of the theory to set up the context using which the description was made compact.

There are many descriptions/theories with this kind of structure. All of them start with "during X" and then proceed to give a description of what happens when X is occurring. The X could range from "winter" to "a revolution" or "an election" or "the operation of a blender".

Let us now try to simulate the structure of the english description using the context mechanism. We use the context WinterMt, which assumes that it is winter.

Enter WinterMt
weather(NorthEast(USA) Snowy)
weather(NorthWest(USA) Rainy)
weather(NorthWest(USA) Foggy)
¬weather(BayArea Snowy)
¬weather(BayArea Rainy)
Exit WinterMt

The operation "Enter WinterMt" puts the system into the WinterMt and has the same effect as the phrase "during winter" in the first sentence in the english description. After that, interactions (assertions and queries) are in the WinterMt. The above exchange in equivalent to asserting.

(A28) ist(WinterMt weather(NorthEast(USA) Snowy) $\wedge$
weather(NorthWest(USA) Rainy) $\wedge$
weather(NorthWest(USA) Foggy) $\wedge$
¬weather(BayArea Snowy) $\wedge$
¬weather(BayArea Rainy)

Having obtained the compaction, let us address the next two questions, (a) how to make the assumptions behind this context explicit, and (b) how to go through the whole inference process without having to make the assumptions explicit.

3.10.3 Making the assumptions explicit

The assumption that the time is winter must be made explicit when an assertion is lifted from this context into one which does not make this assumption, i.e. includes times when it is not winter. The following lifting rule specifies this. (Note that in order to make the assumption explicit, the target context should not leave time implicit.)

(A29) ist(WinterMt p) $\wedge$ not temporalFormalismUsed(ci ImplicitTime) $\rightarrow$
ist(c_i ($\forall$ t_i during(t_i I_i) $\wedge$
allInstanceOf(I_i Winter) $\rightarrow$ holds(t_i p)))

If p is true in the WinterMt, and c_i does not use ImplicitTime, then when lifted p into c_i, convert p to holds(t_i p) and constrain t_i to be during winter.

3.10.4 Exceptions

If the target context also makes the assumption that it is winter, or implicitly fixes the time to be a constant which is during winter, no modification is required to the assertion. If the target context makes some other assumption which makes (during (t_i I_i) $\wedge$ allInstanceOf(I_i Winter)) ill formed, i.e., it cannot express the assumption that the time is winter, then the assertion cannot be lifted into that context. This is of course not a complete list of cases where the assertion cannot be lifted into the target context. It is possible that the target context cannot express p, p could be contradictory with the theory already in the target context, etc.

3.10.5 Generalizing

The lifting rule can be generalized as follows. The WinterMt is part of a more general theory of Weather whose other parts include the SummerMt, FallMt, etc. The WinterMt, FallMt and SummerMt can be seen as "extracts" of this larger context. As explained earlier, there are a good number of theories with the structure "during X <a description of Y>" which are parts of larger theories about Y. Some examples of the larger theories with "extracts" include

- Y = Acceptable social behavior and X = elections, revolution, natural disaster
- Y = Social gathering and X = the beginning, guests departing
- Y = A show (movie, theatre) and X = feature presentation

In an analogy to the function fixTime mentioned in the first example, we introduce the function fixTimeType so that the context c_1 which describes Y during X is related to the context c_0 that contains the more general theory of Y by

(A30) fixTimeType(c_0 X) = c_1

The lifting rule can be written more generally as,

(A31) ist(fixTimeType(c_i X) p) $\Leftrightarrow$
ist(c_i ($\forall$ t_i during(t_i I_i) $\wedge$ allInstanceOf(I_i X) $\Rightarrow$ p))
The definition of fixTimeType.

3.10.6 Inferencing without lifting

Now we consider the situations where the inference may be performed directly on the smaller representation. To begin with, let us first look at some rules for predicting desirable features based on the weather. These assertions are all in the WinterMt.

(A32) weather(x Foggy) $\wedge$ livesIn(p x) $\wedge$ usesCar(p y) $\rightarrow$ hasFeatures(y FogLights)
If its foggy, use fog lights
(A33) weather(x Snowy) $\wedge$ livesIn(p x) $\wedge$ usesCar(p y) $\rightarrow$ hasFeatures(y AntiLockBrakes)
If its snowy, use antilock brakes

Now given that the user lives in Seattle, the system should suggest that the car have FogLights.
Given in CSPSC

(S12) livesIn(user Seattle)
(S13) usesCar(user Car001)

the system should derive hasFeatures(Car001 FogLights) (in CSPSC.)

3.10.7 Long inference path

First derive
(D11) during(t_i I_i) $\wedge$ allInstanceOf(I_i Winter) $\rightarrow$ holds(t_i livesIn(user Seattle))
(D12) during(t_i I_i) $\wedge$ allInstanceOf(I_i Winter) $\rightarrow$ holds(t_i usesCar(user Car001))

Lifted from WinterMt:
(D13) during(t_i I_i) $\wedge$ allInstanceOf(I_i Winter) $\rightarrow$
holds(t_i weather(Seattle Foggy)))
(D14) during during(t_i I_i) $\wedge$ allInstanceOf(I_i Winter) $\rightarrow$
holds(t_i weather(x Foggy) $\wedge$ livesIn(p x) and
usesCar(p y) $\rightarrow$ hasFeatures(y FogLights))

These axioms allow us to conclude that if it is foggy in winter, the car should have fog lights in winter.

Note that the system still hasn't concluded that the car should have fog lights at the time of the sale - only that if it is winter it should have fog lights. After all it is possible that the user might attach fog lights to his car at the beginning of every winter and take them off in spring. However, for the purpose of this application, we ignore this possibility and assume that if the car needs them for a significant part of the year, it should have them now (i.e., at the time of the purchase). The following rule says this.

(A34) (allInstanceOf(I_i Season) $\rightarrow$ (during(t_i I_i) $\rightarrow$ holds(t_i hasFeatures(x y))))
$\rightarrow$ hasFeatures(x y)
If a feature is required throughout a season, then the car should
have that feature.

This rule enables the system to conclude if the user lives in Seattle, the car should have fog lights. i.e. hasFeatures(Car001 FogLights).

The inference path outlined above is an excellent example of the kind of inference path we would like the system to avoid. It took a fairly compact representation in the WinterMt and CSPSC, expanded it considerably (D12-D14), and finally arrived at a conclusion which was again very compact. Many of the steps in the inference were only involved with decontextualizing the given formulas, only to contextualize them again. Our goal is to enable as much of the inference to be performed without having to do extensive decontextualization.

3.10.8 Simpler inference path

The following is an alternative problem solving strategy for the above inference. The goal is that given a query, the system should use a problem solving context which makes the set of assumptions that minimizes the total effort (lifting + inferencing within the context) required. The system uses the context fixTimeType(CSPSC Winter) ($CSPSC_W$) as the context for solving this query.
In $CSPSC_W$

From CSPSC:

(S12) livesIn(user Seattle)
(S13) uses(user Car001)

From WinterMt:

(A32) weather(x Foggy) $\wedge$ livesIn(p x) $\wedge$ usesCar(p y) $\rightarrow$ hasFeatures(y FogLights)

If its foggy where the user lives, then his car should have FogLights.

(A35) weather(Seattle Foggy)

From which the system gets, in $CSPSC_W$

(D15) hasFeatures(Car001 FogLights).

Now, if the car is expected to have a feature for a significant period of time, the system can assume that it will have it at the time of the sale. This is written as a lifting rule.

(A36) ist(fixTimeType(c x) hasFeatures(car feature)) $\wedge$ ist(c allInstanceOf(x Season))
$\rightarrow$ ist(c hasFeatures(car feature))

Using this rule, the system can go directly from the conclusion hasFeatures(Car001 FogLights) in $CSPSC_W$ to the same conclusion in CSPSC.

3.10.9 Discussion of shorter proof

The inference path shown above is significantly less complex, both in the lifting and in the assertions with which it dealt. The assumption that the implicit time was winter is the one that enabled this simplification. Given that WinterMt itself makes this simplification, why couldn't the system have done this inference in WinterMt itself?

The problem with performing the inference in WinterMt is as follows. To conclude that the user needs foglights, the system needs the assertion livesIn(user Seattle). However, this formula in the WinterMt would mean something different from what it means in CSPSC$_W$. Note that CSPSC$_W$ is a projection (during winter) of the contents of CSPSC, which is restricted to the present few years. On the other hand, the temporal scope of WinterMt is over many thousands of years. So, while the assertion livesIn(User Seattle) in CSPSC$_W$ means that for the present few years, the user lives in Seattle during winter, the same formula in WinterMt would mean that the user lives in Seattle during winter for over thousands of years - and this is certainly not what we intend. The system needs to retain and add to the assumptions made by CSPSC, and therefore uses a projection of CSPSC, namely CSPSC$_W$ to perform this inference.

The two inference paths shown in this example illustrate the two different approaches - of lifting all the required assertions into the current context and deriving the answer from these (in the current context) and of switching to an appropriate context, performing the inference within that context and lifting the answer back. The second strategy sometimes leads to much more efficient inferences, but involves the overhead of deciding which context to switch to and when. If lifting the relevant axioms into the current context does not involve much work (as in Examples 1-3), the former strategy is preferred. If the lifting involves significant modifications to the axioms, i.e., involves significant decontextualization, then the later strategy is preferred.

3.11 Example 6: Kludging the temporal aspects of a predicate

Towards the end of the previous example, we showed how the inference leading to the conclusion that the car should have fog lights could be simplified. In this example, we show another means of doing the same thing, only this time, the aim is not to explain a desirable or recommended technique for drawing this conclusion. Rather, the aim is to show how a theory which tends to use a representation that is sloppy but sufficient for its purpose, can be incorporated into a larger framework.

3.11.1 Mixing Time into Weather

There are two different senses in which the predicate "weather" could be used. It could be used in the sense used in the previous example to say that at a particular time (implicit or explicit), the weather of the place is such and such. Alternately it could be used in the sense in which it is used in the following sentences.

"One needs all kind of clothing in Boston. Since the weather is snowy, one needs heavy clothing. Since it (the weather) is also hot and humid, one also needs light clothing"

What is meant by the "the weather is hot and humid, and the weather is snowy" is that for some significant fraction of the time it is hot and humid and for some significant portion of the time it is snowy. This can be literally translated into the representation system as:

In CSPSC:
(A37) weather(Boston Snowy)
(A38) weather(Boston Hot)
(A39) weather(Boston Humid)

This implicitly means that for a significant portion of the year, it is hot, for a significant portion of the year it is Snowy, etc.

Though this is extremely sloppy, this is exactly the sense of "weather" the system needs for this particular application. If the place where the user lives is foggy for a significant part of the year, that needs to be taken into account in the selection of the car, irrespective of exactly which time of the year it is foggy. If the system were to use this sense of weather (Cyc does not do this currently), the lifting rules to obtain these assertions would be,

(A40) ist(WinterMt weather(x y)) $\rightarrow$ ist(CSPSC weather(x y))

The assertion

(A32) weather(x Foggy) $\wedge$ livesIn(p x) $\wedge$ usesCar(p y) $\rightarrow$ hasFeatures(y FogLights)

from the AutoMt (which would use weather in the same sloppy sense) could get lifted without change into CSPSC.

3.11.2 Problems with this approach

These two assertions together with the assertion livesIn(User Seattle) allow the system to conclude hasFeatures(Car001 FogLights). Though this use of weather is adequate for this example, it can very quickly lead to problems.

For example, consider the statement "if the user lives in a place that is hot and damp, he is unlikely to choose a convertible". We translate "the place is hot and damp" as livesIn(user x) and weather(x hot) and weather(x damp). However, this is true even if the place is hot during some portion of the year and damp during a different portion of the year. All it means when we say weather(x c) is that during a significant part of the year, the weather of x is c. We can get around this problem by changing our representation of "hot and damp" to say something like weather(x HotAndDamp). While this scheme works in this case, its sloppiness makes the representation increasingly cumbersome after a certain point.

Though the application does not use this strategy, this example illustrates how systems that use predicates in such a sloppy fashion could be integrated into larger knowledge bases.

3.11.3 Extensions to non-temporal modalities

The last 6 examples all dealt with various forms of simplifications to time. While the simplifications given above don't appear very significant by themselves, it should be noted these examples are mainly expository in nature. Isomorphic simplifications can be made with modalities such as beliefs. Significant savings will be accrued by combining two or more of these simplifications. We now outline how the approach explained above could be used elsewhere.

Consider the problem of specifying how people behave in various situations. In particular, let us focus on the behavior of adults in office settings.

It is clear that there are a number of constraints on what a person can and can't do at work. E.g., no loud noises, nothing indecent, he has to spend most of the time working, etc. These constraints arise out of an agreement/ contract the worker has with the employer. The employer and the employee enter into an agreement at the time of the employment which specifies what each of them will do, the consequences of reneging on part or all of the agreement, etc. This agreement is an important part of the conduct of a person at the workplace.

Similar agreements govern people behavior in other situations. Sometimes this agreement is "official" and sometimes it is not.

Consider the constraint that the worker wears decent clothes. The outline of how this would be represented in terms of agreements is as follows.

$$\begin{aligned}&\text{employer(x y)} \Rightarrow (\exists\ \text{a allInstanceOf(a WorkPlaceAgreement)} \wedge \\ &\qquad\qquad\qquad\qquad \text{agreesTo(a x)} \wedge \text{agreesTo(y z))}\end{aligned}$$

for every employer employee pair, there is a workplace agreement they agree to.

$$\begin{aligned}&\text{allInstanceOf(a WorkPlaceAgreement)} \Rightarrow \\ &\qquad \text{(specifies(a allInstanceOf(x Worker)} \wedge \text{holds(ti inside(x XYZCompanyPremises))} \\ &\qquad\qquad\qquad \wedge\ \text{holds(ti wears(x c))} \Rightarrow \text{holds(ti allInstanceOf(c DecentClothing))))}\end{aligned}$$

Let us write the right hand side of this as, specifies(a P). The inference path in concluding that Fred is wearing decent clothes when he is at work is then as follows:

(a) There is an agreement between Fred and Fred's employee.

(b) This agreement states that Fred should wear decent clothes.

(c) Fred usually follows the agreements he makes.

(d) Therefore Fred must be wearing decent clothes.

While this is a very satisfying proof in that it is very "first principled", there is something discomforting about it. Certainly, one should not have to invoke the concept of agreement, etc., in order to conclude that a person must be decently dressed if he is at work.

The naive inference path "People wear decent clothes at work. If Fred is at work, he must therefore be wearing decent clothes" should be adequate to derive this conclusion.

The standard AI answer to this problem is that the naive path will be obtained from the more first principled path by a compilation procedure (such as explanation based learning (EBL)). There are two problems with this stock answer. The first (which has been raised previously) is that in practice this does not work. EBL and its relatives, in practice, only slow a system down [Steve Minton]. The second problem, the one with which we will be concerned here is as follows.

The learning process seems to occur the other way around with humans. That is, we first learn the simple theory - that people wear decent clothes to work - and then later learn that this and other aspects of a person's behavior at work is because of an agreement. It would clearly be desirable if we could first state the simple theory and then add some assertions which explained that the statements of this theory are really statements of an agreement which has certain properties. This can be done as follows.

The context WorkPlaceBehaviorMt has statements like

wears(x c) $\rightarrow$ allInstanceOf(c DecentClothing).

We then qualify the statements in this context with the following.

(a) The WorkPlaceBehaviorMt assumes that the individuals involved are rational adult workers in the workplace.

(b) For every employee employer pair, there exists an agreement which includes the statements of the WorkPlaceBehaviorMt as the statements agreed upon.

These two qualifications are lifting rules isomorphic to many of the lifting rules given in the last 6 examples.

The interesting point to note is that for most purposes, (b) does not need to be invoked at all. The contents of WorkPlaceBehaviorMt can be directly used to derive the requisite inferences. This approach of stating the simpler theory and then explaining the origins (or underlying principles) behind the theory is easier both from the perspective of doing the representation and from the perspective of doing inference.

3.12 Example 7: Hypothetical Reasoning

As mentioned at the beginning of the chapter, in the car selection process we go about building a model of the car the user wants. We do this by determining what features the car should ideally have, given the users situation and needs (assuming a desire for safety, comfort, economy, etc.). Whenever the system concludes that the car should have a particular feature, it poses a question to the user asking him to verify this.

However, rules mentioned earlier for determining car features seem to go one step further than suggesting features: they conclude that the car does indeed have the features. For example, the rule suggesting fog lights states that

(A32) weather(x Foggy) $\wedge$ livesIn(p x) $\wedge$ usesCar(p y) $\rightarrow$ hasFeatures(y FogLights)

These rules assume that the person does have the car! All he is doing is thinking of buying one! They also conclude that the car does have the features. All the system wants to imply is that under ideal conditions, the user might desire to have these features in the car.

What usesCar(person car) really means is that the person will be using the car if he buys it. Similarly, when the system says hasFeatures(car feature), it means that the user of the car desires the car to have the feature.

For many problems involving deciding what kind of X to buy (or more generally, what kind of X to select), this strategy of using a hypothetical instance of X and concluding these desired features of that X seems useful. In fact it is not just in this situation that this sort hypothetical reasoning is useful. Some other problems for which it is useful to be able to reason with hypothetical situations include planning, designing, etc.

3.12.1 Complications

There are some complications that come with hypothetical objects and situations. For example, if the system were in some other context (unrelated to the car selection problem) and had to determine which car the user has (to find out whether he could transport the neighbors kids to school), the system should not answer that the person uses the car that was instantiated in CSPSC. So the system has to know that the car instantiated in CSPSC isn't a real one and that the user does not really have the car and the car might not even really exist.

On the other hand, the car can't be completely ignored outside of CSPSC. If the system is asked what features the person would desire in a car that he used, it should be able to use the conclusions in the car selection PSC to answer this (even though he might end up buying a car without those features). What this means is that the modality of assertions and objects could change across contexts.

3.12.2 Contexts for *Selecting An X*

Since this strategy of considering a hypothetical X for determining which X the user should purchase is a fairly standard one, we first introduce a class of contexts called MajorPurchaseSelectionPSC, which is the class of Problem Solving Contexts where we deal with selecting objects to buy. Let us first consider some of the properties of this context and then proceed to write the lifting rules for changing the modality of assertions across contexts.

A part of the description of a MajorPurchaseSelectionPSC (such as CSPSC), the system will be given the following.

(a) The kind of object that is being selected. For the present, we assume that this is specified as a category. Since CSPSC involves selecting a car, we have selectionObjectType(CSPSC) = Car. The actual object (the instance of Car) that is used for building up the model of the ideal car (for the user) is equal to selectionObject(CSPSC).

(b) The person for whom the selection is being done. The system distinguishes between the person who is doing the selection and the person who will be using the object. So, if the user of the car in CSPSC is going to be Fred, we write, selectionFor(CSPSC) = Fred.

In addition, the system will know the time, place, etc. where the selection takes place.

3.12.3 Lifting from Hypothetical Contexts

Now we consider the issue of lifting assertions from these MajorPurchaseSelectionPSCs to other contexts. There are two major issues involved.

First, if we "gensym" the object Car001 to be the selection object for CSPSC, then Car001 should only be in the domain of CSPSC and its derived contexts (contexts such as fixTime(CSPSC t_2) and fixTimeType(CSPSC Winter)). If the system were to consider an unrelated context, such as one in which the user is determining how to get his child to school, Car001 should not be one of his possessions in this other context. We do this by specifying that the selection object of a context is not present in the domain of any other context (with the exception of the derived contexts). Also, if a context $CSPSC_X$ is derived from CSPSC, the selectionObject of $CSPSC_X$ should be the same as that of CSPSC. The following rules state this.

Below are some rules related to the predicate derivedContext. Contexts derived from c_i using fixTime and fixTimeType are derived contexts of c_i.

(A41) $(\forall\ c_i\ t_i\ \text{derivedContexts}(c_i\ \text{fixTime}(c_i\ t_i)))$
(A42) $(\forall\ c_i\ x\ \text{derivedContexts}(c_i\ \text{fixTimeType}(c_i\ x)))$

The propagation of selectionObject to derivedContexts

(A43) $\text{derivedContexts}(c_i\ c_j) \wedge \text{selectionObject}(c_i\ o) \Rightarrow \text{selectionObject}(c_j\ o)$

Finally, we restrict the contexts which include the selection object in their domain.

(A44) $\text{selectionObject}(c_i\ o) \wedge (\neg\text{derivedContext}(c_i\ c_j) \vee \neg(=\ c_i\ c_j))$
$\rightarrow \neg\text{presentIn}(c_i\ o)$

If o is the selectionObject of c_i, and c_j is not derived from c_i, then o is not present in c_j.

3.12.4 Using conclusions from Hypothetical contexts in other contexts

In CSPSC
(S14) uses(User Car001)
(S15) allInstanceOf(Car001 Car), etc.

Now if we were to go to another context c_1 and ask if the user makes use of Car001, the system will try to lift this assertion from CSPSC. However, since the system has $\neg$presentIn(c_1 Car001), this lifting will fail and the system will not give the incorrect answer (that the user uses Car001).

It should be noted that the selection object does exist in the outer context (the context that is outer to both c_1 and CSPSC) and the one in which we state the lifting rules. However, in the outer context, it is just an abstract object and is not a car or owned by the user.

Second, let us suppose that at the end of a session, the system has determined the features that the user would like to have in the car, e.g., antilock brakes. In CSPSC the system will have hasFeatures(CSPSC AntilockBrakes). Now consider another context c_1, unrelated to CSPSC. The system should be able to exploit the information acquired in CSPSC to answer the query "what features would the user like in his car" in a different context.

The query in c_1 is

(D16) desires(User uses(User y) $\wedge$ allInstanceOf(y Car) $\rightarrow$ hasFeatures(y ?feature))

where the system has to find bindings for ?feature. The following lifting rule takes care of the lifting from CSPSC to c_1.

(A45) ist(c_i hasFeatures(car feature) $\wedge$ uses(person car)) $\wedge$
selectionObject(c_i car) $\wedge$ selectionObjectType(c_i C) $\rightarrow$
($\forall$ c_j ist(c_j desires(user uses(x y) and allInstanceOf(y C)
$\rightarrow$ hasFeatures(car feature))))

Using this rule, the system derives (in c_1) that the user would like a for car that he owns to have antilock brakes (from the conclusion in CSPSC that the car we gensymed has antilock brakes).

3.12.5 Problem solving strategies specialized for contexts

The important categories of Problem Solving Contexts, such as MajorPurchaseSelectionPSC and DiscourseContexts, may have problem solving strategies associated with them. The major goal in a MajorPurchaseSelectionPSC is to determine features the selection object should have. When the system is in one of these contexts, its behavior is in accordance with this goal. Some of the actions triggered when the system is in one of these contexts include the following.

(a) As soon as any new assertion is obtained, it performs the following:

- **(i)** directed forward propagation, i.e., forward propagation aimed at concluding new features for the selection object, and
- **(ii)** after the propagation following the assertion is complete, if any new information about the car has been obtained, the candidate list of possible makes and models is updated.

(b) Every derivation about the features the car has is nothing more than an argument for the car having that feature. This argument could be blocked if the user were to decide for some reason that he does not want that feature. So a question is posted to the user to verify that he indeed wants the feature.

3.13 Example 8: Domain assumptions

Before we go much further, we should also state some of the basic assumptions made by the car selection program.

(a) It assumes that the purchase is taking place in the late twentieth century and in the United States. If the selection were taking place in 1960, then it would be inappropriate to use this program.

(b) It assumes that the user intends to use the car for normal use, i.e., transportation. If the buyer were thinking of using the car for "monster truck shows" (where it would not be desirable for the car to be very strong or very small and issues such as fuel economy, comfort, acceleration, etc. don't matter at all), the predictions made by the theory would not be useful.

(c) It assumes that the user best knows his own needs. So, even if all the situational parameters suggest that the user not get a convertible (he lives in Seattle where it rains a lot), if the user specifies that he wants a convertible, then the user's specification will be taken as a constraint on the car.

(d) It assumes that the user accurately knows the details about himself - his finances, where he lives, etc.

(e) The user has normal human desires such as self-preservation, desire for comfort, etc.

These are all assumptions made by most of the theories involved - the AutoMt, the weather related contexts, HumanKinshipMt, ... Though the program currently can't offer any advice in situations when these assumptions are not satisfied, it is still important to make these assumptions explicit so the system knows when it is dealing with a situation beyond its sphere of competence. This becomes especially important if the knowledge based system (KBS) is just part of a larger system. If the problem is outside the scope of the KBS, some other part of the system might have a shot at the problem and it is important for the KBS to know what its domain of competence is. In any case, not giving any answer is certainly better than giving incorrect answers.

3.13.1 Kinds of assumptions

Up to this point, we have been using the term "assumption" rather vaguely. Now we will distinguish between some of the basic kinds of assumptions and their effect on a theory.

A context as a whole might assume that the objects it applies to satisfy some assumption. If that assumption were not true of some object, none of the assertions in that context would be applicable to that object. These are usually very fundamental assumptions that are rarely stated explicitly (in human communications). They are so fundamental that most theories would undergo radical changes if these assumptions were violated.

For example, if the user wanted the car for car safety advertisements (involving crashing the car), very little of the AutoMt would be relevant or make sense for him. As another example, consider a theory about the structure and functioning of business enterprises. It assumes that the business will try to achieve its goals. If a particular business does not satisfy this assumption, little or none of the theory will hold for this business.

Some set of assertions – this could even be all of the assertions – in the context make some assumptions that cannot be stated using the vocabulary of that context. Though some set of objects might violate these assumptions, the other assertions in that context would still be applicable to those objects. E.g., at a particular instant in time, the user might not be aware of his financial position. This should not preclude the relevant theories from being applicable to him at other times (when he does know his financial situation).

All the examples cited at the beginning of this section belong to the first category. In the next example, we will consider the details of the second category. We now discuss the strategy followed for stating the first category of assumptions.

3.13.2 The *Scope* of a theory

What does it mean to say that a certain user is "outside the scope of the theory" because he violates the assumption? In order to make this notion of "outside the scope" more precise, we need to be more precise about the notion of "the scope of the context".

The scope of the context is the set of objects over which its predictions hold. If we consider an assertion ($\forall$ x p(x)) in the AutoMt, the set of objects with which this x can be instantiated defines the scope of the context, i.e., the domain of the context. So when we say that an object is outside the scope of a context, what we mean is that it is not an element of the domain of that context. The scope of the context is the set of objects denoted by the symbol "$\forall$" in that context. Using our vocabulary, if Fred is outside the scope of the AutoMt, then $\neg$presentIn(AutoMt Fred) is true.

When an assertion such as ($\forall$ x p(x)) is lifted from the AutoMt to a Problem Solving Context, the system has two possibilities - the problem solving context (PSC) also makes the assumption or it does not. If it also makes the assumption, then the system doesn't have to worry about this assumption during lifting. If some object not satisfying the assertion is introduced into the context, this is a normal contradiction and the appropriate action is taken (the user is notified, etc.). If the Problem Solving Context does not make the assumption, then the formula has to be changed to account for this difference in assumptions.

Earlier we pointed out how a formula might change when lifted from one context to another (to preserve the meaning of the formula). In those examples, when dealing with formulas such as ($\forall$ x p(x)) (where p could be a complicated expression itself), we were primarily concerned with changes to p(x) - adding extra arguments, conjuncts, etc. We were not really concerned about the changes to the meaning of the "$\forall$". We now consider the changes to the $\forall$ during lifting.

3.13.3 Scope and Lifting

As we said above, the concept of the "scope" of a theory is intricately tied to the denotation of the "$\forall$" symbol in that theory. When the system lifts an assertion from the AutoMt to another context which does not make all the assumptions the AutoMt makes, we are moving to a context which has a different scope (possibly larger, since it does not make all the assumptions the origin context makes), i.e., the meaning of the symbol "$\forall$" is different between the two contexts. To preserve the meaning of the assertion being lifted, some changes must be made to it.

Consider the simplest case where the only assumption made by the origin context (which is not made by the target context) is that a particular individual (say Fred) is not the user (Fred has some very bizarre tastes and none of the AutoMts predictions apply to Fred). However, the Problem Solving Context does not make this assumption (it is happy dealing with Fred). The formula under consideration is:

(A46) $(\forall$ x primaryCar(x car) $\wedge$ children(x y) $\rightarrow$ hasFeatures(car FourDoorBody))

i.e., if it is the primary car and he has children, then he might need a four door car.

Since this assertion is in the AutoMt, the scope of the $\forall$ does not include Fred. So, in the AutoMt, this $\forall$ cannot be instantiated with Fred. Now the system lifts this rule into CSPSC, we have to ensure that this rule does not conclude that Fred's car should be a four door based on his having kids, i.e., the change in the meaning of "$\forall$" has to be compensated for. The most straightforward way of doing this is to modify the lifted assertion to,

(A47) $(\forall$ x $\neg$(equal x Fred) $\wedge$
primaryCar(x car) $\wedge$ children(x y) $\rightarrow$ hasFeatures(car FourDoorBody))
Unless the user is Fred, if the user has children, his primary car
should be a four door car.

More generally, if the origin context assumes all the objects in its domain satisfy the condition P (and the target context does not assume this), and P is the strongest such condition, when lifting an assertion such as $(\forall$ x R(x)) will get lifted to

$(\forall$ x P(x) $\rightarrow R_1$(x))

where R_1(x) states the same thing in the target context as P states in the origin context.

3.13.4 Representing Scope Assumptions

Now, how does this technique of adding conjuncts relate to restricting the scope of a context by specifying what is not presentIn a context? The above technique can be derived from the constraints on the scope, i.e., the not presentIn statements we make. Consider a context such as the AutoMt. Given a new person G002 (and his car), unless we know otherwise, the system assumes the theory applies to him, i.e., unless it knows otherwise the system assumes presentIn(AutoMt G002). The restrictions presentIn are stated by the assertions representing the assumptions. These generally have the form

(A48) ist(c_i $\neg A_1$(x)) $\Rightarrow$ $\neg$presentIn(c_j x)

where c_i is a context that is more general than c_j and A_1 is an assumption. Let A be the strongest assumption, i.e., the conjunction of all the assumptions. Since the system is trying to maximize the scope of the context, the only condition under which an object will not be in the domain of the context is when it does not satisfy A (in c_i). So,

(A49) ist(c_i ¬A(x)) ⇔ ¬presentIn(c_j x)
A(x) is a necessary and sufficient condition for an object x to be present in c_j.

When the system lifts a formula from c_j to c_i, let the set of objects which are in the domain of c_i but not in the domain of c_j be S. These are exactly the objects the system is interested in since these are objects outside the scope of c_j, but objects the system might encounter in c_i. How does the system determine whether a given object is one of these? The above formula tells us that these objects are exactly those which satisfy ¬A. So to compensate for the difference in the scopes of the two contexts, it is sufficient to change (∀ x R(x)) to (∀ x A(x) ⇒ R_1(x)).

3.13.5 Obtaining the strongest assumption

There is of course the problem of obtaining the necessary and sufficient condition "A" mentioned earlier, i.e., the strongest assumption.

Since it is usually easier to obtain several simple necessary conditions, the system assumes the strongest assumption is the conjunction of these necessary conditions. This is equivalent to maximizing the extent of presentIn, i.e., assuming a maximal scope for our theories (which at least *a priori* appears reasonable).

3.13.6 Assumptions behind CSPSC

We now present the representations of some of the assumptions (associated with the car selection application) mentioned at the beginning of this example and then discuss how the problem solver deals with these kind of assumptions.

(a) The purchase is taking place in late twentieth century and in the US.

(A50) (∀ t_i ist(CSPSC before(t_i 1970)) → ¬presentIn(AutoMt t_i))
(A51) (∀ p selectionFor(CSPSC p) ∧ ist(CSPSC ¬livesIn(p US)) →
¬presentIn (AutoMt p))

(b) The primary use of the car will be for transportation.

(A52) (∀ c ist(CSPSC ¬primaryUse(c Transportation)) →
¬presentIn(AutoMt c))

3.13.7 Assumptions and Problem Solving

The assumptions are integrated into problem solving as follows. We would like to reduce the number of times the system has to check whether some object satisfies the assumption. Most objects the system encounters are expected to satisfy these assumptions. However, we don't expect most sets of objects to satisfy the left hand sides of most rules. So, after the system finds bindings that satisfy the left hand side of a rule, but before adding any conclusion to the knowledge base using those bindings, the system verifies that the bindings do not violate the assumptions. If the assumption is statable in the vocabulary of the context, the system also asserts that the assumption is satisfied (not simply "not violated") by the objects. The internal representation Cyc uses for these assumptions is briefly covered in Example 10.

3.14 Example 9: Unstatable assumptions

Extending the previous examples, in this example, we consider the second kind of assumption that might be externally associated with a context (or specific assertions in the context). An assertion could make an assumption which is not statable in the limited vocabulary of the context in which that assertion was made. The following example illustrates this.

If a person does not have enough financial liquidity to pay cash for the car, he finances the car. This is stated in a theory of monetary transactions (NaiveMoneyMt) as

$$\begin{aligned}
\text{(A53)}\quad & \mathrm{holds}(t_i\ (<\mathrm{financialLiquidity(person)\ price(x)})) \wedge \mathrm{objectBought(buying\ x)} \\
& \quad \wedge \mathrm{occursAt(buying\ ti)} \wedge \mathrm{performer(buying\ person)} \wedge \\
& \quad\quad \neg\mathrm{ab\text{-}financialLiquidity(person\ buying}\ t_i) \\
& \quad\quad\quad \rightarrow \mathrm{paymentMode(buying\ Credit)}
\end{aligned}$$

Here ab-financialLiquidity is an "abnormality predicate" that stands for all the assumptions that might have been made by this rule (these are usually left implicit in the other rules mentioned in this chapter). While this seems to be a reasonable rule, there is a hidden assumption in it - namely that the person believes that he does not have enough financial liquidity. So if the person were to have a grossly wrong belief about his financial liquidity, he might actually do something different from what this theory predicts. Since our naive theory of monetary transactions does not have a concept of beliefs, this assumption is stated relative to a more general theory as follows

$$\begin{aligned}
\text{(A54)}\quad & \mathrm{ist(NaiveMoneyMt\ financialLiquidity(person) = x)} \rightarrow \\
& \quad \mathrm{ist(GeneralMoneyMt\ believes(person\ financialLiquidity(person) = x))}
\end{aligned}$$

The NaiveMoneyMt does not include modalities such as believes and we therefore have to use the more general theory to state this assumption.

Now consider the case when this assumption is not satisfied for Fred (at a time t_i when he is buying the car). The strategy used in the previous example would be to exclude Fred from the NaiveMoneyMt. However, this might be much stronger than the system needs. If we do this, none of the other assertions in the NaiveMoneyMt will apply to Fred, and that is not what we desire. In fact, it is possible that Fred is under a wrong impression about his financial liquidity just at the time of this particular purchase and we want this very assertion to apply to him at other times when he has a more balanced idea of what his financial liquidity is. We want the system to suspend the conclusions of this particular rule just in cases where the buyer does not know his actual financial liquidity at the instant of the purchase. So,

$$\begin{aligned}
\text{(A55)}\quad & \mathrm{ist(CSPSC\ holds}(t_i\ \mathrm{(believes(person\ financialLiquidity(person) = x)} \wedge \\
& \quad\quad \neg\mathrm{financialLiquidity(person) = x)))} \\
& \Rightarrow \mathrm{ist(NaiveMoneyMt\ ab\text{-}financialLiquidity(person\ y}\ t_i))
\end{aligned}$$

The above axiom blocks the rule (about finances) from concluding anything in the case that the buyer does not know his real financial liquidity.

The NaiveMoneyMt has other rules indirectly related to financial liquidity. For example, it asserts that the balance in a checking account is always greater than zero. Now, if the individual does not know his financial liquidity, it might no longer be reasonable to make the above assertion. So, in addition to blocking the rule about mode of payment, we might want to make this assumption (about knowing one's financial liquidity) a "limited" context wide assumption. For example, we may say that none of the conclusions made by the Naive Money Context about Fred at the time when he was unaware of his financial liquidity hold at the present time. Formally, this means that if we consider any abnormality literal, every ground instance of this literal with arguments which include Fred (or his bank accounts, purchases he takes part in, etc.) and the time of his ignorance, will be true. This results in the NaiveMoneyTheory suspending its predictions about Fred at the time of his ignorance and this is exactly what we want.

3.14.1 Similar examples

This particular kind of assumption, that a person is aware of his situation, i.e., he knows his financial status, where he is, etc., is made by a number of theories. Using this assumption, rules can be stated (and applied) with far less complexity. Rather than saying - if a person believes that X is true of himself, he decides to do Y and does Y - we just assert "if X is true of a person, then he does Y". Some examples of this include the following.

(a) If a person is ill, he will take medication or go to the doctor: this assumes the person knows he is ill. This is a special case of assuming that a person knows about his health and physical state.

(b) If there is someone in immediately in front of a moving vehicle, the driver will attempt to stop the car: this assumes that the driver sees the person in front of the car. This is a special case of the more general assumption that a person is perceptually aware of his surroundings.

(c) If someone calls another person on the telephone and the second person is near the phone, he will answer it: this assumes the phone is working. This is a special case of the assumption that devices are working. Similar assumptions are made about peoples skills - walking, talking, hearing, etc.

3.15 Example 10: Domain assumptions

Here we consider another example of making an assumption that restricts the scope of a theory. We will focus on lifting axioms from the theory with a restricted scope into another context where not all the objects fall within the scope of the theory. In such cases it is important to ensure that the lifted axiom is not wrongly applied to these objects which are not within the scope of the original theory.

In addition to knowledge about desired car features, the system is also being given knowledge about the changes that occur in cars (and car parts) as the car is used. For example, since the charge capacity of the battery decreases with use, batteries don't normally last more than about 7 years and so the age of the battery in a car is less than 7 years. This is represented as,

(A56) $(\forall$ b allInstanceOf(b Battery) $\rightarrow$ ($<$ age(b) years(7)))

This is of course not really true. For example, even after the battery is removed from the car, it is still a battery and may lie around in some garbage dump for many years. Also, one can imagine abandoned cars sitting with their batteries still connected for many years. So by the predicate "battery" we are only referring to batteries in working cars. If a battery does not satisfy this assumption, it is outside the scope of our theory. Similar assumptions hold for other auto parts referred to in our Auto theory.

(A57) ist(CSPSC allInstanceOf(x AutoPart) $\wedge$ $\neg$($\exists$ y allInstanceOf(y Car) $\wedge$ partOf(x y)))
$\Rightarrow$ $\neg$presentIn(AutoMt x)

Lifting into contexts where the assumption does not hold.

Now consider a CSPSC where the buyer might use the car to transport some car parts. So we might have both a battery that is part of the car (the one under the hood) and one that is not a part of the car (the one that he is transporting). The system should still be able to use the rule about batteries being less than 7 years old to conclude that the battery under the hood is less than 7 years old without concluding that the battery he is carrying in the car is less than 7 years old. (This example is isomorphic to the example on codes of conduct mentioned at the beginning of this chapter.)

When the rule about the age of batteries is lifted into the PSC, it is modified to take the next axiom into account (see previous example and also previous chapter for details on this modification). So it becomes

In CSPSC :
(A58) $(\forall$ b allInstanceOf(b Battery) $\wedge$ ($\exists$ y allInstanceOf(y Car) $\wedge$ partOf(x y))
$\rightarrow$ ($<$ age(b) years(7)))

So the battery under the hood must be less than 7 years old while the battery the user is carrying inside the car has not such restriction.

During problem solving, i.e., at Cyc's heuristic level, axioms such as A57 are represented as "domain assumptions". So the domainAssumptions slot of the unit AutoMt contains the entry (LogImplication battery(u) ($<$ age(u) years(7))). In general, it contains a formula with a single free variable such that every object in the domain of that theory must satisfy that formula (in the current context) with that object as the binding for that free variable. The system lifts the rule into the PSC (without modifying it to take the domain assumption into account). If the left hand side of the rule is satisfied, the system then verifies that the assumption is not violated. The heuristic used here is that the left hand side of most of the rules in the mt are likely to be more discriminative that the domain assumption. Using this strategy, the system will have to test the domain assumption far fewer times.

3.16 Example 11: "The terms"

This example illustrates the use of the context mechanism to incorporate some of the means of compaction used by communication languages. In this example, we introduce a mechanism similar to that of definite pronouns for replacing complex descriptions with simpler terms.

The left hand side of the earlier rules have conjuncts such as uses(person car) and objectBought(buying car) ..., which serve primarily to pick out the car, the person, etc. Repeating these conjuncts over and over again seems cumbersome. English (and other natural languages) solve this problem by replacing descriptions with definite and indefinite pronouns. Consider an english version of the auto theory:

> "A person is deciding which car to buy. If the person lives in a place which is foggy, the car might need fog lights. If the person has children and the car will be his primary car, the car will be a four door car ..."

The person and the car are first introduced and later references to them are made as "the car" and "the person". A similar strategy can be found in legal documents, manuals describing complex devices, and in almost every lengthy document describing a set of objects.

3.16.1 "The Term" Syntax

We use a syntax similar to english to obtain a similar compaction of statements. For example, the earlier rule about foggy weather and fog lights could be rewritten as follows.

(A59) livesIn((The Person) place) $\wedge$ weather(place Foggy) $\rightarrow$
hasFeatures((The Car) FogLights)

or even more compactly as,

(A60) weather((The Place) Foggy) $\rightarrow$ hasFeatures((The Car) FogLights)

where (The Place) refers to the geographical region where the user lives.

3.16.2 Ambiguity

Of course, in english communications, there might often be ambiguity as to the referent of these indefinite descriptions. However, in the use described here, these terms are replacing constructs which were unambiguous. So we have to ensure that these "the terms" we use in assertions do not suffer the problem of being ambiguous.

In the english paragraph given earlier, the user, his car, etc., were introduced in the first few sentences which serve to set up the context. These introductory sentences could then be used to determine the referent of "the person" and "the car". We need an equivalent "preamble" for contexts which use such terms.

3.16.3 Kinds of "The terms"

At this point we should distinguish between two kinds of referents that these definite references might have.

- They might refer to a specific individual.
  E.g., There was a blue Toyota with a green interior parked near the medical school building yesterday. The car was towed by the Campus police because it was parked in a no parking zone.

  Here, "the car" refers to a particular car.

- They might stand for a variable which satisfies some constraints.
  E.g., Consider the situation in which a cat walks into a store with a dead fish in its mouth. The cat is likely to be thrown out.

  Here, "the cat" doesn't refer to any specific cat but any cat which walks into a store.

In this example, we are interested in the second kind of use of definite references.

3.16.4 Lifting

Suppose the user in CSPSC is Fred, the car he is considering is Car001 and he lives in Seattle. When lifting the rule

(A61) weather((The Place) Foggy) $\rightarrow$ hasFeatures((The Car) FogLights)

we want (The Place) replaced with Seattle and (The Car) replaced with Car001, so that the lifted formula is

(A62) weather(Seattle Foggy) $\rightarrow$ hasFeatures(Car001 FogLights)

Though it seems tempting to add corefers(AutoMt (The Car) Car001 CSPSC), the mapping between (The Car) in AutoMt and Car001 in CSPSC is weaker than corefers since it is not bidirectional, i.e., facts the system knows about Car001 do not translate into facts about (The Car) in the AutoMt. The following schema captures the substitution behavior we want.

(A63) ist(AutoMt p((The Car))) $\rightarrow$ ist(CSPSC p(Car001))

where p is any formula involving the term (The Car). Similar schemas are required for (The Person), (The Place), etc.

3.16.5 Complication

This approach of substituting the actual referents for "the terms" works only if the target context has a unique instantiation for each of the "the terms." Since these terms are really stand-ins for variables, in the general case, we will need to reintroduce the variables in the lifting process. The following example illustrates this. Consider the following english sentence - "During the school picnic, each child will be accompanied by his/her parent ..." - stated as,

(A64) holdsDuring((The Picnic) accompaniedBy((The Child)(The Parent)))

Now, if we have a context c_1 involving a picnic P_{c1} with many children, $Child_1$, $Child_2$, $Child_3$, ..., each with it's parent, and if the system must determine who was accompanying whom (using the above rule), the strategy of replacing (The Child), etc., with objects from c_1 would not work. There are potentially many possible bindings for (The Child). By chosing one, we don't want the system to exclude the possibility of using this rule with the other children. If the system finds a set of terms [(The Picnic) (The Child)(The Parent)] in a sentence, the system assumes that (The Parent) is the parent of (The Child) who goes to (The Picnic). It does not really matter which particular picnic, child, parent set the system chooses until it satisfies this constraint. So the solution is to replace occurrences of (The Child), (The Parent), etc., with variables and conditionalize the formula on these variables satisfying these constraints.

Going back to the rule about fog lights, we want the formula

(A65) weather((The Place) Foggy) $\rightarrow$ hasFeatures((The Car) FogLights)

to be lifted as,

(A66) user(x y) and allInstanceOf(y Car)
livesIn(x place) $\wedge$ weather(place Foggy) $\rightarrow$
hasFeatures(y FogLights)

We state the following schema for this translation as

(A67) ist(AutoMt ϕ[(The Person) (The Car) (The Place)]) $\rightarrow$
ist(c_i user(x y) $\wedge$ allInstanceOf(y Car) $\wedge$ livesIn(x z) $\rightarrow$ ϕ[x y z])

where ϕ[(The Person) (The Car)(The Place)] is any formula with the only occurrences of "the terms" being (The Person), (The Place) and (The Car).

3.16.6 Problem Solving

For problem solving these "the terms" can be dealt with very efficiently. A set of "the terms" used by a context (such as (The Person), (The Car), (The Place)) induces a set of sets legal bindings for these terms. If a context has the people P001 and P002 who own Car001 and

Car002 respectively and live in Place001 and Place002 respectively, then if (The Person) is unified with P001 in a formula, (The Car) and (The Place) should unify with Car001 and Place001 respectively. This set of sets of legal bindings is computed and cached (on demand.) Then, during the unification process, if one of these "the terms" (i.e., any one of [(The Person), (The Car),(The Place)] is unified with a certain object, the other "the terms" from this set that appears in this formula can be replaced with the corresponding objects that have been precomputed in the list of legal binding sets for these "the terms."

3.16.7 Iota Operator

Certain logics include an operator called *Iota.* The term (Iota x ϕ(x)) denotes *the* x such that ϕ(x) is true. The term is undefined if there is more than one x such that ϕ(x). This (sometimes being undefined) introduces all sorts of problems which is one reason for these terms being used infrequently. Though "the terms" appear similar to Iota terms, they avoid this problem. Being normal nonatomic terms, "the terms" don't require any additional logical machinery. When they are modified during lifting, they are replaced with variables which might be substituted with many different bindings.

3.16.8 Use of "the terms" as *Prototypes*

In the preceding discussion, these "the terms" were converted to variables before (or during) the time inferencing took place using them. In this use, in some sense, these terms could be considered as merely interface simplifications. Now we consider a use where a "the term" might never be replaced with/or be treated as a variable during inference.

The view taken earlier of these terms is as a heavily constrained variable. There is another interpretation of these terms as being *prototypes.* So a term such as *(The Person)* in the AutoMt denotes the prototypical buyer of the car.[5] Under the interpretation of these terms as prototypes, we might find uses for them not just in microtheories such as the AutoMt, but also in problem solving contexts such as CSPSC. Let us now examine one of these uses.

Typical Drives

One of the rich sources of constraints on the car is the type of drives the user is likely to take. There are different categories of drives that the user might make - a drive to work, a family drive, etc. Given a particular user, he might make only certain of these types of drives, e.g., if the user were single and lived alone, he might not make any family drives.

If the system could conclude that on a *typical* family drive he had to ferry around four people, it might suggest that he get a four door car. Note however, if on most of his family drives there are going to be only two people (including him), he might not need a four door car even if he had to drive four people around once every six months. The important point to note here is that the system is interested only in the *typical* drives he takes. Of course this assumes there is enough in common between the different family drives he takes for there to be a typical family drive.

[5]There is yet another interesting view of these 'the terms'. Quantification in logic is "lexical" in scope. These terms introduce the equivalent of "dynamically" scoped variables into the logic.

How do we state "On Fred's typical family drive there are more than two passengers"? The following expression states this.

(allInstanceOf(e FamilyDrive) $\wedge$ performedBy(e Fred) $\rightarrow$
entryCardinalityGreaterThan(e passengers 2))

Now let us write the axiom to suggest that if the number of passengers is typically greater than two the car should be a four door car. Note that the first part of the left hand side states "if there are more than 2 passengers on a typical drive".

(allInstanceOf(e FamilyDrive) $\wedge$ performedBy(e Fred) $\rightarrow$
entryCardinalityGreaterThan(e passengers 2)) $\wedge$ vehicle(DR_1 car) $\rightarrow$
bodyStyle(car FourDoor)

This axiom is quite awkward and inefficient to reason with.

We simplify this by using "the terms" in CSPSC. The system creates a unit corresponding to Fred's typical family drive. This is a "the term" with its driver being Fred and the vehicle used being the car (about which we are trying to gather more information). The above axiom would be restated as

allInstanceOf(e TypicalFamilyDrive) $\wedge$ vehicle(e car)
entryCardinalityGreaterThan(e passengers 2)
$\rightarrow$ bodyStyle(car FourDoor)

which, being a horn clause, is much easier to use when performing inference.

Alternately, to avoid the term TypicalFamilyDrive, the above axiom can be rewritten as

allInstanceOf(e FamilyDrive) $\wedge$ allInstanceOf(e TheTerm) $\wedge$ vehicle(e car)
and entryCardinalityGreaterThan(e passengers 2) $\rightarrow$
bodyStyle(car FourDoor)

The above axiom (in the AutoMt) can be instantiated, with the typical family drive created by the system, to derive that Fred might need a four door car.

In general, the strategy of introducing "Typical X" can be used to simplify reasoning of the form "if the usual/typical X satisfies the constraint C_1, then Y satisfies the constraint C_2." The non-horn axioms which usually result from such statements can be converted into horn rules by the use of "the terms".

It should also be noted that statements made about Fred's typical family drive may be used to derive conclusions about some particular family drive taken by Fred. The technique followed for this is the same as the one described earlier in this example, i.e., "the terms" are converted to variables during lifting. The only difference is that the "lifting" (if it can be called that) need not be to another context, it could be internal (within the context.)

3.17 Example 12: Multiple Conflicting Theories of a domain - 1

One of the central issues in the selection of the car is the price. The price is often one of the parameters used to shorten the set of possible cars. Associated with the car is a number of different prices. There is the retail price, the price that one might actually settle for after some bargaining, the actual sum of money that will be paid over a few years if the buyer takes a loan, the price including tax, registration, etc. One could also associate the cost not with the car, but with the buying/negotiating event.

However, in the initial stages of the selection, we approximate the cost to the buyer of the retail price of the car plus the price of the options he wants, and we use this to shorten the list of possible cars. While the retail price might be an adequate approximation in the initial stages, it certainly is not adequate in the later stages of the purchase.

3.17.1 Adequacy of a theory as a function of the task

In the earlier examples where we dealt with implicit assumptions, we were concerned with whether the assumptions (about the domain) made by a theory were satisfied by the objects included when the theory was instantiated. If the assumptions were satisfied, the predictions of the theory were correct, otherwise the predictions were not correct.

Here it is not so much a case of the theory making the correct or incorrect predictions. There is no hard and fast measure of correctness in determining the price of the car. One could always use a more sophisticated theory which takes into account the effort involved in writing the checks for the monthly car payments, the interest lost on the interest paid, ...We have to replace the notion of "correctness" of the theory with that of the precision of the predictions being adequate. For the initial stages, the results of using the car price as the cost to the user are sufficiently precise, but might not be true in the later stages.

Even in the initial stages, it is insufficient to approximate the cost as just the base retail price of the car since the cost of some of the options (such as air conditioning) could be significant. However, there are some situations where the price of the car could be approximated to the retail price.

3.17.2 Domain assumptions vs. adequacy constraints

When dealing with assumptions about the domain, we used the strategy of excluding the objects that did not satisfy the assumptions from the domain of the relevant contexts. Unfortunately, this strategy will not work here. Here, the issue is not one of the car being (or not being) in the domain of the context with the approximate model of the price. There are some contexts which might find these predictions adequate and so indeed the car must be in the domain.

For example, the NaiveTransactionMt estimates the retail price of an object as its cost. Let us suppose that for some problem addressed in the context c_1, this approximation leads to inaccurate answers. One straightforward approach to avoid the bad answer would be to block the NaiveTransactionMt from making any prediction about the object being dealt with. This could be done by removing this offending object from the scope/domain of the NaiveTransactionMt.

However, there might be another context CSPSC for which this approximation might be adequate and the system should therefore be able to use the predictions made by the NaiveTransactionMt in CSPSC. However, if the system were to remove this object from the scope of the NaiveTransactionMt, it could not make any predictions about it at all. So, the system needs to selectively block the use of the predictions made by the NaiveTransactionMt from being used by problem solving contexts, based on the accuracy desired by the problem solving context. I.e., rather than blocking the NaiveTransactionMt from making predictions about the person (or car), the system blocks importing these predictions to certain contexts.

3.17.3 Solution

We need to block the importing of the predictions that aren't sufficiently precise for the target context. So instead of associating assumptions with the theories that make the predictions, we have to associate estimates of the precision of the relevant conclusions made by the mts. Then based on the precision required for the particular problem the system is trying to solve, the predictions made should be imported (or not imported) into the current Problem Solving Context.

To do this, the system needs an estimate of the precision (or possible error) of the cost predictions made by the NaiveTransactionMt (measured either in absolute terms or percentages). The error in the predictions could vary from the cost prediction of one car to another, and so to be completely accurate, the system needs to use estimates for the error in determining the cost of Car001, the error in determining the cost of Car002, etc. separately. However, it is very unlikely that we will either have such information readily available or be able to compute it. So the system makes an approximation and simply estimates the error in cost predictions in general.

3.17.4 Blocking the Lifting

From the default coreference rule, if cost(Car001)=A in a context M, as a default coreference rule, this will be lifted to cost(Car001)=A in any Problem Solving Context PSC unless $ab - dcf$(M, PSC, cost, Car001) is true. As we mentioned earlier, some of the conditions under which this will not be true are as follows.

1. Cost is not in the vocabulary of PSC (or it has a different representation).

2. Car001 is not in the domain of PSC.

Now we have another condition under which this might be true - if the accuracy of the predictions made by M is insufficient for PSC. The following rule (in the outer context) states this.

(A68) (> error(NaiveTransactionsMt cost) precisionRequired(PSC cost))
$\rightarrow$ ab-dcf(NaiveTransactionsMt PSC cost x)

The lifting process takes care of these rules as follows. When the problem solver tries to access cost(Car001) from the knowledge base, if the error of NaiveTransactionsMt is greater than the precision required in the current Problem Solving Context, no lifting from the NaiveTransactionsMt takes place. Similarly, when the problem solver is trying to access rules that conclude cost, if the error from NaiveTransactionsMt is greater than that allowed by current Problem Solving Context, rules from the NaiveTransactionsMt are not lifted.

3.17.5 Multiple acceptable theories

At times there may be more than one theory such that the error in the prediction made by the theory is acceptable. Consider the program used by Porsche for estimating the annual income of people (in order to determine whether to send them literature about the new Porsche 914). They were only interested in determining the income to an accuracy of about 10,000 dollars. In such a case, it does not matter whether the annual income was predicted as eighty five or eighty six thousand dollars (though the IRS might be concerned with this difference). However, if a theory can estimate the cost to within five hundred dollars, there is likely to be some additional computational cost associated with it. In such cases, the system prefers the cheapest adequate theory, and so the results of the lifting will be ordered to try the rules from the cheaper theory first.

3.18 Example 13: Multiple Conflicting theories of a domain - 2

The discussion in example 12 made it seem as though there is a clear distinction between situations where assumptions made by theories make the theories incorrect and situations where the assumptions lead to the predictions having some error in them. Unfortunately this is not the case. Assumptions (such as the ones explained in Example 9) and errors in the predictions made by theories are two sides of the same coin. The errors are the cost of the assumptions that a theory makes.

3.18.1 Approaches to determining the adequacy of a theory

In general, there are two approaches to determining whether to use the predictions made by a theory. One approach, illustrated in the previous example, is to estimate the error in the predictions of a theory and determine whether this error is acceptable. This approach is useful when the assumptions that have been made by a theory are not available. This is the case especially when the system has access to the predictions of a theory (such as a database) but has to treat the theory itself as a black box, i.e., it doesn't know the internals of the theory or the origins of the data in the database.

In other cases, the system might not have any reasonable way of estimating the error in the predictions of the theory (or the error might vary wildly). However, the system might know some of the assumptions made by the theory and know when it is reasonable to make these assumptions.

For example, the potential error in using Newtonian mechanics (as opposed to Relativistic mechanics) might be very very large in the worst case and so this estimate is not very useful in determining whether Newtonian mechanics is adequate for a particular problem. However, we do know that if the speed of the objects is less than half the speed of light, Newtonian mechanics is sufficient, i.e., the assumption that the ratio of the speed of the object involved to the speed of light being zero is acceptable if this ratio is less than half.

Now we consider an example, taken from the car selection application, of such an approach for determining whether the information in a context is sufficiently accurate. The database associated with the car selection program has information about mileage, acceleration, etc. for the different automobile models. These numbers are obtained from tests performed at sea level. If the car is going to be used at higher altitudes, these numbers will change. We don't have any reasonable estimates of the required level of precision for these numbers, or the error in the mileage estimates of the database as a function of the the altitude. However, we do know that these numbers shouldn't really be used (for the purpose of selecting cars) if the car is going to be used at an altitude of more than 10,000 feet. We want to ensure that if the user lives at an altitude of greater than 10,000 feet, the system does not lift the information about the mileage, from the database.

As in the previous example, the strategy used to block the information in the database (abstracted as AutoDBMt) from being lifted into CSPSC is to derive ab-dcf(AutoDBMt PSC mileage y) (y will be instantiated with the car). This is done using the following rule.

(A69) ist(CSPSC (> altitude((The Place)) ft(10000))) $\rightarrow$
ab-dcf(AutoDBMt PSC mileage y)

To summarize, some of the effects of (and the means of dealing with) the assumptions made by a context are as follows.

(a) The assumption excludes some objects from the scope/domain of the context. This manifests itself during lifting as the addition of constraints on variables.

(b) The assumption results in a degradation of the predictions made by the context. These predictions might be adequate for some purposes, and inadequate for others. (a) is an extreme case of this where the prediction is not adequate for any purpose. When there are some contexts for which the predictions are of acceptable quality, the system needs to include rules for determining when the predictions are acceptable. There are two approaches that can be taken at this point.

  (i) If estimates on the error in the predictions of the context and acceptable error margins of the Problem Solving Context are available, these can be used.

  (ii) Heuristics specifying conditions under which certain assumptions are reasonable might be available and can be used for this purpose.

3.19 Example 14: Non-Numeric Approximations

One of the factors that is used to shorten the list of cars is the storage space that is required by the user. It is often the case that the user would like a sports car, but has to occasionally use the car to transport bulky items. This constraint eliminates small sports cars and roadsters.

The general problem of determining whether a given object will fit into a given region is a very complex one, especially if we consider objects with non-regular shapes such as christmas trees. However, approximate, “safe” answers might be obtained by approximating the shape of the tree to be rigid cone and the interior of the car to be an appropriately sized cuboid. The answer is safe in the sense that if it says that the tree will fit in the car, we can be sure that the tree will fit. The potential error is when it claims that the tree will not fit - in practice one might still be able to fit the tree in the car (by bending the tree, etc.). Abstracting the interior of the car as a cuboid and the tree as a cone raises several issues.

3.19.1 Problems with approximations

While it is adequate to abstract the interior of the car as an empty cuboid for the sake of determining whether something will fit inside, this might be unacceptable for some of the other aspects of selecting cars (like evaluating the tastefulness of the interior). So the bulk of the inference regarding whether the christmas tree will fit in the car will have to be done in some other context (such as the ContainerMt), with only the final answer being imported into CSPSC. Let us elaborate on this a bit further.

Assume that we asserted in CSPSC that the interior of the car was a cuboid. (This is for the purpose of determining whether the tree will fit into the car), that is, ist(CSPSC cuboid(interior(car001))). Later, in evaluating the aesthetics of the car the user indicates that he would like a car whose interior is not “boxy”, i.e., an interior that doesn’t resemble a cube. For this, a more accurate model of the interior of the car (more accurate than abstracting it as a cube) is required. If we had asserted cuboid(interior(car001)) in CSPSC, this precludes the interior from having any other shape (since a rigid object can have only one shape) and the evaluation of the aesthetics will give us a wrong answer.

This is an instance of the following general problem. The system is given a description of a set of objects and has to answer a sequence of queries about these objects. Approximations which are reasonable for answering one query are might not be reasonable for answering another, but if the approximate attributes are going to be asserted for the objects, a query might end up using an approximation it should not use.

Similarly, while it might be acceptable to abstract the shape of a christmas tree as a cone for purposes of determining whether it will fit into a car, for other purposes (such as cutting it), this would not be an acceptable approximation.

The solution for the problem is to make the assertions corresponding to the approximate representation in a different context, one which shares the problem solving specifications of the query for which these approximations are acceptable. This context could be an extant context or a new one created for this purpose.

3.19.2 Context Based Solution

We take the given situation and build an appropriate “model” of it, i.e., a representation suited for the task of determining whether the object the user wants to transport will fit in the car being considered. Since this model could be contradictory with the given information (because of approximations), it will have to be constructed in a different context. This construction is done by a class of lifting rules called modeling axioms.

Let us assume the user wants to be able to use the car for carrying small christmas trees (of a given height). The ContainerMt is concerned with issues such as determining whether one object will fit into another and approximates the interiors of containers to regular polyhedra. The system builds the model of the christmas tree as a cone and the interior of the car as a cuboid in the ContainerMt. It then uses the axioms in the ContainerMt to determine whether the cone will fit into the cuboid. If it does, we assume that the tree will fit into the car and if not, we assume that the tree will not fit into the car (even though it might still fit in!). The rules for this are

(A70) ist(CSPSC mic(x y)) $\rightarrow$ ist(cgmt interior(x y))
The interior of an object (in the container mt) is the biggest
cuboid that will fit into an object. mic is maximal-interior-cuboid.

(A71) ist(CSPSC mep(x y)) $\rightarrow$ ist(cgmt exterior(x y))
The exterior of an object (in the container mt) is the minimal exterior
polyhedron around that object.

(A72) ist(cgmt interior(x y) $\wedge$ exterior(a b) $\wedge$ fitsIn(b y) $\rightarrow$ fitsIn(a x))
If the exterior of an objects fits into the interior of another object,
the first object fits into the second.

(A73) ist(cgmt fitsIn(x y)) $\leftrightarrow$ ist(CSPSC fitsIn(x y))
If the container microtheory predicts that one object fits inside another,
this prediction is adequate for the car selection context.

The first 2 (A70 and A71) are called modeling rules, the third (A72) is the rule in the container gmt and the final rule (A73) is for lifting the results of the computation on the model back into the original context.

The term "perspective" has been sometimes used in Artificial Intelligence literature [KRL] to describe this phenomenon. We could say that in the above example, the tree was described from the perspective of a regular polyhedron or that the interior of the car was described from the perspective of a regular polyhedron. However, we will reserve the term perspective for something else (Example 17 has more details on this).

3.20 Example 15: Doing approximation with and without contexts

Let us consider an example similar to Example 14 that is related to the problem of getting from one place to another. Before driving from one place to another, some sort of planning involving choosing the route is often done. One of the canonical examples of the use of multiple models involves making simplifications in the geometric shape of the road, the destination, etc. for purposes of this planning [Hobbes]. Let us now examine how this modeling could be done with and without contexts.

When determining which roads are to be taken going from one place to another, we are not concerned with the dimensions of the road such as its width or its curvature and can therefore abstract the road as a line. Similarly, we are not usually concerned with the dimensions of the origin and destination and can abstract them as points. This reduction of the roadway to a set of points connected by lines reduces the roadway system to an undirected graph, therefore graph search methods (which are much more efficient than general inference) may be used to determine the path.

3.20.1 Approximation without contexts

First let us consider how we might go about this reduction without the use of contexts. If we have a road such as PalmDrive (PD), we cannot simply assert that it is a line - it is a road and the system already has information about the dimensions of a road. This problem is identical to the one we faced in the previous example. Approximating the interior of the car as a cube to solve one problem got us into trouble with another problem. Since there might be other problems for which approximating Palm Drive as a line would not be appropriate, we cannot just unassert the information the system already has about Palm Drive and assert that Palm Drive is a line.

Since we don't have any means of separating sets of assertions about an object (without contexts), the incriminating attribution (i.e., of being a line) will have to be made of some other object. Of course this object will be closely related to Palm Drive and we might even assume the relation to be a one to one relation. Let us denote this new object as approx(PD). Then approx(PD) can be thought to denote the abstraction of PD that is appropriate for solving our problem at hand. Later we might want to add other arguments to approx to distinguish between different approximations of Palm Drive. Once we have approx(PD) we can say that approx(PD) is a line.

Similarly, if the origin is MargaretJacksHall, since MargaretJacksHall isn't really a point, we introduce an abstract object approx(MargaretJacksHall) and say that this is a point. Then, if we know that the Palm Drive connects two places A and B, we say that approx(PD) connects approx(A) and approx(B). The rules required are

(A74) connects(r, p_1, p_2) $\Rightarrow$ connects(approx(r), approx(p_1), approx(p_2))
If r connects p_1 and p_2, the approximation of r connects the approximation of p_1 and p_2.

(A75) road(x) $\Rightarrow$ line(approx(x))
The approximation of a road is a line.

(A76) place(x) $\Rightarrow$ point(approx(x))
The approximation of a place is a point.

Then, if we need to determine a sequence of roads to take us from A to B, we determine the sequence of lines that connect approx(A) and approx(B) and from this sequence of lines, map back to the roads they denote.

3.20.2 Problems with this approach

This approach, though feasible, has a number of shortcomings. The central problem is the need to introduce abstract objects such as approx(PD). Since approx(PD) is not equal to PD, the representation states that there are actually two different objects. Not only is this cumbersome, it is also completely wrong. It is not the case that there are two different objects - there is one object and two ways of looking at it. If this were just a philosophical complaint, it would not have much significance. However, there are some more serious problems that arise from this.

(a) A framework for reasoning with approximate models, etc., should not allow one to make nonsensical statements using constructs introduced to do the approximation. Consider the formula near(PD approx(PD)) - it is nonsensical in that it does not make sense to talk about whether Palm Drive is near Palm Drive approximated as a line. Is approx(PD) connected to the real Margaret Jacks Hall or is it only connected to Margaret Jacks Hall approximated as a point? That these questions arise seems to indicate that there is something wrong with this approach. More pragmatically, these questions are all potential subgoals in problem solving and tend to increase the size of the search space.

(b) There are some queries that will simply get the wrong answer using this scheme. For example, is approx(PD) a road? If it is not, we have a problem since cars can't traverse it. If it is road, consider the query "how many roads does Stanford campus have"? PD and approx(PD), being distinct entities count as two roads. This is true for the other roads on the campus also and so the answer to the query is twice the number of actual roads. At this point it seems tempting to go back and say that approx(PD) is not a road - it is only a line, and cars can't traverse it. But there are still many problems remaining. Assume that we want to avoid driving under palm trees. Are there palm trees on approx(PD)? If not, this approximation is not adequate. If we do accept that there are palm trees, we are back to the original kind of problem - how many roads are there on Stanford campus with palm trees - the system's answer will include Palm Drive and approx(PD) and this is clearly wrong.

The problems seem to stem from the fact that other than three axioms mentioned earlier [A74, A75, A76] relating "real" objects (such as Palm Drive) to abstract object (such as approx(PD)), there should not be any other axioms relating these approximate objects to real objects. However, the framework outlined above does not accord the axioms mentioned earlier any special status or prohibit us from making these statements relating approximations to real things. The crucial problem with the above approach is that there is really only one object - Palm Drive - and there are two sets of assertions about this object. It is not the case that there are two distinct objects as the above scheme portrays. Unfortunately, without contexts there is no way of capturing the notion that there could be two mutually inconsistent theories (sets of assertions) about the same object.

3.20.3 Context based solution

With contexts, this problem is vastly simplified by not having to worry about these abstract objects or the approx function. We have a context, the MapContext, in which roads remain roads, but one of the properties of a road is that it is a line. Similarly, places remain places and one of the properties of a place is that it is a point. Given a query asking for the path between a place A and place B, the system solves this query in the MapContext, and lifts the same answer back to the original context.

In MapContext

(A77) road(x) $\rightarrow$ line(x)
A road is a line (in the map context).

(A78) place(x) $\rightarrow$ point(x)
A place is a point (in the map context).

The following conclusion is provided by the default coreference rule (i.e., no new lifting rule is required).

(D17) ist(CSPSC connectedTo(x y)) $\leftrightarrow$ ist(MapContext connectedTo(x y))

3.20.4 The Role of Contexts in this solution

The distinction between the two approaches is illustrative of the power of contexts. Without contexts, introducing approximations is made difficult by the fact that the approximate representation could contradict the more accurate one. Since there is no mechanism for "keeping apart" different sets of assertions that constitute a representation of an object, it is necessary to posit the approximation as being true of some other object and then map back and forth between the two things. The context mechanism provides a means of keeping the assertions about an object (or sets of objects) different. This in turn allows us to focus on a single model of the system without having to worry about inter-model inconsistencies.

3.21 Example 16: Polysemous use of predicate symbols

One of the factors important to many people in the purchase of a car is who they buy the car from. In colloquial english we come across statements like "I bought the car from Fred" where Fred is the actual salesman dealt with, or "I bought the car from McDavid Acura" or "I bought the car from Acura". These are all different senses of "buy from" (seller). There are different situations in which we come across these different uses of the predicate seller and for each, there is a class of situations in which each is useful.

3.21.1 Cyc representation of actions

In Cyc, actions (such as Joe buying Car001 from Fred on 14th January 1991) are rich objects. Predicates such as seller, buyer, objectBought, etc., are called "Actor Slots" and relate the action to the different individuals involved in the action. If Buying001 denotes the above action, we have buyer(Buying001 Joe), objectBought(Buying001 Car001), seller(Buying001 Fred).

3.21.2 Multiple senses of Actor Slots

The issue we are concerned with in this example is the following. Different contexts, following the different uses of the phrase "bought from" in english, might use the predicate "seller" to refer to the dealership the car was bought from, the company the car was bought from, the salesman the car was bought from, etc. For different purposes, different uses are preferred. For example, if one were considering which dealership from which to buy a car, "seller" might be used to refer to the dealership. If one had decided on the dealership and was determining which salesperson to approach, one might use "seller" to refer to the salesperson. The symbol "seller" in these different occurrences is used to denote different concepts. We call these the different senses of the predicate symbol.

There is a metonymical relationship between these different senses of the slot seller. If seller(Buying001 Fred) is true, then the system knows that there is some shop X such that worksFor(Fred X) then seller(Buying001 X) is true for a different sense of the slot seller.

3.21.3 The General Phenomenon

This phenomenon of a symbol being used to denote one of many closely related things with the actual referent varying with the context occurs frequently. Some other examples of this include,

- The term "USA" might be used to refer to the country, the government, the administration, the military, ...
- Actions initiated by an individual might be attributed to that individual. For example, one might say "Bush bombed Iraq". Bush did not actually fly any plane, he initiated the action by ordering the bombing.
- Properties that hold for objects might be attributed to larger objects in which they are a part. For example, one might say that a car has 189 horse power. It is not really true that the car has that much power, it is the engine of the car that has 189 horse power.

The objects denoted by these different uses (of a term) are different and it is important to distinguish between them, e.g., we certainly want to distinguish Bush from the pilots who flew the planes, the government of the USA from the military, etc.

One approach would be to introduce distinct terms for each of these denotations and use a term uniformly to denote the same thing.

However, natural language utterances use polysemous words and depend on the context for disambiguation. In keeping with our earlier stated strategy of admitting translations (of natural language utterances) which have context dependencies left in them, we should provide some mechanism for allowing a term/symbol in a discourse context to denote something other that what it usually denotes.

Also, as mentioned at the beginning of this chapter, natural language often influences representations in the sense that the representation has some of the properties of natural language utterances left in it. This often leads to a predicate symbol being used in different theories with different intended denotations. Again, in keeping with our strategy, we should try to provide translation rules to map between these theories (as opposed to redoing the theories completely).

In this example, I will focus on these actor slots and attempt to provide a general solution for mapping between theories using an actor slot in different senses.

3.21.4 Standard kinds of actor slot senses

To begin with, let us analyze the senses in which the predicate seller may be used. If we state seller(Buying001 B), the B might denote,

(a) The person actually doing the selling. E.g., the counter person at a McDonald's.

(b) The legal entity involved. E.g., the McDonald's Corporation.

(c) The person under whose immediate direction the action is taking place. E.g., the manager on duty at McDonald's.

These might all be the same or different agents. Interestingly, almost every actor slot involving an agent can be used in any of the above senses (performer, victim, buyer, observer, referee, etc.). This suggests that these senses might have an existence independent of the individual predicates. If a context c_1 uses seller in the first sense, this is written as wordSense(c_1 seller) = ActualPerformerSense. Similarly, (b) is called LegalEntitySense and (c) is called ControllingAgentSense.

Axioms using seller (or any of the other actor slots) in the ActualPerformerSense can be lifted to contexts using seller in the LegalEntitySense using the following lifting rule.

(A79) $\quad \mathrm{wordSense}(\mathrm{mt}_1\ \mathrm{s}) = \mathrm{ActualPerformerSense} \wedge$
$\quad\quad \mathrm{wordSense}(\mathrm{mt}_2\ \mathrm{s}) = \mathrm{LegalEntitySense} \wedge \mathrm{ist}(\mathrm{mt}_1\ \mathrm{representsAgent}(\mathrm{y}\ \mathrm{z}))$
$\quad\quad\quad \Rightarrow \mathrm{ist}(\mathrm{mt}_1\ (\mathrm{s}\ \mathrm{x}\ \mathrm{y})) \Leftrightarrow \mathrm{ist}(\mathrm{mt}_2\ (\mathrm{s}\ \mathrm{x}\ \mathrm{z}))$

3.21.5 Second order reformulation

The concept of different senses of a predicate symbol can be captured without contexts in a second order language. The different predicates related to "seller" are obtained using the function predFunction. To say that Fred is the actual agent involved in Buying001, we write

(A80) ((predFunction seller ActualPerformerSense) Buying001 Fred)

Similarly, to say that McDonald's is the legal entity involved in Buying001, we write

(A81) ((predFunction seller LegalEntitySense) Buying001 McDonalds)

If a set of axioms uses seller in the same sense, say the LegalEntitySense, the system can replace (predFunction seller LegalEntitySense) with just seller and put these axioms into a single context. This makes the axioms first order (within the context).

3.21.6 Determining the sense

The NaiveMoneyMt is concerned with monetary transactions and since monetary transactions are usually concerned with legal agents, seller is used in the LegalEntitySense. On the other hand, the HumanRelationsMt which is concerned with interactions between humans, attempts to describe aspects of the relation between the human buyer and human seller such as the code of conduct each follows, and therefore uses seller in the ActualPerformerSense.

Axioms from both these microtheories might be used during the course of a car selection session. Since each context (such as CSPSC) must use "seller" in a uniform sense, the axioms from at least one of the above two microtheories will need modification during lifting. Lifting axioms such as the one given above are adequate for taking care of the lifting once the system knows the sense in which CSPSC is using "seller". The problem therefore is that of determining the sense in which CSPSC (or any other problem solving context) uses "seller" or any other such predicate.

The primary strategy used is that of comparing the statements known in the Problem Solving Context with those that would have been made if it had been using seller in a certain sense.

The sense (in which CSPSC uses seller) imposes certain constraints. If these are violated, the system knows that CSPSC can not use seller in that sense. E.g., if the sense is ActualPerformerSense, the seller must be a person. If a shop or company is specified as the seller, then the system knows that the sense in which it is used is the LegalEntitySense. (Note however even if the seller had been specified as a person, the sense could be LegalEntitySense.)

More precisely, the strategy used is as follows. The system knows that if "seller" is in the vocabulary of CSPSC, it must use it in some sense. The system minimizes the number of possible senses of a predicate, i.e., assume that the system knows of all the possible senses. Given this, wordSense(CSPSC seller) has to be one of ActualPerformerSense, LegalEntitySense or ControllingAgentSense. With this constraint, what is logically entailed is that the sense in which CSPSC uses "seller" is the one whose predictions show least deviation from the assertions the system already has in CSPSC.

In practice, the system checks the domain, the range, and a few other such constraints on "seller" which these senses impose and keep an updated list of possible candidate senses for each context predicate pair.

3.22 Example 17: Functional vs. Structural Definitions

In Example 16, we considered the different senses in which a predicate symbol such as "seller" could be used. In this example, we consider the different concepts a term such as "bridge" or "screwdriver" might denote.

Depending upon the context, the term "screwdriver", might be used to refer to anything that can be used to screw in a screw or anything with the shape of a standard screwdriver (even if the object with that shape cannot be used for this purpose).

In the context of assembling a device, if one person were to ask another person for a screwdriver and the other were to hand him a pen knife capable of performing the task, this might be acceptable, i.e., the pen knife can be considered a screwdriver. Here we are using a "functional" definition of a screwdriver. Anything which can perform the function of tightening a screw is a screwdriver.

Now consider the context of teaching a child the shape of various hand tools and what they are used for. To teach the child what a screwdriver looks like, the teacher might ask someone to bring him a screwdriver. Certainly, returning with a penknife would not be acceptable in this context. However, returning with a toy screwdriver (that could not actually be used to tighten screws) would be acceptable (though this would not be acceptable in the previous context). Here we are using a "structural" definition of a screwdriver. There is a certain structure associated with a screwdriver and anything that has this structure is a screwdriver.

3.22.1 Comparison with Actor Slot Senses

The names for most devices (tables, bridges, houses, ...) can be used in a functional or structural sense. As with the actor slots (Example 16), these senses exist independent of the particular term. If a context c_1 uses the term ScrewDriver to refer to anything which can perform the function of a screwdriver, we write wordSense(c_1 ScrewDriver FunctionalSense) and if it uses ScrewDriver to refer to anything with the shape of a standard screwdriver, we write wordSense(c_1 ScrewDriver StructuralSense). As with the actor slots, we minimize the number of possible senses, i.e., given a context which mentions the term ScrewDriver, the system assumes that it uses it in the StructuralSense or the FunctionalSense (and not in any as yet unknown sense).

In the case of the actor slots, the different senses were mutually exclusive. If a context used "seller" in the ActualPerformerSense, the system knows that it cannot use it in the LegalEntitySense also. However, with devices, a context might use the ScrewDriver to denote something that was both functionally and structurally a screwdriver. In fact, as a default, if someone were to use the term ScrewDriver, the system assumes that he is referring to something that can both perform the function of a screwdriver and has the shape of a screwdriver. So, the system has the default,

(A89) $\quad$ wordSense(c_i x FunctionalSense) $\leftrightarrow$ wordSense(c_i x StructuralSense)

If most contexts use a term both in its functional sense and structural sense as a default, what exactly can be concluded by knowing that a context uses ScrewDriver in a functional sense? If the system does know that c_1 uses ScrewDriver in a functional sense, then it knows that any object which is an instance of ScrewDriver must (i.e., not just as a default) be usable for the task of driving in a screw. Similarly, if a context uses the term ScrewDriver in a structural sense, any instance of ScrewDriver (in that context) must have the shape of a screw diver. Based on this, we have

(A90) $\quad$ wordSense(c_i ScrewDriver FunctionalSense) $\Rightarrow$
ist(c_i allInstanceOf(x ScrewDriver) $\Rightarrow$ deviceCapableOf(x ScrewingAScrew))

On the other hand, it is only a default that the object has the shape of a screwdriver. From this it follows that if something is called a screwdriver, it has to be either capable of driving in screws or must have the shape of a screwdriver. So, it is alright to call a broken screwdriver a ScrewDriver or a pen knife (which can be used to drive in a screw) a ScrewDriver, but it is not reasonable to call a broken pen knife (which cannot be used to drive in a screw) a ScrewDriver.

3.22.2 Determining sense using Problem Solving Goals

Given a Problem Solving Context (such as CSPSC) which uses a certain term (such as ScrewDriver or TireJack), the system should be able to deduce the sense in which this term is being used by that context. In addition to the technique based on minimal disparity with the predictions made by assuming each of the possible senses, the system can use the goals of the Problem Solving Context to constrain the possible senses of a term. The previous two examples focused on the issue of comparing the predictions made by assuming a sense with the assertions in the Problem Solving Context. Here we discuss the use of problem solving goals to determine the sense of the term.

Consider the problem of buying a power screwdriver. It would be extremely unlikely that the buyer will settle for something that only looks like a power screwdriver and does not perform like one. It is also unlikely that he will buy a blender motor to which the appropriate tips can be attached which could potentially be used as a power screwdriver. So in this context, the term PowerScrewDriver refers to one which is both structurally and functionally a power screwdriver.

(A91) $\text{PurchaseSelectionPSC}(c_i) \wedge \text{selectionObjectType}(c_i\ x) \Rightarrow$
$\text{wordSense}(c_i\ x\ \text{FunctionalSense}) \wedge \text{wordSense}(c_i\ x\ \text{StructuralSense})$

Similarly, if the problem solving goal were to design a power screwdriver we would expect the term PowerScrewDriver to denote something that was usable as a power screwdriver. However, since the design might involve some new innovative structure, we do not constrain the relevant context to use the term in a structural sense also.

In the above two examples the use of the term PowerScrewDriver was constrained to be the functional use of the term. There are situations where we might be interested in a purely structural use. Some examples of this include the following.

- Consider the problem of designing a toy "medical kit" for children. The kit includes a toy stethoscope. While it is not important for this toy to perform the function of a real stethoscope, it is very important that it have the shape of a real stethoscope.

- Assume that we are interested in opening a lock (and we don't have the key). Knowing that hairpins may be used to pick locks we go about searching for a hairpin. There are many hair accesories which perform the job of a hairpin (which are functionally hairpins). However, in this context, we are interested only in things that are structurally hairpins.

3.23 Example 18: Perspectives

Actions/events in Cyc (and most other systems) are spatio-temporal regions where something of interest is happening. Let us consider describing (in english) a common spatio-temporal region that is of interest for this application. This spatio-temporal region includes everything that happens when a person (Fred) gets into a car and drives to a shop. Note that this includes not just the person's actions (getting into the car, driving, etc.), but also that of the car itself (the engine running, etc.). Some english descriptions of this event include,

(a) Fred got into the car and drove to the shop.

(b) The car (under the control of Fred) went to the shop.

Both these (and other) descriptions inevitably slant the description towards the actions performed by one of the objects involved in the event. This seems to be the case not just in english but also in most Indo-European and Dravidian languages. Let us refer to this "slant" a description might have as the "perspective" of that description. Given that most languages seem to have this property, it is tempting to conclude that there must be some psychological reason for this. If there does exist some such reason, this would be a good case for incorporating some such facility in knowledge representation languages also.

Even if one did not buy the story that there is some psychological basis to this phenomenon, the fact that it is so prevalant in natural language utterances is enough reason for allowing perspective dependent representations.

3.23.1 Formalizing Perspectives

We are given an event (the one described above) involving a person going from one place to another in a car. Call this event E_1. The context TPD-E_1 contains a neutral or "third person" description of E_1. The different actors involved in E_1 are Fred (the driver), Car001 (the car being used), etc. So the system has (in TPD-E_1)

(A82) driver(E1) = Fred
(A83) vehicle(E1) = Car001

Associated with TPD-E_1 are a number of perspective contexts containing the description of E_1 from these different perspectives. The description from Fred's perspective is in perspectiveMt(E_1 Fred) and that from Car001's perspective is in perspectiveMt(E_1 Car001). In general, if s is an actor slot for an action A, perspectiveMt(A s(A)) contains a description of A from the perspective of s(A). Similarly, TPD(A) contains a neutral or perspective independent description of A.

The term "perspective" has been used in Artificial Intelligence literature in a different sense. For example, we might view a battery as a "container" or as a "current source", i.e., describe the battery from the perspective of a container or of a current source. This sense of perspective was dealt with in Example 14 and we are not using the term perspective in that sense here.

There are a couple of interesting questions that arise with the use of perspectives.

- What simplifications/changes may be made by using perspectives?
- How does the system detect the perspective associated with a Problem Solving Context?

(a) Simplifications/changes related to perspectives. We list a few of the simplifications commonly made with perspective contexts.

(i) Since the system is viewing the action from the perspective of a particular actor, that actor is referred to as the performer. This use is related to the third person context as follows.

(A84) ist(perspectiveMt(A s(A)) performer(A x)) $\Leftrightarrow$ ist(TPD(A) s(A x))

So, for example, in a description of a sale from the perspective of a seller, the seller might be referred to as the "performer" of the action.

(ii) Consider the description of a hair cutting event from the perspective of the barber (C_1) and from that of the person whose hair is being cut (C_2). C_1 and C_2 will contain descriptions of the event which specify how tiresome the action is, what resulting monetary profit/loss, how careful the person needs to be, etc. All these are about the performer which happens to be different across the two contexts. In the third person context, we have to add an argument to explicitly specify the person incurring the profit/loss, the person who needs to be careful, etc. We have,

(A85) binaryPred(TPD(A) p) $\wedge$ ist(perspectiveMt(A s(A)) p(x))
$\rightarrow$ ist(TPD(A) p(x A))

i.e., if a perspective context suppresses an argument, that argument usually defaults to the actor from whose perspective that event is described.

(iii) Not every actor is privy to everything that takes place in an event. For example, in a car servicing event, there are actions the mechanic knows about which the owner of the car does not. Similarly, with the driving event, the driver might not know anything about the engine running. In this case, the description of the event from the driver's perspective cannot possibly include the engine running, etc.

More generally, perspective contexts contain only partial a description of events. The third person context can usually be formed by combining the information from all the perspective contexts. Though the perspective context has only a partial description, the description still satisfies certain properties.

For example, the system can assume that perspectiveMt(E A) contains all the subevents of E performed by A, and that A is aware of everything that is happening in E in this context.

(b) Detecting perspective. There are two broad strategies available for detecting perspectives.

(i) Based on the role played by the narrator: The system associates a narrator with the context. If the narrator is the performer in the event, the system assumes that the description is from the perspective of the narrator.

$$\text{(A86)} \qquad \text{narrator(c x)} \wedge \text{ist(c performer(e x))} \rightarrow \text{c} = \text{perspectiveMt(e x)}$$

(ii) Based on consistency of the description which might be associated with that type of perspective: For example, the system expects a certain kind of description of a driving event from the perspective of the driver, a different kind of description from the perspective of the car, and yet another description from a neutral perspective. If the description the system is given closely corresponds to one of these, it can use this to determine the perspective.

Up to this point, we have been considering only the description of specific events such as Driving001 from the perspective of one of the actors. Let us now consider the description of the general action of Driving from the perspective of the driver. We want to say that mental activity level required for Driving is high. This could be stated as,

$$\begin{aligned} \text{(A87)} \qquad & \text{ist(c allInstanceOf(e Driving)} \wedge \text{driver(e x))} \rightarrow \\ & \qquad \text{ist((perspectiveMt e x) mentalActivityLevel(e High))} \end{aligned}$$

We then simplify (perspectiveMt e x) to TheDriverPerspectiveMt which is "the term" (described in Example 11), to get

$$\begin{aligned} \text{(A88)} \qquad & \text{ist(TheDriverPerspectiveMt allInstanceOf(e Driving)} \rightarrow \\ & \qquad \qquad \text{mentalActivityLevel(e High))} \end{aligned}$$

Using this technique, not just Problem Solving Contexts, but also microtheories can use perspectives to simplify the description of an event.

3.24 Example 19: Granularity in language natural language queries

In some of the previous examples, we have considered how translations of language natural utterances, which have certain context dependencies left in them, might be dealt with in the representation system. In this example, we consider some context dependent queries posed to the system.

Consider the question "Where is Fred?". The person asking the question (assuming he is really requesting some information and not just posing a rhetorical question) might want to know

- the city where Fred is at present, or
- the building or room he is in, or
- his telephone number or e-mail address, or

- which company he works for,

  ⋮

Using purely linguistic knowledge, the NL front end can at best translate this query into something like location(Fred ?x) where we want the system to find bindings for ?x. Depending on why he is asking the question, he might want one of the above as an answer. The task he is trying to achieve forms the context for the question and needs to be taken into account when answering it.

It is not just questions of the type "where is X?" which have this problem. Questions such as "what is x?", "tell me about x", etc., also have a similar problem in that they could mean one of a fairly large set of questions, with the context providing the information for disambiguation.

There are two approaches that can be taken for obtaining the right answer to such questions.

(a) Modify the question. Rather than answering location(Fred ?x), answer the question phoneNumber(Fred) = ?x.

(b) Use a context which uses the predicate location in the same sense in which it is used in the question.

The second approach has the advantage that there is no decontextualization involved. However, there might be cases where the sense in which "location" is used in the question is too restrictive and it is not possible to answer the question with this restriction. Therefore in this example, we will take the first approach and describe how the question might be reformulated to obtain the correct answer.

As mentioned earlier, the user has some goal in mind and is asking the question in order to fulfill this goal. This goal sets up the context for this (and possibly other) question(s). So for example, if the user is trying to reach Fred by telephone, this sets up the context and the question "where is Fred" should be interpreted as "what is the telephone number where I can reach Fred?". The predicate "location" is a generalization of "phoneNumber" and may sometimes be used instead of the function phoneNumber to refer to the phoneNumber, assuming the context in which the utterance is made establishes this. If the system knows that the user is trying to telephone Fred, since the user needs Fred's phone number to call him, the system can assume as a default that by "location" he means "phoneNumber". The following axioms attempt to capture this context dependence.

We associate the action the user is trying to perform with the context by writing

(A92) contextTask(PSC Phone(User Fred)).

Phone(User Fred) is the action of the user calling Fred on the telephone. In order to perform the task of telephoning Fred, the user needs to know Fred's telephone number. This is written as,

(A93) informationRequirement(Phone(User Fred) phoneNumber(Fred))[6]

Further, the system knows that the concept of location is a generalization of the concept of phoneNumber. So,

genlSlots(phoneNumber location).

Then there is the default rule,

(A94) informationRequirement(a s(x)) $\wedge$ contextTask(PSC a) $\wedge$ genlSlots(s s1) $\Rightarrow$ ist(PSC s1(x y) $\rightarrow$ s(x) = y)

Given the query location(Fred ?x), using the above default, the system reformulates the query to phoneNumber(Fred ?x).

3.25 Example 20: Integrating Databases

The next two examples will be concerned with the use of contexts for integrating multiple databases (DBs). The problem of integrating information from different DBs has been receiving increasing interest over the last couple of years. With many organizations having numerous databases, each with its own schema and design, the problem has gone from one of not having the data to one of finding the data. In this example, we describe how a uniform interface might be constructed for a heterogenous set of databases.

Much of the work on integrating databases has been focussed on relating the semantics of different data models, E.g., relating the Relational data model with the Entity Relationship data model, etc. However, even if we restrict our attention to databases using the same data model, there seem to be significant differences which make the task of integrating them difficult. These differences are at the schema level.

Different databases use different but conceptually related schemas. E.g., A car dealership might have one database recording information about different makes and models. The objects in this data base would be things like AcuraIntegra and HondaCivic. They might have another database to record the repair and maintenance information about particular cars this dealership has sold. The objects in this database would include DougsAcura, McCarthysBMW, etc. The content of these two databases is certainly closely related, but because of they use completely different schemas, information from one cannot easily be combined with information from the other. What we need is some means of relating the schemas used by different databases, i.e. some means of writing translation rules that go from one database to another.

The approach taken by the database community has been to provide translation rules that take us directly from one database to another. However, this approach has the drawback that given a set of N databases, we have to write N^2 sets of translation rules. Our approach will be to write only N sets of translation rules. We use a single general vocabulary and translate the databases into and out of this common vocabulary. We use the vocabulary of Cyc as this general vocabulary and the mechanism of contexts to write the translation rules. These translation rules (which turn out to be nothing but lifting rules) will be called Articulation Axioms (AA).

3.25.1 Context Based Solution

Lifting rules can be used to translate sentences from one context to another. However, we are trying to translate information stored in a relational (or other) database. So, before we can write these articulation axioms, we need to have the content of the database available as a theory in a context. We now introduce the function DBContext.

Given a database D, DBContext(D) (DBCD) is a context whose theory is the information contained in D. The assertions in DBCD are logical sentences and not relational tuples (or links in an Entity Relationship database). So, if the database D contains a tuple with the information I, there will be a formula F in DBCD which states the same thing. It should be noted that the formula F is in DBCD only "conceptually", i.e., the system does not go about actually converting all the tuples from the database into formulas in DBCD. Let us first see how the translation from the database tuple to the formula F might be done. For the purpose of this example, we will consider only relational databases (extension to other data models is straightforward).

3.25.2 Database → DBContext(Database)

Let us first consider how the information in the database D could be converted into information in DBCD. The basic idea is that the assertions in DBCD should be translatable to and from D in a database independent procedure, i.e., without any knowledge about the content of the databases.

Let us assume that we have an n-ary relation R with the fields (f_1, f_2, ..., f_n) in D with the key (f_i, f_j, ..., f_l). Corresponding to every tuple TP_i in R, there exists an object TP_iO in DBCD. Corresponding to every field f_i, we introduce a binary predicate cf_i in DBCD. Corresponding to every possible entry of any field in R, we have a corresponding object in DBCD. It should be noted that the object in DBCB could be the same as the entry. If an entry e in D corresponds to the object O in DBCD, we write, dbObject(D DBCD e O). If the f_i field of TP_i contains e, in DBCD we have $cf_i(TP_i\mathrm{O}\ \mathrm{O})$. We have the following axiom in DBCD that establishes the meaning of the key.

$$\text{(A95)} \qquad \begin{array}{l} cf_i(\mathrm{x\ a}) \wedge cf_j(\mathrm{x\ b}) \ldots \wedge cf_l(\mathrm{x\ l}) \wedge \\ cf_i(\mathrm{y\ a}) \wedge cf_j(\mathrm{y\ b}) \ldots \wedge cf_l(\mathrm{y\ l}) \Rightarrow (= \mathrm{x\ y}) \end{array}$$

An atomic formula in DBCD can be converted into the appropriate database encoding as follows. Given $cf_k(\mathrm{O\ l})$, the system first obtains the formula $cf_i(\mathrm{O}\ Ocf_i)$ And $cf_j(\mathrm{O}\ Ocf_j)$ .... And $cf_l(\mathrm{O}\ Ocf_l)$ where Ocf_i is the value of the cf_i slot of O, etc. From this, by using the dbObject relation, we can obtain the key for the tuple corresponding to O in R and hence the tuple TP_n. If D_l is the object corresponding to l in D, the system can translate $cf_k(\mathrm{O\ l})$ as stating that the f_k field of TP_n is D_l.

Let us now consider an example of a database and see the mapping between D and DBCD. We will then show some articulation axioms for translating from DBCD to the general Cyc vocabulary. The particular database we will consider is one on cars. This is a database encoding of some of the information found in the Consumer Guide databook on new cars. One of the relations (FeatureTable) in the database specifies the standard features of cars sold in the United States. The relation has four fields - make, model, version, and features. Some example tuples of this relation are,

[Acura Integra GS ALB]   ALB stands for Anti lock brake
[Acura Legend LS DSAB] DSAB stands for Driver side air bag

The names of the fields are make, model, version, and feature. The DBContext for this table is called CFT, i.e., CFT = DBContext(FeatureTable). The assertions in CFT corresponding to the above tuples are

(A96) make(AcuraIntegraGS Acura) $\wedge$ model(AcuraIntegraGS Integra)
$\wedge$ version(AcuraIntegraGS GS) $\wedge$ features(AcuraIntegraGS AntilockBrakes)

(A97) make(AcuraLegendLS Acura) $\wedge$ model(AcuraLegendLS Legend)
$\wedge$ version(AcuraLegendLS LS) $\wedge$ features(AcuraIntegraGS DSAirBag)

We have dbObject(CFT FeatureTable ALB AntilockBrakes) to map from the term ALB in the database to the term AntilockBrakes in the corresponding context. For the other terms in the database, the system uses the same term in the corresponding context. It should be noted that in a database, often the same term is used in different places to denote completely different things. For example, in the Kelly Blue book database, a number such as 80, if it appears in the model-year field could stand for the model year of the car, or if it appears in the features field, could represent the fact the the car is a four seater. In such cases, the simple scheme based on dbObject is insufficient for converting the DB encodings into more meaningful terms. Therefore, we postpone the decoding and let the articulation axioms do the job.

3.25.3 DBContext(Database) $\rightarrow$ Other contexts

Now that we have the information from FeatureTable in CFT, we turn our attention to translating this into the more general Cyc ontology. As seen in some of the earlier examples, in the more expressive contexts such as the AutoMt, it does not make sense to say that something like "AcuraIntegraGS has antilock brakes". AcuraIntegraGS is a category and categories can not have brakes. That is, it is meaningful to say hasFeatures(GuhasAcura AntilockBrakes) but not hasFeatures(AcuraIntegraGS AntilockBrakes). What is meant by the database tuple is that all Acura Integra GS's have antilock brakes.

So we have the articulation axiom

(A100) ist(CFT features(car AntiLockBrakes)) $\Leftrightarrow$
ist(AutoMt ($\forall$ x allInstanceOf(x car) $\Rightarrow$
($\exists$ y allInstanceOf(y AntilockBrakes) $\wedge$ parts(x y))))

A query posed to Cyc is translated into queries to the relevant database contexts. Within a database context, the query is then converted into an appropriate SQL query. Different parts of a query to Cyc might result in different queries being addressed to different databases. The answers from these databases are then combined to give the final answer to the original query. For example, there is a Consumer Reports based database (associated with the car selection application) that deals with aspects such as the reliability of the car, etc. So a query to Cyc which searches for reliable cars with airbags might result in a query to the Consumer Guide database and the Consumer Reports based database.

A number of interesting issues arise when combining answers from different databases and these are dealt with in the next example.

3.26 Example 21: Mutual inconsistencies between databases

In this example, we extend the treatment given in the previous example to cases where the information available in the different databases is mutually inconsistent. There are different kinds of inconsistencies which might occur and these require different treatments.

The simplest (and possibly most common) kind of inconsistency is due to the closed world assumption. Consider two databases DB_1 and DB_2 which contain car features. Each of them makes a closed world assumption, i.e., if the database does not explicitly specify that a certain car has a certain feature, then the database implies that the car does not have that feature. DB_1 has no information about Acuras having air bags. So, DB_1 implies that Acuras do not have air bags. However, DB_2 states that Acuras do have airbags. Intuitively, given that DB_2 has this information, the lack of this information in DB_1 should not be regarded as stating that Acuras do not have air bags. It should simply be regarded as having some missing information. However, if neither of the databases had said anything about Acuras having airbags, then it would be reasonable to assume that Acuras did not have airbags. Clearly, the traditional approach as regards the closed world assumption (CWA) needs modification.

In the traditional formalization of the CWA, if a database D does not contain the fact F(x), then it implies ¬F(x). In our scheme, this would be equivalent to minimizing the extent of F in the DBContext corresponding to D. However, this will result in unintended contradictions between the different database contexts. Instead the system needs to make a closed world assumption for F in the common context into which the sentences in the different database contexts are lifted. In the case of the car related databases, since they are all lifted into the AutoMt, the closed world assumption is made in the AutoMt.

The next case where the system might have contradictory information is where one database (i.e., the context corresponding to the database) contains the formula p(a) = b and another contains the formula p(a) = c. We have to ensure that such inconsistency does not render the whole knowledge base inconsistent. If the articulation axioms we write are too strong, this might happen. Therefore, the articulation axioms will be defaults. A couple of the databases related to the car selection application contain the base price of cars. Let the corresponding database contexts be CDB_1 and CDB_2. The cost of an Acura Integra GS (AIG) in CDB_1 is A and in CDB_2 is B. The lifting axiom for lifting the relation price from cdb1 to the AutoMt is

$$\text{(A101)} \quad \neg\text{ab-price}(\text{car } CDB_1) \wedge \text{ist}(CDB_1 \text{ price(car)} = \text{y}) \Rightarrow \\ \text{ist}(\text{AutoMt price(car)} = \text{y})$$

In cases where we have this kind of mutually contradictory information, there are a couple of possible courses of action.

(a) One of the databases is more reliable and the information from that database is preferred. Logically, this can be stated as a priority ordering on the relevant "ab" predicate. So in this case, if we prefer CDB_1 to CDB_2, we minimize ab-price(x CDB_1) before minimizing ab-price(x CDB_2). This is stated as ab-price(x CDB_1) $<$ ab-price(x CDB_2).

In practice, a query given to a context such as the AutoMt might be translated into queries to one of many contexts. In this case, when asked for the price of a car, the system can translate this into a query to CDB_1 or CDB_2 (from where it will be translated into an SQL query). Preference information can be used at this stage to determine which DB to pose the query to.

(b) If one database claims that the price is A and another claims that the price is B, and the system doesn't have any information about which of the two is more reliable, all it can logically conclude is that the price is either A or B. However, in such as case, we might want something even weaker and conclude that the price is between A and B. The following rule states this.

$$\text{(A102)} \quad \text{ist(CDB1 price(x) = y)} \wedge \text{ist(CDB2 price(x) = z)} \wedge (\text{y} < \text{z})$$
$$\text{ist(AutoMt y =< price(x) =< price(z))}$$

(c) The more general contexts might use a variable unit of measure to "gloss over" such disrepencies. For example, the price of the car might be measured in thousands of dollars (rounded), i.e., [15k - 16K], [16K - 17K], etc. If the disrepency between the price from CDB_1 and CDB_2 is less than a few hundred dollars, and the general context measured the price in thousands of dollars, the system does not have a contradiction. If the user were simply trying to get a rough estimate of the price, this accuracy might be adequate.

Finally, the domain knowledge in the general contexts (such as the AutoMt) might cause the information in two DBs to be mutually contradictory. For example, one database might claim that a car X has cruise control and another claims that the car X has airbags. *A priori* this is not contradictory. However, the general context might have an assertion stating that a car cannot have both airbags and cruise control. Given this axiom, the system has a contradiction. If one of the databases is known to be more reliable, the information from it can be used. Otherwise, all the system can really conclude is the disjunction, i.e., either the car has airbags or it has cruise control.

Chapter 4

Current work and comparison with related work

In this chapter we first briefly describe some ongoing work using contexts and then compare the work presented here with related work in AI.

4.1 Current work on contexts

Several different directions are being taken with the current work on contexts. Two of the more interesting ones are described here. The first is the use of contexts in building device models for qualitative simulation and the second is the use of contexts for natural language understanding. The work described here is currently underway and so the discussion will be more sketchy than the other discussions in this thesis.

4.1.1 Use of Context for building Device Models

[Forbus and Falkenheiner] consider the problem of 'modeling' a device for performing a qualitative simulation. The problem they address is the following. We have a detailed description of a device (the particular device described by them was a steam power plant) and a question (or set of questions) we would like to answer. Answering the question might require performing a simulation (qualitative) of the device. However, simulating the device to the level of detail provided in the input description is both too expensive and unnecessary. Therefore, we would like to derive a partial description of the device which can then be used for the simulation. The new description may be partial in the sense that it might ignore certain components, abstract certain components as black boxes, ignore certain aspects (such as the thermal properties) of certain components, make assumptions about operating conditions (e.g., assume that it is a steady state operating condition), etc.

This problem is a generalization of the problem considered in example 14 in the previous chapter. In that example, we were concerned with setting up the model for a single query (and not a complete simulation). In that example, we were concerned only with the shape of the interior of the car and of the christmas tree. We were not concerned with modeling the whole car. On the other hand, here we are interested in modeling the whole device.

Forbus's Approach

The approach taken by [Forbus] can be briefly summarized as follows. Each axiom describing the device and its (potential) behavior is disjoined with a statement of the form (not consider(...)). So, in order for a component to be included in the partial description (model), the relevant consider statement should be true. Similarly, to include the thermal properties in the model, the relevant consider statement should be true. Of course, there are constraints between these consider statements. E.g., one cannot include the subcomponents of a component in a model without including the component itself. If a consider statement cannot be proved, it is assumed to be false, i.e. we minimize the extent of consider.

The mechanism of consider statements performs the job of contexts to a limited extent. The relationship between contexts and these consider statements is exactly like the relationship between contexts and nonmonotonic logic. The incorporation of consider statements is almost exactly like the incorporation of 'ab' literals in axioms. In fact, the problem solving technique described in that paper treats the consider statements very similar to how 'ab' literals are treated.

Context based approach

At the beginning of the previous chapter we discussed the problem of determining the set of axioms required for answering a query. There were two orthogonal issues, that of determining whether the approximations and assumptions made by a model (i.e. set of assertions) are suitable/ acceptable for the problem at hand, and of identifying the most relevant axioms required for solving the query. We come back to the same problem here, except, in this case, we are interested in obtaining the set of relevant axioms up front (as opposed to generating the set using clues provided by the problem solver.)

We outline how the mechanism of contexts might be used for the modeling task. Our approach is as follows. We introduce a new context (mc) in which we will build a 'model' of the given device that is adequate for the task at hand. The rules used for this purpose are similar to lifting rules and are called *modeling rules.*

Unfortunately, there is no simple solution of deriving the model required for a problem, i.e., there isn't a single set of modeling rules that is universally adequate. However, even at a general level, we would like to do better than just say 'use modeling rules'.

Here we present some general ideas on the topic of model generation and discuss how these can be implemented using contexts.

The Model

We are given a device consisting of a number of component devices. Each component (or set of them) can exhibit certain behaviors. For each component and each behavior, we have a set of axioms (model) describing it. The overall device model is the conjunction of the models used for the different components and behaviors.

For each device, the model specifies the components, their interconnections, the possible behaviors the device may take part in, etc. We include or exclude a component from consideration by including or excluding it from the model of the device. This is done recursively for each component. Similarly, for each behavior, the model for that behavior can decide which subactions to consider, which aspects (e.g., mechanical, thermal) of the behavior to consider, etc.

Device Model

Conventional philosophy has been to develop and use context independent models of devices [deKleer]. The models are context independent in the sense that the model used for a device is independent of the overall system it is a part of, the query being answered, etc. While this is a very laudable goal, it has certain problems. If we need a universally adequate model of a device, the model will have to incorporate all the details that any potential application might require. Not only is it very difficult to build *the* model of a device, even if we succeeded in building these models, they are likely to be detailed than is required for most applications. As a result, using these models for answering queries is likely to result in very poor performance.

Instead of using a single model for each device/ behavior, our approach will be to have several models for each device. The problem of model building is that of selecting the right one of these models for each device. The conjunction of the appropriate model for each component /behavior then forms the model of our overall device.

While this approach allows us to build models of varying granularities, etc., it does complicate matters significantly. Not only do we have to build more models, we also have to select the appropriate model for each device.

Constraints between different models of a device

The different models of a device are not completely unrelated sets of axioms. There are some very strong constraints between these different models which might allow us to derive some of the models of a device automatically from others.

There are also constraints between the models used for the different components of a device. One cannot arbitrarily pick models for the different components and get a meaningful model of the overall device. We can exploit these two sources of constraints to simplify the model building task.

If we look at the different models of a device, some of the models are usually more detailed or accurate than the others. The cruder and simpler models can often be derived from these more accurate models. We certainly have to manually construct the accurate models. However, the construction of the less accurate models might be at least partially automated. In general, the approximate models can be derived from the accurate ones by incorporating assumptions. (Though the introduction of assumptions might allow us to use much simpler representations, discovering these simpler representations and reformulating the representation is a very hard problem [Devika] and is beyond the scope of this thesis.) Based on the assumptions incorporated, we can derive many different approximate models. Some issues related to this are as follows.

(i) Assumptions : We can distinguish between the following two kinds of assumptions.

- **(a)** Assumptions that are consistent with the accurate model. E.g., the accurate model of a steam plant might include a parameter for the variation in the temperature (of the steam) across the cross section of the steam outlet pipe (the steam at the periphery of the pipe is typically a little cooler.) We can assume that this variation is Zero. While this does simplify the model, it is perfectly consistent with the more accurate model. Such assumptions are usually easy to incorporate. However, determining when it is reasonable to make the assumption is a different question altogether. [Weld] provides a theory which provides some conditions to determining when it is reasonable to make these kinds of assumptions.

(b) Assumptions that are inconsistent with the accurate model. E.g., the accurate model of the steam plant might specify that the capacity of the boiler is X cubic meters. If X is is sufficiently large, one might assume the the boiler has infinite capacity. The accurate model might specify the existence of certain valves in the steam path (resulting in pressure loss which in turn complicates calculations). We might decide to ignore the valves, i.e., assume they don't exist. This is directly contradictory with the information in the more accurate model. When an assumption contradicts the more accurate model, some of the axioms in the more accurate model will have to be retracted to ensure internal consistency of the approximate model. In the general case, there might be more than one way of restoring consistency. In such cases, it might be worthwhile to ask a human to fix the problem.

(ii) In principle, any formula could be regarded as a potential assumption that could be added to an accurate model to derive an approximation. However, in practice, there are a relatively small number of kinds of assumptions that are useful. Some of these include,

(a) Letting a variable (i.e., a numeric valued non-atomic term) take an extreme value. [Weld] provides a nice analysis of when the value of a variable may be assumed to take on an extreme value.

(b) 'Regularizing' some property. E.g., assuming that some object has a regular shape (cuboid, cylinder, etc.) when it is not. Assuming that some curve is smooth and/or has a regular shape when it is not.

(c) Ignoring components.

(d) Ignoring certain behaviors (such as transients.)

Later, when discussing how this scheme might be represented using contexts, we will see how having these standard kinds of assumptions helps.

(iii) Certain sets of assumptions go together. Making one assumption might preclude us making certain other assumptions (even if the model does not become inconsistent). E.g., we might ignore the mass of an object (if we are considering it purely for its geometric properties) or the volume (if we are interested purely for its dynamics). But ignoring both the mass and the volume (without ignoring the object itself), does not make sense.

Automating device model generation

We first take up the issue of automating the generation of some of the coarser models from the more accurate ones. This can be at least partially automated for the models generated by making one or more of the standard assumptions. We have a kind of device DT. The microtheory CDT0 contains an accurate model of this kind of device. From now on, we will refer to CDT0 (or more generally, the context containing the model) as the model. We have,

(a) If f(d) (d is the device) is a variable in CDT0, and ef is an extreme value for this variable, fixVariable(CDT0 f ef) is an approximation model that assumes f(d)=ef. As a default, everything that is true in CDT0 is true in fixVariable(CDT0 f ef).

ist(CDT0 allInstanceOf(d DT)) $\Rightarrow$ ist(fixVariable(CDT0 f ef) f(d)=ef)

ist(CDT0 p) $\rightarrow$ ist(fixVariable(CDT0 f ef) p)

Note that the second axiom is a default, but the first is not.

This can also be used to define approximation models where certain properties of objects are 'regularized'.

(b) If CDT0 predicts that each DT has a component which is an instance of DT1, then an approximation model can be obtained by ignoring this component. This approximation model is ignorePart(CDT0 DT1).

ist(ignorePart(CDT0 DT1) allInstanceOf(d DT) $\Rightarrow$
$\neg$(Exists d1 allInstanceOf(d1 DT1) and parts(d d1)))

ist(CDT0 p) $\rightarrow$ ist(ignorePart(CDT0 DT1) p)

(c) If a set of properties of the device (such as f1(d), f2(d) ...) form a coherent set (such as thermal properties, electrical properties), etc. we write aspectType(f1 ThermalProperty), aspectType(f2 ThermalProperty), etc. We can decide to completely ignore a certain aspect of the device. What does it mean to ignore an aspect or more specifically, to ignore the property f1 of the device? It means that we don't (and cannot) give the device that property in our model. The approximation model obtained by ignoring the aspect A is ignoreAspect(CDT0 A).

ist(CDT0 allInstanceOf(d DT) And aspectType(f A) $\Rightarrow$
$\neg$ist(ignoreAspect(CDT0 A) (f(d) = x))

ist(CDT0 p) $\rightarrow$ ist(ignoreAspect(CDT0 A) p)

Using the above functions and axioms we derive a set of more approximate models from the accurate model. Note that the above functions can be applied recursively. So, not just ignorePart(CDT0 DT1) but also ignoreAspect(ignorePart(CDT0 DT1) A), etc. also define models.

The model CDT0 is related to ignorePart(CDT0 DT1), fixVariable(CDT0 f e), etc. by the relation approxModel. So we have,

- approxModel(CDT0 fixVariable(CDT0 f e)),
- approxModel(CDT0 ignorePart(CDT0 DT1)) and
- approxModel(ignorePart(CDT0 DT1) fixVariable(ignorePart(CDT0 DT1) f e))

$\vdots$

The relation approxModel is transitive and so we have,
approxModel(C_1 C_2) $\wedge$ approxModel(C_2 c3) $\Rightarrow$ approxModel(C_1 c3).

As stated by some of the above axioms, as a default, the assertions from the more accurate model get lifted into the approximate model. However, there are cases when we want this to be more than just a default. There might certain aspects of the accurate model that have to be part of even the approximate models. If p is such an assertion, this is enforced by writing axioms such as the following.

ist(c p) $\wedge$ approxModel(c ci) $\Rightarrow$ ist(ci p)

Specifically, if an approximate model ci is derived by making a certain assumption, this assumption should not be violated in further approximations derived from this one.
approxModel(fixVariable(c f e) ci) $\Rightarrow$ ist(ci allInstanceOf(d DT) $\Rightarrow$ (= f(d) e))

To say that a context c is a model of a device type DT, we say, deviceModel(DT c). Finally, we minimize the extent of deviceModel so we assume that we know of all the models there are.

Many of the models defined above will not be very useful. For example, the model fixVariable(fixVariable(CDT0 f a) f b) is inconsistent since it assigns both a and b as the value of f(d). We might also incorporate axioms to ensure that models we use are plausible. E.g., we might insist that if a model considers mechanical aspects of device (i.e. mechanical aspects is not ignored), then an object cannot be assumed to be without both mass and volume.

aspectIncluded(ci MechanicalAspect) $\Rightarrow$
ist(ci (not (mass(x)=0)) and volume(x)=0)

The incorporation of such modeling heuristics will also make a number of models internally inconsistent. If a model is internally inconsistent, we can derive False in this model (actually in the corresponding context.) We will ignore such models.

Building the model for the Simulation

We go about building the model for the problem solving activity as follows. Given that the particular device D is an instance of DT in the context psc, we want to build the appropriate model (of D) in psc. We first pick a model of DT that we use. This makes predictions about what components D has. We then incorporate an appropriate model for each of these components. This process is carried out recursively.

ist(psc allInstanceOf(d DT)) $\Rightarrow$
(Exists ci deviceModel(DT ci) and baseMt(psc ci))

$\text{baseMt}(C_1\ C_2)$ and $\text{ist}(C_2\ p) \Rightarrow \text{ist}(C_1\ p)$

Note that the above rule applies recursively. Using it once gives us the parts of the device. The rule can again be applied with these parts. We want to exclude the 'bad' models, i.e., models with internal inconsistencies. So,

ist(c False) $\Rightarrow$ (not baseMt(ci c))

If a model is excluded, so are models which are approximations of that model.

(not baseMt(ci c)) and approxModel(c cj) $\Rightarrow \neg$baseMt(ci cj)

Model selection heuristics

The above axioms only tell us that we should pick some model for each kind of device. It does not address the crucial question of determining which model we should pick. The approach we take is to write axioms for eliminating certain candidate models and for ranking models in a preference ordering. We now consider some factors influencing this. We use the most preferred model that has not been eliminated.

(a) We should not assume the answer away. So if the problem solving task is to determine the weight of an object, we should start off assuming that the object is massless! So if the assumptions behind a model answer the query, that model should be eliminated.

(b) As in example (11), we might be able to associate error estimates with models (or even assumptions). Based on this and the precision required in the answer, we might be able to eliminate some of the models.

(c) Using a particular model might not give us any answer, i.e. if we want to know whether Q is true, a model M might imply neither Q nor (not Q). Such models are eliminated.

(d) We prefer simpler models. If we have approxModel(c0 C_1), we prefer C_1 to c0.

4.2 Use of Contexts for Natural Language Understanding

In this section we discuss some issues related to the application of contexts in natural language understanding.

Natural language (NL) utterances exploit context in the most obvious fashion. Many of the recognized semantic problems of NL processing involve using a context to determine the meaning of an utterance. In this section, we will outline a framework for using the representation machinery of contexts to help with natural language understanding.

4.2.1 Utterances vs Sentences

At the outset, I should clarify that by the term NL understanding, I refer to the problem of understanding the meaning of utterances and not the problem of understanding the meaning of sentences taken in isolation. The distinction is an important one because understanding utterances - sentences (or fragments of sentences) uttered in particular situations to convey something - is in some ways harder but in many ways easier than understanding sentences considered in isolation. Let us first consider some issues involved in understanding utterances and distinguish this from understanding sentences.

4.2.2 Understanding Utterances as Constraint Satisfaction

There are a number of constraints imposed by a situation on what might possibly be conveyed by a speaker to a listener. In some cases these constraints are enough to sufficiently narrow down the range of possibilities so that a simple nod or gesture suffices to convey the intended information. In most cases something more complex is needed and a sentence may be uttered. This sentence in conjunction with the other constraints conveys the intended information.

Example : Consider the utterance 'lets talk about the weather in California'. It is ambiguous as to whether 'in California' modifies where the talking should be done or whether it modifies the weather. Now we might happen to know something about the speaker and listener or what they were talking about earlier that might help disambiguate this. If we knew that one of them was thinking of moving to California, we would know that the phrase 'in California' modifies the weather. On the other hand, if they were planning on meeting in California and were drawing up an agenda of things to do there, then it probably modifies the planned talking event.

The problem of understanding an utterance is easier since there are many constraints that can be used to determine the information conveyed by the utterance. The problem is harder because of the need to integrate constraints from many different sources. It is difficult to specify *a priori* the sources and types of constraints that may need to be considered. Therefore we need a framework for integrating these different constraints. We already have a tool for dealing with a very wide variety of information, i.e., the knowledge base. We will try to adapt this for integrating constraints on the information conveyed by an utterance.

The traditional model of NL understanding (NLU) has been that the NL front end takes natural language utterances as inputs and produces completely decontextualized logical sentences as outputs. As we have repeatedly claimed in the previous sections, completely decontextualized formulas are very difficult to come by even when a human is manually writing them. Expecting the natural language front end to produce these using only linguistic information (possibly with the help of some questions the natural language front end asks the KB) might be setting up too difficult a task for it to perform. If the natural language front end were translating into a language in the logic of contexts, some of these problems might be handed off to the KB to be dealt with (at a later time).

The previous chapter gave examples of encoding and using theories which made contextual assumptions. Most of these assumptions were fairly coarse grained in nature. The only exception to this was the use of "the terms" explained in example 11. In this section we take the approach taken in that example to an extreme and exploit contexts for resolving some relatively fine grained context dependencies.

4.2.3 Framework for Context based NLU

The overall framework is as follows. We have a hierarchy of contexts based on granularity. At the lowest level, we have a set of "utterance contexts" one for each utterance. These are ordered based on the temporal ordering of the utterances. Given an utterance, the NL front end creates a new utterance context and asserts the translation of the utterance into this new context. At the next level, we have a "discourse context" which corresponds to the discourse of which the utterances are a part. The formulas in the utterance context are partially decontextualized and lifted into the discourse context. Depending on the kind of contextual assumptions made, the discourse context might also be a microtheory and/or a problem solving context. In the previous chapters we discussed decontextualization of assertions in microtheories and problem solving contexts. Here we discuss some issues associated with the decontextualization of utterance contexts.

We assume the natural language front end, using the lexicon and purely linguistic knowledge about grammars, etc., will be able to rephrase the NL utterance as a formula. This formula might be heavily context dependent in a manner we will illustrate shortly. There might be multiple parses of the given NL sentence. In this case, there are two alternatives. The first is to use a formula that is an exclusive-or of the formulas corresponding to the different parses. The other is for the NL front end to rank the different parses and use the formula corresponding to the most likely parse, allowing for the KB to later ask it for an alternate translation. The implementation currently underway uses this second strategy.

The language used by the utterance contexts allows for a number of contextual dependencies. We now briefly examine some of the vocabulary of this language.

(a) Pronouns and Indexicals: He, She, It, Now, I, etc. are terms in the language.

(b) Definite and indefinite references: The language includes the functions The and A. The function A is similar to the article A. The sentence "the lady owns a bag" would be translated into owns((The Lady)(A Bag)).

(c) We use a variadic function Etc. The sentence "Fred likes ice cream, softees, etc." would be translated as likes(Fred (Etc IceCream Softee)).

(d) We have a set of predicates such as "to", "with", "for", etc. The sentence "Fred bought the rose for Jane" would be translated as, (exists e allInstanceOf(e Buying) and object(e (The Rose)) and for(e Jane)).

(e) The formulas might refer to the lexicon. Given the sentence "Joe is at the bank", if the NL front end feels overwhelmed by the number of possible denotations of bank, this sentence may be translated as (exists c at(Joe (The c)) and englishWord(c "bank")).

(f) A formula might refer to other utterance contexts. Reference to an utterance context might be used to specify something about the context itself or to somehow qualify a previous utterance.

The following are more examples of utterances and their corresponding translations in the language of the utterance contexts.

i) "I want to send a package to Karen. Where is she?"

(want I ($\exists$ e allInstanceOf(e Sending) $\wedge$ object(e (A Package)) $\wedge$ to(e Karen)))
(location She ?x)

ii) "Fred bought a ticket to the show. He lost it."

$(\exists$ e performedBy(e Fred) $\wedge$ before(e Now) $\wedge$ allInstanceOf(e Buying) $\wedge$
object(e (A Ticket)) $\wedge$ to(e (The Show)));
$(\exists$ e performedBy(e He) $\wedge$ before(e Now) $\wedge$ isa(e Losing) $\wedge$
object(e it))

iii) "He bought a ticket to give Mary"

$(\exists$ e performedBy(e He) $\wedge$ before(e Now) $\wedge$ isa(e Buying) $\wedge$
object(e (A Ticket)) $\wedge$ to(e (Give Mary)))

iv) "A dog has an appendix. It does not use the appendix."

has((A Dog) (A Appendix));
$\neg(\exists$ e performedBy(e It) $\wedge$ uses(e (The Appendix)).

There might be no simple correspondence between the formulas and the elements of a natural language (sentence, word, etc.), i.e. one wff could be the translation of a single word, another of a sequence of sentences, etc. The intuition is that using techniques such as context free grammars, we might be able to regularize the syntax of the english sentence to obtain a sentence with a simpler syntax.

Consider an utterance such as "The lady owns a car", which is translated as ist(c owns((The Lady) (A Car))). Resolving the referent of (The Lady) can be formulated as finding the binding for ?x in (= (The Lady) ?x). Any information that is representable in the KB can be used to solve this. The contrast between this approach and the more algorithmic approaches (where the semantics of a definite reference is an algorithm for determining the referent) is most clearly illustrated in the case where the logic (encoding the information) is undecidable. In such cases, in our approach, we can still talk about the referent entailed for (The Lady) though we might not have a general algorithmic equivalent.

The term "the lady", translated as (The Lady), might be used in a later utterance to refer to some other lady. To avoid problems, we have to ensure that these two utterances have different contexts associated with them. In fact, a single utterance might use the same term to refer to different things. Consider the sentence "he put it on it". There are contexts in which this makes perfect sense with the two occurrences of "it" denoting different objects. Since it is important that a context use each term to denote exactly one object, we need to ensure that the formula obtained by translating a single utterance does not use the same term to denote different individuals. Because of this restriction, what we consider an utterance need not correspond exactly to a single natural language utterance. If the NL front end produces the formula F as the translation of a NL utterance U, we assume that the all the subexpressions of F have the same dependency on the context, i.e., F does not use a term such as (The Lady) twice, intending to denote different individuals in different uses of that term.

Though all the examples given in this chapter are of descriptive statements, the theory outlined here is not intended to be limited to descriptive statements and should be applicable to other kinds of speech acts as well. Since the issues we are concerned with have little to do with the kind of speech act with which one is dealing, to simplify the presentation I shall be considering only descriptive utterances.

4.2.4 Utterance Contexts vs Discourse Contexts

Since the scope of the discourse context is much larger than single utterances, the language used by it does not include the above features. Therefore, when lifting a formula from the utterance context to the discourse context, the formula might need to be significantly changed. More specifically, the language of the discourse context will not have any of the earlier mentioned features of the utterance context. Many different sources of information might affect the translation from the utterance to discourse context. To illustrate this, let us examine some of the different types of heuristics that might be used to determine the referent of 'it' and try to formalize these. We will then show a detailed trace of the derivation of the denotation of "it".

Constraints on resolving 'it'.

(a) Linguistic semantic information.

(i) The denotation of 'it' cannot be a male or female human.

(forall c_i utteranceContext(c_i) $\Rightarrow$
ist(c_i (forall y (= y it) $\Rightarrow$ $\neg$(male(y) or female(y)))))

(b) Linguistic context information

(ii) The denotation of it is an object present in one of the previous contexts.
(forall x ist(c_i (it = x)) $\Rightarrow$ (Exists c_j allPreviousUC(c_i c_j) and presentIn(c_j x)))

(iii) The total number of possible referents of 'it' is to be minimized. We use a slightly more general heuristic and minimize the extents of the utterance contexts. i.e., the utterance contexts use a domain closure assumption. This reduces the existential in (ii) to a disjunct.

Every object referred to in an utterance is put into the domain of the corresponding context. Later we will consider other factors which might require us to add an object to the domain of an utterance context.

(iv) Given two candidate denotations, the one which occurs in a closer context is preferred. This is an example of something we will be trying to do fairly regularly. Given a set of possible solutions, we want to induce a partial ordering of them and then select the best fit. In this case, the problem for which we are seeking a solution is, (= it ?x). In such a case, i.e., where there is only a single solution, we use the following encoding strategy.

- first specify the set of possible candidates and assert that any of them can be the referent (i.e., is the referent as a default).
allPreviousUC(c_i c_j) and presentIn(c_j x) $\wedge$ $\neg$ab-it(c_i c_j x)) $\Rightarrow$ ist(c_i x = it)
It should be noted that this implies the axiom in (ii).

- specify the partial order as the prioritization (or preference) ordering on the default. So in this case,

$$\text{allPreviousContext}(c_i\ c_j) \text{ and } (\text{not allPreviousContext}(c_j\ c_i)) \Rightarrow \\ \text{ab-it}(c_i\ c_x x) < \text{ab-it}(c_j\ c_x y)$$

There might be more straightforward ways of representing this.

Note that allPreviousContext is the transitive closure of previousContext and is reflexive.

The heuristic stating that the reference in the more temporally proximate context is preferred has several deficiencies. The most glaring one is that it does not take into account any structure the discourse might have and instead relies on a purely temporal ordering. Given a set of utterance contexts $\{ uc_1, uc_2, \ldots, uc_n \}$, in addition to the temporal order on them, there might be other useful structurings which might be imposed on these. For example, the first three utterances might be about some topic, the next four a digression, and the next three back on the first topic. In such a case, the logical precedent of the eighth utterance is not the seventh one but the third one.

In addition to allPreviousContext, other relations, such as those described in [Grosz&Sidner] might be used to structure the utterance contexts. The structure of the discourse itself may be a topic of an utterance in that discourse. "Cue statements" such as "Going back to ..." are translated as assertions about the structuring of the utterance contexts.

Introducing a new partial order of contexts (call this one priorContext) is quite easy in this framework (we simply introduce a new relation that can hold between contexts) since contexts are objects in the domain of discourse. However we have to figure out some way of deducing this new partial order. We can assume that as a default, priorContext is the same as previousContext. So we have previousContext($c_i c_j$) And not ab-pc($c_i c_j$) $\Rightarrow$ priorContext($c_i c_j$). However, one of the effects of the utterance could be to change what the priorContext of the next utterance is. So (in computer-deutsch) we can have an axiom like (where P denoted the statement - "Going back to ...")

$$\text{ist}(c_i\ P) \wedge \text{contextTopic}(c_j\ X) \wedge \neg(\text{exists } c_k \text{ contextTopic}(c_k\ X) \wedge \\ \text{nearer}(c_i c_k c_j) \rightarrow \text{priorContext}(c_i c_j)$$

It is interesting to note that it is not just the prior context that can be changed by an utterance. In this framework, it is possible to make statements that will completely change the way following statements are to be interpreted. I.e., it is possible to use language to set up new 'language games'. If we included descriptions of actions that the speaker and hearer performed as the discourse proceeded, it might be possible to write axioms to predict the changes in the language (and maybe even predict how things like strategies for referring to objects change) as the conversation proceeds (based on whether the hearer does want the speaker wants him to do, etc.)

(c) Backgrounds of the speaker and listener. This encompasses a very wide variety of heuristics and we shall consider only a simple example. Consider the statement, "The transformer in my amplifier is broken. How do I fix it?" If this statement were being made to a transformer repair person, the "it" might be taken to refer to the transformer. However, if the statement was addressed to the sales representative of the shop where the amplifier was bought, the "it" would probably refer to the amplifier.

(d) The situation of the utterance. This might bring objects into the domain of the utterance contexts. E.g., the speaker and listener are working together to fix a car. The listener has a screw driver in his hand and the speaker needs this. If the speaker says "Give it to me", the "it" probably refers to the screw driver (even though it might never have been explicitly referred to). In general, if a set of people are working together on something, there are objects of shared focus and these are included in the domain of the utterance contexts as valid denotations of "it".

(e) Felicity constraints. We assume that utterances obey the conversational postulates. E.g., as a default, they provide information the listener does not know. Consider the statement "see that bird on the tree - it is made of wood". The "it" might refer to the bird or the tree. However, the interpretation which assigns the denotation as the tree does not yield any new information. We expect the listener to know that trees are made of wood. However, since birds are not usually made of wood, the other interpretation, that the bird is made of wood, would indeed provide the listener with new information and we prefer this interpretation.

(f) Background common sense. This is probably the biggest and most important source of constraints. Let us examine where this fits in the translation process. Consider the utterance, "he brought me a cup of coffee. I drank it." Common sense tells us that the "it" does not refer to the cup, but to the coffee. Let the translation of "I drank it" be P and the corresponding utterance context be c_p. When P is lifted from c_p to the discourse context c_d, linguistic and other constraints identify two possible candidates for the denotation of "it" - the cup and the coffee. The theory of c_d includes among other things, basic common sense information (only liquids can be drunk, cups are solids, ...) If we picked the cup as the denotation of "it", P would be contradictory with the other information in c_d. So the lifting process eliminates this as a candidate and we are left with only possibility of the coffee being the denotation of "it".

Common sense may be used not just to eliminate potential interpretations but also to impose a preference ordering on them. For example, consider the utterance - "Mary liked the watch in the shop. She went back and bought it." Based on common sense we know that it is much more likely that Mary would buy the watch than that she would buy the shop and this information can be used to order the likelihood of "it" referring to the watch and not the shop.

We now present a very simple example of the resolution of 'it'. This example is intended to be merely illustrative of the steps involved in determining the referent of "it" and does not claim to include an adequate theory of "it".

Example

We have the following two utterances.
1) Bobo the pig is in that cage.
2) It is asleep.

We need to determine that the 'It' in the second utterance refers to the pig and not the cage.

The first utterance corresponds to the wff

(U1) $\quad$ in(Bobo Cage1) $\wedge$ Pig(Bobo) $\wedge$ Cage(Cage1)

The second utterance corresponds to the wff

(U2) $\quad$ asleep(It).

I will refer to asleep(x) as p(x). I use another predicate Living. Living(x) means that x is alive. Living(x) will be abreviated to q(x).

U1 holds in the context c_1 and U2 holds in the context c_2.

There is a more general context DC and we lift the axioms from c_1, c_2, etc. to DC. In lifting U2 (from c_2 to DC), we want to replace the symbol *It* with some symbol that in DC denotes what *It* denotes in c_2, i.e. we need to determine the referent of "It'' when lifting from c_2 to DC. This is because there could be other context (at the level of c_1 and c_2) which also use the symbol *It*, but to denote different objects.

In the above case, the "It" could denote the pig or the cage.

The relation between c_1 and c_2 is priorContext(c_2 c_1).
The relation between c_1 and DC is discContext(c_1 DC).
The relation between c_2 and DC is discContext(c_2 DC).

The following axioms are stated in the context OC which is the outer context to c_1, c_2 and DC.

1. ($\forall$ cc pc ($\exists$ x priorContext(cc pc) $\rightarrow$ presentIn(pc x) $\wedge$ ist(cc (= x It)))

   This says that in every utterance context which has some previous context, there is something that was in the domain of the previous context, that can be referred to using "It".

2. ($\forall$ cc c_1 c_2 priorContext(cc c_1) $\wedge$ priorContext(cc c_2) $\rightarrow$ (= c_1 c_2))

   There exists a unique previous context.

3. Minimize presentIn.

4. ($\forall$ ci cj x discContext(ci cj) $\wedge$ ist(ci p(x)) $\rightarrow$ ist(cj p(x)))

   Actually, the DCR will imply this. But lets take it as given here. Similarly, we have axioms for Pig, in and Cage.

5. ist(DC $\neg$q(x) $\rightarrow$ $\neg$p(x))

6. non-living things can't be asleep.

7. corefer(DC Bobo c_1 Bobo), corefer(DC Cage1 c_1 Cage1)

   c_1 and DC use the same names to refer to Bobo and Cage1. Again, just the DCR.

8. presentIn(c_1 Bobo), presentIn(c_1 Cage1)

   By the definition of presentIn and 6.

9. ($\forall$ x presentIn(c_1 x) $\rightarrow$ ((= x A) $\vee$ (= x B))

   From the minimization of presentIn and 7.

10. ist(DC in(Bobo Cage1) $\wedge$ Pig(Bobo) $\wedge$ Cage(Cage1))

    From U1 and 4.

11. ist(DC Pig(x) $\rightarrow$ q(x)), ist(DC Cage(x) $\rightarrow$ $\neg$q(x))

12. ist(DC $\neg$q(Cage1) $\wedge$ q(Bobo))

    From 9, 11 and 5.

13. ($\forall$ x ist(c_2 (= x It) $\rightarrow$ ((= x Bobo) $\vee$ (= x Cage1)))

    From which ($\forall$ x ist(c_2 (= x It) $\rightarrow$ ist(DC (= x Bobo) $\vee$ (= x Cage1))))
    From 1, 6, 9 and prevContext(c_2 c_1)

14. ($\forall$ x ist(c_2 (= x It)) $\rightarrow$ ist(DC asleep(x))

    From 4.

15. ist(DC (asleep(Bobo) $\vee$ asleep(Cage1)))

    From 12 and 13.

16. ist(DC asleep(Bobo)) from 15 and 10.

Comparison to other work in NLU

This whole picture of an utterance being interpreted in a context and changing the context seems very similar to Kampfs discourse theory. There are however the following significant differences.

(a) Unlike in discourse theory (DT), the contexts being referred to here are objects in the domain of discourse and not merely objects that store state in the translation process. DT treats contexts as data structures used in the process of understanding for obtaining the translation of an utterance. This theory treats contexts as part of the translation and not just as an appendix to the translation to be used in translating the next utterance.

(b) The discourse sentences of DT primarily allow for information about which objects were referred to, etc. in one utterance to affect the interpretation of the next utterance. The interpretation of an utterance is still independent of the actual content of the utterance of the following utterance. However, in this scheme, each utterance could have an almost arbitrary impact on the interpretation of the next utterance. In fact it's not only the previous utterances but it is almost about any information in the KB that can be used in the lifting process.

(c) The contextual aspects of the utterance on which DT focuses are quite limited. They deal mostly with issues such as pronominal reference, anaphora, etc. That is, they restrict their attention to contextuality of a relatively fine granularity. The context based scheme on the other hand attempts to address (and exploit) contextuality at a wide range of granularities.

(d) DT still has the notion of *the correct* translation of an utterance, i.e. the contextuality is seen as a problem and is considered as something we can eliminate completely - there is no concept of a translation that retains certain contextual elements. It is in this aspect that the proposal in this thesis most radically differs from traditional approaches. Not only are utterances assumed/allowed to have a much greater contextual aspect to them, but the translations produced retain some of this contextual aspect.

The theory of discourse structures (DST) proposed by Grosz and Sidner is slightly closer to the formalism proposed here though its focus is almost entirely orthogonal to that of the DT theory. The following are some of basic differences between the DT theory and that of Grosz and Sidner (DST).

a) As with DT, DST treats contexts (or discourse structures) as data structures used in the process of determining the translation and not as part of the translation.

b) DST is a theory of the structure of discourses and not a theory of how the different aspects of the context affect the meaning of an utterance. Here we are concerned more with the meaning of utterances than with the structure of discourses. At certain points we shall introduce some concepts related to the structure of a discourse in order to determine aspects of the meaning of an utterance. Even here, these aspects of discourse structure are meant largely to illustrate the framework and should not be interpreted as constituting a theory of discourse sentences.

c) Though Grosz and Sidner point out that understanding a discourse is much like a constraint satisfaction procedure, they do describe how different sources of constraints can be put together to obtain the meaning of an utterance. One of the primary goals of the work described here is to provide a framework for doing this.

The material presented here can be considered meta to DST in the sense that it should be possible to formalize DST in the framework provided here. Since there are too many fundamental differences between the framework presented here and DST, I will resist the temptation to draw analogies between some of the concepts presented here to those in DST.

4.3 Related work in AI

The concept of context as a representation tool was first mentioned by McCarthy in [jmc 59]. Though not much work has been directed towards the vision of contexts outlined in that paper, there have been a number of pieces of work that are related to the concept of contexts developed in this thesis. In this section, we compare contexts to some of these.

Contexts and ATMS [deKleer] A context in an ATMS is a consistent set of propositional assumptions. This is similar to our notion of contexts only in the most general way which is that both deal with assumptions. In addition to the assumptions being propositional in an ATMS (which is not the case in our system), they are also *a prori* enumerated in an ATMS (which is also not the case in our system.) Finally, the ATMS does not concern itself with issues such as relative decontextualization which are of prime concern to us.

FOL and Descendents [Weyrauch, Fausto] FOL and its descendents are similar to our system in that both support the idea of having multiple theories with certain theories being "meta" to the others. The main differences are that the "meta" theories and the "base" in FOL do not share any ontology. The meta theory is concerned with the expressions in the base theory. As a result, concepts such as quantification across theories (a tool which we made extensive use of in the previous chapter) are not meaningful in FOL. Finally, FOL (and its descendents) are not concerned with the issue of decontextualization.

Packages [Steele] In a crude sense, packages are meant to do for programming what contexts are supposed to do for representation. Packages in common lisp are similar to our notion of contexts in that different packages might have different symbols and the same symbol might have different values in different packages. However, being a programming language construct, packages are not concerned with issues such as assumptions or relative decontextualization.

Paritioned Semantic Nets and its descendents Following [Henricks], many "representation systems" have provided tools for having multiple theories (including KEE, MRS

[Russell], etc.) This is the only similarity they share with our notion of contexts. The theories in these systems are really very separate beings and there is no interaction between them. Consequently, the concepts of assumptions behind theories and relative decontextualization have not been dealt with by these systems.

Forbus The work by Forbus is probably the closest in spirit to the work described here. In the previous chapter we compared contexts to his "consider" statements. The similarity between contexts and his work is that both attempt to capture the assumptions, etc. made by a theory. The differences are that his approach is to associate assumptions with individual statements while we associate them with theories. He does not deal with the problem of the assumptions changing the representation to a point where the assumption cannot be stated. Finally, since contexts are not explicitly represented in his system, he does not address the problem of relative decontextualization.

Chapter 5

Appendix A : Sample session with the Car Selection Application

This section contains an annotated script of a session of the car selection application.

5.1 The Interface

The application uses the graphical interface shown in the accompanying diagram. The interface has 5 panes for the following functions.

(a) The top left pane contains a list of questions the user may answer, in any order. Based on the answers he provides, it might become useful to ask him certain new questions, certain questions might become redundant, etc. So this list is continually updated. Beneath (or beside) each question is a list of possible answer templates. Symbols in angle brackets $(< \ldots >)$ are blanks which need to be filled in as part of the answer.

(b) The top right window is used for two purposes.

- **(i)** If the user would like to get a sample of the possible entries for filling in a blank in an answer template, he can click on the blank and a list of possibilities is displayed in this window.
- **(ii)** This window is also used for presenting summaries of the information available in the databases on a type of car.

(c) The left bottom window is used to record the statements made by the user. Most or all of such statements are made not by typing them in, but rather by clicking on a template (in [a]) and filling in the blanks (which itself is usually done by clicking on a blank, then clicking one of the choices displayed in [b].) A statement may be retracted by clicking on it in this window.

(d) The right bottom window is used to display the current list of candidate types of cars. It starts off with just the entry Automobile. As the system accumulates more constraints on the car, the list of candidate cars is generated and pruned.

(e) The bottommost window is an interactor used for accepting answers. (e.g., typing in a number to fill a blank.)

5.2 Notes on this script

Since the medium being used in the presentation in this thesis is not as dynamic as a computer screen, the script below is really just a record of what the system does. The script will consist of a sequence of answers provided by the user. For each, we specify the new questions that are raised and the changes in the possible selection of cars.

Also, since the application is not meant for demoing to an AI audience, much of what the system does is hidden to the user. This is especially true for the context related inferences. As the user provides information, the system draws a number of conclusions about the user, the car, and other relevant objects. Some of these conclusions will also be presented along with each answer.

Finally, this script is intended to be a supplement, not be a discussion of contexts and lifting. Hence discussion will be kept to a minimum and pointers to the relevant sections of the previous chapters will be provided.

For each answer, the question that is being answered and also the assertion generated by the interface as the answer (i.e. the translation of the key clicks, etc.) is provided .

5.3 More on Questions

Each question is a structure with the following parts.

Question Template : The english phrase to be used as the question. This phrase may include variables.

Question Formula : A formula which must be satisfied by by bindings used for the variables in the question.

Precondition : A formula which is the precondition for the question to be asked. This formula must be satisfied for the question to be posed to the user.

Answer : A set of answer templates. Each includes the english template and the formula which is the translation of that template.

As soon as the variables in the question part can be instantiated, the preconditions are checked. If the preconditions are satisfied, the question is added to the question pane of the interface. Similarly, if the preconditions stop being true, the question is retracted. An example of a question is given below.

Question *where-does-primary-driver-live*

Template : "Where does primarydriver live?"
primary-driver is a variable that needs to be instantiated.

Formula : (primaryDriver *the-car* ?primary-driver)) for generating binding for primary-driver. *the-car* is a special variable bound to the selectionObject for the current problem solving context. See example 7 for details.

Precondition : (FalseOrUnknown (residesInGeographicRegion primary-driver ?place))

Answer : The answer consists of two parts.

Answer Template : "She/he lives in ?US-state"

Answer Formula : (residesInGeographicRegion primary-driver ?US-state) Here ?US-state is the variable that is to be supplied by the user.

5.4 Initialization of the application

To initialize the application, the following are done.

1. Objects (units) are created for the problem solving context corresponding to the current session, the user, the car, the buying event and the discourse.
2. The different databases on cars and list of possible questions are loaded.
3. The interface is initialized.

In the session given below, the user's name is Chris. Chris-Car is the hypothetical car he is thinking of buying. Even before we begin the actual session, a number of conclusions have been drawn about the user and the car. These units corresponding to the user, the car and the problem solving context look like the the following. (Note that we are using the frame syntax below.)

5.5 The Initial State

The information on the important units at the beginning is as follows.

```
In Mt : CarBuyingForChrisContext
;; in the context of chris buying a car,

Unit : Chris
;; here are the assertions about Chris.

instanceOf : (Person)
heightOfObject : ((Foot-UnitOfMeasure nil :max 7.25 :min 1))
  ;; This is Cyc's notation for interval terms. It is default about people.
  ;; It states that Chris is at least a foot tall and less than 7.25 feet.
age : ((YearsDuration nil :max 115 :min 15))
  ;; the 115 is a default about people. 15 is a default from the fact that
  ;; Chris is buying a car.
```

```
entryCardinalityLessThanOrET : ((Person cohabitants 10))
  ;; this is the frame syntax for a quaternary predicate namely
  ;; entryCardinalityLessThanOrET(Chris Person cohabitants 30) and is a
  ;; default about people in the AngloAmericanKinshipMt. It says that there
  ;; are probably less than 10 people cohabiting with Chris.
owns : (Chris-Car)
  ;; it is a good default people usually buy things
  ;; for themselves (i.e., not as a gift, as someone's agent, etc.)
buyerIn : (Chris-CarBuyingEvent)
  ;; this is not a default but monotonically true.
```

At this stage in the session, there isn't much we can infer about Chris. The above inferences were made by the system because it determined that these were potentially relevant for the current application. (A rule may be specified to be forward propagated in certain classes of contexts and backward propagated in certain others.)

```
In Mt : CarBuyingForChrisContext

Unit : Chris-Car

instanceOf : (Automobile)
cost : ((Dollar-UnitedStates nil :max 100000 :min 5000))
 ;; default about new cars.
slotValueHasInsOf : (RoadVehicleBody externalParts)
            (CarInterior interiorParts)  \ldots
 ;; the default parts, etc. of a car
 ;; slotValueHasInsOf(Chris-Car RoadVehicleBody externalParts)
 ;; means that the externalParts slot of Chris-Car has an instance
 ;; of RoadVehicleBody as one of its entries.
ownedBy : (Chris)
usedBy  : (Chris)
drivenBy : (Chris)
 ;; its a good default that if Chris is buying the car, he is probably
 ;; its user and driver. Note the absence for the need for time here.
 ;; Because of the earlier temporal qualification of CarBuyingForChrisContext,
 ;; these statements really mean ``the car would be driven by Chris, at
 ;; around the present time,  \ldots See example 4 for details.''
```

```
In Mt : BaseKB

Unit : CarBuyingForChrisContext
```

```
instanceOf : (HypotheticalContext DiscourseContext MajorPurchaseSelectionPSC)
 ;; see example 8 for why this has to be a HypotheticalContext.
 ;; Example 9 explains the concept of MajorPurchaseSelectionPSC
pscExecFeatures : (car-selection-psc-exec-features)
 ;; this defines certain special problem solving actions to be taken
 ;; when an assertion is made in this context. See example 9.
hasDiscourseContextOf : (Cyc-Chris-Discourse)
 ;; this is asserted by the initialization process
importsFrom : (BuyingGMt cost) (AngloAmericanKinshipMt cohabitants)  \ldots
 ;; during the initialization process, the system has decided to use
 ;; the concept of cost as defined by the BuyingGMt, the concept of
 ;; cohabitants from the AngloAmericanKinshipMt, etc.
selector :  (Chris)
 ;; Chris is the person doing the selection. See example 7 for details
selectionObject : (Chris-Car)
selectionObjectType : (Automobile)
 ;; See example 7 for details
domainAssumptions : ((livesIn x UnitedStatesOfAmerica))
 ;; the assumption that the buyer et. al. live in the US. Cyc could
 ;; deduce this from the fact that the discourse is taking place in
 ;; the US (the system does not do this now.) See example 10.
defaultImplicitTime : (TheYear1991)
 ;; See example 3.
precisionRequired : ((cost (Dollars 300)))
 ;; this is again a default assumption about the buyer (Chris).
 ;; It is possible that even at such an early stage in the purchase
 ;; he insists on knowing the cost with an accuracy of $50. In this case,
 ;; the value of this slot would be different. See example 12.
```

The initial state of the questions and candidate cars is as follows.

At the start, there are 324 different possible types of cars and two questions the user may answer.

(Q1) How much are you planning on spending on this car?
;; this is of course a principal concern!

(Q2) Will you be the (or a) primary driver of the car?
;; this question is really to confirm the default conclusion reached earlier.

5.6 Application Session

(A1) Answer Q1 : Between 15 and 20 thousand dollars.

```
Assertion : (LogAnd
```

```
(entriesGreaterThanOrEqualTo Chris-Car cost nil
    (Dollar-UnitedStates 15000))
(entriesLessThanOrEqualTo Chris-Car cost nil
    (Dollar-UnitedStates 20000)))
```

(Note : If an unreasonable answer such as "about 50 dollars" is given, the system complains at this point.)

Inferences : Cyc transforms this assertion into its interval representation.
(cost Chris-Car (Dollar-UnitedStates nil :max 20000 :min 15000))

Possible selections : Associated with CarBuyingForChrisContext are some special problem solving actions. One of these is to use the database to update the list of possible cars. Based on the cost information, the list of cars is narrowed down from 324 to 112 types of cars. Details of this are given in example 20. To estimate the cost of the car, the naive model of cost (that ignores financing charges, etc.) is used. Details of this are in example 12 and 13.

(A2) Answer Q2 : Yes (i.e., I will be a primary driver of the car).

Assertion : (primaryDriver Chris-Car Chris)

New questions : Now that the system has confirmation as to who the driver is, Q2 is removed and a number of new questions are presented. This action takes place because of the preconditions associated with the questions. The new questions include,

(Q3) Will there be more than one primary driver?

(Q4) Will you use your car for your commute (to work, school, etc.)?

(Q5) What kind of image do you wish this car to project?

(Q6) Do you sometimes drive just for the fun of it?

(Q7) Is there any particular kind of car you want?

(Q8) Do you want a car made in a particular country?

(Q9) What are some features you want in the car?

(Q10) What are some features you want to avoid in the car?

(Q11) Where do you live?

(Q12) Do you live in an urban area?

(Q13) What is your job?

(Q14) How old are you?

(Q15) Do you have any disabilities?

(Q16) What is your marital status?

(Q17) Who else resides in your household?

(Q18) Will you drive this car on vacations?

(Q19) What hobbies do you have?

Inferences : A number of inferences are drawn by about Chris based on his being the user of the car. A few of these include -

(D1) Chris is bodily able, i.e.,

- he is not blind or quadriplegic (this is monotonically true).
- he is not paraplegic (this is a default conclusion)
- his height is greater than four feet

  ...

(D2) He is older than 15 years and younger than 75 years. The system had already concluded that he was older than 15 based on his purchasing a car. This now gives us a second independent and even stronger argument for this conclusion.

...

In deriving these conclusions, axioms were lifted from the AutoMt. The techniques described in examples 1-4 are used here.

(A3) Answer Q13 (What is your job?) : RealEstateAgent

Assert : (allInstanceOf Chris RealEstateAgent)

New conclusions :

(D3) Chris is an adult (> 18 years old), a salesperon and less than 65 years old. This is from information in the JobMt. The system might have already guessed at his being less than 65 years old, but this additional argument makes this much more certain.

(D4) Chris might use his car in his work. This is based on the information in the JobMt that one of the typical activities of a RealEstateAgent is to drive customers around. However, we don't know for sure as to whether Chris is planning on using this car for his/her job yet. So the following question is posed.

New questions : Based on (D4), we have a new question.
(Q20) Will you use this car in your job?

(A4) Answer Q20 : Yes. (i.e., Chris will use this car as part of his/her work.)

New conclusions and questions : This information might potentially impose a number of constraints on the car. Based on information from the JobMt, the system constructs a model of a typical drive as part of his work - Chris-WorkDrive.

```
In Mt : CarBuyingForChrisContext

Unit : Chris-WorkDrive
```

```
instanceOf : (DriveAsPartOfWork TheTerm)
 ;; See example 11 for details on TheTerm
spatialExtent : (CitySized)
 ;; the drives taken by real estate agents usually does not
 ;; take them outside the city.
speedOfObject : ((MilesPerHour 45  :max 55))
 ;; based on the fact that this is city driving.
performedBy  : (Chris)
 ;; Chris is doing the driving - note that the description of
 ;; this event is from the perspective of Chris. See example 18.
vehicle : (Chris-Car)
entryCardinalityGreaterThanOrEqualTo : ((passengers nil 2))
 ;; a default that Chris will be taking at least 2 people around
 ;; on a typical drive
 ;; in which  he/she takes someone to see a house.
```

Chris-WorkDrive is an example of the use of "the terms" and perspectives. Details of this are found in examples 11 and 18. At this stage, the system is interested only in the properties of the drive from the user's perspective. Since Chris will be the driver, the description of this drive is from the perspective of the driver. See example 18 for more details on perspectives. The conclusions drawn about Chris-WorkDrive can be used to derive the properties of any particular work related drive the Chris might make.

New inferences : The following constraint on the car is inferred.

(D5) The car should not be a subcompact or a sports car. There are many arguments supporting this. Sports cars and subcompacts are not meant for taking many people around. They are difficult to get in and out of. subcompacts are not very comfortable to ride in.

Given this new constraint on the car, the system goes through the currently possible selections and prunes the list to exclude sports cars and subcompacts. Interestingly, not many cars are excluded by this new constraint. This is because most of the subcompacts were excluded by the constraint that the cost be greater than 15 thousand dollars and most of the sports cars were excluded because of the constraint that the cost be less than 20 thousand dollars.

Since Chris will be using this car to drive customers around, the system tentatively concludes certain desirable features of the car. Since these are only tentative conclusions, the following new questions are posed.

(Q21) Do you want a car air conditioner?

(Q22) Do you want a smooth ride?

(Q23) Do you want a quiet interior?

(Q24) Do you want four doors?

...

These are all comfort related features. Based on the requirement that the drive be comfortable (for Chris and the passengers), these suggestions are made by the system.

It should be noted that Cyc's own internal list of possible questions does not contain separate questions for each of Q21, Q22, etc. Instead, it contains the generic question "Do you want X?". When the system makes a tentative suggestion that the car have a certain feature (such as an airconditioner), this question is instantiated with that feature and the instantiated question is posed to the user. A slot on the Cyc unit CarAirConditioner contains the string "car air conditioner" which is used in the question template. Do derive these suggestions, the system uses rules stated using "the terms."

(A5) Answer Q21 : Yes (i.e., Chris wants an air conditioner)

Given this new constraint on the car, the system goes through the list of candidate cars and makes sure that the it is possible to get an airconditioner with the car. It also recomputes the price of the car with an airconditioner and excludes those whose cost is now greater than 20 thousand dollars. Though all the cars have an airconditioner as an optional feature, the cost of the airconditioner pushes some of the cars beyond the $20,000 budget. Note that the system is still using the naive model of cost. The list of candidate cars is reduced to 105.

(A6) Answer Q5 : SafetyConscious (i.e., Chris is safety conscious)

The system suggests safety features such as airbags and antilock brakes and rear seat shoulder belts. Though all the cars currently under consideration have shoulder belts, the system does not notice this and still offers this as a suggestion. This is a deficiency of the system that needs to be fixed.

Based on Chris being safety conscious, further conclusions are made about the maximum speed he drives at, etc. This has no impact on the list of candidate cars.

I won't bother listing all those new safety feature questions. The user now accepts the suggestion for having airbags. This significantly reduces the possible cars to those that are offered with airbags. At this stage, we have 16 cars -

```
ToyotaCelicaGtS Volvo240Car ChevroletCapriceClassic
ChryslerLebaronCoupeGtc ChryslerLebaronCoupeLx DodgeDynastyLe
DodgeSpiritR/t FordLtdCrownVictoriaLx FordTaurusLx
FordLtdCrownVictoriaLxCountrySquire MercuryGrandMarquisColonyParkGs
MercuryGrandMarquisColonyParkLs MercuryGrandMarquisGs
MercuryGrandMarquisLs MercurySableGs MercurySableLs
```

(A7) Answer Q16 (what is your marital status) : Single.

The system now concludes that Chris has no spouse. This is of course just the definition of Single (from the NaiveKinshipMt). To make the discourse interesting, let us now answer Q17 (who else resides in your household).

(A8) Answer Q17 : (who else resides in your household) : My wife.

The system creates a unit for the wife and tries to place this as an entry on the wife slot of Chris. From the HumanKinshipMt, this is known to be contradictory with the earlier monotonic conclusion that Chris has no spouse. Since this is a monotonic constraint violation, the system can't just leave this as an unresolved conflict. The system offers us a choice - retract A7 or A8. We choose to retract A7. (See examples 1-4 for a discussion of the context related issues involved.)

After this, the system can successfully place Chris-Wife as an entry on the the wife slot of Chris. It concludes that Chris is male, older than 18, etc. See examples 1-4 for details.

The system knows that Chris might need to take his whole family out in the car. Now that we know that he has a family, the following question becomes relevant.

(Q25) Will this be your family's main car?

We answer yes to this question and Cyc creates a unit for the typical drive for Chris with his family (again using the technique of 'The terms' - see example 11) and makes conclusions about who will be driving it, who the passengers will be, etc. In this case, since the system has already been told that Chris will be using this car for driving his customers around, no new suggestions are made. If Chris had said he lives with his wife and four kids, then the system would suggest using a larger car.

(A9) Answer Q11 (Where do you live?) : Peru.

Assertion : (livesIn Chris Peru).

This contradicts the assumption made by the current context that the discussion is taking place in the United States. Many of the heuristics used earlier assumed that Chris lived in the United States (what is involved in being a real estate agent, the kinds of cars available, the number of people living with him, etc.) The system (as it is today) cannot do much if the user lives in South America. Since it can't really go ahead if the user really lives in Peru, the system complains.

It is quite possible that the user asked for some sample answers to this question and picked Peru of by mistake, e.g, he lives in Peru, Indiana or he was just testing the system. So it gives us a choice and we have to choose between retracting the statement and making the statement in a different context. We choose the former. The concept of domain assumption is explained in example 8.

We next try to answer the same question again.

(A10) Answer Q11 : CityOfSeattleWA

Note that the assertion (livesIn Chris CityOfSeattleWA) does not mean that Chris has been living in CityOfSeattleWA forever. All it means that he is living in Seattle during the implicit temporal period associated with the session, i.e. 1990. We are using the technique explained in example 3 here.

The system has information about the weather in the major US cities. It knows that it is foggy in Seattle in winter and that fog lights are useful in fog. Given that Chris is safety conscious, it suggests Fog lights as a feature.

(Q26) Do you want Fog lights?

The reasoning behind this question is described in examples 5 and 6.

It also derives the constraint that the car should not be a convertible.

(D6) It also derives that the car should not be a convertible.

However, as none of the cars under consideration are convertibles, this does not affect the list of cars.

Another source of constraints on the car is the hobbies and other activities of the user.

(A11) Answer Q19 : (What hobbies do you have?) Camping.

The system gensyms a unit for a typical Camping related drive that Chris might take. Since this might involve driving on harsh terrains, the system tentatively derives a number of new features the user might want. Since these are only tentative, they manifest themselves in the form of questions.

(Q27) Do you want large cargo capacity?
;; to carry camping related equipment.

(Q28) Do you want manual transmission?
;; for greater control in driving on harsh terrain.

(Q29) Do you want automatic locking differential?
;; for driving in mud, ice, etc.

We answer yes to Q28, i.e., Chris wants manual transmission. As it turns out, most of the American cars in the running at this point are not available with manual transmissions. So this reduces the number of cars to three.

DodgeSpiritR/t, Volvo240 and ToyotaCelicaGtS.

Now that we have a fairly short list of cars, the user might take a closer look at each. We ask the system to summarize the information found in the different databases about the DodgeSpiritR/t. Here is the output.

```
Dodge Spirit r/t:
List Price:: $17820 to $17871
Body Type:  NotchbackCar
Transmission: one of
    ManualTransmission5Speed
    OverdriveAutomaticTransmission
```

```
Drive:  FrontWheelDrive
Doors:  4
Displacement: 2.2
Horsepower  : 224
Mileage -    City: 19
          Highway: 27
Weight: 2801
Length: 181
Width: 68
Seating Capacity::  6
Headroom -   Front: 38  Rear: 38
Legroom  -   Front: 42  Rear: 38
Cargo Volume: 14.4
Fuel Capacity: 16.0

Standard Features: DiskBrakeSystem
    CruiseControlSystem TiltSteeringColumn TintedCarWindow CarFogLights
    IntermittentWindshieldWiper CarAM/FMRadio TripOdometer
    VoltMeter OilPressureGauge TemperatureGauge Tachometer
    ReclinableCarSeats FrontBucketCarSeat ClothSeats CarAirbags PowerSteering

Available Features:
    DividedCarSeats RearFoldingCarSeats
    AutomaticCarWindow TiltSteeringColumn TintedCarWindow
    CruiseControlSystem CarSunRoof CarAirConditioner AntilockBrakeSystem

General Attributes:
    RearWindowWithDefroster  CompactCar MediumEngine
    ModeratelyPricedCar GasolineEngine Inline4Engine RunsPerfectly GoodReliability
    SlightDamage HandlesSmoothly  HandlesPrecisely
    BrakesPerformVeryWell ExcellentClimateControl
    ShiftsSmoothly  ModeratelyNoisyInterior
    FirmRide  CrampedDrivingPosition
    MediumFrontHeadRoom  MediumRearHeadRoom
    MediumFrontLegRoom  MediumRearLegRoom
```

Notice that the price is given as a range since the price information in the Consumer Reports database is conflicting with the price information in the consumer Guide database. Also, each of these databases reports different features as being available and some care needs to be taken with the closed world assumption. These aspects are discussed in examples 20 and 21.