PREFERENTIAL ORDERING OF BELIEFS FOR DEFAULT REASONING

by

Bharat Bhushan

April 28, 2003

A thesis submitted to
the Faculty of the Graduate School of
State University of New York at Buffalo
in partial fulfillment of the requirements for
the degree of
Master of Science

Department of Computer Science and Engineering

Dissertation Committee:
Dr. Stuart C. Shapiro
Dr. William J. Rapaport

Acknowledgments

My sincerest gratitude to my advisor, Dr. Stuart C. Shapiro, who has been most patient with me. He has been a source of constant support and guidance when I lacked in spirits and ideas. I thank him for giving me a chance to work on the topic of my choice. I am not sure how I will ever be able to repay him for his endurance of my occasional passiveness and dumbness. Thanks Stu.

This work was supported in part by the U.S. Army Communications and Electronics Command (CECOM), Ft. Monmouth, NJ, through a contract with CACI Technologies. Their support is gratefully acknowledged; without them this research would not have been possible.

Thanks to my co-advisor, Dr. William J. Rapaport, whose ideas have helped me understand the problems of the thesis in better light. His input on the documentation were most valuable.

Thanks to my lovely officemates, Fran, John, and Haythem, who have guided me through as seniors in the business of research.

Last, but not the least, to all my friends — Thanks for your moral support.

Contents

List of Figures

Abstract

Preferential Ordering of Beliefs for Default Reasoning

by Bharat Bhushan

Major Professor: Stuart C. Shapiro, Ph.D.

This thesis investigates the application of ordering of beliefs to default reasoning. An important component of intelligence is decision making with the help of given information. If the given information is not sufficient, default results should be assumed. If the derivation of a proposition results in a contradiction, belief revision should occur about some of the propositions that led to the contradiction. A step before belief revision, however, is to see if one of the contradictory results is more preferred than the others. In the case where one result is more preferred, it may be believed by default and no beliefs may be revised. The concept of preferential ordering of beliefs for default reasoning has been used in other representations of defaults. This thesis claims that no special representations are necessary and any special syntax is in fact futile to represent defaults. This thesis provides a theory of the preferences of beliefs and an implementation of the theory to carry out default reasoning.

To Mr. and Mrs. Sukhija

Chapter 1

Introduction

One important component of intelligent activity is being able to do assumption of relevant knowledge efficiently. For example, to make a decision if a bird flies or not believing in advance about it that it flies, without requiring any more specific knowledge, saves an effort to acquire more knowledge about the bird. This requires representation and reasoning with "background" knowledge. Thus, in the given example, the background knowledge is the assumption that usually birds fly. Standard first-order-logic-based inferencing techniques are well developed and could potentially be used except for the fact that they are monotonic in nature. In monotonic logic, the set of legal conclusions grows monotonically with the set of facts appearing in the database of facts. Monotonic classical logic is not sufficient to deal with the situations where implicit assumptions have to be made when there is lack of enough knowledge but after the information is available, the opposite may be derived. There exist cases where sometimes it is reasonable to maintain beliefs that are contrary to what may be logically entailed using inference procedures of the classical logic.

Generalization is such a case. In classical logic, the two quantifiers are the ones that capture the notions of "all" and "some". A universally quantified statement holds true of "all" things in the domain. An existentially quantified statement holds true for "some" of the things in the domain. For example, "All birds are feathered" is a universally quantified statement. "Some birds fly" is an existentially quantified statement. A generalization is a non-classical quantifier that is unlike a universal or an existential quantifier. Generalizations may be statistical claims of "more than half" or any such variant. They are the claims that although sometimes wrong may still be presupposed

in the absence of complete information. Generalizations of such a kind are usually represented by what are called defaults. Following are some of the cases where generalizations are useful.

1. Typicality: The overhead of asking followup questions in case of missing information may be avoided by assuming what is normally the case.

2. Assume to be up to date: If we assume that things do not change till we notice them, we may be able to make some useful deductions. Otherwise, in the ever-changing world, making deductions may become a very complex task.

3. Heuristics: Rules of thumb may not be successful in all cases but propose good solutions without much search. Therefore, they may be treated as default information.

The list is incomplete and there are many other similar situations. Most of these situations are dealt with in natural language by the use of defaults that are just "usually" true. Defaults are not "TRUE" in a strict sense and are "defeasible" when exceptions arise.

1.1 Examples of default reasoning

People generalize all the time. The first sentence in this section is not a generalization! Defaults are used in our daily language and reasoning. The world is somewhat structured[1] so as to have exceptions for every rule.[2] Thus the generalized rule/statement is a default and the exceptions are the special cases. The first sentence in this section, *"People generalize all the time"*, is a generalized belief of mine - a personal belief that does not mean it holds for "all" people and for "all" times. The second sentence in this section, *"The first sentence in this section is not a generalization"* is a negative generalized statement. It is a negative statement because it asserts about something that is **not** the case. It is a generalized statement because it is not as categorical as it seems. It is a matter of perspective or is context specific that the first sentence shown above is a generalization. The intent of the statement is purely rhetorical and to show that defaults exist as positive as well as negative statements.

[1] Alternatively, we structure the world.
[2] Well... not "every" rule, and so this is a default statement also!

Now, consider a better, non-self-referential, example for better understanding — a widely held notion that birds fly. While this statement is true for all practical purposes, there are exceptions to this. Penguins do not fly. Dead birds and baby birds do not fly. Hence there exist exceptions to this default rule. Also, when we say penguins do not fly, we cannot discount the possibility of some special kinds of penguins (say, the imaginary super-penguin!) that can fly. Thus, that penguins do not fly is a default statement also. As is argued later in section 3.1, there are hardly any statements that have no exceptions in any context. In fact, it is hard to think of any sentence that is not a default in some context.

Now let us assume that all we know is that birds fly and that Opus is a bird. Then, it can be reasoned out that Opus flies, by default. This is default reasoning. If we get to know some additional information that penguins do not fly and penguins are birds and Opus is a penguin, we should be able to reason that Opus does not fly. This is an issue in the field of default reasoning and has been solved through various representations and reasonings with knowledge.

Based on the same idea is the following narration that is inspired from a narrative in the book *Gödel, Escher, Bach: An Eternal Golden Braid*[Hof79], and is a common puzzle for young children. The question of the puzzle "Am I alive?" is asked and a decision has to be made after every statement. Default reasoning takes place at each step and an exception occurs at every "next" step.

I was going to France on a plane
Unfortunately, the plane's engine died, —-> *Not Alive*
Fortunately, I had a parachute, —-> *Alive*
Unfortunately, it did not open, —-> *Not Alive*
Fortunately, there was a haystack right below, —-> *Alive*
Unfortunately, there was a pin on it, —-> *Not Alive*
Fortunately, I missed the pin, —-> *Alive*
Unfortunately, I missed the haystack. —-> *Not Alive*

This example shows that one can never be sure of one's decisions and can at best only make

default inferences with the given information because of the uncertainty in the world. Defaults and default reasoning are an important part of any intelligent system.

1.2 Background

This section contains a brief overview of default reasoning and where it lies in the field of knowledge representation and reasoning (KRR).

It is hard to define what knowledge is. Philosophers usually characterize knowledge as "justified true belief": i.e., you can be said to know that some proposition P is the case if and only if you believe P, your belief is justified (e.g., by rational argument or experimental evidence), and P is, in fact, true. Computer scientists, on the other hand, seem to use the term as being more or less equivalent to "data" or "information", whether or not it is true or even justified !

Knowledge may be considered as a higher abstraction of data or information. Thus, what data-structures are to data, representation is to knowledge. Similarly, what algorithms are to data, reasoning is to knowledge. The analogies hold only to distinguish the levels of abstraction by similar amounts. They do not talk about the similarities or differences in the design issues involved at the two levels.

A

Representation deals with ontological commitments for the domain of a problem it is used to solve. Hence, it has bearing on the reasoning procedure just like the choice of data-structures affects the efficiency of an algorithm. Reasoning deals with expansion of the knowledge base (KB).[3] This also means that some representations are better suited for a kind of knowledge base than other representations. Research in KRR is directed towards finding good representations and reasoning procedures for typical knowledge bases.

One such system studied is the Truth Maintenance System (TMS). A truth maintenance system is a mechanism whereby a knowledge-based system can keep reasonably consistent and truthful as

[3] A knowledge base may be analogous to a database! It is a repository or storage place for knowledge.

its knowledge changes. For example, if facts have been added to the KB through inference based on a set of premises, and one of the premises is later removed from the KB, any conclusion that depends on that premise should also be removed from the KB.

1.2.1 Non-monotonic Reasoning

It is commonly agreed that most knowledge bases are going to be dynamic. This means that the content and size of the knowledge base would keep changing with time. As new information is added to the KB, the need might arise to revise older beliefs or disbelieve some. Thus, these systems with typical knowledge bases would need to employ non-monotonic reasoning. Non-monotonic reasoning aims to capture, among other things, the notion of common sense reasoning and reasoning under uncertainty. It departs from the dictates of classical logic in that the addition of further premises may result in the retraction of certain inferences that could have been drawn previously. Often this is done by sanctioning inferences based on what is "usually the case" in certain circumstances but "disabling" them in light of further information. Some of the major non-monotonic reasoning problems are belief revision, multiple inheritance, diagnosis, and default reasoning. Of these, belief revision and default reasoning are somewhat related.

Belief Revision

Belief revision or belief change studies the way in which an agent's beliefs about its domain should be altered to accommodate new information. Important problems in the field of belief revision include: iterated belief change, revision by conditional information, abductive belief change, ontology revision, and the relationship between two important forms of belief change known as revision and update. Belief revision is a kind of non-monotonic reasoning involving the problem of changing the knowledge as a result of a contradictory observation.

Default Reasoning

Default reasoning is a kind of non-monotonic reasoning. In default reasoning, we want to draw conclusions based on what is most likely to be true. The notion of "most likelihood" may or may not be probabilistic. We have already seen examples of this in section 1.1.

Belief revision and default reasoning are related in that both of them tackle the problem of disbeliev-

ing some classically derivable propositions. [Eli98] argues that default reasoning, in fact, arises out of belief revision concerns. According to [Eli98], there are five conditional types for propositions, illustrated with the help of examples below. In the following, the classification of propositions comes from a prevalent part of everyday reasoning by which we formulate and revise situational models of our world.

1. Causal-few disabler: If Mary jumps in the pool, then she gets wet.
2. Causal-many disabler: If John studies hard, then he does well on the test.
3. Promise: If Susan completes the report by the weekend, then her boss will give her a day off next week.
4. Familiar Definition: If a mineral is a diamond, then it is made of compressed carbon.
5. Unfamiliar Definition: If a plant is equisetium, then it spreads by creeping horizontal root stems.

Based on a statistical study of human responses to exceptions to the above conditional types, it is argued in the same paper that some of the beliefs that have fewer exceptions are the kinds that are considered as default rules. Other rules are considered for a revision if an exception arises. Thus, Causal Disabler rules are generally revised and disbelieved when a contradiction arises, while Definitions are considered as defaults under similar circumstances. This means that the Definitions are still believed albeit not as strongly as before.

Formal representation of default statements has been attempted by many. [Rei80] developed default logic. [MD80] developed non-monotonic logic. Both of these logics are based on the concept of "consistency". That means, if it is reasonable, by any means, to believe something, it is believed. Other notable theories of representation of default rules include Circumscription [McC80] and Autoepistemic Logic [Moo84].

Non-monotonic Logic Non-monotonic Logic is basically an extension of first-order predicate logic to include a modal operator, M. The purpose of this operator is to allow for the expression of

the notion of necessary consistency. For example: $\forall$(x): bird(x) $\wedge$ M $\neg$abnormal(x) $\Rightarrow$ fly(x) states that for all x, if x is a bird and if the fact that x is not abnormal (as in, dead or flightless) is consistent with all other knowledge, then we can conclude that x flies. Since, the requirement of consistency is a necessary one for the implication to hold, the operator M is a modal.

How do we define consistency? One common solution (consistent with PROLOG notation) is, if we cannot prove $\neg$P, we may say that P is consistent (since $\neg$P is false).

Default Logic [Rei80] extends the standard first-order logic with a special syntax for defaults that is of the form $\frac{\alpha:\beta}{\gamma}$ and reads much like the non-monotonic-logic default rules, i.e., if α is true and β is consistent with the current knowledge, then infer γ. It differs from non-monotonic logic in the sense that default rules cannot be inferred in the language. That means, non-monotonic expressions are rules of inference rather than derivable propositions. In other words, $\frac{\alpha:\beta}{\gamma}$ is just a rule of inference and not a wff of the language while all propositions in non-monotonic-logic are wffs of the language. Hence, there are two kinds of rules in the default logic language, namely, the defeasible rules and the non-defeasible rules. The defeasible rules are the defaults. The default rules can be used as rules of inference, but cannot appear in proofs nor be subexpressions of expressions. Another difference is in the reasoning procedures of the two. Default logic maintains multiple consistent extensions in some cases when the multiple extensions are mutually inconsistent. On the other hand, in such a case, reasoning in non-monotonic logic might lead to an inconsistent knowledge base.

Circumscription Circumscription was formulated as a form of non-monotonic reasoning in [McC80]. It follows the closed world assumption (CWA), which means that it assumes the assertion of nothing more than what is explicitly asserted. All other possible propositions are assumed to be negatively asserted until they are explicitly asserted in the system.

The original formulation of circumscription was augmented in [McC86] with the notion of "abnormal" to represent defaults specially. Thus, the default "Birds fly" is represented as "Non-abnormal birds fly". Specific information contradictory to a general rule, like "Penguins do not fly" is stated by stating that "Penguins are abnormal birds". Thus, for every *kind* of exception for a rule, a new "abnormal" predicate is introduced. The distinction between default logic and circumscription is

that in circumscription, defaults are sentences in the language itself, whereas in default logic, default rules are not sentences of the language.

Autoepistemic Logic Autoepistemic logic is a modal logic, just like default logic and non-monotonic logic. It deals with "internal" beliefs of an agent. Thus, in autoepistemic logic, the modal operator L applied to a proposition α is Lα and is read as "α is believed". It combines the ideas of circumscription and non-monotonic logic so as to encode "If it is a bird, it flies" as $bird(x) \wedge \neg L ab(x) \Rightarrow flies(x)$. Autoepistemic logic is like default logic in the sense that it maintains the notion of stable extensions that are possible consistent belief spaces.

1.2.2 Summary

To summarize and relate all the ideas in this section, default reasoning is an important issue in KRR because it is an important component of a typical knowledge base. A typical intelligent knowledge base would create rules based on its *experiences* using inductive reasoning. It would carry out deductive reasoning to answer specific queries using the rules created. If some rules or propositions are found to be contradictory at any stage, it would either revise them or would carry out default reasoning.[4]

The problem of representation and reasoning with defaults has been dealt with in various ways. Major ideas among them ([Rei80] and [MD80]) propose a separate syntax for expressing default rules. Some theories like [McC80] propose the use of different predicates for the special case.

We claim that no such separate syntax is required. One of the ways to reason with defaults is to order them according to a preference. The notion of ordering beliefs is discussed in section 3.2.

1.3 Prospectus

This thesis is concerned with a study of some of the major approaches to default reasoning. It argues against some of their shortcomings and hence proposes a new representation and reasoning for

[4] In some suitable way that we discuss later.

defaults based on previous ideas. This new representation and reasoning is carried out in the SNePS knowledge representation and reasoning system. A brief account of that is also presented in this thesis.

Chapter 2 contains the work that has influenced this research. This thesis is based on some foundation work by Delgrande, Touretzky, and Grosof. Their ideas are presented.

Chapter 3 is a chapter about the motivating ideas. It contains the arguments for the implementation of the ideas that form this thesis. It contains an introduction to what preference orderings are and why they are required. It also contains our approach to representing default rules.

Chapter 4 presents an overview of the SNePS knowledge representation and reasoning system.

Chapter 5 contains a brief account of the implementation. Based on the ideas due to the foundation works, an algorithm for carrying out default reasoning with preferential ordering of beliefs is presented also.

Finally, Chapter 6 summarizes the whole thesis. It contains a discussion of the problems encountered during implementation. It also gives a few directions for future work.

Appendix A contains demonstration of a few examples that were worked out in SNePS

Chapter 2

Survey

Following are the works that form the basic building blocks of this thesis. These works have influenced the research of this thesis in a very direct way.

2.1 Delgrande

Delgrande's paper [DS99] discusses a method of using Default Logic [Rei80] to make inferences that involve preference amongst rules. [Rei80] defines default rules with syntax $\frac{\alpha:\beta}{\gamma}$, with the semantics, if α holds and there is no reason to disbelieve β, then γ holds too. [DS99] introduces three new predicates $ok(n)$, $ap(n)$ and $bl(n)$ that are used to enable or disable the "firing" of a rule.[1] Inferencing is done by deriving a consistent extension.[2]

A set of default rules D and a set of formulas W form a default theory (D,W). An ordered default theory (D, W, $<$), as defined in [DS99], is an augmentation to the default theory (D,W) and is defined to be a 3-tuple, where D = set of default rules (rules that are normally valid), W = set of formulae (initial knowledge of the system, hard facts) and (δ_1,δ_2)$\in <$ when $\delta_1, \delta_2 \in$D and the default rule δ_1 is less preferred than the default rule δ_2. The relation $<$ has the ordered pair (δ,δ_T) for every $\delta \in$ D, where $\delta_T = \frac{True:True}{True}$. [DS99] describes a way to build a theory (D$\prime$, W$\prime$) from the ordered default theory, which shall have preference as defined by the relation $<$ "compiled" into it. In the procedure, the newly introduced symbols have the following semantics

[1] To be technically correct, it is the inference that is enabled or disabled, and hence it is the ground rule that is enabled or disabled from "fi ring". Rules are and should always be enabled

[2] An extension is a closed set of all the formulae that can be derived from the theory

- Predicate ok(n): It is ok to apply the default rule identified by n.
- Predicate ap(n): The default rule n has been found to be applicable.
- Predicate bl(n): The default rule n has been found to be inapplicable and is hence blocked.
- Relation $\prec$: $n_1 \prec n_2$ means that default rule n_1 is less preferred than the default rule n_2 and the number n_T refers to the number corresponding to trivial maximal default rule δ_T.

This procedure is as follows.

1. Enumerate the default rules so that there is a unique name for each of the rules. Thus, we have n:$\frac{\alpha:\beta}{\gamma}$, where n$\in \mathcal{N}$, the set of natural numbers.
2. Let D$\prime$ = $\{\frac{\alpha \wedge ok(n):\beta}{\gamma \wedge ap(n)}, \frac{ok(n) \wedge \neg\alpha:}{bl(n)}, \frac{ok(n) \wedge \beta:}{bl(n)} \mid n : \frac{\alpha:\beta}{\gamma} \in D\} \cup \{\frac{:\neg(x \prec y)}{\neg(x \prec y)}\}$
3. W$\prime$ = W $\cup$ $\{ (n_1 \prec n_2) \mid (\delta_1, \delta_2) \in < \} \cup \{ok(n_T)\} \cup \{\forall x \in \mathcal{N}[\forall y \in \mathcal{N}, (x \prec y) \Rightarrow (bl(y) \vee ap(y))] \Rightarrow ok(x)\}$

The first part of W$\prime$ contains the prior world knowledge W. The second part specifies that $\prec$ is a predicate whose positive instances correspond to those of the strict partial order $<$. ok(n_T) asserts that it is always valid to apply the maximally preferred default. The last part of W$\prime$ ensures that ok(n) is asserted before ok(m) if m$\prec$n and thus controls the application of defaults. It is proved in [DS99] that this new default theory is the same as the ordered default theory in the sense that the extensions of the two are equal. This is a result out of the fact that ok(n_δ)$\in$E, for every $\delta \in$D.

[DS99] has influenced this thesis by introducing the notion of applicability and blocking of the default rules. In contrast with belief revision concepts where a rule may be disbelieved if it leads to contradiction, in default reasoning, a rule may continue to stay in the KB even if it leads to contradictions. It stays in the KB because it may be applicable for certain instances and may be blocked for certain others. Thus, some propositions are *derivable* but not *derived* to maintain consistency.

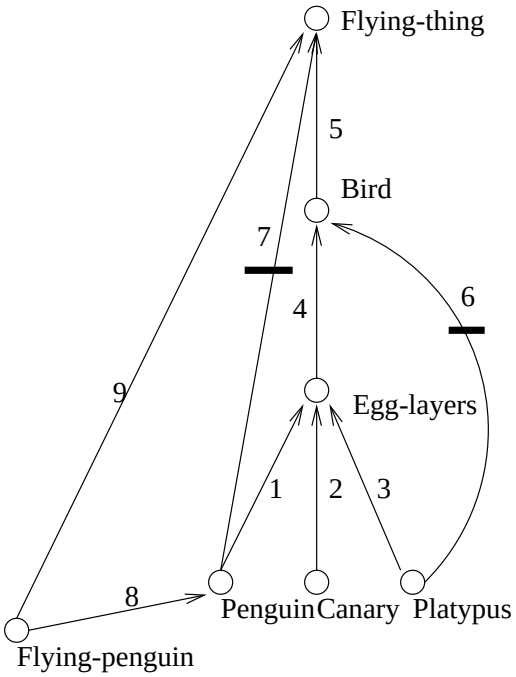

Figure 2.1: Representation of Example 1 in NETL

2.2 Touretzky-Fahlman's NETL Projects

One of the unique ways of expressing defaults, in which no separate logic has been defined, has been described by [Fah79] through the NETL project. In this scheme, the contents of the knowledge base (KB) are represented graphically. Nodes represent propositions. Edges between nodes represent relation linking the propositions. Consider the example below that is represented in NETL as in figure 2.1

Example 1:
1:"Penguins are egg-layers"
2:"Canaries are egg-layers"
3:"Platypus are egg-layers"
4:"Egg laying creatures are birds"
5:"Birds are flying-things"
6:"Platypus are not birds"
7:"Penguins are not flying-things"
8:"Flying-penguins are penguins"
9:"Flying-penguins are flying-things"

The links in the figure 2.1 are IS-A links but may represent other kinds of links in general. The

ones with single heavy bars through their shaft are negative links, i.e. the negation of some of the other links. They represent negative default rules as described in section 1.1. Disregard the numbers on the links for the time being. Their existence is described later in the text. [Tou86] describes a reasoning procedure based on this graphical representation.

2.2.1 Net Notation

The representational structure of the KB is called a net. For a net Γ, the following terms are defined. For a rigorous syntax of net notation, refer to [Tou86]

- KB(Γ): set of links in Γ.
- WF-L(Γ): superset of KB(Γ) that contains all possible links that can be formed, as allowed by the syntax of nodes and links.
- WF-paths(Γ): set of sequence of links drawn from WF-L(Γ) such that if there is any negative link in a sequence then it's the the last one in the sequence. Intuitively, a path refers to a proof and hence no link after a negative link in a sequence amounts to a proof.
- A-types(Γ): set of possible answers to queries on Γ
- A(σ): For each path σ in WF-paths(Γ), an associated assertion from A-types(Γ).
- Contradiction: A subset of WF-paths(Γ), Φ, contradicts a path σ if A(σ) and A(τ) negate each other for some $\tau \in \Phi$
- Preclusion: Let a path from node x to node q (not necessarily distinct) be denoted as σ(x,q) and a link between node x and q be denoted as <x⇒q>. Also let the path formed by the concatenation of two paths be denoted by lexically concatenating the denotations of the paths. A path $\sigma_1(x,q)\sigma_2$(q,p) is precluded, relative to a set of paths Φ, by another path τ_1(x,r)<r⇒p> iff τ_1(x,r)$\in \Phi$ and <r⇒p>$\in \Gamma$ and $\tau_1(x,r)\tau_2(r,q) \in \Phi$ for some $\tau_2 \in \Phi$. For example in figure 2.2 (which is a modified version of figure 2.1), if the node x is the predicate "Penguin", node q is the predicate "Bird", node r is the predicate "Special" and node p is the predicate "Fly", then the path "Penguins do not fly" precludes "Penguins Fly" because there exists a path from node r ("Special") to node q ("Bird") and is hence more specific. A more general

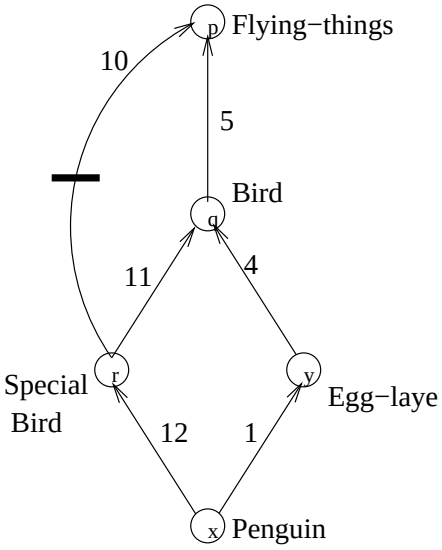

Figure 2.2: Example showing preclusion

figure showing preclusion is in figure 2.3. In figure 2.3, the dashed lines/links denote paths while solid line/link denotes a relation and the nodes are propositions.

- Extensions:[3] An extension of Γ is a subset of WF-paths(Γ) in which all paths are free of contradiction or preclusion and none of whose proper superset is an extension itself. A net may have multiple extensions.

2.2.2 UPSCAN/DOWNSCAN Algorithms

The NETL project defines preference amongst rules/propositions based on specificity in case of inheritance structures. The two algorithms are the UPSCAN and DOWNSCAN algorithms and are correct for a limited class of networks that are consistent and "orthogonal". Preclusion, as defined in previous section, is the criterion for preferring a proposition over another. The UPSCAN algorithm finds for a node x, the nodes that it inherits from. The DOWNSCAN algorithm is the exact opposite and finds for a node x, the nodes that inherit from it. Briefly, the UPSCAN algorithm works as follows.

[3]To be specific, this definition refers to credulous extensions as opposed to skeptical ones. For details refer to [Tho92]

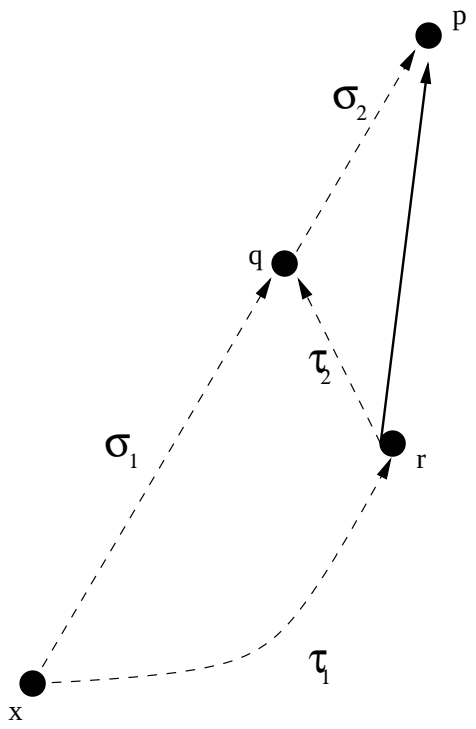

Figure 2.3: Formal representation of Preclusion

Beginning at x, going from the specific to the general proposition, the algorithm finds a transitive closure of IS-A links of node x and marks them T. If in the traversal process an IS-NOT-A link is found, the node at the head is marked F.

While these algorithms are very useful in principle,[Tho92] finds to be computationally very expensive and find a restricted acceptance with us. The main contribution of this reference thus, is the introduction of the concept of preclusion that can be mathematically formalized based on the structure of the knowledge in KB. However, we realized that it is worthwhile to have redundancy of preference information in the form of preference rules for efficiency. Some of the preference rules that are based on inheritance however may be computed using the criterion of inheritance.

2.3 Grosof

A work found to be very similar to ours involving Logic Programs (LPs) is by Grosof[Gro97]. [Gro97] talks about courteous LPs, which are an extension (not to be confused with extensions defined in the previous section) of normal LPs. A special predicate symbol *"Overrides"* is introduced,

and rules formed using this predicate augment the normal LP rules to form an extended LP. Rules formed with the *"Overrides"* predicate are meta-rules that put a partial order on the existing rule-base. This helps in conflict resolution if and when it arises. The *"Overrides"* predicate captures the notion of preference ordering that is discussed in section 3.2.

The algorithm described in [Gro97] employs a topological sort on the rules. This means, if a implies b then a comes before b. To deduce the truth value of a ground instance of a predicate P, all the ground instances that are used to prove P are listed. All the preference rules (the *Overrides* rules) are listed first, followed by ground facts. Ground facts are followed by the instances of predicates that are directly derivable from the ground facts followed by the instances derivable using the ground instances in the list so far. This is how the topologically sorted list of instances is built. The list is then traversed, rule by rule in the order they exist in the list and the truth value of each predicate is determined. For each predicate, its positive and negative literal contest. The literal which is derived by a rule that is not "Overridden", is declared as winner.

The main power of Grosof's solution lies in producing a "courteous" answer set. For each atomic literal the algorithm consistently makes a decision and puts the literal in the answer set. If a conflict is not resolved even with the *"Overrides"* rules, it adopts skepticism and leaves the literal out of the answer set. We retain this property in our solution. We extend this work by providing an algorithm for multiple levels of preference orderings.

2.4 Summary

To summarize this chapter, major ideas of this thesis have come from the foundation works by Delgrande[DS99], Touretzky[Tou86] and Grosof[Gro97]. [DS99] describes an algorithm that uses preferential ordering on rules represented in Default Logic. It describes an algorithm that can be used to test the applicability of a default rule to an instance. A rule is blocked from carrying out inference if there is a more preferred rule that is applicable. We believe in the same idea of blocking some undesirable results. However, we find a problem with the work by Delgrande. A major problem that we find with this work is that it uses default logic for representing default rules. Our claim, as we argue later in section 3.1, is that no special representation is necessary for default rules.

[Tou86] describes the concept of preclusion that implicitly orders the beliefs on the basis of the position of a proposition in the net that represents the KB. Generally, more specific facts are more preferred than the less specific ones. This predefined preferential ordering rule can be used to create the preference rules for KB.

Finally, [Gro97] describes an algorithm that can be used to carry out reasoning with rules and preferential ordering on rules. The implementation of this algorithm in SNePS is a part of the current work. Another part is extending the algorithm to support multiple levels of rules that assert preferences over other rules. Thus, our algorithm supports nested preference rules that allow us to derive the preference rules based on the context. Default preference rules may be provided for the contexts that do not have sufficient information for making decision as to which preference rules are asserted in the context. This is illustrated more clearly in the demo 2 in appendix A.

Chapter 3

Motivation

This chapter contains the motivating ideas of this thesis. One of the ideas is to use explicit rules of preference ordering among beliefs. Another idea is to have no separate representation for default rules.

3.1 No Default Rules

One of the salient points of this thesis is a representation system for default rules in which there is no distinct syntax for them. All rules look alike. No distinct sets of rules are maintained, as is done in default logic[Rei80] and SWMC (Shapiro, Wand, Martins and Cravo) [CM92], for classical rules and default rules. In fact, there are no default rules of any kind. There are just **rules**. All rules are treated as default rules and are simply called rules. This is reasonable, since in a dynamic system, [1] all rules either have an exception or are **known** to have no exceptions. There might arise exceptions to rules that are known to have no exceptions as changes are made to the KB.

To begin with, most rules have an exception. Only the rules that state scientific definitions very precisely are generally considered exceptionless. The claim of this thesis is that it is not only unnecessary to differentiate between default rules and exceptionless rules but also it is inconvenient. It is unnecessary, because the same reasoning can be done by maintaining preference ordering among rules. Rules that have fewer exceptions may be more preferred compared to ones that have more

[1] A typical system would be dynamic

exceptions.[2] The rules that have no exceptions are, thus, the most preferred rules, and, if they are used for inferencing in comparison to any other less preferred rule, then the reasoning would be sound. Thus, preference ordering among rules obviates the need for distinction among classical and default rules. All rules may belong to the same hierarchy, with the most true rules at the top. This also means that all rules are wffs of the language and there are no special rules of inference. The preference ordering amongst rules may not be total. Thus, only the rules that lead to mutual contradictions may be more preferred or less preferred amongst themselves.

It is inconvenient to distinguish default rules from other rules in a dynamic system, because, as soon as an exception to a hitherto exceptionless rule is observed, a modification would be required in the set of default rules and also in the set of other rules. The extent of truth of any statement changes with time. "Birds fly" was a good approximation to defining birds initially before modern man chanced upon some creatures that were more like birds than any other creatures in all respects except for the ability to fly. Ever since, the correct definition of birds has evolved beyond the vocabulary of common man. Consider a classic rule like "Every person has a father who is a person". It is categorically TRUE only until biotechnology comes up with exceptions. Even for the mathematical facts like "2+2=4" that are considered as non-defeasible truths, there exist mathematical contexts like the base-3 system where the symbol "4" does not exist and hence the statement is not "TRUE" in a strict sense. Hence, finding exceptions to existing rules is a matter of time or finding an appropriate context or perspective.

Scientific definitions like "The atomic number of carbon is six" and "Birds are feathered" are considered to be exact and absolute. This means that, by definition, that is how a thing is defined and hence there is no question of exceptions to it. Still, as time goes by, even scientific definitions change. The best example of this comes from the "as time changes, time changes" anecdote. This is the true story about the evolving scientific definition of one second, the basic unit of time. It is taken from one of the reference articles on the website of the US Naval Observatory. Historically, the second was defined in terms of the rotation of the Earth as 1/86,400 of a mean solar day. By the 1950s,

[2] A rule may be more preferred due to any reason other than having few exceptions. Whatever be the criterion, as long as the preference orderings make correct default reasoning, they are justified.

it was recognized that the Earth's rotation was not sufficiently uniform and hence could not be used as a standard of time. Thus, the second was defined based on the motion of the Sun at the epoch 1900 based on astronomical observations made during the eighteenth and nineteenth centuries. It was defined as the fraction 1/31,556,925.9747 of the tropical year for January 1st, 1900 at 12 hours ephemeris time. In the late 1960s, the inconvenience of defining time in terms of motions that are astronomical in space and distant in time led to the currently accepted definition — the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium 133 atom. It may seem far-fetched to design the AI systems for such changes in definitions which are very rare but so did the design that ultimately led to the Y2K problem. There is no harm in designing systems that would be flexible when required. There is especially no harm in this particular case, because such a common representation system for rules does not even increase resource requirements. As a post-script to the story narrated above, scientists have found that this definition of second leads to some lag because of constant deceleration of Earth. The solution to this could be to redefine the second in terms of the deceleration factor or alternatively, introduce a patch regularly. This patchwork is currently called the leap second!

The point of this discussion is that to begin with, every rule does act like a default rule in some context with less information, and even in stable contexts where the rule is an absolute truth, default reasoning can be done by defining a notion of preferences among those rules. An important distinction between previous solutions and the solution of this thesis is that in the solution of this thesis, all rules are alike and the rules that represent generalizations or default statements are not treated or represented differently as done previously. Since the context of a KB is dynamic, we cannot be sure at the time a rule is inducted into the KB whether it will end up having some exceptions. Defaults are implemented implicitly through the preferences. A default case fires when all exceptional rules (or the more specific rules or the more preferred rules) fail to fire.

3.2 Preference Orderings

If all rules are to look alike, default reasoning would require some mechanism different from separate representation for default rules. One of the ways to reason with defaults is to "infer" all that

is derivable through classical logic but not "believe" all of it, if required. This means that we must distinguish the notion of the *derivable* and the *derived* propositions. If there are two propositions that are contradictory to each other, elimination of the contradiction might be done by *prefer*ring one result over the other. Thus, both the contradictory results are derivable, but only one of them is a part of the current belief space. This implementation requires a preference ordering amongst beliefs.

Most beliefs are incomparable with respect to the extent of their truth values. Most beliefs have an exception, and that is why they are true only to an extent. Beliefs are not just "True" or "False" but partially (or possibly completely) "True" or "False". This is so because, as argued earlier, defaults exist and they are not "True" in a strict sense. Beliefs can be ordered according to how true or specific they are. Another way to look at this is to say that some propositions are more *believable* than others and hence may be considered as *more* true than them.[3] A proposition may be more believable than another because its source might be more credible. This means that in case of multiple sources of information, the concept of "credibility of sources" becomes tantamount to "believability of information" from the sources which in turn is equivalent to "extent of truth value" of the proposition. The concept of "extent of truth value" may or may not be based on their probabilistic or statistical valuation.

One of the ways to order beliefs is to rank the beliefs. The idea is to prefer the results that are derivable through the higher ranked beliefs. To rank beliefs means to associate an ordinal number with each of the beliefs. The criterion of the association could be domain-specific and could be based on a statistical or a probabilistic model. Thus, for example, let us assume that on statistical evidence, 97 of percent birds fly and 99 percent of penguins do not fly (taking into account the imaginary super-penguins that we know fly!). Thus, if the rule "Penguins do not fly" is ranked higher than "Birds fly", "Opus, the penguin, does not fly" could be *derived* in preference to "Opus, the penguin and the bird, flies".[4]

However, rankings are not appropriate, because ranks order the beliefs fully, while there exist

[3]Note that since we are talking about various levels of truth, we are moving away from the conventional/classical two valued logic now

[4]Of course, "Penguins are birds" is a rule of the system also.

beliefs that cannot be compared at all either because of the domains they deal in or for the lack of objective understanding of the extent of their truth-values. There might be awareness of just the qualitative comparison amongst some beliefs.

3.2.1 Inappropriateness of hard numbers

In the real world, numbers are used to rank various beliefs. Tennis players are seeded in a tournament to tell who the better players are and hence are more likely to win. Thus, there exists a generalized belief Pete Sampras, ranked 1 in the ATP rankings, is a better player than Tim Henman, ranked 10 in the ATP rankings. The American Film Institute (AFI) has ranked the all-time greatest movies. Thus, there is generalized belief that Citizen Kane (rank 1) is a better movie than My Fair Lady (rank 77). Beauty contests rank the participants to declare the winners. Hence, the belief Miss Aruba (first runner up and hence rank 2) is more beautiful than Miss Brazil (rank 13) and so on.

The number one tennis player is incomparable to the best movie ever made. In fact the whole idea of comparing the players with movies is ridiculous, and yet both of these are associated with the number “1”. It is as if the numeral “1” is the common property between the two and, at some abstract level, equates them. Similarly, number “1” player is not better than the number “3” beauty queen. Clearly, such a use of numbers is not intended. Real life use of numbers is vague and uses just a limited semantics. However, AI/computer systems are exact, and vagueness has to be formalized through some appropriate representation of knowledge. Hence, it is inappropriate to rank beliefs by assigning numbers to them.

To understand why this is not only a problem at a philosophical level but also may have a bearing on the process of reasoning, let us observe an example.

Consider again the Truth Maintenance System, briefly introduced earlier in section 1.2. In this system, let us assume that there are two independent agents this time that are sources of information. The object of the system is again to identify an observed entity. One of the sources, say agent1, contains in its knowledge base (KB) the beliefs that the entity is either a tank or a ship, with ranks

5 and 7 respectively.[5] The second source, say agent2, contains in its KB the beliefs that the entity is either a tank or an airplane, with ranks 5 and 8. Independent queries on the two KBs work fine, and reasonable deductions are possible. The problem arises when the two are combined to form a single joint KB. If the criteria of ranking in the two KBs is not equivalent, the system could end up with rankings that are not justified. For example, in the case of the agent described above, the joint KB would have the beliefs that the object observed was a tank or a ship or an airplane with ranks 5, 7, and 8, respectively. It seems from this merger that the belief "The object is an airplane" (rank 8) is ranked higher than the belief "The object is a ship" (rank 7). This is not justified, because none of the two agents have compared these two beliefs internally and independently. Why, then, should a joint KB of the two be able to provide a comparison? In general, rankings in a KB are relative, and standardization of ranks over independent KBs is not guaranteed. Hence, the rankings could be unjustified, and inferences could be erroneous. A workaround to this problem would be to make the act of merging two KBs more complex by normalizing ranks of the beliefs from various sources. This is not only unnecessary, because a cheaper alternative exists, but also unsound, since in some cases the ranks are really not comparable. For instance, it would not make sense to assign ranks and hence compare two semantically unrelated propositions like those relating to persons and movies.

3.2.2 Partial Orders

[Hal01] argues that beliefs in default reasoning may be viewed as a plausibility space. A plausibility space is a 3-tuple (W,X,π) where W is a set of possible worlds, π(w) is a truth assignment to primitive propositions for each world w$\in$W, and X can be viewed as a "measure" on W. The "measure" could be anything like χ-rankings[GP96], possibility measures, etc. A probability space is an instance of a plausibility space that maps sets in X to numbers in [0,1]. A plausibility set maps them to some arbitrary partially ordered set. The crux of the argument is that beliefs can be partially ordered according to a preference criterion.

Thus, the semantics of the notion of preference is as follows

[5]The ranks might be given on some domain-specific criteria and could be based on a statistical model. Without loss of generality, let us assume, high rank means more preferred.

"If A$\prec$B, read as proposition A is less preferred than proposition B, and both the propositions, A and B, are applicable to an instance, then prefer the result derived using B, if the two results derived using A and B are contradictory to each other".

The point to note here is that the preference rule, the rule that says which proposition is preferred over which other proposition like A$\prec$B, may be a statement of preference over "rules", while the actual act of preference acts on the "inference" or the instance. The implication of this is that two contradictory "rules" can coexist in the current belief space (CBS) because they are just general statements. However, two specific statements that are inferred and are contradictory cannot coexist. Thus, a choice between "Opus flies" and "Opus does not fly" has to be made, and only one of them could stay in the CBS. However, "Birds fly", "Penguins are birds" and "Penguins do not fly" can all be asserted in the CBS at the same time. This is consistent with [DS99] that states that contradictory default rules can stay in a belief space simultaneously. Coexistence of contradictory rules just means that contradictory inferences are derivable. It does not mean that contradictory inferences are derived. The above notion of preference also means that ground instances of rules can be more preferred or less preferred than a rule or another ground instance of a rule. This last statement about the notion of preferences may lead to assertions of preferences that do not have a deep semantic value. This means that comparing two absolutely unrelated propositions or rules may not have any semantics. This brings us to the need for explicit preference ordering relations and is explained and illustrated later, in the example 2 in section 3.2.3.

3.2.3 Preclusion versus Explicit Preference Ordering

The NETL project was described in section 2.2. As stated earlier, the net approach of [Tou86] orders the rules implicitly. This arises out of defining inheritability on the basis of preclusion. The graph structure for knowledge representation clearly depicts a preference ordering of rules. Thus, by defining a criterion of ordering the rules based on specificity, it is already determined which rules are preferred in case of contradictory inferences. That we need to specify an explicit ordering of rules to be able to carry out preferences is a redundancy. Explicit preference rules are redundant information because the rules are computable on the basis of preclusion as defined in chapter 2.

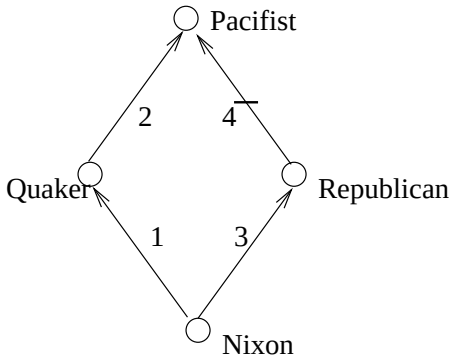

Figure 3.1: Representation of Example 2 in NETL

However, we still need the preference rules because of efficiency and to be able to say more than what the concept of preclusion would allow us to.

[Tou86] describes two procedures to decide the question of specificity between two nodes namely, the UPSCAN algorithm and the DOWNSCAN algorithm. [Tho92] asserts that both these techniques are NP-complete and hence computationally intractable. Tractability is the first reason for which the redundant information of explicit preference order rules may be used. The second reason is to specify preference ordering between two rules whose preference ordering is not deducible from the structure of the graph itself. Such cases arise when no implicit preclusion is a part of the KB. For example, in the famous Nixon diamond problem, which is restated below with its NETL representation in figure 3.1, it is not possible to say if Nixon is a pacifist or not because none of the paths precludes the others.

Example 2:
1:"Nixon was a Quaker"
2:"Quakers are pacifists"
3:"Nixon was a Republican"
4:"Republicans are non-pacifists"

There are many ways of deducing that Nixon was not a pacifist, using preference ordering. One of them is to simply state that the statement "Nixon was a non-pacifist" is a more preferred statement than the statement "Nixon was a pacifist". By stating this, what is meant is that if both of these statements were in the current belief space then the latter is precluded by the former. However, if we did not want to assert such specific preference ordering, we may may still be able to derive one

of them by precluding its proof by the proof of its contradiction. The intuition of precluding proofs comes from the following situation.

If all the statements above were told to an independent jury and it was added that the statement "Nixon was a Republican" is more credible than "Quakers are pacifists", the jury would be inclined to concluding that "Nixon was a non-pacifist". Now, the proof of "Nixon was a pacifist" is the set of assertions $\{1,2\}$, where the numbers are the numbers associated with the assertion as in example 2 above. The proof of "Nixon was a non-pacifist" is the set of assertions $\{3,4\}$. Hence, precluding proofs is a way of concluding the desired result. Proofs may be precluded in many ways. One of them is to assert that "*Quakers are pacifists* is a less preferred rule than "*Republicans are non-pacifists*". Another solution is to assert that "*Republicans are non-pacifists* is a more preferred rule than "*Nixon was a quaker*". In the second solution, the preference ordering is between a rule and a proposition. Also it is a preference ordering involving no common propositions or nodes. Such a preference ordering is possible only through explicit preference rules. There is no algorithm that can deduce such preference rules because there are no common nodes and hence presumably no related semantics. Hence, our procedure involves adding the explicit preference rules in the net.

Chapter 4

SNePS

The Semantic Network Processing System or SNePS ([SR92], [SG02] and [Sha79]) is the semantic-network based KRR system built by the members of SNePS research group in the computer science and engineering department at the University at Buffalo. It has been a result of continued research on projects for several years and has been implemented in Lisp.

In SNePS, a proposition is represented in the network with a node. Some other nodes represent entities that assist building propositions. Nodes are connected through arcs that have labels and a well-defined semantics. Each propositional node has a support that is used to decide if a node is a part of the current context or not. The support of a proposition can be used to build a proof for the proposition. The support of a proposition contains the assumptions used to prove the proposition. SWM[MS88] introduced the concept of supported wffs. A supported wff is a 3-tuple $< P, \tau, \alpha >$ where P is a wff, τ is the origin tag taking values from the set $\{hyp, der\}$, and α is the origin set. The origin set is a set and consists of all the hypotheses that are used to prove P (ATMS[1] style). A proposition P may have multiple origin sets, each one containing the hypotheses underlying one of the multiple proofs of P. $(\tau = hyp)$ implies that P exists as a hypothesis in the knowledge base. In this case, α contains just the wff P. $(\tau = der)$ implies that P is a wff that is derivable using the hypotheses in the origin set α. The pair (τ, α) is called the support of P denoted as $Supp(< P, \tau, \alpha >)$ = (τ, α).

[1] Assumption Based Truth Maintenance System. Such systems store for a proposition, the sets of assumptions that imply the proposition.

4.1 User's View

SNePS can be used to represent mental entities. Mental entities include all entities that Cassie (the cognitive agent) can think about. Each entity is represented in SNePS by a node. Nodes are connected to each other through arcs that represent a relation between two nodes. Arcs are directed from one node to another, and nodes and arcs together form a directed graph or a network. There are four kinds of nodes in SNePS, namely

1. Base node - Represents an entity conceptually different from other entities. They are atomic and have no structure i.e. no arcs emanate from base nodes.

2. Molecular node - Represents a proposition, a rule, or a structural individual. Molecular nodes are similar to base nodes except that they dominate other nodes i.e. directed arcs begin from molecular nodes.

3. Variable node - Represents any arbitrary individual or proposition.

4. Pattern node - Is like a molecular node with free variables (dominating a variable node)

The network consists of asserted and unasserted molecular nodes. The asserted nodes represent the current beliefs of the cognitive agent.

SNePS can be used to carry out forward and backward inferencing on the existing knowledge base (KB). The KB comprises the propositions represented by the asserted molecular nodes.

The knowledge engineer or the user of SNePS typically uses it as follows:

1. Define the arc labels or the relations for the context(s) to be represented in SNePS.

2. Using these relations and some of the basic ones provided by SNePS (for example the quantifiers like forall, ant-cq pair, etc.), build the rules for the system.

3. Assert the base propositions

4. Activate inferencing for required queries.

4.2 Detailed Design of SNePS

SNePS may be divided into five major subsystems.

1. SNePS-SNePSUL interface : This provides a user interface for building and finding information in the network. In other words, this package is responsible for creating nodes and connecting arcs.

2. Multi : This is a simulation of a multi-processing system for controlling inference. A SNIP (see later) process is an activation of a node. Each node is associated with a process. The multi package maintains a queue of processes and schedules them in a first-in first-out (FIFO) order. Processes are of three kinds namely - rule, non-rule, and act. Rule processes correspond to the rules of inference like or-entailment, and-entailment, nor, etc. The activation of three kinds of processes are handled differently. Whenever a proposition is to be inferred, its pattern node is built (if the node corresponding to the proposition does not already exist). The node to be inferred associates with a process and each such process in an inference procedure goes through the process of testing and matching and unification. The processes return the resultant molecular nodes if they are inferred.

3. Match : This is the network pattern matcher

4. SNIP : This is the inference package that handles forward and backward inferencing using the rules in the network. The node processes communicate with each other through “channels” that run parallel to rule arcs or between matched nodes. In a true OO style, processes send and receive “messages” . Messages can be “request” or “reports”. Request messages contain necessary information to form further channels through which reports can be sent. Report messages contain “substitutions” that represent instances which have been determined to be derivable in the network. Reports are generated by forward inference or as responses to requests initiated by backward inference. Reports are scheduled ahead of request in the queue maintained in the Multi package for any-time reasoning and are returned to the user as soon as possible.

5. SNeBR : This is the belief revision system. It has traditionally not been considered as a “core” part of SNePS, but it forms an independent subsystem just like the ones mentioned above.

SNeBR is triggered when an explicit contradiction is introduced into the SNePS belief space, either because of a user's new assertion, or because of a user's query. SNeBR then makes the user decide what belief to remove from the belief space in order to restore consistency, although it provides information to help the user in making that decision. SNePSwD [CM93], that is described briefly in the next section, extends SNeBR to handle non-monotonic logic.

The code for SNePS 2.5 was used as the basic framework for developing the default reasoning module for this project.

4.3 Default Reasoning in SNePSwD

Default Reasoning has been implemented earlier in one of the previous versions of SNePS and is called SNePSwD. SNePSwD is based on the SWMC (Shapiro, Wand, Martins and Cravo)[CM92] logic which is an extension of the SWM[MS88] logic. SWMC is a nonmonotonic logic that supports ATMS-like systems with reasoning capabilities. SWMC allows for default reasoning, i.e., the use of default rules, which are not universally true (rules with exceptions). SWMC allows expression of default rules and exceptions to these rules. It also presents a semantics for SWMC, for which the logic is sound and complete. It uses a particular quantifier, named default quantifier, and presents a theory of inference rules to deal with the default quantifier. The implementation uses supported wffs as described earlier in this chapter.

For supporting default reasoning, the origin tag takes values from the set {*hyp*,*asp*,*der*}: *hyp* identifies hypotheses, *asp* identifies assumptions and *der* identifies derived wffs. Here, the new addition to the set is the assumptions. They differ from hypotheses. Hypotheses are wffs that represent the available knowledge from which we want to draw conclusions, using SWMC. Assumptions, on the other hand, do not represent any knowledge from which conclusions can be drawn. They do not represent any knowledge that is explicitly asserted in the KB. They represent a piece of knowledge that helps forming a weak argument to a deduced proposition that may be defeated in wake of stronger, and more specific or explicit knowledge.

Based on the default logic formalism, SNePSwD maintains a set of default rules $\mathcal{L}_D$ that is dis-

joint from the set of wffs from classical logic, denoted by $\mathcal{L}_{FOL}$.[2] To express default rules the default quantifier, represented by symbol $\bigtriangledown$, is introduced. If A(x), B(x)$\in \mathcal{L}_{FOL}$, then $\bigtriangledown$(x) A(x)$\Rightarrow$B(x) is a wff and is called a default rule. Default rules are neither believed nor disbelieved in a system. They are applied to follow the line of reasoning they suggest. They are presumed knowledge that can easily be defeated if a contradiction arises. The assumptions as described in the previous paragraph are used every time some defeasible conclusion is to be drawn using a default rule D by applying to a particular individual c. Each such defeasible conclusion has assumptions in its support.

Another disjoint set maintained by SWMC logic is the set of extended wffs, denoted by $\mathcal{L}_E$. Following are the two kinds of extended wffs -

- If D is a default rule and c is an individual, then *Applicable*(D,c) is an extended wff and is an *assumption* that the default rule D is applicable to a particular individual c. Thus, if A(c) and *Applicable*(D,c) where D = $\bigtriangledown$(x) A(x)$\Rightarrow$B(x), the wff B(c) is a conclusion of the assumption. The treatment of propositions of this kind as assumption is crucial. If the assumption were not associated with the derivation of B(c) and some time later, the system came to believe $\neg$B(c), it would conclude that the set $\{\bigtriangledown$(x) A(x)$\Rightarrow$B(x), A(c), $\neg$B(c)$\}$ is inconsistent, which might be wrong in case the default rule is not applicable to c. Assumptions are easily defeasible and hence, would take care of the new order as a consistent behavior.

- If A(x)$\in \mathcal{L}_{FOL}$ and D$\in \mathcal{L}_D$ then $\forall$(x) A(x)$\Rightarrow$ $\neg$*Applicable*(D,x)$\in \mathcal{L}_E$. This kind of proposition is used to express exceptions to default rules. They block the applicability of default rules in certain cases.

As is clear from the brief description of the implementation, SNePSwD is based on the notion of applicability. For every individual for which a default rule is applicable, an *Applicable* proposition exists. This means that for a default case that applies presumably, to a majority of population of individuals, a large number of such *Applicable* rules have to be a part of the KB. This is one of the problems with this implementation. It would be much more efficient to block the applicability of the default rules in the exceptional cases that may be safely assumed to be much lesser in number. There is however, a much bigger problem with this implementation that was argued against in the previous chapter in section 3.1. This happens to be the problem with most other representation

[2]FOL stands for First Order Logic

systems like Nonmonotonic Logic, Default Logic and Autoepistemic Logic. This is the problem of existence of a special syntax for default rules. It is argued that this is not only unnecessary but also cumbersome for maintaining dynamism, which is a key feature of a typical KB.

Chapter 5

Implementation of Default Reasoning in SNePS

This chapter presents an algorithm and the description of an implementation of default reasoning in SNePS. The idea is based on previous foundation work described in chapter 2 and implementation of SWM logic. Specifically, preference ordering is used to decide the applicability of a rule in a context.

5.1 Implementation Ideas

This section contains some ideas that were initially thought of before implementing preferential ordering of beliefs for default reasoning. These ideas even though not implemented in direct form in SNePS currently because of some constraints, still direct the current implementation. They may be used in future.

5.1.1 Preclusion Sets

Our first attempt at implementation involved modification to the existing support for a node. A preclusion set was added to the support of a proposition.

Touretzky's NETL project ([Fah79], [Tho92], [Tou86] and [TT90]) introduced the concept of preclusion. Preclusion inherently orders beliefs in order of specificity. The less preferred conclu-

sions are derivable but not derived in a context with enough information for the most preferred rule to "fire". Grosof uses the concept of ranked beliefs in Logic Programs[Gro97] to implement defaults. Essentially, his procedure decides iteratively, for each proposition, starting from the one with the shortest longest-proof.[1] While apparently the procedure seems to be working on propositions, it actually comes out with a solution that has a proof consistent with current context. This is fine since a proposition may be derivable through multiple proofs not all of which may be precluded. In that case, the proposition should be derived. If on the other hand, preclusion were to act only on propositions, we would not be able to derive it if there existed some proof that derived its contradiction, even if that proof itself could be precluded in the current context.

Combining these two ideas we realize that -

1. A support should have some information to say that the wff is derivable but is not derived i.e. is precluded under certain circumstances.

2. Preclusion should act on proofs rather than propositions. That means a proposition is derived even if there exists a proof for its contradiction as long as that proof itself is precluded in the current belief space. Just because there is some proof that contradicts a proposition does not imply that the proposition is not-believed.

For this, a new definition of a supported wff was proposed. A supported wff, would now be defined as a 2-tuple $< P, S >$ where P is a wff and S is a set of supports defined as (τ, α, β). where, τ is the origin tag as in SWM logic and takes values from the set $\{hyp, der\}$. α is the origin set as in SWM logic and is the set of assumptions underlying a proof for wff P. β is a new addition to the support and is called the preclusion set. It is a set of sets and each element set is a set of wffs. For some set $\Psi \in \beta$ if all the members of Ψ hold in the current belief space (CBS), the corresponding origin set α (that represents a proof) is rendered useless/precluded. The support (τ, α, β) is then said to be precluded.

[1] For all the wffs, choose their longest proof and order the wffs on it. Now choose the wff in the increasing order. By "length" of a proof it is meant, the number of implication rules or alternatively, the path size of topologically sorted atom dependency graph as described in [Gro97].

P is derived if it is derivable in the CBS and there exists a support that is not precluded. Thus, even though *P* with support (τ, α, β), "may" be a part of CBS through multiple supports due to $\alpha \subset$ CBS, it "would not" be, if for every such support its preclusion set disllaows it. The sufficient condition for *P* to be derived is $\alpha \subset$ CBS and $\neg\exists\gamma : (\gamma \in \beta) \wedge (\gamma \in$CBS).

For a wff P, for each origin set α that it has, the corresponding preclusion set β is created as follows.

1. Initialize β to a null set ϕ.
2. For every proposition x$\in \alpha$,
   (a) find an origin set in $\neg$P that contains y such that x$\prec$ *y*.
   (b) $\beta = \beta \cup$ { y, x$\prec$y }

The preclusion set consists of the more preferred rule for each of the rules in the origin set and thus precludes the support if it holds in the CBS.

To see how this works, consider again the example 1 in section 2.2 and illustrated in figure 2.1. In this example, let us augment the CBS with the following statement.

Example 1 (continued):

10: "Opus is a flying-penguin".

11: 5$\prec$*7*

12: 7$\prec$*9*

With the preclusion sets, the support for "Opus flies" and "Opus does not fly" are as follows.

Opus flies:	$\alpha_1 = \{1,4,5,8,10\}$	$\beta_1 = \{\{7,11\}\}$
	$\alpha_2 = \{9,10\}$	$\beta_2 = \{\{\}\}$
Opus does not fly:	$\alpha = \{\{7,8,10\}\}$	$\beta = \{\{9,12\}\}$

From the supports given above, there are two justifications (represented through two origin sets) for "Opus flies". The first justification is precluded if the statements 7 and 11 are in the CBS and that is why its preclusion set contains the statements 7 and 11. The second justification is not precluded and hence its corresponding preclusion set is empty. There is only one justification for "Opus does

not fly" that is precluded if statements 9 and 12 are in the CBS. Thus, both, "Opus flies" and "Opus does not fly", are derivable in contexts that contain all the statements except 9 or 12 or both. In a context that contains both, statement 9 and statement 12, along with others, Opus flies not because Opus is derivable as a bird and birds fly. This justification is precluded by a justification that leads to the result that Opus does not fly. Finally, Opus flies in a context that contains all of the above statements because "Flying-penguins fly" is not precluded by any statement. Thus, proofs preclude proofs. A demonstration of the modified version of this example is shown in demo 1 in appendix A also.

5.1.2 Preference Graphs

The idea of ordering beliefs based on preferences is observed to be useful for default reasoning. This is so because the specific rules may be viewed as the rules more preferred than the default rules, if both are applicable. However, the scope of preference orderings is much wider in general. This means that, in general, any rule may be stated to be more preferred than another rule and the reasoning should honor that. This characteristic of preference orderings has led to a better understanding of the concept of preclusion and led us to the concept of preference graphs.

Specifically, as briefly hinted earlier in section 3.2.2, preference orderings allow coexistence of contradictory rules. Thus, contradictory rules like "Birds fly" and "Birds do not fly" can coexist. The two rules are default rules for two different contexts. However, preference ordering would allow reasoning in the two contexts without having to talk about contexts. To see how this works, consider the following example.

Example 3:
1: "Birds fly"
2: "Birds do not fly"
3: "Betty is a bird"
4: $1 \prec 2$
5: $2 \prec 1$
6: $4 \prec 5$

It should be possible in the above belief space to deduce that Betty, the bird, flies. This is so because the decisive preclusion (statement 6) acts on preclusion rules and effectively leads to preclusion of statement 2 by statement 1. This is a case where preclusion sets in the support would not be able to help make a deduction. The statement 6 is what we term as level-2 preclusion. In general, preclusion could act on any kind of rules including the preclusion rules of higher levels. The introduction of multiple levels of preference rules enables reasoning and efficient representation of certain pieces of knowledge. For example, the knowledge base of example 3 actually represents knowledge of two contexts. Statement 1 is a default rule in the default context of places outside New Zealand(NZ). Statement 2 on the other hand, is a default rule in the context of NZ. Thus, in reality, statement 1 and statement 2 would not contradict because they would be in two disjoint contexts of inside NZ and outside NZ. Preference rules allow representation of knowledge of these two contexts without explicitly talking about contexts. Following is the restatement of knowledge base of example 3 that uses the NZ predicate to combine the two knowledge bases.

Example 3a (Example 3 restated):
1: Bird(x)⇒fly(x)
2: Bird(x)⇒ ¬fly(x)
3: Bird(Betty)
4: 1 ≺ 2
5: 2 ≺ 1
6: InNewZealand⇒4
7: ¬InNewZealand⇒5
8: InNewZealand≺¬InNewZealand

The difference between example 3 and example 3a is that in the former, there is a level-2 default while in the latter the same effect is achieved by "conditional implication" of a preclusion rule. In example 3a, statements 4 and 5 may be conditionally a part of the current belief space and hence there would not be a problem of contradictory deductions due to these two. Preference graphs are needed to be able to reason in KBs in which representation is done using level-2 preference rules. Following is an algorithm that does the same.

The algorithm in this section is for the KB in a NETL representation system and uses its terminology. For our purpose, without loss of generality, we may assume the A-types(Γ) values, as

defined for a proposition in the NETL project, to be limited to only two values - the positive and negative literal forms of the propositions. The algorithm is based on [Gro97]. It finds a "most preferred" answer value for a proposition with the given rules and preference rules. Identification of rules is done by association of a unique natural number with each of the rules or links. Thus, a link *l* may now be denoted as *n:l* where, n$\in \mathcal{N}$. The numbers on the arcs in figures 2.1 and 2.2 owe their existence to this. Ideally, rules in the preference graph can be labeled and meta-meta-rules can be formed also. Demo 2 in appendix A illustrates how a second level default may be used to derive preclusion relations. In principle, any number of levels of preference graphs may exist. Hence, a hierarchy of preference graphs exist and they are defined iteratively as follows.

Definition 1: Preference Graphs

The Preference Graph $\mathcal{P}_0(V_0, E_0)$ of net Γ, denoted as $\mathcal{P}_0(\Gamma)$, is a directed graph such that
$V_0 = \{n \mid n \in \mathcal{N}$ and l is a link such that n:l$\in \Gamma\}$
$E_0 = \{(x,y) \mid x,y \in V_0$ and $x \prec y\}$, where $x \prec y$ has the semantics that rule x is less preferred than rule y.
Subsequent preference graphs $\mathcal{P}_i(V_i, E_i)$ for i>0 are defined recursively as
$V_i = \{n \mid n \in \mathcal{N}$ and n:l$\in \mathcal{P}_{i-1}(\Gamma)$ and l is some link$\}$
$E_i = \{(x,y) \mid x,y \in V_i$ and $x \prec y\}$, where $x \prec y$ has the semantics that rule x is less preferred than rule y.

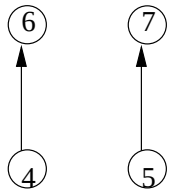

Figure 5.1: Preference Graph

Figure 5.1 shows the preference graph for the net shown in Figure 2.1. As is clear, links in a preference graph $\mathcal{P}_i$ are represented as nodes in the preference graph $\mathcal{P}_{i+1}$. To resolve conflicts when they arise, the links of the path of each inference must be stored with the inferred proposition. Conflict resolution as to which literal form of a proposition, positive or negative, is acceptable is made on the basis of comparison of these sequences. This scheme is similar to the ATMS that is already implemented in SNePS, ultimately.

Definition 2: Contradiction on sets of nodes

Two sets of nodes S_1 and S_2 that contain nodes of $\mathcal{P}_i$ for some i, contradict each other if either the

set of nodes $S_1 \cup S_2$ is empty or all the subgraphs of $\mathcal{P}_i$ that contain the set of nodes $S_1 \cup S_2$ are cyclic.

Definition 3: Order on sets of nodes

For two sets of nodes of $\mathcal{P}_i$, S_1 and S_2, $S_1 \prec\prec S_2$, read as set S_1 is less preferred than S_2 if either one of the following holds.

1. *S_1 and S_2 do not contradict in $\mathcal{P}_i$ and, there exists a pair (x,y) such that $x \in S_1$, $y \in S_2$ and there is a path from x to y in $\mathcal{P}_i$ or,*
2. *S_1 and S_2 contradict in $\mathcal{P}_i$ and $S_{21} \prec\prec S_{12}$ in $\mathcal{P}_{i+1}$ where, S_{pq} is the set of nodes that have links from the nodes in S_p to the nodes in S_q.*

The sets of nodes in above definitions would refer to the ones consisting of all the links of the path of each inference that leads to deduction of a proposition. Thus, the sets represent a proof and hence are similar to the origin sets used in SNePS. Note that the above stated definition of order on sets is recursive. Conflict resolution among various inferences of a single query is possible only if there exists exactly one literal type whose set is more preferred than set of the other literal type.

The algorithm for finding an answer to a query Q in a KB with various levels of preference graphs is as follows. In the following algorithm, X is a sequence of predicates, and predicates P, Q, Y etc. are in their atomic forms, i.e. no matter whether they exist positively or negatively in a rule, the predicate symbol is the only portion that is taken into consideration. The operator + denotes the concatenation operator and - is the delete operator.

```
module:createSortedSequence(i, P_i)
input:  integer i, Preference graph P_i
output:A sequence S_i, of nodes representing propositions,
in topologically sorted order.
algorithm:
S_i = X = arbitrary sequence of maximal elements of P_i.
(The literal L is a maximal element if there does not
exist in P_i, a literal M such that L≺M.)

while X is not empty
      for every element Y, where Y is atomic, in X {
          X = {X - Y};
          if Y is not a leaf (i.e.  not a hypothesis/fact) {
             for every Z such that Z≺Y is a rule {
                 if Z is in S_i
                    S_i=(Z) + {S_i - Z};
                 else
                    S_i=(Z) + S_i;
```

```
                X = X + Z;
            }
        }
    }
```

`createSortedSequence` module returns a topologically sorted sequence of nodes for a preference graph. This sequence is then used to decide the applicability of rules in the next lower level preference graph. The preference graph $\mathcal{P}_0$ contains the propositions and rules of the KB. Each proposition stores the hypotheses and rules that are used to deduce it. These hypotheses/rules form a set of propositions. If the set is more preferred than that of the proposition's negation, then the proposition is deduced.

module:**isBelieved(Q)**
input: Query Q, and the net Γ consisting of preference graphs $\mathcal{P}_1$ to $\mathcal{P}_{max}$
output:An answer to query Q.
algorithm:
forevery preference graph $\mathcal{P}_i$,
   S_i = createSortedSequence(i, $\mathcal{P}_i$);
(ie. form set of sequences S_1, S_2 etc. for each of the preference graphs. S_1 is for meta-rules for P_0, S_2 is for meta-meta-rules and so on.)

Let Answer set A = {set of hypotheses/facts} in S_{max}

```
for i = max downto 1 {
   for each atom Z in S_i {
       RULES(Z) = the set of rules for which Z is a consequent and
                  the antecedent is a literal from the answer
                  set A (i.e.  Y≺Z for some Y).
   }
   Using the "order on sets of nodes" for all the RULES(Z),
   find the "most preferred" set.  If a "most preferred" set exists,
   put the corresponding literal, Z in the answer set A
}
For Q and ¬Q,
   for each of its preclusion sets β,
       if no set B∈β is such that B⊂(P0∪ A),
       then deduce the corresponding proposition.
```

The `isBelieved` module in the algorithm invokes the `createSortedSequence` module to sort the nodes in each preference graph. Starting from the highest level preference graph, it iteratively decides for the nodes in each preference graph their preculsion status. The step involving

ordering on sets of nodes is really at the heart of the computation and needs some explanation as to how "most preferred" answer is found. Essentially, the step contains an algorithm that checks, for each node a more preferred node by finding meta-rules that decide between the two contentious forms. Iteratively, this check is made unto the highest level of meta-rules. Computationally, this is intractable, and efficient methods to implement this may be a research matter. However, this allows provisions for a hierarchy of default rules and their preferences and deductions based on them. The unambiguous answer, if it exists, is finally present in the answer set A, returned by the algorithm. The working of the algorithm becomes clearer with the following example that is a continuation of the example 3 shown earlier.

Example 3(contd.):
To find an answer to query Q = Fly(Betty)? following is arrived at.

$P_2 = \{6\}$	$S_2 = (6)$	$RULES(6) = ()$	$A = \{6\}$
$P_1 = \{4,5\}$	$S_1 = (1,2)$	$RULES(1) = 5, RULES(2) = 4$	$A = \{6,5,1\}$
$P_0 = \{1,2,3\}$			
P_0	$\neg Fly(Betty)$	$\alpha = \{2,3\}$	$\beta = \{\{1,5\}\}$
	$Fly(Betty)$	$\alpha = \{1,3\}$	$\beta = \{\{2,4\}\}$

In P_0 ¬Fly(Betty) is precluded since its preclusion set is in the answer set A.Thus, Fly(Betty) is deduced.

This example shows why preference graphs are necessary. It shows that two contradictory rules may co-exist in a belief space. Rule 1 and Rule 2 are contradictory and still they may exist simultaneously. This could happen if the two rules come from two independent sources and there is no way to decide which one is correct. Since default reasoning has been found to be acting not upon the default rules but on their instantiations, it is perfectly valid to have two contradictory rules. Now let us say, rule 4 and rule 5 make their way into the CBS also and again the stalemate of deciding between whether Betty, the bird flies or not continues. The rule 6 that asserts preclusion relation between two preclusion rules, should enable us to deduce that Betty flies. In this example, the contents of P_0 alone cannot decide if Betty, the bird, flies. The contents of P_1 are insufficient to make a deduction also.

In the current implementation in SNePS, second level of preference ordering is supported. Demo 2 in appendix A demonstrates a second-level nested default at work. Extension of this work to

support arbitrary number of levels of preference graph could be a research issue.

5.2 SNePS Implementation

5.2.1 New Preclusion Relation

SNePS has some predefined relations like `forall, min, max, ant, cq` etc. This set of basic relations are used for inferencing in SNIP. As argued in the section 3.2.3, explicit preference rules are required in the KB for efficient and required deductions in default reasoning. To this end, two new relations - `precludes` and `precluded` have been defined. The two are always used together. The semantics of the following usage of the two relations

```
(assert precludes prop1 precluded prop2)
```

is that proposition prop1 is more preferred than the proposition prop2. If the two propositions, prop1 and prop2, contradict each other directly, then prop1 is believed while prop2 is not. If there exists a contradiction in the belief space such that one of the answers is *derivable* using prop1 as a hypothesis and the other using prop2 as a hypothesis, then the one using prop1 is *derived*. The use of this case-frame is illustrated in the demos in appendix A.

5.2.2 The Code

The implementation of default reasoning with preference ordering has been done in Common Lisp. It uses the already existing code from SNePS 2.5. The two files modified for the implementation are as follows.

- `context3.lisp` in the `ds` directory of the `sneps` package. This file earlier contained modules for checking, printing and enacting the assert status of a node. The module that checks the assert status of a node/proposition is modified to now check if the node is precluded or not. The algorithm for checking if a node N is precluded or not is as given below.

```
result = True;
for every origin set α in the support of N
    partialResult = False;
    for each node A in α,
        find the node B that precludes A ie.  A≺B
```

```
        if B is asserted in the current context,
           partialResult = partialResult ⋁
                           is−node−in−support(B, Support(¬N))
        result = result ⋀ partialResult
return result;
```

In the process of implementation, it was observed that SNePS is built in such a way that it would be futile to store the data contained in preclusion sets permanently. The reason for keeping the preclusion sets was to cut the inference processes for the propositions that are known to be precluded short or in other words to disallow their inference. SNePS was built with a view to support multiple processes running in parallel for carrying out inferencing. Thus, each node being inferred has a process associated with it. Each such active process creates pattern nodes to search for possible bindings for the free variables that it dominates. Some of these pattern nodes that have successful bindings are finally matched with existing molecular nodes or lead to creation of new ones. It is because of this existence of temporary pattern nodes that preclusion sets cannot be used in the SNePS implementation of default reasoning with preferential ordering of beliefs. This is because preclusion sets facilitate inferencing by making it more efficient during the process of inference. They provide the information that can be used at inference time to pause the process of deduction of a node if all of its supports are found to be precluded. However, temporary nodes in SNePS do not have supports. At inference time in SNePS, there is no information attatched to the nodes involved in inference that can be used to curb the inference process. If the implementation of the inference procedure were such that molecular nodes involved in the deduction were used, we could use the preclusion sets.

However, the concept of preclusion sets may be used in any other implementation of ATMS. An implementation, in which the intermediate phase of the node being inferred can extract relevant information from the rule and its antecedent, can potentially use preclusion sets. Such information attatched to each node can make dynamic inferencing faster. This is made possible as, every time it needs to be determined if a node is asserted or not, all that needs to be done is to check if the preclusion set is a part of the current belief space or not. Preclusion sets would obviate the need for creation of such a set at assertion time.

In the current implementation, preclusion of propositions represented by molecular nodes is done dynamically. This means, every time a node is checked if it is asserted or not, it is checked if it is precluded or not. The proposition is asserted if it is derivable in the current context. A proposition is derivable in the present implementation if it were derived in the previous implementation. In the current implementation, if a node is derivable and is not precluded, it is derived. The inference with the module for preclusion works as follows.

Before deduction process for a proposition starts, deduction of all the preclusion rules that hold in the current belief space is done. Since all the preclusion rule nodes are determined before any deductions, it is known which preference rules are going to be tried for application.

Deduction of preference rules is followed by the deduction of all the derivable nodes without taking into consideration the preferences. Finally, the propositions that are precluded by others are left out from the final answer and a message stating why they were left out is printed. The algorithm above takes its inspiration from the algorithm for creation of the preclusion sets described in section 5.1.1. Thus, the vestiges of the concept of preclusion set in the current implementation are visible. It is implemented as volatile data. This preclusion set needs to be calculated every time the assert-status of a node needs to be determined. The `isassert.n` function has been modified to return T only if the input parameter node is derivable and is not precluded in the CBS. The logic for determining whether a node is derivable or not has been modified a little also. A node is derivable if either of the following two is true.

- The node exists as a hypothesis (*hyp*) in the CBS.
- The node exists as a derived node and one of its origin sets is a subset of the CBS and all the elements of the origin set are asserted in the CBS.

This modification ensures that derivability of a node does not depend on itself and hence does not lead to any circular reasoning which in turn leads to an infinite loop. The following functions/macros were defined or redefined for supporting preclusion based assertions.

- `isassert.ct:` A context is said to be asserted in CBS if all the hypotheses in it are asserted. This function returns true if the given context is asserted.
- `isassert2.n:` An origin-set with origin-tag ‘hyp’ is asserted if the origin-set is a subset of the CBS
- `isassert1.n:` An origin-set with origin-tag ‘der’ or ’ext’ is asserted if the origin-set is a subset of the CBS and the origin-set is asserted.
- `is-node-derivable:` A node is defined to be derivable if all the origin sets in its support (hypothesis, derived and extended) are asserted
- `is-node-precluded:` It returns true if the given node n is precluded in a given belief space. The node is precluded if the following conditions are met -
  1. The negation of the node n-dash exists in the current belief space AND
  2. One of the two holds -
     (a) There exists a rule node in CBS that states explicitly that ”n-dash precludes n” AND n-dash is derivable in CBS
     (b) For each origin set in the support of n, for each hypothesis in the origin set, check if the node that precludes it exists in the support of n-dash. If it does then the current origin-set is precluded. If all the origin sets in the support of n are precluded then n is precluded
- `prec-in-sup2:` This function returns true if the given node exists in some contextset of the given support else it returns NIL
- `isassert.n:` A node is said to be asserted or derived if it is not precluded and is derivable. This function checks the two conditions and returns the true status.

- `deduce.lisp` in the `fns` directory of the `snip` package. This file contains the functionality for the top level sneps command `deduce`. The following function/macros have been defined or redefined for the current implementation
  - `print-preclusion-info:` This function is very similar to the function that determines if a given node is precluded or not. If it is precluded it prints out the message

that says which node precludes which other node. It also prints their respective supports that are involved in preclusion

– `deduce*:` This function has been modified to print the preclusion statements after the deduction processes are over. The previous version of this function collected the results from the various deduction processes and print them. The new version, after collecting all the molecular nodes that have been found to be *derivable*, finds out which ones are *derived*. For the *derivable* nodes that preclude other nodes, a message stating which node precludes which, is printed. The information that a proposition is precluded or not is available no sooner than the corresponding molecular node comes into existence. As soon as it needs to be determined if a molecular node is asserted, the nodes preclusion status is determined also. Thus, at printing time, the `deduce*` function just extracts information as to which node precludes which other node and prints that. The preclusion of a node, however, is determined as soon as the supports for the node are created or modified.

– `deduce:` Before the inference processes are initiated, the deduce function infers all the preclusion rules. The new version of the `deduce` function contains a module to do just that. Deduction of preclusion rules is done to implement second level nested defaults. The function recursively deduces the preclusion rules. An extension to this project would be to support more levels of preclusion graphs that contain preclusion rules. In the current version up to two levels only are supported. This is so because only the preclusion rule nodes are deduced beforehand and not the rule nodes that assert preclusion over preclusion nodes and so on. To extend this into multiple levels, preference graphs would need to be defined as data structures and the `deduce` function would need to be modified.

Chapter 6

Discussion

This thesis is a confluence of the following ideas.

- Default reasoning without special representation for default rules.
- Preference ordering amongst beliefs/propositions for resolving conflict in inferences.
- Use of ATMS to carry out reasoning.
- Extension of Grosof's idea of deductions using preference ordering in logic programs by introducing multiple levels of preference rules.

The algorithm in the module `isBelieved` that was introduced in chapter 5 supports multiple levels of preference rules. The implementation described in chapter 5 is different from the theory proposed about the preclusion sets but follows the same fundamental philosophy and ideas as itemized above. The implementation currently supports and has been tested for two levels of defaults. An arbitrary number of levels may be supported in the future. The implementation also differs because of the way the legacy code of SNePS 2.5 was. This is explained below.

6.1 Caveats

The implementation in SNePS does not include preclusion sets. This is because the implementation of ATMS in SNePS would not be able to use the preclusion set information during the process of inference. Preclusion sets, if implemented in an ATMS, could be used to make inferences faster by stopping the inferences if they were known to be producing propositions that are precluded. Since

SNePS does inferencing by creating temporary nodes that do not have supports, inferences cannot be stopped in the middle. Thus, it was decided to do a workaround. Hence the preclusion of a node is decided at assertion time.

Since preclusion sets could not be implemented in SNePS, it is not possible to stop deduction processes that lead to derivable nodes that are finally precluded. Thus, the messages output show all the nodes that are derivable. It would have been desirable to output the messages for only the derived nodes followed by the ending messages containing the preclusion information. The current implementation displays messages for all the derivations and declares the final results, taking preclusions into consideration, in the end.

Multiple levels of preferences to be implemented using preference graphs is not implemented in the current implementation also. This implementation would require an extra data structure and the extension to the existing code is an extended project. Thus, it was decided to leave it as a separate project and show the capabilities of the related algorithm through the given structure. Thus, in the given implementation, preference ordering up to level 2 is supported. For higher levels of preference ordering, paths made up of preclusion links would be required to reason with.

In the implementation it is assumed at certain places for simplicity that there is only one precluding node per node. This assumption may easily be eliminated by treating the precluding nodes as a set of more than one elements and traversing through each element of the set.

6.2 Future Work

Future work in this direction could broadly be classified into two categories: theory and implementation.

At the theoretical level, it remains to be seen if there are any complicated cases that would increase the complexity of the reasoning procedure described. For example, [GPLT91] highlighted the

issue of disjunctive defaults that raise a problem in default logic representation. Disjunctive defaults contain a disjunction that is observed to be true. The problem arises when each of the literals in the disjunction is assumed to be true by default and leads to an inconsistent extension. It is predicted that our theory would not have any problems with this because the only assumptions made in any deduction are supported by the existence of a default in the belief space. Thus, no such inconsistent assumptions would ever be made in the first place so as to lead to an inconsistent extension. The second issue is of automated creation of preclusion rules. This should be possible and could derive its theory from [Tou86]. Efficient algorithms for finding preclusion relations is a research issue.

At the implementation level, this theory needs to be implemented for the fully object-oriented version of SNePS, SNePS 3[Sha00]. A SNePSLOG interface for the present implementation as described in chapter 7 of [SG02] would be a desirable extension also.

Demo 3 in appendix A shows the result of a KB in which the preclusion rules contradict each other. In general, detection of cycles of such kind would require computation that is beyond the scope of this project and hence remains to be implemented. This task would require implementation of transitivity of preference ordering that is not implemented yet. Preclusion relations may or may not be transitive. If they are transitive, then a cycle should not be detected. For example, as in demo 3 in appendix A, if

r1: all(x)(Quaker(x)⇒Pacifist(x))

r2: all(x)(Republican(x)⇒ ¬Pacifist(x))

r2≺r1

r1≺Republican(Nixon)

Quaker(Nixon)

Republican(Nixon)

then, ¬Pacifist(Nixon) should be deduced since it is supported by the most preferred rule. However, this might also be detected as a cycle since the same proof for ¬Pacifist(Nixon) is precluded by a proof for Pacifist(Nixon). Thus, the corrected notion of preclusion is that proofs are precluded by proofs that contain the most preferred rule of a transitive closure of preference rules.

As mentioned earlier, the current implementation supports two levels of nested defaults. Support for multiple levels of preference graphs will be a good extension to this project. Also, reasoning in multiple contexts are expected to bring to fore some interesting research issues. The present implementation, though designed for multiple contexts, has not been tested comprehensively for very complicated cases. Finally, it will be interesting to see the performance of preclusion sets in another implementation of ATMS.

Appendix A

Examples

Following are a few examples that were run for demonstration of the working of the algorithm. The transcription shown here are the actual outputs from the SNePS engine except that they are formatted to fit in the page here and a few other comments are added for more explanation.

****

Demo 1: Proofs preclude proofs. The following demonstration shows that the most preferred proof takes precedence. A node may have multiple proofs, some of which may be precluded by the node's negation. Some other proofs may not be precluded. The decision whether a node or its negation is believed in a context is based on which one of them has a proof that is not precluded. In the following example,

$$
\begin{array}{lll}
r1 = & M1 & : \ \forall x(Bird(x) \Rightarrow Fly(x)) \\
r2 = & M2 & : \ \forall x(Penguin(x) \Rightarrow \neg Fly(x)) \\
 & M3 & : \ \forall x(Penguin(x) \Rightarrow Bird(x)) \\
r3 = & M4 & : \ \forall x(Flying\text{–}Penguin(x) \Rightarrow Fly(x)) \\
 & M5 & : \ \forall x(Flying\text{–}Penguin(x) \Rightarrow Penguin(x)) \\
 & M6 & : \ \forall x(Canary(x) \Rightarrow Bird(x)) \\
 & M7 & : \ Flying\text{–}Penguin(Opus) \\
 & M8 & : \ Penguin(Chilly) \\
 & M9 & : \ Canary(Tweety) \\
 & M10 & : \ r1 = M1 \prec r2 = M2 \\
 & M11 & : \ r2 = M2 \prec r3 = M4
\end{array}
$$

Tweety is an arbitrary bird and hence flies. Chilly is an exceptional bird (a penguin) and hence does not fly. Opus is an exceptional penguin that is supposed to fly because it is a bird but should not fly because it is a penguin. However, the most specific rule that governs it is that "Flying-penguins fly". This proof for flying precludes the proof for not flying even though the proof that contains "Birds fly" is precluded.

****

```
* (resetnet t)
Net reset
```

```
* (define member class precludes precluded object ability)
(MEMBER CLASS PRECLUDES PRECLUDED OBJECT ABILITY)

* ;; Birds fly
(= (assert forall $x
     ant (build member *x class bird)
     cq (build object *x ability fly)) r1)
(M1!)

* ;; Penguins do not fly
(= (assert forall *x
     ant (build member *x class penguin)
     cq (build min 0 max 0 arg (build object *x ability fly))) r2)
(M2!)

* ;; Penguins are birds
(assert forall *x
  ant (build member *x class penguin)
  cq (build member *x class bird))
(M3!)

* ;; Flying-penguins fly
(= (assert forall *x
 ant (build member *x class flying-penguin)
     cq (build object *x ability fly)) r3)
(M4!)

* ;; Flying-penguins are penguins
(assert forall *x
     ant (build member *x class flying-penguin)
     cq (build member *x class penguin))
(M5!)

* ;; Canaries are birds
(assert forall *x
  ant (build member *x class canary)
  cq (build member *x class bird))
(M6!)

* ;; Opus is a flying-penguin
(assert member opus class flying-penguin)
(M7!)

* ;; Chilly is a penguin
(assert member chilly class penguin)
(M8!)
```

```
* ;; Tweety is a canary
(assert member tweety class canary)
(M9!)

* ;; The rule "Penguins do not fly" is more preferred than "Birds fly"
(assert precludes  *r2 precluded *r1)
(M10!)

* ;; The rule "Flying-penguins fly" is more preferred than "Penguins
do not fly"
(assert precludes  *r3 precluded *r2)
(M11!)

* ;; Who flies and who does not?
(describe (deduce object $x ability fly))

I wonder if
((P7 (PRECLUDED V4) (PRECLUDES V5)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M10!
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
  (PRECLUDES
   (M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
    (CQ (P4 (MIN 0) (MAX 0)
    (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))))

I know
((M11!
  (PRECLUDED
   (M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
    (CQ (P4 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
  (PRECLUDES
   (M4! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))))

I wonder if
((P8 (ABILITY (FLY)) (OBJECT V6)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I wonder if
((P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
I wonder if
((P1 (CLASS (BIRD)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M7! (CLASS (FLYING-PENGUIN)) (MEMBER (OPUS))))

Since
((M4! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P5 (CLASS (FLYING-PENGUIN)) (MEMBER (V1 <-- OPUS))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))

I wonder if
((P3 (CLASS (PENGUIN)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M8! (CLASS (PENGUIN)) (MEMBER (CHILLY))))

Since
((M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P3 (CLASS (PENGUIN)) (MEMBER (V1 <-- CHILLY))))
I infer
((P4 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- CHILLY))))))

I know it is not the case that
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- CHILLY))))

I wonder if
((P6 (CLASS (CANARY)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT

Since
((M3! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P1 (CLASS (BIRD)) (MEMBER V1)))))
and
((P3 (CLASS (PENGUIN)) (MEMBER (V1 <-- CHILLY))))
I infer
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- CHILLY))))
```

```
Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- CHILLY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- CHILLY))))
```

****

Comment: As is visible here, “Chilly flies” is derived in the middle of the inference, but there is no way to stop this, because the pattern node created cannot have any supports and hence it cannot have preclusion sets.

****

```
Since
((M5! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
  (CQ (P3 (CLASS (PENGUIN)) (MEMBER V1)))))
and
((P5 (CLASS (FLYING-PENGUIN)) (MEMBER (V1 <-- OPUS))))
I infer
((P3 (CLASS (PENGUIN)) (MEMBER (V1 <-- OPUS))))

Since
((M3! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P1 (CLASS (BIRD)) (MEMBER V1)))))
and
((P3 (CLASS (PENGUIN)) (MEMBER (V1 <-- OPUS))))
I infer
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))

Since
((M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0)
      (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P3 (CLASS (PENGUIN)) (MEMBER (V1 <-- OPUS))))
I infer
((P4 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))))

I know it is not the case that
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))
```

```
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))

I know
((M9! (CLASS (CANARY)) (MEMBER (TWEETY))))

Since
((M6! (FORALL V1) (ANT (P6 (CLASS (CANARY)) (MEMBER V1)))
  (CQ (P1 (CLASS (BIRD)) (MEMBER V1)))))
and
((P6 (CLASS (CANARY)) (MEMBER (V1 <-- TWEETY))))
I infer
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
```

****

Conclusions so far:

M12: Fly(Opus)	*from {M4, M7} and from {M1, M3, M5,M7}*
M14: ¬ Fly(Chilly)	*from {M2, M8}*
M13: Fly(Chilly)	*from {M1, M3, M8}*
M18: ¬ Fly(Opus)	*from {M2, M5, M7}*
M20: Fly(Tweety)	*from {M1, M6, M9}*
but,	*r2=M2 ≺ r3=M4 so M12 precludes M18*
	r1=M1 ≺ r2=M2 so M14 precludes M13
so the answers are:	*M20: Fly(Tweety)*
	M14: ¬ Fly(Chilly)
	M12: Fly(Opus)

****

```
Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))
Since
((M11!
  (PRECLUDED
   (M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
    (CQ (P4 (MIN 0) (MAX 0)
    (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
  (PRECLUDES
   (M4! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((M4! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
Therefore
((M18 (MIN 0) (MAX 0) (ARG (M12! (ABILITY (FLY)) (OBJECT (OPUS))))))
containing in its support
((M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
is precluded by
((M12! (ABILITY (FLY)) (OBJECT (OPUS))))
that contains in its support
((M4! (FORALL V1) (ANT (P5 (CLASS (FLYING-PENGUIN)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))

Since
((M10!
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
  (PRECLUDES
   (M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
    (CQ (P4 (MIN 0) (MAX 0)
    (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))))
and
((M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
holds within the BS defined by context DEFAULT-DEFAULTCT
Therefore
((M13 (ABILITY (FLY)) (OBJECT (CHILLY))))
containing in its support
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
is precluded by
((M14! (MIN 0) (MAX 0)
(ARG (M13 (ABILITY (FLY)) (OBJECT (CHILLY))))))
that contains in its support
((M2! (FORALL V1) (ANT (P3 (CLASS (PENGUIN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
(M20! (ABILITY FLY) (OBJECT TWEETY))
(M14! (MIN 0) (MAX 0) (ARG (M13 (ABILITY FLY) (OBJECT CHILLY))))
(M12! (ABILITY FLY) (OBJECT OPUS))

(M20! M14! M12!)
```

****

Demo 2: Level two rules - nested defaults. This example shows the power of the solution of this thesis. The fact that birds fly is a default rule only if we are talking of contexts outside New Zealand(NZ). Inside New Zealand, however, birds do not fly, by default. The twist in the example is due to a second level default which asserts that by default, we talk of contexts outside New Zealand. Thus by default, a bird flies but in New Zealand, birds do not fly by default. In the following example,

$$
\begin{array}{lll}
r1 = & M1 & : \ \forall x(Bird(x) \Rightarrow Fly(x)) \\
r2 = & M2 & : \ \forall x(Bird(x) \Rightarrow \neg Fly(x)) \\
 & M3 & : \ \forall x(Canary(x) \Rightarrow Bird(x)) \\
 & M4 & : \ \forall x(Penguin(x) \Rightarrow Bird(x)) \\
 & M5 & : \ Canary(Tweety) \\
 & M6 & : \ Penguin(Opus) \\
 & M7 & : \ Bird(Betty) \\
r3 = & M8 & : \ \forall x(Canary(x) \Rightarrow Fly(x)) \\
r4 = & M9 & : \ \forall x(Penguin(x) \Rightarrow \neg Fly(x)) \\
 & M10 & : \ r1 = M1 \prec r4 = M9 \\
 & M11 & : \ r2 = M2 \prec r3 = M8 \\
 & M14 & : \ NZ(we) \Rightarrow (r1 = M1 \prec r2 = M2) \\
 & M17 & : \ \neg NZ(we) \Rightarrow (r2 = M2 \prec r1 = M1) \\
 & M18 & : \ \neg NZ(we) \prec NZ(we) \\
 & M15 & : \ \neg NZ(we)
\end{array}
$$

Note that M1 and M2 are both rules of the KB. M1 precludes M2 in the default context of the world outside New Zealand. M2 precludes M1 in the context of NZ. “Canaries fly” all over the world and hence preclude M2 in NZ. “Penguins do not fly” all over the world and hence precludes M1 outside NZ. Thus Tweety, the canary, flies. Opus, the penguin, does not fly. Betty, the bird, flies since we have not specified any context (default assumption is that we are outside NZ and is explicitly stated).
****

```
* (resetnet t)
Net reset

* (define member class precludes precluded object ability resident
  place)
(MEMBER CLASS PRECLUDES PRECLUDED OBJECT ABILITY RESIDENT PLACE)

* ;; Birds fly
(= (assert forall $x
     ant (build member *x class bird)
     cq (build object *x ability fly)) r1)
(M1!)

* ;; Birds do not fly
(= (assert forall *x
     ant (build member *x class bird)
     cq (build min 0 max 0 arg (build object *x ability fly))) r2)
(M2!)
```

```
* ;; Canaries are birds
(assert forall $y
  ant (build member *y class canary)
  cq (build member *y class bird))
(M3!)

* ;; Penguins are birds
(assert forall $y
  ant (build member *y class penguin)
  cq (build member *y class bird))
(M4!)

* ;; Tweety is a canary
(assert member tweety class canary)
(M5!)

* ;; Opus is a penguin
(assert member opus class penguin)
(M6!)

* ;; Betty is a bird
(assert member betty class bird)
(M7!)

* ;; Canaries fly
(= (assert forall $x
     ant (build member *x class canary)
     cq (build object *x ability fly)) r3)
(M8!)

* ;; Penguins do not fly
(= (assert forall *x
     ant (build member *x class penguin)
     cq (build min 0 max 0 arg (build object *x ability fly))) r4)
(M9!)

* ;; The rule "Penguins do not fly" is more preferred than "Birds fly"
(assert precluded *r1 precludes *r4)
(M10!)

* ;; The rule "Canaries fly" is more preferred than "Birds do not fly"
(assert precluded *r2 precludes *r3)
(M11!)
```

```
* ;; If we are in NZ, the rule "Birds do not fly" is more preferred
than "Birds fly"
(assert
   ant (build resident we place NZ)
   cq (build precludes *r2 precluded *r1))
(M14!)

* ;; If we are not in NZ, the rule "Birds fly" is more preferred than
"Birds do not fly"
(assert
   ant (build min 0 max 0 arg (build resident we place NZ))
   cq (build precludes *r1 precluded *r2))
(M17!)

* ;; The rule "We are in NZ" is more preferred than "We are not in NZ"
(assert precluded
    (build min 0 max 0 arg (build resident we place NZ))
  precludes (build resident we place NZ))
(M18!)

* ;; We are not in NZ (by default)
(assert min 0 max 0 arg (build resident we place NZ))
(M15!)

* ;; Who flies and who does not?
(describe (deduce object $x ability fly))

I wonder if
((P12 (PRECLUDED V7) (PRECLUDES V8)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M10!
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
  (PRECLUDES
   (M9! (FORALL V4) (ANT (P10 (CLASS (PENGUIN)) (MEMBER V4)))
    (CQ (P11 (MIN 0) (MAX 0)
     (ARG (P9 (ABILITY (FLY)) (OBJECT V4)))))))))
I know
((M11!
  (PRECLUDED
   (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P3 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
  (PRECLUDES
   (M8! (FORALL V4) (ANT (P8 (CLASS (CANARY)) (MEMBER V4)))
    (CQ (P9 (ABILITY (FLY)) (OBJECT V4)))))))
```

```
I know
((M18!
  (PRECLUDED (M15! (MIN 0) (MAX 0)
   (ARG (M12 (PLACE (NZ)) (RESIDENT (WE))))))
  (PRECLUDES (M12 (PLACE (NZ)) (RESIDENT (WE))))))

I wonder if
((M12 (PLACE (NZ)) (RESIDENT (WE))))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know it is not the case that
((M12 (PLACE (NZ)) (RESIDENT (WE))))

I know
((M15! (MIN 0) (MAX 0) (ARG (M12 (PLACE (NZ)) (RESIDENT (WE))))))

Since
((M17! (ANT (M15! (MIN 0) (MAX 0)
 (ARG (M12 (PLACE (NZ)) (RESIDENT (WE))))))
  (CQ
   (M16
    (PRECLUDED
     (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
      (CQ (P3 (MIN 0) (MAX 0)
       (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
    (PRECLUDES
     (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
      (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))))))
and
((M15! (MIN 0) (MAX 0) (ARG (M12 (PLACE (NZ)) (RESIDENT (WE))))))
I infer
((M16
  (PRECLUDED
   (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P3 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
  (PRECLUDES
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))))

I wonder if
((P13 (ABILITY (FLY)) (OBJECT V9)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I wonder if
((P1 (CLASS (BIRD)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
I wonder if
((P8 (CLASS (CANARY)) (MEMBER V4)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M7! (CLASS (BIRD)) (MEMBER (BETTY))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- BETTY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- BETTY))))

Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- BETTY))))
I infer
((P3 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- BETTY))))))

I know it is not the case that
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- BETTY))))

I wonder if
((P10 (CLASS (PENGUIN)) (MEMBER V4)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M5! (CLASS (CANARY)) (MEMBER (TWEETY))))

Since
((M8! (FORALL V4) (ANT (P8 (CLASS (CANARY)) (MEMBER V4)))
  (CQ (P9 (ABILITY (FLY)) (OBJECT V4)))))
and
((P8 (CLASS (CANARY)) (MEMBER (V4 <-- TWEETY))))
I infer
((P9 (ABILITY (FLY)) (OBJECT (V4 <-- TWEETY))))

I wonder if
((P6 (CLASS (PENGUIN)) (MEMBER V3)))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
I know
((M6! (CLASS (PENGUIN)) (MEMBER (OPUS))))

Since
((M4! (FORALL V3) (ANT (P6 (CLASS (PENGUIN)) (MEMBER V3)))
  (CQ (P7 (CLASS (BIRD)) (MEMBER V3)))))
and
((P6 (CLASS (PENGUIN)) (MEMBER (V3 <-- OPUS))))
I infer
((P7 (CLASS (BIRD)) (MEMBER (V3 <-- OPUS))))

Since
((M9! (FORALL V4) (ANT (P10 (CLASS (PENGUIN)) (MEMBER V4)))
  (CQ (P11 (MIN 0) (MAX 0)
(ARG (P9 (ABILITY (FLY)) (OBJECT V4)))))))
and
((P10 (CLASS (PENGUIN)) (MEMBER (V4 <-- OPUS))))
I infer
((P11 (MIN 0) (MAX 0)
      (ARG (P9 (ABILITY (FLY)) (OBJECT (V4 <-- OPUS))))))

I know it is not the case that
((P9 (ABILITY (FLY)) (OBJECT (V4 <-- OPUS))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))

Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0) (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))
I infer
((P3 (MIN 0) (MAX 0)
      (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))))

I know it is not the case that
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))

I wonder if
((P4 (CLASS (CANARY)) (MEMBER V2)))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
I know
((M5! (CLASS (CANARY)) (MEMBER (TWEETY))))

Since
((M3! (FORALL V2) (ANT (P4 (CLASS (CANARY)) (MEMBER V2)))
  (CQ (P5 (CLASS (BIRD)) (MEMBER V2)))))
and
((P4 (CLASS (CANARY)) (MEMBER (V2 <-- TWEETY))))
I infer
((P5 (CLASS (BIRD)) (MEMBER (V2 <-- TWEETY))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))

Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0)
(ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
I infer
((P3 (MIN 0) (MAX 0)
      (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))))

I know it is not the case that
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))

(M24! (MIN 0) (MAX 0) (ARG (M23 (ABILITY FLY) (OBJECT OPUS))))
(M21! (ABILITY FLY) (OBJECT TWEETY))
(M19! (ABILITY FLY) (OBJECT BETTY))

(M24! M21! M19!)
```

****

Demo 2 (continued): Level two rules - nested defaults. Still moving further in this example, dynamic behavior is tested. If new information is now input in the current belief space, i.e., specific information that we are in New Zealand, the default results change but the specific results remain the same. In the following demonstration Tweety, the canary, still flies. Opus, the penguin, still does not fly. However, Betty, the bird, now does not fly because we are now in the context of NZ.

****

```
* ;; We are in NZ
(assert resident we place NZ)
(M12!)

* ;; Clear previous inferences
(clear-infer)

(Node activation cleared. Some register information retained.)

* ;; Now, who flies and who does not?
(describe (deduce object $x ability fly))

I wonder if
((P14 (PRECLUDED V12) (PRECLUDES V13)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M10!
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
  (PRECLUDES
   (M9! (FORALL V4) (ANT (P10 (CLASS (PENGUIN)) (MEMBER V4)))
    (CQ (P11 (MIN 0) (MAX 0)
  (ARG (P9 (ABILITY (FLY)) (OBJECT V4)))))))))

I know
((M11!
  (PRECLUDED
   (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P3 (MIN 0) (MAX 0)
  (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
  (PRECLUDES
   (M8! (FORALL V4) (ANT (P8 (CLASS (CANARY)) (MEMBER V4)))
    (CQ (P9 (ABILITY (FLY)) (OBJECT V4)))))))
```

****

Note how this works. The node M15 (for the context outside NZ) is precluded by M12 (for the context in NZ) and this leads to the preclusion of M1 by M2. Both the rules are still in the CBS since they are general rules and hence are not contradictory. The specific facts can be contradictory and not the rules.

****

```
I know
((M18!
  (PRECLUDED (M15 (MIN 0) (MAX 0)
(ARG (M12! (PLACE (NZ)) (RESIDENT (WE))))))
  (PRECLUDES (M12! (PLACE (NZ)) (RESIDENT (WE))))))
```

```
I know
((M12! (PLACE (NZ)) (RESIDENT (WE))))

Since
((M14! (ANT (M12! (PLACE (NZ)) (RESIDENT (WE))))
  (CQ
   (M13
    (PRECLUDED
     (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
      (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
    (PRECLUDES
     (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
      (CQ (P3 (MIN 0) (MAX 0)
    (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))))))
and
((M12! (PLACE (NZ)) (RESIDENT (WE))))
I infer
((M13
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
  (PRECLUDES
   (M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
    (CQ (P3 (MIN 0) (MAX 0)
    (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))))

I wonder if
((M15 (MIN 0) (MAX 0) (ARG (M12! (PLACE (NZ)) (RESIDENT (WE))))))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M12! (PLACE (NZ)) (RESIDENT (WE))))

Since it is the case that
((M12! (PLACE (NZ)) (RESIDENT (WE))))
I infer it is not the case that
((M15 (MIN 0) (MAX 0) (ARG (M12! (PLACE (NZ)) (RESIDENT (WE))))))

I wonder if
((P15 (ABILITY (FLY)) (OBJECT V14)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M21! (ABILITY (FLY)) (OBJECT (TWEETY))))

I know it is not the case that
((M19 (ABILITY (FLY)) (OBJECT (BETTY))))
```

```
I know it is not the case that
((M23 (ABILITY (FLY)) (OBJECT (OPUS))))

I wonder if
((P1 (CLASS (BIRD)) (MEMBER V1)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I wonder if
((P8 (CLASS (CANARY)) (MEMBER V4)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M7! (CLASS (BIRD)) (MEMBER (BETTY))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- BETTY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- BETTY))))

Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0)
  (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- BETTY))))
I infer
((P3 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- BETTY))))))

I know
((M22! (CLASS (BIRD)) (MEMBER (OPUS))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))
```

```
Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0)
  (ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- OPUS))))
I infer
((P3 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- OPUS))))))

I know
((M25! (CLASS (BIRD)) (MEMBER (TWEETY))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P2 (ABILITY (FLY)) (OBJECT V1)))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
I infer
((P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))

Since
((M2! (FORALL V1) (ANT (P1 (CLASS (BIRD)) (MEMBER V1)))
  (CQ (P3 (MIN 0) (MAX 0)
(ARG (P2 (ABILITY (FLY)) (OBJECT V1)))))))
and
((P1 (CLASS (BIRD)) (MEMBER (V1 <-- TWEETY))))
I infer
((P3 (MIN 0) (MAX 0)
     (ARG (P2 (ABILITY (FLY)) (OBJECT (V1 <-- TWEETY))))))

I wonder if
((P10 (CLASS (PENGUIN)) (MEMBER V4)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M5! (CLASS (CANARY)) (MEMBER (TWEETY))))

Since
((M8! (FORALL V4) (ANT (P8 (CLASS (CANARY)) (MEMBER V4)))
  (CQ (P9 (ABILITY (FLY)) (OBJECT V4)))))
and
((P8 (CLASS (CANARY)) (MEMBER (V4 <-- TWEETY))))
I infer
((P9 (ABILITY (FLY)) (OBJECT (V4 <-- TWEETY))))
```

```
I wonder if
((P6 (CLASS (PENGUIN)) (MEMBER V3)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M6! (CLASS (PENGUIN)) (MEMBER (OPUS))))

Since
((M4! (FORALL V3) (ANT (P6 (CLASS (PENGUIN)) (MEMBER V3)))
  (CQ (P7 (CLASS (BIRD)) (MEMBER V3)))))
and
((P6 (CLASS (PENGUIN)) (MEMBER (V3 <-- OPUS))))
I infer
((P7 (CLASS (BIRD)) (MEMBER (V3 <-- OPUS))))

Since
((M9! (FORALL V4) (ANT (P10 (CLASS (PENGUIN)) (MEMBER V4)))
  (CQ (P11 (MIN 0) (MAX 0)
   (ARG (P9 (ABILITY (FLY)) (OBJECT V4)))))))
and
((P10 (CLASS (PENGUIN)) (MEMBER (V4 <-- OPUS))))
I infer
((P11 (MIN 0) (MAX 0)
      (ARG (P9 (ABILITY (FLY)) (OBJECT (V4 <-- OPUS))))))

I wonder if
((P4 (CLASS (CANARY)) (MEMBER V2)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M5! (CLASS (CANARY)) (MEMBER (TWEETY))))

Since
((M3! (FORALL V2) (ANT (P4 (CLASS (CANARY)) (MEMBER V2)))
  (CQ (P5 (CLASS (BIRD)) (MEMBER V2)))))
and
((P4 (CLASS (CANARY)) (MEMBER (V2 <-- TWEETY))))
I infer
((P5 (CLASS (BIRD)) (MEMBER (V2 <-- TWEETY))))

(M24! (MIN 0) (MAX 0) (ARG (M23 (ABILITY FLY) (OBJECT OPUS))))
(M21! (ABILITY FLY) (OBJECT TWEETY))
(M20! (MIN 0) (MAX 0) (ARG (M19 (ABILITY FLY) (OBJECT BETTY))))

(M24! M21! M20!)
```

****

Demo 2 (continued): Level two rules - nested defaults. Dynamic behavior is further tested. This time the current belief space is subject to deletion of information. The specific information that we are in New Zealand is removed and results are probed. Finally, the context is switched again and the deductions are checked again. In the following, node M12 (for the context in NZ) is removed. This leads to instant assertion of M15 (which is the default context) that was asserted earlier but was precluded. This leads to preclusion of M2 and immediate assertion of M19 (Betty flies). That is because M19 was asserted before and has a support. This support was precluded earlier because M1 was precluded by M2 (M2 precludes M1 is supported by M12). As soon as M12 is unasserted, the preclusion reverses because M16 (M1 precludes M2) is supported by M15 which is now not precluded.

****

```
* ;; Remove the fact that "we are in NZ" from the KB
(remove-from-context m12)

The context (M18! M17! M15 M14! M11! M10! M9! M8! M7! M6! M5!
M4! M3! M2! M1!)
that you are trying to build is inconsistent.

You have the following options:
   1. [y]es, create the context anyway, knowing that a
      contradiction is derivable;
   2. [n]o,don't create the context.
  (please type y or n)
=><= y

((ASSERTIONS
  (M18! M17! M15! M14! M11! M10! M9! M8! M7! M6! M5! M4! M3!
   M2! M1!))
 (RESTRICTION (NIL)) (NAMED (DEFAULT-DEFAULTCT)))

* ;; Does Betty fly or not, now ?
(full-describe m20)

(M20 (MIN 0) (MAX 0)
 (ARG
  (M19! (ABILITY FLY) (OBJECT BETTY)
   (DER
    ((ASSERTIONS (M7! M1!)) (RESTRICTION ((M9! M6! M4!) (M2!)))
     (NAMED NIL)))))
 (DER
  ((ASSERTIONS (M7! M2!)) (RESTRICTION ((M8! M5! M3!) (M1!)))
   (NAMED NIL))))

(M20)
```

****

Demo 3: Nixon Diamond. The famous Nixon Diamond problem may be solved by applying precedence of rules at various levels. However, if the preference rules are such that it is not possible to make conclusive deduction, the output is skeptical and an appropriate message is displayed. In the following demonstration,

$$
\begin{array}{lll}
r1 = & M1: & \forall x(Quaker(x) \Rightarrow Pacifist(x)) \\
r2 = & M2: & \forall x(Republican(x) \Rightarrow \neg Pacifist(x)) \\
 & M3: & r1 = M1 \prec r2 = M2 \\
 & M4: & Quaker(Nixon) \\
 & M5: & Republican(Nixon) \\
 & M6: & Republican(nixon) \prec Quaker(Nixon) \\
 & M7: & Pacifist(Nixon) \\
 & M8: & \neg Pacifist(Nixon)
\end{array}
$$

M7 (“Nixon is a pacifist) has a proof that precludes M8 (“Nixon is a non pacifist) because M4 (“Nixon is a Quaker”) precludes M5 (“Nixon is a Republican”).
M8 (“Nixon is a non-pacifist) has a proof that precludes M7 (“Nixon is a pacifist) because M2 (“Republicans are non-pacifists”) precludes M1 (“Quakers are pacifists”).
There is no reason to conclude anything and hence there is a possibly that the preclusion rules are wrong. Detection of cycles in a preference graph that contains preclusion rules is, in general, a non trivial task and not a part of this thesis. hence, we adopt skepticism and display a message that the preclusion rules need to be looked into for deductions to be made.
****

```
* (resetnet t)
Net reset

* (define member class precludes precluded object quality)
(MEMBER CLASS PRECLUDES PRECLUDED OBJECT QUALITY)

* ;; Quakers are pacifists
(= (assert forall $x
     ant (build member *x class Quaker)
     cq (build object *x quality pacifist)) r1)
(M1!)

* ;; Republicans are non-pacifists
(= (assert forall *x
     ant (build member *x class Republican)
     cq (build min 0 max 0
arg (build object *x quality pacifist))) r2)
(M2!)

* ;; The rule "Republicans are non-pacifists" is more preferred
than "Quakers are pacifists"
(assert precludes  *r2 precluded *r1)
(M3!)
```

```
* ;; The fact "Nixon was a Quaker" is more preferred than the fact
"Nixon was a Republican"
(assert precludes  (build member Nixon class Quaker)
precluded (build member Nixon class Republican))
(M6!)

* ;; Nixon was a Quaker"
(assert member Nixon class Quaker)
(M4!)

* ;; Nixon was a Republican
(assert member Nixon class Republican)
(M5!)

* ;; Was Nixon a pacifist?
(describe (deduce object Nixon quality pacifist))

I wonder if
((P5 (PRECLUDED V4) (PRECLUDES V5)))
holds within the BS defined by context DEFAULT-DEFAULTCT

I know
((M3!
  (PRECLUDED
   (M1! (FORALL V1) (ANT (P1 (CLASS (QUAKER)) (MEMBER V1)))
    (CQ (P2 (OBJECT V1) (QUALITY (PACIFIST))))))
  (PRECLUDES
   (M2! (FORALL V1) (ANT (P3 (CLASS (REPUBLICAN)) (MEMBER V1)))
    (CQ (P4 (MIN 0) (MAX 0)
    (ARG (P2 (OBJECT V1) (QUALITY (PACIFIST)))))))))

I know
((M6! (PRECLUDED (M5! (CLASS (REPUBLICAN)) (MEMBER (NIXON))))
  (PRECLUDES (M4! (CLASS (QUAKER)) (MEMBER (NIXON))))))

I wonder if
((M7 (OBJECT (NIXON)) (QUALITY (PACIFIST))))
holds within the BS defined by context DEFAULT-DEFAULTCT

I wonder if
((P1 (CLASS (QUAKER)) (MEMBER (V1 <-- NIXON))))
holds within the BS defined by context DEFAULT-DEFAULTCT

I wonder if
((P3 (CLASS (REPUBLICAN)) (MEMBER (V1 <-- NIXON))))
holds within the BS defined by context DEFAULT-DEFAULTCT
```

```
I know
((M4! (CLASS (QUAKER)) (MEMBER (NIXON))))

Since
((M1! (FORALL V1) (ANT (P1 (CLASS (QUAKER)) (MEMBER V1)))
  (CQ (P2 (OBJECT V1) (QUALITY (PACIFIST))))))
and
((P1 (CLASS (QUAKER)) (MEMBER (V1 <-- NIXON))))
I infer
((P2 (OBJECT (V1 <-- NIXON)) (QUALITY (PACIFIST))))

I know
((M5! (CLASS (REPUBLICAN)) (MEMBER (NIXON))))

Since
((M2! (FORALL V1) (ANT (P3 (CLASS (REPUBLICAN)) (MEMBER V1)))
  (CQ (P4 (MIN 0) (MAX 0)
  (ARG (P2 (OBJECT V1) (QUALITY (PACIFIST)))))))
and
((P3 (CLASS (REPUBLICAN)) (MEMBER (V1 <-- NIXON))))
I infer
((P4 (MIN 0) (MAX 0)
     (ARG (P2 (OBJECT (V1 <-- NIXON)) (QUALITY (PACIFIST))))))

I know it is not the case that
((P2 (OBJECT (V1 <-- NIXON)) (QUALITY (PACIFIST))))

Since
((M6! (PRECLUDED (M5! (CLASS (REPUBLICAN)) (MEMBER (NIXON))))
  (PRECLUDES (M4! (CLASS (QUAKER)) (MEMBER (NIXON))))))
and
((M4! (CLASS (QUAKER)) (MEMBER (NIXON))))
holds within the BS defined by context DEFAULT-DEFAULTCT
Therefore
((M8 (MIN 0) (MAX 0)
     (ARG (M7 (OBJECT (NIXON)) (QUALITY (PACIFIST))))))
containing in its support
((M5! (CLASS (REPUBLICAN)) (MEMBER (NIXON))))
is precluded by
((M7 (OBJECT (NIXON)) (QUALITY (PACIFIST))))
that contains in its support
((M4! (CLASS (QUAKER)) (MEMBER (NIXON))))

Some Preclusion Rules contradict each other. M8 and its negation
are both precluded. Please check your Preference Ordering graph
and convert it to a DAG
```

Bibliography

[CM92] Mario R. Cravo and Joao P. Martins, *Defaults and belief revision*, Tech. report, Instituto Superior Tecnico, Av Rovisco Pais, 1000 Lisboa, Portugal, 1992.

[CM93] Maria R. Cravo and João P. Martins, *SNePSwD: a newcomer to the SNePS family*, Journal of Experimental and Theoretical Artificial Intelligence (JETAI) **5** (1993), no. 2&3, 135–148.

[DS99] James P. Delgrande and Torsten Schaub, *The role of default logic in knowledge representation*, Logic Based Artificial Intelligence (Jack Minker, ed.), Kluwer Academic Publishers, 1999, pp. 107–126.

[Eli98] R. Elio, *How to disbelieve p→q: Resolving contradictions*, Proceedings of the Twentieth Annual Conference of the Cognitive Science Society, Lawrence Erlbaum, 1998, pp. 315–320.

[Fah79] Scott Fahlman, *NETL: A system for representing and using real-world knowledge*, MIT Press, Cambridge, MA, 1979.

[GP96] M. Goldszmidt and J. Pearl, *Qualitative probabilities for default reasoning, belief revision and causal modeling*, Artificial Intelligence **84** (1996), no. 1-2, 57–112.

[GPLT91] Michael L. Gelfond, Halina Przymusinska, Vladimir Lifschitz, and Miroslaw Truszczyński, *Disjunctive defaults*, KR'91: Principles of Knowledge Representation and Reasoning (James F. Allen, Richard Fikes, and Erik Sandewall, eds.), Morgan Kaufmann, San Mateo, California, 1991, pp. 230–237.

[Gro97] Benjamin N. Grosof, *Courteous logic programs: Prioritized conflict handling for rules*, Tech. report, IBM, Dec 30 1997, Research Report.

[Hal01] Joseph Y. Halpern, *Plausibility measures: A general approach for representing uncertainty*, Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01) (Seattle, WA), vol. 2, Morgan Kaufmann, 2001, pp. 1474–1483.

[Hof79] Douglas Hofstadter, *Godel, Escher, Bach: An eternal golden braid*, Vintage Books, 1979.

[Leh92] Fritz Lehmann (ed.), *Semantic networks in artificial intelligence*, Pergamon Press, Oxford, 1992.

[McC80] John McCarthy, *Circumscription - a form of non-monotonic reasoning*, Artificial Intelligence **13** (1980), 27–39.

[McC86] John McCarthy, *Applications of circumscription to formalizing commonsense knowledge*, Artificial Intelligence **28** (1986), 89–116.

[MD80] Drew McDermott and Jon Doyle, *Non-monotonic logic 1*, Artificial Intelligence **13** (1980), 41–72.

[Moo84] Robert C. Moore, *Possible-world semantics for autoepistemic logic*, Proceedings of AAAI Workshop on Nonmonotonic Reasoning (New Paltz), 1984, pp. 396–401.

[MS88] João P. Martins and Stuart C. Shapiro, *A model for belief revision*, Artificial Intelligence **35** (1988), 25–79.

[Rei80] Raymond Reiter, *A logic for default reasoning*, Artificial Intelligence **13** (1980), 81–132.

[SG02] S. C. Shapiro and The SNePS Implementation Group, *SNePS 2.6 user's manual*, Buffalo, NY, 2002.

[Sha79] Stuart C. Shapiro, *The SNePS semantic network processing system*, Associative Networks: The Representation and Use of Knowledge by Computers (Nicholas V. Findler, ed.), Academic Press, New York, 1979, pp. 179–203.

[Sha00] Stuart C. Shapiro, *An Introduction to SNePS 3*, Conceptual Structures: Logical, Linguistic, and Computational Issues. Lecture Notes in Artificial Intelligence 1867 (Bernhard Ganter and Guy W. Mineau, eds.), Springer-Verlag, Berlin, 2000, pp. 510–524.

[SR92] Stuart C. Shapiro and William J. Rapaport, *The SNePS family*, Computers & Mathematics with Applications **23** (1992), no. 2–5, 243–275, Reprinted in [Leh92, pp. 243–275].

[Tho92] Richmond H. Thomason, *Netl and subsequent path-based inheritance theories*, Computers Math. Applications **23** (1992), 179–204.

[Tou86] D. Touretzky, *The mathematics of inheritence systems*, Morgan Kaufmann, San Mateo, CA, 1986.

[TT90] Richmond H. Thomason and D. Touretzky, *Inheritence theory and networks with roles*, Principles of Semantic Networks (1990), Morgan Kaufmann.