1985

Basic characteristics to achieve common sense reasoning

A. Raul Maselli

Basic Characteristics to Achieve Common Sense Reasoning

I A. Raul Maselli I. hereby grant permission to the Wallace Memorial Library, of R.I.T., to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Rochester, March 9th, 1985.

Approvals

Mrs. Margaret Reek Margaret M. Reek

Mr. Al Biles Al Biles

Dr. Peter Anderson Peter G. Anderson

Acknowledgement

I want to express my deepest thanks to the members of my comittee, Mrs. Margaret Reek, Mr. Al Biles and Mr. Peter Anderson. Without their worthy help, this thesis would have been impossible.

Also my thanks go to my dear parents and sister, Raul, Carolina, and Marissa for their moral support and their love; Thanks a lot to all the people who gave me his/her friendship, and also to all the persons who kept in touch through mail or phone.

Finally, I also want to offer my sincere thanks to R.I.T. and to the Government of the United States of America for this unforgetable and useful experience.

Table of Contents

1.0 Introduction

The prime objective of this thesis is to review some important aspects of first-order logic and different methods of reasoning in order to show a possible way of achieving a part of common sense knowledge in a computer.

Common sense reasoning differs from first-order logic in the need to draw conclusions from partial and changing knowledge. In logic, a conclusion is not accepted unless it can be proved according to the rules of inference, but in real life, conclusions without a proof are accepted because they seem plausible. Thus, something more powerful than logic is needed.

Two of the most outstanding researchers in Artificial Intelligence, J. McCarthy and M. Minsky, disagree on how to achieve common sense in a computer. McCarthy [1980] believes that the solution lies in designing computer programs to reason according to mathematical logic, whether or not that is the way people think. The major alternative is a psychological approach called the Frame System developed by Minsky [1975].

First-order logic began to show its usefulness as a knowledge representation scheme as a result of research into mechanical theorem proving. Concern has been expressed, however, about the lack in logic of an explicit

scheme for indexing relevant knowledge, its incompetence in dealing with incomplete and changing knowledge, and the limitations of deductive monotonic inference.

Frames represent a way to organize knowledge by breaking it into highly modular pieces. Frames are very useful in representing knowledge of stereotypical concepts or situations, analyses of visual perceptions, means of reasoning, etc.

The organization of this thesis centers on adaptations of logic and frames to common sense reasoning. Chapter two shows the basic characteristics of common sense reasoning. Chapter three traces the historical development of logic and reviews its basic principles. Chapter four describes several ways to achieve common sense reasoning through the use of non-monotonic logic and circumscription, and through the Frame System. Chapter five explains how an application of non-monotonic logic works, giving its functional description and explaining a simple example.

2.0 Logical Reasoning

Logical Reasoning is the method whereby propositions of a system are derived from postulates. The principles of logical reasoning are based upon evident 'canons' of logical procedure. A proposition is a statement which can be either true or false, and that can be asserted or denied. A postulate is a proposition or statement whose truth is taken for granted, or is self-evident. A postulate meets the following criteria: coherence, belonging to the system; contributiveness, having consequences in the system; consistency, not contradicting any other implied postulate or proposition; and, independency, not being itself implied by any other postulate or group of postulates.

Logical reasoning is the method of following rules of inference in the manipulation of the premises, which are the statements that constitute the starting point of an inference -statements which express a belief. Reasoning is logical if the inference rules are sound and if they are applied correctly. The fact that a line of reasoning is logical does not imply its correctness or truth; a line of reasoning will be true and correct only if the reasoning is logical and the premises are true and correct. In other words, if 'true' terms are manipulated according to the rules, the result will be correct and true. Logic, then, is concerned with consistency, not with value judgements.

2.1 Principles of Logical Procedure

The following principles define an algebra of Logic as a series of theorems. A theorem is defined to be a statement containing nothing that cannot be proved. It must be entirely implied by propositions other than itself, and it may contain no assumptions not made in the postulates.

2.1.1 Principle of Identity

This principle asserts that if any statement is true, then it is true. It dictates that every statement of the form $p \supset p$ is true, that is, that every such statement is a tautology.

2.1.2 Principle of Contradiction

This principle asserts that no statement can be both true and false. It dictates that every statement of the form $p \cdot \sim p$ is false; that is, that every such statement is self-contradictory.

2.1.3 Principle of Excluded Middle

This principle asserts that any statement is either true or false. It dictates that every statement of the form p v ~p is true; that is, that every such statement is a tautology.

2.1.4 Principle of Substitution

This principle asserts that identical terms may be substituted for one another; therefore, if x is equal to y, then x plus y may be substituted for y plus z.

Similarly, equivalent propositions may be substituted for one another. This is expressed as follows: if a ⊃ b == (a+b = b), then (a+b = b) may be substituted for (a ⊃ b), and vice versa.

2.1.5 Principle of Application

This principle asserts that a statement about a part, applies to the whole. For example, if it is granted that a+b = b+a, and if it is known that a certain element x and a certain element y exist, then it is true of this x and this y, that x+y = y+x.

2.1.6 Principle of Inference

A proposition that may be asserted assures that any proposition thereby implied may also be asserted. Thus, if (a= ~b) and (a= ~b) ⊃ (b= ~a), then it may be asserted thereafter that (b= ~a) is an independent proposition.

This principle makes possible the process of passing from given premises to inferred conclusions, that is, the process of deductive reasoning.

2.2 Rules of Manipulation

The following rules are characteristic of the algebra of logic, and they provide means for substituting equivalent propositions. The ability of interchangeable forms is the central requirement for a deductive system. Some of these rules are

2.2.1 Commutative Law

The Commutative law dictates the equivalency between sequential changes in disjunctions or conjunctions as it may be convenient. Thus

$$(p \vee q) == (q \vee p), \quad (p \wedge q) == (q \wedge p).$$

2.2.2 Distributive Law

The distributive law consists of two parts; the first is related to the disjunction of an element to a conjunction, and the second , concerns the conjunction of an element with a disjunction. P is disjuncted to a conjunction q ^ r by distributing its value to each member of the conjunction. Thus, p v (q ^ z) == (p ^ q) v (p ^ z). Similarly, p is conjuncted to q ^ r, distributing its value to each member of the disjunction. Thus, p ^ (q v r) == (p v q) ^ (p v r).

2.2.3 De Morgan's Laws

There are two logical equivalences that express the interrelationship among conjunction, disjunction, and negation. Since the disjunction (p v q) asserts that at least one of its two disjuncts is true, it is not contradicted by asserting that at least one is false, but only by asserting that both are false. Thus asserting the negation of the disjunction (p v q) is equivalent to asserting the conjunction of the negation of p and q, that is, ~(p v q) == (~p ^ ~q). Similarly, since asserting the conjunction of p and q assserts that both are true, to contradict this, it is needed to assert that at least one is false. Thus asserting the negation of the conjunction (p ^ q) is

equivalent to asserting the disjunction of the negation of p and of q, that is, ~(p ^ q) == (~p v ~q). These two tautologies are known as De Morgan's theorems.

De Morgan's theorems can be given a combined formulation as: The negation of the disjunction/ conjunction of two statements is equivalent to the conjunction/ disjunction of the negations of the two statements.

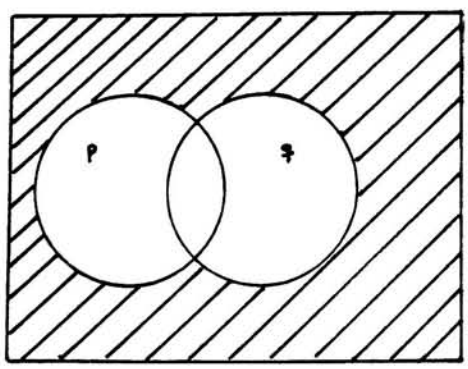
As a result of this, any disjunction may also be formed by conjunction and negation; and any conjunction may also be formed by disjunction and negation.

The exact relation between disjunction and conjunction is shown by these two laws

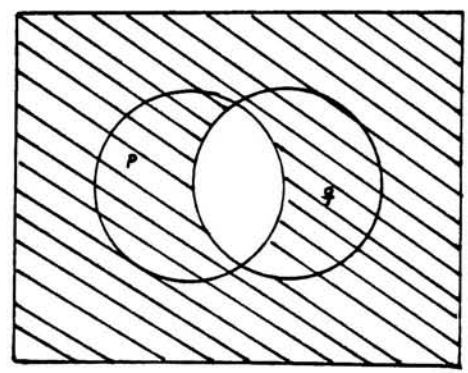
$$\sim(p \vee q) == \sim p \wedge \sim q$$

$$\sim(p \wedge q) == \sim p \vee \sim q$$

The previous laws can be represented graphically by the following Venn Diagrams



~(p v q) == ~p ^ ~q



~(p ^q) == ~p v ~q

Both Theorems are of great significance because they permit any conjunction to be written in the form of a disjunction and vice versa. It follows then, that all laws applying to disjunctions also apply to conjunctions, and vice versa.

Since disjunctions are expressed as conjunctions and vice versa, substitutions may be done whenever convenient, a very important tool for designing circuits and simplifying arguments.

2.3 Postulates as Formal Definition of Relations

If a universe K of unspecified elements x,y,.... is considered, and an unspecified relation r exists among the elements of K, a number of abstract propositions called postulates determine all that is known about either K or r. In other words, the postulates determine the kind of relation r can be, and how r operates on the elements of K.

The possible meanings of r are restricted to relations that "behave" like r, where the behavior of r is specified by the postulates. Thus, the postulates constitute a formal definition of r, which means that the properties of a certain type of relation are known, that the symbol r stands for this type of relation rather for a

specific relation, and that it may be interpreted to mean any concrete relation of the given type.

Working with logical concepts, such as relations, and propositions, involves dealing with kinds of constituent relations, and with kinds of systems rather than with the systems themselves. In other words, logical concepts deal with abstract patterns rather than with concrete things. Abstract patterns have the advantage of being applicable to many concrete things.

The postulates that define a relation, must describe the properties of that relation. These postulates or primitive propositions can be expressed symbolically, and, taken together, they constitute a formal definition of a relation in some universe. The following set of properties define an important class of relations

1. Irreflexiveness:

The relation r is said to be <u>irreflexive</u> if no element of x is in the same relation r to any other element of x. As example, let r stand for the relation "taller than". If r(a,b) means "a is taller than b", then r(a,a) is not possible, i.e., a cannot be taller than itself.

2. Transitivity:

The relation r is said to be transitive if x is in relation r to z, as a result of being x in relation r to y, and y in the same relation r to z. Following the previous example, if a is taller than b and b is taller than c, then a is taller than c.

3. Asymmetry:

The relation r is said to be assymetric if x is in relation r to y, but y is not to x. Thus, continuing with the same example, if a is taller than b, b cannot be taller than a.

4. Connexity:

The relation r is said to be connexive if x is in relation r to y, or y in relation r to x, but not both relations can happen simultaneously. Concluding with the example, a may be taller than b, "or" b taller than a, but a may not be taller than b "and" b taller than a, where a and b are different.

The previous rules designate relations of great importance in the formulation of an algebra.

3.0 The Algebra of Logic

Logic is the discipline concerned with making generalizations and abstractions by a system of orderly thinking. To do this, logic deals with two processes of reasoning, induction and deduction.

Induction is the process of reasoning from general observations to specific truths. It is also known as 'probable inference'. In inductive reasoning, the evidence is not supposed to support the conclusion with logical necessity, but only to support it with a definite positive degree of probability. That is, inductive arguments are intended to support their conclusions with probability only, because it is possible that the premises of an inductive argument could be true while its conclusion is false.

Deduction is the process of reasoning from specific truths to general principles. Deductive reasoning is also known as 'formal inference'. In deductive reasoning, the conclusion is supposed to be necessarily implied by the premises; it is more reliable because a valid deduction holds by necessity. A deductive system is a system wherein a small number of propositions or postulates determines all other propositions.

To illustrate a deductive system, Boolean Algebra will be examined in the next section of this chapter, not only because it constitutes a good example in computer science and logic design, but because it was also the first attempt to formalize the principles of logic as an algebra.

The second section of this chapter deals with propositional calculus, which provides a means for operating on propositions or statements by logical combinations. Thus, new statements can be deduced from an initial set of propositions.

The last section examines predicate calculus, which handles propositions in more detail and allows some analysis of the inner structures of propositions.

The purpose of this chapter, then, is to show how reasoning is achieved in classical or monotonic logic. This will serve as a "base line" for the comparison with non-monotonic logic to follow in chapters four and five.

3.1 Historical Review of Logic

The need for finding a general method of solving logical problems was satisfied with the development of an algebra of Logic. In 1847 George Boole invented the first workable algebra that obeyed the principles of logic. The theory of Boolean algebra was called "Algebra of Logic" in Boole's treatise "An Investigation of the Laws of Thought" [1847], hence the name of this chapter.

The history of logic has two peculiarities that distinguish it from other disciplines. First, formal logic was created by Aristotle virtually out of nothing. Second, for over two thousand years this subject matter was believed to have been exhausted by its creator.

Although the history of formal logic begins with Aristotle whose thinking has had a profund influence throughout the world, logical thinking was known before his time. There is proof of this in the Golenishev Papyrus written by an unknown Egyptian priest [*], and in the Rhind Papyrus written by A'h Mose [*]. There also is evidence that the Babylonians knew logical procedures [*].

The logic of Aristotle, with its rules of syllogisms, defined the elementary processes for all reasoning. A

---

* See Alice M. Hilton [1963], Ch. 1

syllogism is a deductive argument in which a conclusion is 'mediately' inferred from two premises. It is mediate inference because the conclusion is supposed to be drawn from the first premise through the 'mediation' of the second. Syllogisms will be explained in 3.3.3

Many contributions to logic were made between Aristotle's time and the nineteenth century. One exceptional contributor was Gottfried Wilhem von Leibnitz [1666], who created a precise symbolic language reflecting the structure of thinking. Another important contributor was Immnanuel Kant [1781] who introduced the principles of inductive reasoning.

John S. Mill [1843] made a profound contribution to inductive logic and introduced the distinction between connotation and denotation. August De Morgan [1847] also quantified the predicate. In addition, he also made several studies in non-traditional modes of inference. G. Boole [1847] invented a method to deal in an easier way with syllogisms and also invented his famous algebra. W. Hamilton's [1852] "Quantification of the Predicate" suggested a way of dealing with propositions as equations of terms. This quantification constituted the basis of what is currently known as predicate calculus.

The algebra of classes was developed by many logicians, among them, W. Jevons [1874], who solved several problems in Boole's Algebra by interpreting the "v" symbol as an inclusive "or" rather than Boole's "exclusive" interpretation. J. Venn [1876] and C. Peirce [1876] adopted Jevons's interpretation and eliminated the operations of subtraction and division in the Boolean Algebra. Venn, particularly, developed a spatial represeentation for propositional Logic. Peirce and E. Schroder [1909] developed the logic of propositions, propositional functions and relations.

G. Frege [1893] is considered the founder of modern logic because he was the first to describe the propositional calculus in its modern form. He also was the first to introduce the notion of a propositional function, the use of quantifiers, and the logical analysis of proof by mathematical induction.

By the end of the nineteenth century Giuseppe Peano [1895-1908] greatly increased the range of symbolic logic by introducing symbols for other logical notions such as "is contained in", "there exists", "is a", etc., symbols which were later adopted by Russell and Whitehead.

Between 1910 and 1913, B. Russell and A. Whitehead developed their Principia Mathematica in which they incor-

porated most of mathematics into a logistic development. They also created a calculus of relations.

Paradoxes have always played a vital role in logic and philosophy. They help to delineate the boundaries of individual knowledge and human comprehension. One of the most interesting paradox is Zeno of Elea's paradox about motion [+]. It presents three problems: infinitesimal, infinity, and continuity. The first problem was solved by K. Weirstrass [**]. The solutions of the other two were proposed by R. Dedekind [**]. G. Cantor [1915] solved these two problems. Another important contributor to logic is L. von Wittgenstein [1953] who invented truth tables.

The purpose if this brief history is to give a feel for the long tradition of logic in western civilizations. The "stretching" of logic to fit common sense problems is the latest in a long line of work aimed at coming up with a formalization for knowledge.

** See R. Mattesich [1978], Ch. 3

+ Suppose a race between the fleet-footed Achilles and a tortoise. Achilles is given a handicap of one hundred meters. The gun indicating the start of the race is shot at time T. When Achilles reaches the starting point of the tortoise, at time T + T1, the tortoise is not there, having moved a short distance beyond. Achilles continues in pursuit of the tortoise, and at time T + T1 + T2 comes to the place the tortoise was at T + T1. The tortoise is not there. Achilles continues in pursuit and so on, and on. Achilles will never catch up with the tortoise.

3.2 Boolean Algebra

A Boolean algebra is a tuple ⟨B, O⟩, where:

B: is a non-empty set.

$0, 1 \in B$, where $0 \neq 1$.

O: is a set containing three operations

$\vee$: $B \times B \rightarrow B$. It is a binary operation called disjunction.

$\wedge$: $B \times B \rightarrow B$. It is a binary operation called conjunction.

$\sim$: $B \rightarrow B$. It is a unary operation called negation or complement.

The operations $\vee$ and $\wedge$ are characterized by the following identities which are satisfied for all x, y, and z e B.

a) Idempotency: $x \vee x = x$

$x \wedge x = x$.

b) Commutation: $x \vee y = y \vee x$

$x \wedge y = y \wedge x$.

c) Association: $(x \vee y) \vee z = x \vee (y \vee z)$

$(x \wedge y) \wedge z = x \wedge (y \wedge z)$.

d) Distribution: $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$.

e) Absorption: $x \vee (x \wedge y) = x$

$x \wedge (x \vee y) = x.$

f) Universe Class: $x \vee 1 = 1$

$x \wedge 1 = x.$

g) Null Class: $x \vee \emptyset = x$

$x \wedge \emptyset = \emptyset.$

h) Complement: $x \vee \sim x = 1$

$x \wedge \sim x = \emptyset.$

i) De Morgan: $\sim(x \vee y) = \sim x \wedge \sim y$

$\sim(x \wedge y) = \sim x \vee \sim y.$

j) Involution: $\sim \sim x = x.$

This algebra has the following characteristics.

If there is a class of elements x and a class of elements y, disjunction forms a class composed of elements x "or" elements y, or both. A class is a collection of objects which are identified by some means or some proper characteristic which its members have.

If there is a class of elements x and a class of elements y, conjunction forms a class composed of elements x "and" elements y.

Negation means that if there is a class of elements x, then there is also a class of elements not x.

The laws of idempotency imply synthesis; that is, if both elements are equal, it is the same to consider one, or the other, or both.

The laws of commutation mean that a sequential change does not affect either disjunction or conjunction. They were also explained in chapter two.

The laws of association mean that disjunction and conjunction are not affected by the ordering of the elements.

The laws of distribution imply that the disjunction of an element with a conjunction is achieved by distributing this element with each element of the conjunction. Similarly, the conjunction of an element with a disjunction means that this element is distributed with each element of the disjunction. In chapter two they were also described.

The laws of absorption are the result of combining the laws of idempotency and distribution

Given $x \vee (x \wedge y) = (x \vee x) \wedge (x \vee y)$,

since $(x \vee x) = x$

then $x \wedge (x \vee y) = x$.

Similarly for the product

Given $x \wedge (x \vee y) = (x \wedge x) \vee (x \wedge y)$,

since $(x \wedge x) = x$

then $x \vee (x \wedge y) = x$.

The truth value of x, Ø or 1, alone determines the truth value of the entire expression.

The laws of universe class state that the truth value of the disjunction of x and the universe is always 1, and the truth value of the conjunction is x.

If x=Ø: $Ø \vee 1 = 1$, $Ø \wedge 1 = Ø$; and

if x=1: $1 \vee 1 = 1$, $1 \wedge 1 = 1$.

Similarly, the laws of null class state that the truth value of the disjunction of x and the null class is x, and the truth value of the conjunction is Ø.

If x=Ø: $Ø \vee Ø = Ø$, $Ø \wedge Ø = Ø$; and

if x=1: $1 \vee Ø = 1$, $1 \wedge Ø = Ø$.

The laws of complement assert that x, or its complement, or both, form the universe and that the conjunction of x with its complement is the null class.

The laws of De Morgan were discussed in chapter two.

Finally, the law of involution is obvious; it dictates that the complement of the complement of x is x.

It is important to see the concept of Boolean Algebra as a calculus of propositional functions, and not only as a calculus of classes. Propositions can either be true or false, 1 or Ø respectively.

Regardless of how different two propositions seem, they are equivalent if they have the same truth values, and if this occurs, their negations have the same truth values too. The important issue for the calculus is the condition of truth and falsehood, and not what is true or false. The calculus of logical relations shows the formal properties of logical relations but does not sanction them, i.e., it demonstrates something, but it does not approve or disapprove.

3.3 Propositional Calculus

Propositional calculus is a method for calculating with propositions that are either true or false, by combining them and deducing other propositions from them. These sentences are treated in an abstract way, so that, if "p" denotes a proposition, there is no interest in knowing what "p" means, but rather what happens if "p" is a true or a false proposition.

A simple statement is one which does not contain any other statement as a component, for example, "This is logic". A compound statement is one which contains another statement as a component. The other statement may be compound too. Examples are "A city of Italy is Rome or a city of Spain is Seville", "If you break it, then you pay it", etc.

The first kind of compound statement to consider is conjunctive compound statements. The conjunction of two statements is achieved by placing an "and" between them. Example:

a) Alice is a girl and lives in Wonderland.

p: Alice is a girl.

q: Alice lives in wonderland.

---

p ^ q

Conjunctive statements are true if and only if both constituent statements are true.

The disjunction of two statements is achieved by placing an "or" between them. Example:

b) Tomorrow is Friday or Saturday.

p: Tomorrow is Friday.

q: Tomorrow is Saturday.

---

p v q

Disjunctive statements are false if and only if both constituent statements are false, otherwise, they are true.

The negation of a statement is formed by inserting a "not" into the original statement. Negative statements are true if and only if what is negated is false. As an example consider the following compound statement.

c) To be or not to be....

p: To be

---

p ⊕ ~p

This example shows a special case of disjunction. The statement can be represented logically as (p v ~p),

but Shakespeare's idea is a "mutually exclusive" or and not an "inclusive" one. (p v ~p) is true if one or both of its disjunctives is true. Therefore, it is impossible "to be" and "not to be" at the same time. If the disjunction is represented as (p ⊕ ~p), which is a mutually exclusive "or", then to be true, only one and not both of its disjunctives may be true at a time. That is precisely Shakespeare's idea.

Conditional statements are formed by placing an "if" before the first statement and inserting a "then" between the statements. The statement between the "if" and the "then" is called the "antecedent", and the statement after the "then" is called the "consequent". In other words, conditional statement asserts that its antecedent implies its consequence. An example is.

d) If I think, then I am.

p: I think

q: I am

---

p ⊃ q

ar wording for this is "I think, there-

ergo sum).

Conditional statements are false only when the consequent is false, and true otherwise. The conditional

statement p ⊃ q, then, is equivalent to (~p v q).

These are the most important connectives since the other commonly used can be defined in terms of these.

3.3.1 Well-Formed Formulas

The first step in problem solving is to observe and identify relevant data, that is, to represent knowledge in the language of the logical system. This language describes strings of symbols that are composed according to certain grammatical rules. These strings are called the well-formed formulas , or wff's, of the logic. The rules would, for example, accept such an expression as "~p v q", but would not permit an expression like "p q v ~". The wff's, then, are those symbol strings that make sense grammatically.

It is necessary to prove both theorems about the language and theorems within the language, two distinct levels called the "language" or "theory," and the "metalanguage" or "metatheory" respectively.

Certain symbols within the language have already been introduced, such as those representing the logical connectives. In addition, the parenthesis and propositional variables like p, q, r, etc. have been introduced. Within

the metalanguage, the letters A, B, C, etc. denote entire well-formed formulas. Thus, (A) v (B), for example, is an expression which denotes any desired member of the class of formulas obtained by substituting expressions for the letters A and B.

A formal definition of a wff is the following:

a) A propositional variable is a wff.

b) If A is a wff, ~(A) is a wff.

c) If A and B are wff's then any logical connection of them is a wff.

d) A symbol string is wff if and only if it is produced by the application of the previous rules.

Examples:

a) A: ((p) ^ (~(q))) ⊃ ((~(~(p))) == (q)) is a wff.

A: (A1) ⊃ (B1) where:

A1: (A2) ^ (B2)

A2: p is a wff

B2: ~(q) is a wff

B1: (A3) == (B3)

A3: ~(A4)

A4: ~(p) is a wff

B3: q is a wff :. A is a wff.

b) A: ((p) ^ (q)) ⊃ ((~(q)) ^ (p) v (r))

is not a wff.

A: (A1) ⊃ (B1) where:

A1: (A2) ^ (B2)

A2: p a wff.

B2: q a wff.

B1: (A3) ^ B3 v C3

:. A is not a wff because B1 has 3 elements instead of 2, contradicting part c) of the definition.

3.3.2 Truth Tables

The truth value of an expression depends on those values that its variables may take. An expression with n variables represents a function of 2^n elements which may be represented in a truth table. Thus, a truth table shows the truth values of a wff for each assignment to its variables. As an example, consider the following expression (p ^ q) ⊃ (p v ~r).

p	q	r	p ^ q (1)	p v ~r (2)	(1) ⊃ (2)
t	t	t	t	t	t
t	t	f	t	t	t
t	f	t	f	t	t
t	f	f	f	t	t
f	t	t	f	f	t
f	t	f	f	t	t
f	f	t	f	f	t
f	f	f	f	t	t

A compressed truth table of the same expression is

p	q	r	p ^ q (1)	p v ~r (2)	(1) ⊃ (2)
t	-	-	-	t	t
f	-	-	f	f	t

If the result of the truth table for any assignment of values is true, it is called a tautology. If the expression always takes the false value, it is called a contradiction. If the final value depends on the assignment of truth values to the variables, the expression is called a contingency.

Two propositions are equivalent if both have the same truth tables. One method for proving theorems, then, is through truth tables by examining all possible combinations for both propositions. This method works but is inefficient because if n variables occur in the premises, then two tables, each with 2^n rows must be examined.

3.3.3 Methods of Proof

A syllogistic argument, or syllogism, is an argument consisting of three propositions, the first two being the premises and the third, the conclusion or theorem. This kind of argument is also called "inference schema" and is valid if the premises are accepted as true, and the conclusion is also true. The form is

premise 1

premise 2

.: conclusion

which is equivalent to {(premise 1) ^ (premise 2)} ⊃ conclusion. In other words, the antecedent is formed by the conjunction of the premises and the consequent is the conclusion. As an example consider

{(m ^n) ^ o) ⊃ p}

q ⊃ {(o ^ m) ^ n)}

:. ~q v p

This kind of argument represents the most common way of reasoning, and there are several methods to prove the validity of such an argument, including the following.

3.3.3.1 Rules of Inference and Equivalence

As was mentioned above, truth tables are not a good method for testing the validity of an argument because of the inefficiency of exhaustively examining all the rows of the table. A more efficient way is to infer the conclusion of an argument from its premises by a sequence of substitutions of valid elementary arguments. The process of substitution means to substitute statements for statement variables and not statements for statements. The rules of inference are the following tautologies:

1. Modus Ponens

$$\begin{array}{l} p \supset q \\ p \\ \hline \therefore\ q \end{array}$$

2. Modus Tollens

$$\begin{array}{l} p \supset q \\ \sim q \\ \hline \therefore\ \sim p \end{array}$$

3. Hypothetical Syllogism

$$\begin{array}{l} p \supset q \\ q \supset r \\ \hline \therefore\ p \supset r \end{array}$$

4. Disjunctive Syllogism

$$\begin{array}{l} p \vee q \\ \sim p \\ \hline \therefore\ q \end{array}$$

5. Constructive Dilemma

$$\begin{array}{l} (p \supset q) \wedge (r \supset s) \\ p \vee r \\ \hline \therefore\ q \vee s \end{array}$$

6. Absorption

$$\begin{array}{l} p \supset q \\ \hline \therefore\ p \supset (p \wedge q) \end{array}$$

7. Simplification

$$\begin{array}{l} p \wedge q \\ \hline \therefore\ p \end{array}$$

8. Conjunction

$$\begin{array}{l} p \\ q \\ \hline \therefore\ p \wedge q \end{array}$$

9. Addition

$$\frac{p}{\therefore\ p \vee q}$$

Many arguments cannot be proved using only the previous rules of inference; additional rules are required. These new rules are logical equivalences, i.e., they replace a statement by another one having the same truth value according to the following tautologies:

10. De Morgan's Theorem

$$\sim(p \wedge q) \equiv (\sim p \vee \sim q)$$
$$\sim(p \vee q) \equiv (\sim p \wedge \sim q)$$

11. Commutation

$$(p \vee q) \equiv (q \vee p)$$
$$(p \wedge q) \equiv (q \wedge p)$$

12. Association

$$\{p \vee (q \vee r)\} \equiv \{(p \vee q) \vee r\}$$
$$\{p \wedge (q \wedge r)\} \equiv \{(p \wedge q) \wedge r\}$$

13. Distribution

$$\{p \wedge (q \vee r)\} \equiv \{(p \wedge q) \vee (p \wedge r)\}$$
$$\{p \vee (q \wedge r)\} \equiv \{(p \vee q) \wedge (p \vee r)\}$$

14. Double negation

$$p \equiv \sim\sim p$$

15. Transposition

$$(p \supset q) \equiv (\sim q \supset \sim p)$$

16. Material Implication

$(p \supset q) == (\sim p \vee q)$

17. Tautology

$p == (p \vee p)$
$p == (p \wedge p)$

18. Material Equivalence

$(p == q) == \{(p \supset q) \wedge (q \supset p)\}$
$(p == q) == \{(p \wedge q) \vee (\sim p \wedge \sim q)\}$

19. Exportation

$\{(p \wedge q) \supset r)\} == \{p \supset (q \supset r)\}$

Taken together these lists constitute a complete system because they permit the construction of a formal proof for any truth argument. The first nine rules can be applied only to whole lines of a proof and the last ten, can be applied either to whole lines or to parts of the lines. Several of these rules were included in the Boolean Algebra.

Example:

1) $\{(m \wedge n) \wedge o\} \supset p$

2) $q \supset \{(o \wedge m) \wedge n\}$

$\therefore \sim q \vee p$

3)	$\{o \wedge (m \wedge n)\} \supset p$	Commutation	(1)
4)	$\{(o \wedge m) \wedge n)\} \supset p$	Association	(3)
5)	$q \supset p$	Hipothetical Syllogism	(2,4)
6)	$\sim q \vee p$	Material Implication	(5)

3.3.3.2 Invalidity

If formal attempts to prove the validity of an argument fail, this does not mean that the validity of the argument cannot be proved by other means neither that the argument is invalid. The method of invalidity is closely related to truth tables, but is shorter.

An argument is invalid if a single case can be found in which truth values are assigned to the variables in such a way that the premises are true and the conclusion is false. Assigning truth values to the variables such that the premises are true and the conclusion false is sufficient to prove the invalidity of an argument. This kind of assignment is equivalent to the one done by truth tables, but it is shorter. Consequently, it is not necessary to examine all rows in a truth table to prove the

invalidity of an argument; it is sufficient to find only one row in which the premises are true and the conclusion is false.

Example:

1) $\{(m \wedge n) \wedge o\} \supset p$

2) $\sim q \supset \{(o \wedge m) \wedge n)$

$\therefore (\sim q \vee p) \supset \sim p$

Assigning T to m, n, o, and p, and assigning F to q, the premises become true. Following these assignments the conclusion is false. Since the conclusion is false, the argument is invalid. Expanding the previous example.

$\{(m \wedge n) \wedge o\} \supset p$ T

T T T T

$\sim q \supset \{(o \wedge m) \wedge n\}$ T

F T T T

$\therefore (\sim q \vee p) \supset \sim p$ F

F T F

3.3.3.3 Inconsistency

An argument is valid if its premises are mutually inconsistent. If a truth table is constructed for such an argument, it will show that in every row, at least one of the premises is false. In other words, there is no row in which all the premises are all true and the conclusion false, hence, the truth table establishes the validity of the argument.

The case of inconsistency is an attempt to negate the principle of contradiction (see 2.1.2). The real problem with inconsistencies is that any and every conclusion follows logically from inconsistent premises.

Example:

1) $a \supset b$

2) $\sim a \supset c$

3) $\sim(b \vee c)$

$\therefore d$

This argument is valid, because its variables are inconsistent. The inconstency is shown in variables a and b, because both are affirmed in premise 1, and both are negated in premises 2 and 3. The conclusion d is vacuously true, since it follows from inconsistent premises.

3.3.3.4 Propositional Resolution

Propositional resolution is an iterative process of proof which at each step compares or resolves two clauses yielding a new clause inferred from them. Propositional Resolution produces proofs by refutation. That is, to prove a theorem it attempts to show that the negation of the theorem produces a contradiction with the known premises.

Before explaining how this process works, it is necessary to define some new concepts. The conjunctive form of a proposition is one in which negation symbols apply only to variables and not to parenthesized expressions, and 'or' symbols connect only variables and not parenthesized expressions. A clause , then, is a simple wff or a wff in conjunctive form.

The resolution procedure says

a) Convert all premises and the negation of the theorem to clause form.

b) Select two clauses that contain a variable and its negation.

c) Generate a new clause containing all the 'or'-ed elements of the selected clauses except for the variable chosen and its negation.

d) Continue the process until it either leads to a contradiction, in which case the theorem is proved, or until no more clauses can be generated, in which case a falsehood is proved.

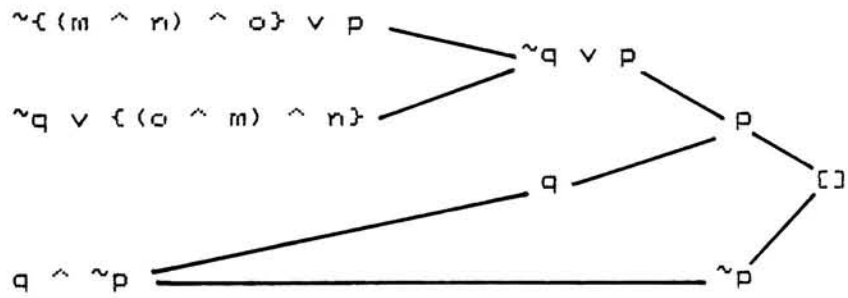
Example:

$\{(m \wedge n) \wedge o\} \supset p$

$q \supset \{(o \wedge m) \wedge n\}$

$\therefore \sim q \vee p$

Converting this syllogism to clause form and negating the theorem



3.3.3.5 Wang's Algorithm

Wang's algorithm is a syntactic proof method, created by Hao Wang [1970], which produces the same results as truth tables but requires less computational effort.

It consists of writing down a series of lines, each simplier than the previous one, until a proof is completed or it is shown to be impossible to continue. Each line consists of any number of premises separated by commas on each side of an arrow.

The rules say the following

1) The initial form is:

   premise 1, premise 2, .., premise n -> conclusion.

2) If the principal connective of a premise is a negation, drop the negation symbol and move the premise to the other side of the arrow.

3) If the principal connective of a premise on the left side of the arrow is an "and", or on the right is an "or", replace it by a comma.

4) If the principal connective of a premise on the left side of the arrow is an "or", or on the right is an "and", produce two new lines, each with one of the two sub-premises replacing the premise.

5) If the same premise occurs on both sides of the arrow, the line is proven.

6) If no connectives remain in a line or no propositional variable occurs on both sides of the arrow, the argument is not provable.

Example:

{(m ^ n) ^ o} ⊃ p

q ⊃ {(o ^ m) ^ n}

.: ~q v p

Rule 1:

~{(m ^ n) ^o} v p, ~q v {(o ^ m) ^ n} -> ~q v p

Rule 4:

a) ~{(m ^n) ^ o}, ~q v {(o ^ m) ^ n) -> ~q v p

b) p, ~q v {(o ^ m) ^ n} -> ~q v p

Applying rule 3 to part b) :

p, ~q v {(o ^ m) ^ n} -> ~q, p

By rule 5, this branch is proved, because "p" is on both sides of the arrow.

Applying rule 2 to part a) :

~q v {(o ^ m) ^ n} -> ~q v p, {(m ^ n) ^ o}

Applying rule 4:

aa) ~q -> ~q v p, {(m ^ n) ^ o}

bb) {(o ^ m) ^ n} -> ~q v p, {(m ^ n) ^ o)}

Applying rule 5 in bb) it is proved.

Applying rule 3 to aa), it becomes:

~q -> ~q, p, {(m ^ n) ^ o)}

Applying rule 5, the argument is valid.

3.4 Predicate Calculus

Propositional calculus deals only with true and false sentences; there is no reasoning about individual entities and their properties. For purposes of artificial intelligence, this is not very useful because in order to get knowledge, more than just true and false sentences are needed. It is also necessary to talk about objects, to postulate relationships between them, and to generalize these relationships over classes of objects. These objectives can be accomplished through predicate calculus.

Predicate calculus is an extension of propositional calculus. The meaning of the logical connectives is retained, but the focus of the logic is changed. Instead of looking at the truth value of the sentences, predicate calculus is used to represent statements about specific objects or individuals.

Consider the example

All humans are mortal. P

Socrates is human. Q

Socrates is mortal. .: R

This notation appears to be invalid even when the argument is valid.

The validity of an argument does not depend upon the way in which simple statements are compounded but rather upon the inner logical structure of the non-compound statements involved.

3.4.1 Basic Concepts

A predicate is a statement about individuals, both by themselves and in relation to other individuals. It has a value of either true or false depending upon which individual it is applied to. If the predicate "is Greek" is considered as an example, by applying it in the singular proposition "Socrates is Greek," it is true, but applied to "Caesar is Greek," it is false.

Particular entities, such as Socrates, Caesar, etc. are designated as individuals, and lower-case letters (excepting x) are used to represent them. Generally, the first letters of the vocabulary are employed. General predicates such as "is Greek," "is human," "is mortal," etc. are represented with upper-case letters.

Having two sets of symbols, one for particular entities or individuals, and the other for attributes of the individuals, the convention is adopted of writing an attribute symbol immediately to the left of the individual symbol. Thus, propositions such as "Socrates is human" or

"Socrates is mortal," for example, are represented as Hs and Ms respectively. When reference to no one in particular is required, it is written Hx to represent the attribute H(uman) of the individual x, no one in particular, and so forth. A letter x is a place marker.

The propositions Hs, Ms, etc. are either true or false, but a proposition using x such as Hx, is neither true nor false, it is a propositional function which contains a variable and becomes a proposition when a constant is substituted for this variable. Propositional functions such as Hx, Mx, etc. are called "simple predicates" to distinguish them from the more complex propositional functions discussed later. A simple predicate is a propositional function having some true and some false substitution instances, each of which is a simple proposition.

3.4.2 Quantifiers

Each predicate defines a set or a sort, i.e., for any predicate P, all individuals X can be sorted in two disjoint groups, one containing those objects that satisfy P (for which P(X) is true), and the other containing those objects that do not satisfy P. It is very important to be able to reference both parts and the whole universe of discourse.

To deal with this situation, there are two different quantifiers, the universal (∀) and the existential (∃). The first one means "For all....," and the second one means "There exists... ." Thus, the general example "All things are mortal" is represented as (∀x)Mx, and "Something is mortal," as (∃x)Mx. The universal quantification of a propositional function is true if and only if all of its substitutions are true. The existential quantification of a propositional function is true if and only if it has at least one true substitution. Thus, propositions also can be obtained by generalization, placing a universal or existential quantifier before a variable, and not only by substituting individual constants for individual variables.

Propositions such as Mx or Hx can be denied by ~Mx and ~Hx respectively. The universal proposition "Everything is mortal" is denied by the proposition "Something is not mortal." Both equivalences are represented as

$$\sim(\forall x)Mx \equiv (\exists x)\sim Mx.$$

Since one is the denial of the other, these two propositions are equivalent

$$(\forall x)Mx \equiv (\sim \exists x)\sim Mx.$$

Similarly, "Nothing is mortal" is denied by "Something is mortal" and represented by

$$(\sim \forall x)\sim Mx \;==\; (\exists x)Mx.$$

Since one is the denial of the other, these two propositions are equivalent

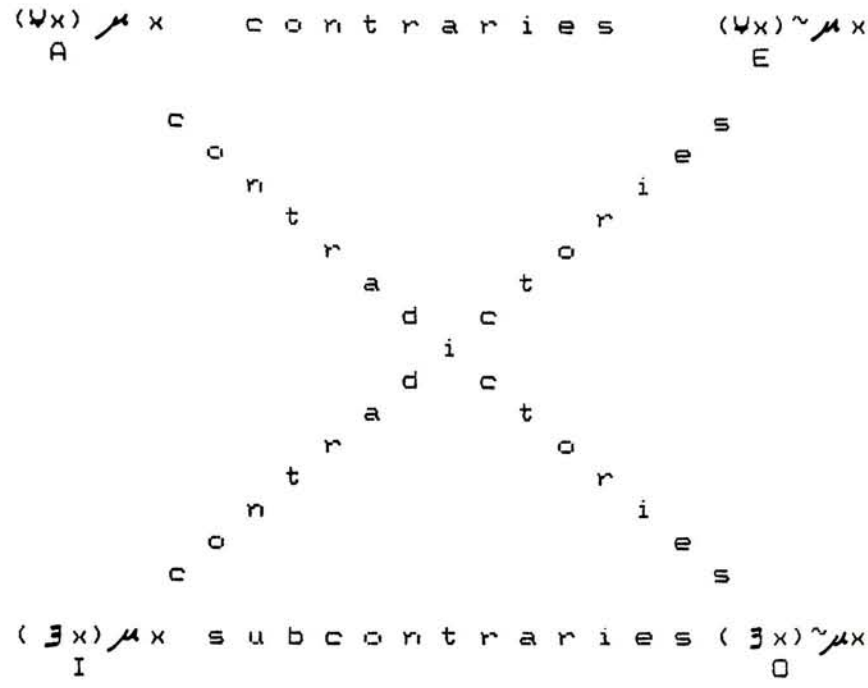
$$(\forall x)\sim Mx \;==\; (\sim \exists x)Mx.$$

Generalizing the previous relations between quantifiers

$$(\forall x)\mu x \;==\; (\sim \exists x)\sim \mu x.$$
$$(\exists x)\mu x \;==\; (\sim \forall x)\sim \mu x.$$
$$(\forall x)\sim \mu x \;==\; (\sim \exists x)\mu x.$$
$$(\exists x)\sim \mu x \;==\; (\sim \forall x)\mu x.$$

where μ represents any simple predicate.

The graphical representation of the relations between quantifiers can be expressed as follows



Two propositions are contraries if they might both be false but cannot both be true. Two propositions are subcontraries if they can both be true but cannot both be false. When one proposition is true and the other false, both are contradictories.

Propositions of types A, E, I, and O, are classified as universal affirmative, universal negative, particular affirmative and particular negative respectively.

Consider as an example the following type A proposition "All Humans are Mortals." This can be represented as (∀x)(Hx ⊃ Mx). A similar type E proposition says "No Humans are Mortals," which can be represented as (∀x)(Hx ⊃ ~Mx). Applying the same example to a similar I type,

"Some humans are mortal," can be represented as (∃x)(Hx ^ M x). Finally, the O type proposition says "Some Humans are not Mortals," which is translated to (∃x)(Hx ^ ~Mx).

An A type proposition may be true while its corresponding I type proposition is false. Similarly, an E proposition may be true while its corresponding O proposition is false.

3.4.3 Rules of Inference

To construct formal proofs of validity, it is necessary to expand the previous list of rules of inference and equivalence. Four additional rules are required. The first one is the Universal Instantiation, UI, which dictates

$$\text{UI:}\quad \frac{(\forall x)(\phi\, x)}{\therefore\ \phi\, w}$$

where ϕ represents any simple predicate and w, any individual symbol.

The second rule is the Universal Generalization, or UG for short, which implies

$$\text{UG:}\quad \frac{\phi\, w}{\therefore (\forall x)(\phi\, x)}$$

The third rule is the Existential Instantiation, EI, which establishes

$$\text{EI:}\quad \frac{(\exists x)(\phi x)}{\therefore\ \phi w}$$

The last rule is the Existential Generalization, EG, which denotes

$$\text{EG:}\quad \frac{\phi w}{\therefore\ (\exists x)(\phi x)}$$

As examples of the application of these rules consider

I) All Humans are Mortals

Greeks are Humans

.: Greeks are Mortals.

1) $(\forall x)(Hx \supset Mx)$

2) $(\forall x)(Gx \supset Hx)$

$(\forall x)(Gx \supset Mx)$

3)	$Hw \supset Mw$	UI (1)
4)	$Gw \supset Hw$	UI (2)
5)	$Gw \supset Mw$	Hypothetic Syllogism (4,3)
6)	$(\forall x)(Gx \supset Mx)$	UG (5)

II) All philosophers are Greek

Some mariners are philosophers

Some mariners are Greek.

1) $(\forall x)(Px \supset Gx)$

2) $(\exists x)(Mx \cdot Px)$

$(\exists x)(Mx \cdot Gx)$

3)	$Ma \cdot Pa$	EI (2)
4)	$Pa \supset Ga$	UI (1)
5)	$Pa \cdot Ma$	Commutation (3)
6)	Pa	Simplification (5)
7)	Ga	Modus Ponens (4,6)
8)	Ma	Simplification (3)
9)	$Ma \cdot Ga$	Conjunction (8,7)
10)	$(\exists x)(Mx \cdot Gx)$	EG (9)

3.4.4 Invalidity

To prove the invalidity of an argument, the same technique of assigning truth values to the variables used in propositional calculus can be employed. This method is based on the general assumption that there is at least one individual in the quantified class of objects.

If there is exactly one individual, say a, then

$$(\forall x)(\phi x) \equiv \phi a \equiv (\exists x)(\phi x).$$

If there are exactly two individuals, say a and b, then

$(\forall x)(\phi x) \equiv \phi a \cdot \phi b$, and $(\exists x)(\phi x) \equiv \{\phi a \vee \phi b\}$;

and so on.

Thus, an argument involving quantifiers is valid if and only if it is valid no matter how many individuals there are, provided there is at least one.

Hence, the procedure to prove invalidity is the following. First, consider a one-element model, writing out the logically equivalent truth-function argument. If this argument can be proved invalid by assigning truth values to its statements, that suffices to prove the invalidity. If that cannot be done, consider a two-element model based on the previous equivalences. If the current argument cannot be proved invalid, continue adding elements.
As an example, consider

$$\begin{array}{l} (\forall x)(Mx \supset Nx) \\ (\exists x)(Mx \cdot Ox) \\ \hline \therefore\ (\forall x)(Ox \supset Nx) \end{array}$$

For a model containing only one element, this is equivalent to

$$\begin{array}{l} Ma \supset Na \\ Ma \cdot Oa \\ \hline \therefore\ Oa \supset Na \end{array}$$

Assigning T to Ma, Na, and Oa, the argument is valid. But considering the equivalent of two elements

$$(Ma \supset Na) \wedge (Mb \supset Nb)$$
$$(Ma \wedge Oa) \vee (Mb \wedge Ob)$$
$$\therefore (Oa \supset Na) \wedge (Ob \supset Nb)$$

which is proved to be invalid by assigning T to Ma, Na, Oa, and Ob, and F to Mb and Nb. Hence, the original argument is not valid for a model containing two elements, and is therefore invalid.

3.4.5 Asyllogistic Inference

All the previous arguments were of a form called categorical syllogism which consists of two premises and one conclusion. To evaluate more complicated arguments, more rules than those already developed are not required.

There are three locutions of natural English that deserve special attention for the construction of more complex arguments. The first one can be observed in statements like "All politicians are either honest or liars." Although it contains an "or" as connective, it is not a disjunction. This statement is very different from "Either all politicians are honest or all politicians are

These statements are represented by

$(\forall x)\{Px \supset (Hx \vee Lx)\}$ and

$(\forall x)(Px \supset Hx) \vee (Px \supset Lx)$ respectively.

The second locution to be observed can be stated in the following sentence "Socrates and Plato are Greeks." This sentence translated to a disjunctive statement

$$(\forall x)\{(Sx \vee Px) \supset Gx\}$$

because Greek is Socrates or Plato but not someone who is both Socrates and Plato.

The last locution is related with different ways of representing excepting propositions, such as "All except....," "All but...," etc. Consider as an example "All except Greeks were Ignorants" which can be represented as

$$(\forall x)(Gx \supset \sim Ix) \wedge (Vx)(\sim Gx \supset Ix)$$

which is equivalent to

$$(\forall x)(Ix \equiv \sim Gx)$$

This is translated into "Anyone who was ignorant was not a Greek."

3.4.6 Resolution

Before explaining how resolution in predicate calculus proves theorems, it is important to know how to determine the contradiction of two literals. In propositional calculus it is very easy to determine that x and ~x is a contradiction, but in predicate calculus it is not simple since the binding of variables is considered.

For Example:	Ms,	~Ms	is a contradiction.
	Ms,	~Mp	is not a contradiction.
	Hs,	~Ms	is not a contradiction.
	Hs,	~Hx	is a contradiction.

The first three cases are obvious, but in the last case, ~Hx claims that there is no x for which Hx is true, and Hs claims that there is an object s for which Hs is true (Similarly to Hx and ~Hs).

This method of resolution is similar to the propositional resolution previously discussed; that is, this method assumes that the negation of the theorem to be proved is true, and attempts to deduce a contradiction from that negation and the original premises.

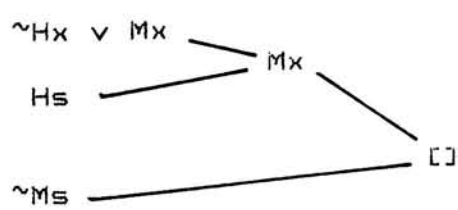
The negation of theorems can be achieved simply by adding the negation symbol to the proposition and looking for its equivalent representation where the quantifier is

not negated or through the square of contradictories, contraries, etc. previously defined.

Existential quantifiers are eliminated by replacing a variable for a constant. For example (∃x)(Mx) is replaced by Ms, claiming that an x exists by choosing a particular s to take its place. However, if an existential quantifier is within the range of a universal quantifier, that constant must depend on the identity of the universally quantified variable. Thus, the replacement is a function of the universally quantified variable. For example (∀x)(∃y)(Mxy) is replaced for (∀x)(Mxf(x)). The function "F" is called a skolem function and f(x), a skolem expression. Universal quantifiers are simply dropped. The resulting expressions are called the quantifier-free form of the Predicate Calculus language. As examples consider

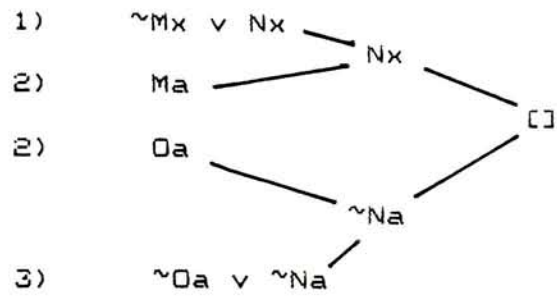
a) (∀x)(Hx ⊃ Mx)
Hs

∴ Ms which is translated into



b) $(\forall x)(Mx \supset Nx)$
$(\exists x)(Mx \wedge Ox)$

$\therefore (\exists x)(Ox \wedge Nx)$ which is translated into:



1) By dropping quantifier and applying Material Implication.

2) Dropping quantifier and assigning a value to x.

3) Dropping quantifier, assigning a value to x, negating the theorem and applying De Morgan's theorem.

4.0 Alternative Ways to Achieve Common Sense Reasoning

It is not at all clear how to represent common sense reasoning in a computer. Common sense reasoning differs from the reasoning achieved through classical logic in the need to draw conclusions from partial or incomplete knowledge. In classical logic, a conclusion is not accepted unless it can be proved according to the rules of inference. In common sense reasoning, however, conclusions without a formal proof are accepted just because they seem plausible.

The artificial intelligence problem consists of how to make true thinking machines. Two opposite philosophical points of view seek to solve this problem, a logical approach and a psychological one.

The logical approach, advocated by McCarthy, considers that the way to solve the AI problem is to design computer programs that reason according to mathematical logic, whether or not that is the way people think.

In Classical logic, the addition of new axioms never decreases the set of theorems. The new axioms produce new theorems, so that the set of theorems grows monotonically with the set of axioms, that is, as more premises are added, the conclusions keep changing.

What is needed is a kind of logic with a set of theorems that may lose members as well as gain them when new axioms are added. This non-monotonic logic that would be able to adapt itself to an incomplete knowledge. Two methods of non monotonic reasoning are reasoning by circumscription, developed by McCarthy, and reasoning by default. Within the latter method, two sorts of detailed formalizations have been proposed non-monotonic logic, developed mainly by McDermott and Doyle, and logic by default, developed by Reiter.

The psychological approach, advocated by M. Minsky, proposes that mathematical logic is almost certainly not the way the human mind works. Minsky calls his approach the frame system. The idea is to put large collections of information into a computer, more information that is ever required to solve any particular problem, and then for each particular situation, to define which details are optional and which are not. A collection of frame definitions set the scene for common sense reasoning, but the importance of the details in a frame can change if there is a change in purpose or goal. In a sense, then, frame systems are like logic with one important difference. Ordinarily, logic would not say which pieces of knowledge are more important than others.

All efforts to solve the AI problem share two major obstacles, first to decide what knowledge to represent, and second, to get answers out of the computer in a reasonable time.

The first part of this chapter explains in more detail the characteristics of reasoning by default, the second part describes the process of reasoning by circumscription and the third part presents the frame system.

4.1 Reasoning by Default

There are two important cases of reasoning by default: non-monotonic logic developed by D. McDermott and J. Doyle, and default reasoning developed by R. Reiter.

Both formalizations interpret a default, S, as provable unless and until S can be disproved. What can be inferred depends on what inference rules are applicable. Simultaneously, what inference rules are applicable depend on what can be inferred. The difficulty with this concept, then, is its circularity.

Both approaches also agree in the way they interpret defaults and in their major theoretical properties, but they differ in their logical forms. Non-monotonic logic defines defaults as modal formulas, and default reasoning defines them as inference rules.

Non-monotonic logical rules are more expressive than those of the logic of defaults because it is possible to make statements about defaults in terms of non-monotonic logic. This is why reasoning by default is not covered in this thesis. However, some applications of it are explained in reference to frames (See 4.3.4).

4.1.1 Non-Monotonic Logic

Traditional logic suffers from the monotonicity problem, which is that new axioms or premises never invalidate old theorems or conclusions. Non-monotonic logic, on the other hand, may decrease the set of theorems when new axioms are added. This kind of logic is important in modeling the beliefs of active processes which, acting with incomplete knowledge, must make and revise predictions in light of new observations.

Non-monotonic logic may be applied to many problems in AI. Its main advantage over other approaches is that it factors out problems of resource limitation and allows a representation of "consistency" to appear in any context. The notation of consistency is a central concept in non-monotonic logic.

4.1.2 Monotonic Logic

Monotonic or classical logic has no tools for describing how to revise a formal theory to deal with inconsistencies caused by new information. This is because the problems of recognizing inconsistencies and of finding and selecting among alternate revisions is very hard. In this contenxt, an axiom is equivalent to a premise, a theorem is equivalent to a conclusion, and a theory

is equivalent to a set of axioms.

Classical logic is called monotonic because the axioms of a theory are always a subset of the axioms of any extension of the theory. For example, if A and B are two theories where A **c** B, if A |- q, then B |- q, where |- denotes monotonic inferability and q is an axiom. A proof from A, then, also can serve as a proof from B.

In other words, if an axiom S is a logical consequence of any theory B, then S is a logical consequence of any theory that includes B. If B embodies a set of initial beliefs, the addition of new beliefs cannot lead to the repudiation of old consequences. Monotonic logic, therefore, cannot adapt to incomplete knowledge, changing situations, or the generation of new assumptions during the process of solving problems.

4.1.3 Basic Characteristics

Non-monotonic logic refers to first-order theories in which new axioms can invalidate old theorems. It can be obtained from classical logic using the modality M in the following inference rule

Infer Mp from the inability to infer ~p

where Mp is an operator that means "consistency" and that

forms new formulas out of existing formulas. Mp can be read as "p is consistent with the theory." As an example consider

1) (winter ^ M(snow)) ⊃ snow

2) winter

3) summer ⊃ ~ snow

in which

4) snow

can be proved.

Now, if

5) summer

is added, then 4) is inconsistent. So, by 1), 4) is not a theorem.

4.1.4 First-Order Modal Theory

A <u>first-order modal theory</u> is defined as a set of proper axioms, logical axioms, and inference rules. For all formulas p, q and r the <u>logical axioms</u> are

a) p ⊃ (q ⊃ p)

b) (p ⊃ (q ⊃ r)) ⊃ ((p ⊃ q) ⊃ (q ⊃ r))

c) (~q ⊃ ~p) ⊃ ((~q ⊃ p) ⊃ q)

d) ∀xp(x) ⊃ p(t)

e) ∀x(p ⊃ q) ⊃ (p ⊃ ∀xq)

where p(x) is a formula, t is a constant or a variable

free from x in p(x), and p(t) denotes the result of substituting t for every free occurrence of x in p(x). All other axioms are called proper or non-logical axioms. The theory with no proper axioms is called the predicate calculus. The theory consisting of axioms which are instances of a), b) and c) only, is called the sentential calculus. In addition, each theory contains all instances of various subsets of the following axioms

A1: $Lp \supset p$
(Everything provable is true.)

A2: $L(p \supset q) \supset (Lp \supset Lq)$
(Description of the Modus Ponens rule)

A3: $(\forall v)Lp \supset L(\forall v)p$
(Description of the Universal Generalization rule)

A4: $Lp \supset LLp$
(P is provable only if it is provably provable)

A5: $Mp \supset LMp$
(p is unprovable only if it is provably unprovable. This assertion is true in non-monotonic systems.)

Lp is equivalent to ~M~p and can be read "not p is inconsistent" with the theory; v is a variable; and p and q are formulas. An atomic formula is an expression P(x1,...xn) where P is a predicate symbol and x1,...xn are terms. A formula is either an atomic formula, an expression ~p, an expression p q, an expression Mp, or an expression $(\forall v)p$, where v is a variable, and p and q are formulas.

The inference rules for this system are

Modus Ponens: $p,\ p \supset q,\ \vdash q$

Universal Generalization: $p \vdash (\forall v)p$

Necessitation: $p \vdash Lp$

These axioms and inference rules are intended to be a plausible account of the logic of "consistency" which is supposed to describe provability in itself. It is essential to the concept of provability that something proven be provable, and this is what Necessitation says.

Before making the system non-monotonic, a few more definitions are needed:

a) $Th(A) = \{p: S \vdash p\}$. $Th(A)$ is a set of theorems of the modal theory with the proper axioms A. $Th(A)$ has the following properties

i) $A \subseteq Th(A)$.

ii) if $A \subseteq B$, then $Th(A) \subseteq Th(B)$.

iii) $Th(Th(A)) = Th(A)$.

where A and B are theories, i) and ii) together are called "monotonicity," and iii) represents idempotency.

b) $As_A(S) = \{Mq: q \in L \text{ and } \sim q \notin S\} - Th(A)$ where L is the set of all formulas and S is the set of formulas for the theory A, and $S \subseteq L$. $As_A(S)$ is the set of assumptions allowed by S in the modal theory with proper axioms A.

c) $NM_A(S) = Th(A \cup As_A(S))$. $NM_A(S)$ is the set of theorems of the modal theory that are derivable from the proper axioms and assumptions.

A <u>fixed point</u> of NM_A is a set, X, such that $X = NM_A(X)$. Such a fixed point is a set containing A and a large set of assumptions, $As_A(X)$, such that no assumption Mp in X is invalidated by ~p being provable from X, and every other element of X has a proof from the assumptions and axioms. In other words, a fixed point is a "belief" that does not change, that is consistent.

d) $TH(X) = L \cap (\bigcap \{X: X = NM_A(X)\})$. TH is the intersection of all fixed points of A, or the whole language L if there are no fixed points. That is, TH is the set of all formulas that are in all fixed points.

A graphical representation of the previous definitions is shown in the figure below.

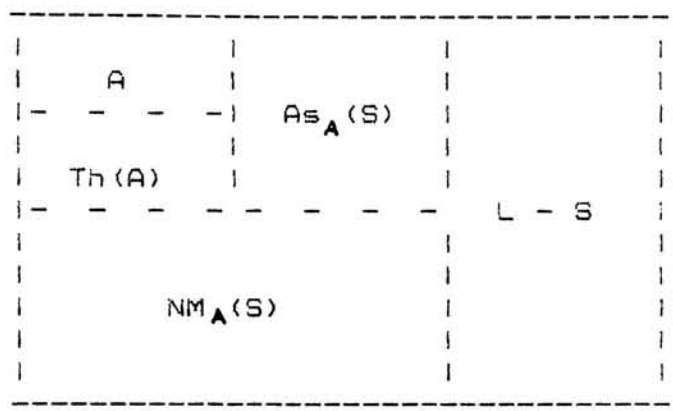


Figure 4.1.1

L is the set of all formulas, S the set of formulas for the theory A. A is the theory in discourse, Th(A), the set of theorems obtained from S by applying the previous inference rules to A. $As_A(S)$ is the set of assumptions from the theory A using the set of formulas S. $NM_A(S)$ is the set of theorems obtained from $(A \cup As_A(S))$, and TH(A) is the smallest fixed point of NM_A .

The picture expresses the following

$$A \subseteq Th(A)$$

$$(Th(A) \cup As_A(S)) \subseteq NM_A(S)$$

$$(Th(A) \cap As_A(S) = \phi$$

$$S = NM_A(S)$$

An application of the elements of the picture will be shown in detail in the next example.

Non-monotonic inference is defined as

set 1 |~ set 2,

to mean

set 2 $\subseteq$ TH(set 1)

If there are no fixed points, every formula in the language is provable, but if L is a fixed point, then it is the only fixed point. Either way, TH(A) = L, and the theory is said to be inconsistent.

As an example consider the following list of "facts"

1) All Birds can fly
2) All penguins cannot fly (Penguins c Birds).
3) Rocky is a bird.
4) Tom is a penguin.

which, when translated into the language of logic, is equivalent to the following proper axioms

1) (Vx)(Bird(x) ^ M Can-fly(x) $\supset$ Can-fly(x))
2) (Vx)(Penguin(x) $\supset$ (Bird(x) ^ ~Can-fly(x)))
3) Bird(Rocky)
4) Penguin(Tom)

This theory has one fixed point which contains the axioms listed above and the following formulas

a) Bird(Tom)

b) ~Can-fly(Tom)

c) M Can-fly(Rocky)

d) Can-fly(Rocky)

a) and b) follow by applying Modus Ponens to 2) and 4), c) follows because ~Can-fly(Rocky) is not a member of the fixed point, since M Can-fly(Rocky) is in the fixed point. Filling in Figure 4.1.1 from this example gives

A: Axioms 1), 2), 3) and 4).

Th(A): Bird(Tom) ^ ~Can-fly(Tom),
Obtained by applying Modus Ponens to 2) and 4);
~M ~Bird(Rocky), ~M ~Penguin(Tom),
obtained by Necessitation.

$As_{\blacktriangle}(S)$: - The set of assuptions is empty because the only candidate, M Can-fly, belongs to L and ~ Can-fly belongs to S. So the conditions stated by the definition are not satisfied.

$NM_{\blacktriangle}(S)$: Bird(Tom) ^ ~Can-fly(Tom).

S: Bird(x), Can-fly(x), Penguin(x), Bird(Rocky), Penguin(Tom), ~Can-fly(x).

L: Language of all formulas.

4.1.5 Semantics for First-Order Modal Theories

A <u>modal interpretation</u> is the algebraic system

$$\langle W, alt, D, V \rangle$$

where

- W is a set of possible worlds.
- alt is a reflexive relation on W called the "alternativeness relation" where w1 alt w2 means that w2 is possible with respect to w1.
- D is a domain of objects.
- V is a function, (L aug D) x W -> {0,1}, which produces the truth value of every expression in the language in every world. L aug D is the language obtained by using Trm aug D, which is the set of all terms, aug D. Trm aug D is the set of all terms obtained by adding D to the set of constants. Aug stands for "augmented."

V has the following constraints

$V(\sim p, w) = 1$ iff $V(p, w) = 0$,

$V(p \supset q, w) = 1$ iff $V(p, w) = 0$ or $V(q, w) = 1$,

$V((\forall v), p, w) = 1$ iff $V(subst(d, b, p), w) = 1$ for all d in Trm aug D

$V(Mp, w) = 1$ iff $V(p, r) = 1$ for some r such that w alt r

where p,q denote formulas of the object language, subst(d,b,p) is the result of substituting term d for

variable b in formula p, v is a variable, and w c W.

A modal model of a first order modal theory is a modal interpretation ⟨W, alt, D, V⟩ such that V(p, w) = 1 for every proper axiom, p, and every world, w, of the theory.

4.1.6 Semantics of Non-monotonic Modal theories

Simple modal models are not adequate for non-monotonic systems. Their purpose is to strengthen a logic by ruling out some of its models, changing the rules of semantic interpretation so that fewer cases qualify as making the desired formulas false.

To do this, "accidentals" of V with respect to the theory A are defined as

V acc A = { Mp: p is a statement in L, V(Mp, w) = 1, and some model V' of A exists, where V'(Mp, w) =|= 1}.

The accidentals are the possible statements that do not have to be.

The most desireable models needed are the noncommittal models, those with as many accidentals as possible. A noncommittal model V of a theory A is a modal model of A such that Mp is true in all worlds of V whenever p is true in any world of any model of A u acc(V, A). The reason

that such models are called noncommittal is that they exclude models with unfounded necessities.

Basically, a noncommittal model is one in which as many things as possible are "possible", where "possible" is a "meta-level" above the first and makes sense if the totality of models of a theory is contemplated.

4.1.7 A Proof Procedure for the Sentential Calculus

In non-monotonic logic, provability is defined without reference to proof; although a theorem will have a proof in any given fixed point of NM_A, there is no obvious way to generate, or to describe each fixed point.

One place to start solving this problem is with the sentential calculus, which defines a finite sentential theory as a theory that contains as proper axioms a finite list of variable-free formulas.

For the classical sentential modal calculus, deciding the provability of a formula, p, consists of a search for a model in which p is false, thereby creating a model with a world in which the proper axioms are true and p is false, and then exploring the consequences of the value assignments. When a disjunction is assigned a true value, it is necessary to split the investigation into branches

with a different disjunct made true in each branch. For example consider the following formula

```
(C v D) ⊃ (C ^ D)
   1     0     0
```

The 1's and Ø's under the operators label the truth values of the constituent expresssions, with 1 standing for true and Ø for false. For this statement to be false, the antecedent must be true and the consequent must be false. The complete reasoning is the following.

```
Ø)   (C v D) ⊃ (C ^ D)
        1     0     0

1)   (C v D) ⊃ (C ^ D)
      1 1     0     0

        (C v D) ⊃ (C ^ D)
         1 1     0   0 0          Closed

        (C v D) ⊃ (C ^ D)
         1 1 0   0   1 0 0        Open

2)   (C v D) ⊃ (C ^ D)
        1 1   0     0

        (C v D) ⊃ (C ^ D)
         0 1 1   0   0 0 1        Open

        (C v D) ⊃ (C ^ D)
           1 1   0     0 0        Closed
```

The table has been split twice for a total of four branches. Two branches are closed, meaning that, some formula (C in the first branch and D in the second) is labeled both 1 and Ø, true and false. A branch is closed

if it has an atomic formula labeled both 1 and Ø, otherwise it is open. Two branches are open, meaning that, there is an exhaustively consistent labeling of formulas. Thus, this formula is not valid, and by completness, it is not a theorem.

To handle non-monotonic logic, the procedure to search for a noncommittal model must be changed. When a formula, Lp, is labeled 1 in any world, it is necessary to create a new table in which p is labeled Ø. If this new table is open, p is not provable, so Lp can be labeled Ø, thereby closing its branch. If Mp is labeled Ø, a new table is created with p labeled 1. If this table is open, Mp must be labeled 1 in all other tables.

Deciding the labeling of a table for a non-monotonic case, as open or closed, may depend on the states of other table entries, so it is necessary to try all combinations applied to the table. A labeling is admissible if, after applying the previous rules, the table entries labeled closed have all their branches closed, and the table entries labeled open have at least one open branch apiece.

As an example consider

A0: MC v MD⊃ E

A1: MC⊃ ~D

A2: MD⊃ ~C

Branch/World	Theorem	Axiom	Labelings	
A0) B0/W0	E 0	M C ∨ M D 0 0 0 0 0	closed	closed
A1) B0/W0	D 1	M C 0 0	closed	open
A2) B0/W0	C 1	M D 0 0	open	closed

There are two admissible labelings. If A1 is labeled open, then A2 is labeled closed, because the open label on A1 enables MD to be labeled 1 in all other A's. Similarly, A2 may be labeled open and A1 closed. Either way, A0 is closed, because one element of MC or MD will be labeled 1. Therefore, since for both labelings A0 is closed, E is a theorem in this theory. There are two fixed points, F1 and F2. MD, ~C ∈ F1, and MC, E ∈ F2.

The proof method can be summarized as follows: to test provability of a formula p in a theory with axioms A, it is necessary to create a table with one branch containing a world in which p is labeled 0 and every axiom in A is labeled 1. The following rules then can be applied repeatedly:

a) Truth Functional Propagation rule: If the label on a formula implies labels on its subparts, then label the subparts accordingly. Such labelings apply throughout the world in which the labeling takes place.

b) Possibility Propagation rule: If Mq is labeled 1 in some world, then create a new alternative world in which q is labeled 1 and every axiom in A is also labeled 1.

c) Necessity Propagation rule: If Mq is labeled Ø in some world, label q Ø in all alternative worlds.

d) Disjunction Splitting rule: If q⊃ r is labeled 1 in some world, split the branch in which it occurs into two branches, each a copy of all the worlds, formulas, and labels of the original branch, except that one branch has q labeled Ø in that world and the other branch has r labeled 1 in that world. This rule comes from the equivalence between (q ⊃ r) and (~q v r).

e) Repetition Elimination rule: If all the labelings in a world are duplicated in a world of which it is an alternative, delete it.

f) Consistency Testing rule: If Mq is labeled Ø in some world, create a new table with one branch, containing a world in which ~q is labeled Ø and every axiom in A is labeled 1. This will be the table for ~q in A.

These rules are applied repeatedly until they do not change anything. Then every possible labeling is tried. If there is an admissible labeling in which the original table is labeled open, then the original formula is not a theorem, otherwise it is a theorem.

This proof procedure works only for the Sentential Calculus, but it gives important hints that are useful in constructing a heuristic prover for first-order theories. An application of non-monotonic logic is explained in more detail in chapter 5.

4.1.8 Conclusions

Classical logic is extended by the inclusion of a modal symbol M representing "consistency." Thus, the formula Mp is read as an assertion that p is consistent, and not as an assertion of p. This follows a well-developed tradition of modal logic, and has parallels in the use of modal 'tense' operators for axiomatizing facts involving temporal sequences. This extension requires proofs of a non-constructive fixed point character. In the present situation is used the notion of fixed point explicitly.

Many researchers have not liked the idea of studying logics of this type. They consider it a mistake to investigate the abstract notion of non-monotonic provability,

since any real program will be constrained by partial knowledge. The real issue is when does it become reasonable to stop trying to prove something and start acting on the assumption that it cannot be proved. It is important to state the fact that non-monotonic rules of inference are rules about changing theories, and not inference rules within a theory.

4.2 Reasoning by Circumscription

Circumscription is another type of non-monotonic reasoning, that handles defaults. It is a parsimonious type of reasoning and can be viewed as the assumption that all qualifications to a problem have been stated explicitly.

Circumscription is a rule of conjecture for jumping to certain conclusions, so that, those objects that can be shown to have a certain property, P, by reasoning from certain facts, A, are all the objects that satisfy P. More generally, circumscription can be used to conjecture that the tuples, ⟨x,y,..z⟩, that can be shown to satisfy a relation, P(x,y,..z), are all the tuples satisfying this relation. Circumscription is used to restrict a predicate as much as possible to be compatible with the facts that are being taken into account. After this, the desired conclusions may follow from mathematical logic.

It can be postulated, for example, that "birds can fly" unless "something" prevents them from flying. In this "something" predicate, circumscription conjectures that the only entities that can prevent the flight of birds are those whose existence follows from the facts at hand. Such facts could include, for example, birds that are penguins, ostriches, maltese falcons, dead, etc. It then can be reasoned that if Rocky is a bird and he is not

a member of the predicate "prevented from flying," then Rocky can fly. The correctness of this reasoning depends on having "taken into account" all relevant facts when the circumscription was made.

An important characteristic to consider is the fact that it is not possible to get a circumscriptive reasoning capability by adding sentences to an axiomatization or by adding an ordinary rule of inference to mathematical logic. This is because the systems of mathematical logic are monotonic. Thus, non-monotonic reasoning is needed.

Circumscription is very useful when it is necessary to avoid excessive qualification, because it allows conjectures that no relevant objects exist in certain categories except those whose existence follows from the statement of the problem and common sense knowledge.

When the first-order logic of a problem is circumscribed with the common sense facts about a boat crossing a river, for example, it is possible to conclude that there is no bridge, or helicopter, or other means at hand to cross the river. It also may be concluded that all the requirements needed to use the boat exist, like oars, no leaks, etc. Thus, the reasoning is the following. It is a part of common knowledge that a boat can be used to cross a river unless there is something wrong with the

boat or something else prevents its use. If the facts do not require that there be something that prevents crossing the river, circumscription will generate the conjecture that there is not something that prevents crossing the river. That is, using circumscription requires that common sense knowledge be expressed in a form that says a boat can be used to cross rivers unless there is something that prevents using a boat.

4.2.1 Types of Circumscription

Two types of circumscription have been developed, an earlier form called domain circumscription, and the form that will be discussed here, called predicate circumscription. Briefly, Domain circumscription conjectures that the known entities are all the entities that exist. Predicate circumscription assumes that entities satisfy a given predicate only if they have to on the basis of a collection of facts.

4.2.1.1 Domain Circumscription

Domain circumscription is also known as minimal inference, and minimal inference has a semantic counterpart called minimal entailment. A sentence q is minimally entailed by an axiom A, written $A \models_m q$, if q is true in

all minimal models of A, where one model is considered less than another if they agree in common elements, but the domain of the larger one contains elements not included in the domain of the smaller one.

The domain circumscription of A may be expressed as

$$Axiom(\emptyset) \wedge A^{\emptyset} \supset \forall x\ \emptyset(x).$$

where Axiom ($\emptyset$) is the conjunction of sentences $\emptyset(a)$ for each constant a and sentences $\forall x(\emptyset(x) \supset \emptyset(f(x)))$ for each function f and the corresponding sentences for functions of higher arities, $A^{\emptyset}$ is the relativization of A with respect to $\emptyset$, and is formed by replacing each universal quantifier $(\forall x)$ in A by $(\forall x \emptyset(x) \supset)$, and each existential quantifier $(\exists x)$ by $(\exists x \emptyset(x) \wedge)$.

4.2.1.2 Predicate Circumscription

As mentioned above, predicate circumscription says that a tuple X satisfies the predicate P only if it has to. The formalization of predicate circumscription is the following. Let A be a sentence of first-order logic containing a predicate symbol, P(x1,....xn), which will be written as P(X). $A(\emptyset)$ is the result of replacing all occurrences of P in A by the predicate expression $\emptyset$. Thus, the circumscription of P in A(P) is the schema

$$A(\emptyset) \wedge \forall X(\emptyset(X) \supset P(X)) \supset \forall X(P(X) \supset \emptyset(X)).$$

This means that the only tuples (X) that satisfy P are those that have to, assuming the sentence A. The first conjunct A(∅) expresses the assumption that ∅ satisfies the conditions satisfied by P. The second conjunct $(\forall x)(\emptyset(X) \supset P(X))$ expresses the assumption that the entities satisfying ∅ are a subset of those that satisfy P. The conclusion of this implication asserts the converse of the second conjunct, telling that in this case, ∅ and P must coincide.

$A \mid\!\sim_{P} q$ means that the sentence q can be obtained by deduction from the result of circumscribing P in A. $\mid\!\sim_{P}$ is a non-monotonic form of inference called circumscriptive inference.

It turns out that domain circumscription can be subsumed under predicate circumscription by relativizing A with respect to a new place predicate, for example 'all', then circumscribing 'all' in A ^ Axiom(all) getting:

$$Axiom(\emptyset) \wedge A^{All} \wedge \forall x(\emptyset(x) \supset all(x)) \supset \forall x(all(x) \supset \emptyset(x)).$$

Circumscription is not deduction in disguise because every form of deduction has two properties that circumscription lacks, transitivity and monotonicity. The first property says that if a |- b and b |- c, then a |- c. The second property says that if A |- a and A $\subset$ B, then B |- a for deduction, where A and B are sets of sen-

tences. Circumscription should be non-monotonic since it is the conjecture that the ways it is known of generating A's are all the ways there are. The next two subsections describe how to circumscribe conjunctive and disjunctive sentences respectively.

4.2.2 Circumscription of the Conjunction

As an example consider the sentence

$$\text{is-color } B \wedge \text{is-color } W \wedge \text{is-color } G$$

meaning that B, W and G are colors. The circumscription of this sentence gives

$$\phi(B) \wedge \phi(W) \wedge \phi(G) \wedge \Psi x(\phi(x) \supset \text{is-color } x) \supset \Psi x(\text{is-color } x \supset \phi(x))$$

Assumming that

$$\phi(x) == (x = B \vee x = W \vee x = G),$$

and substituting this sentence into the previous one, produces

$$\Psi x(\text{is-color } x \supset (x = B \vee x = W \vee x = G))$$

which says that the only colors are B, W, and G, which are the colors that the first sentence requires.

Although this example is trivial because the original sentence to be circumscribed provides no way of generating new colors from old ones, it shows that circumscriptive inference is non-monotonic since if is-color A is adjoined to the original sentence, it will no longer be able to infer the resulting sentence.

4.2.3 Circumscription of the Disjunction

As an example consider

is-color W v is-color B

The circumscription of this sentence gives

$(\emptyset(W) \vee \emptyset(B)) \wedge \forall x(\emptyset(x) \supset \text{is-color } x) \supset \forall x(\text{is-color } x \supset \emptyset(x))$

Substituting successively

$\emptyset(x) == (x = W)$ and $O(x) == (x = B)$

produces respectively

$(W = W \vee W = B) \wedge \forall x(x = W \supset \text{is-color } x) \supset \forall x(\text{is-color } x \supset x = W)$

$(B = W \vee B = B) \wedge \forall x(x = B \supset \text{is-color } x) \supset \forall x(\text{is-color } x \supset x = B)$

Simplifying both terms gives

$$\text{is-color } W \supset \forall x(\text{is-color } x \supset x = W)$$

$$\text{is-color } B \supset \forall x(\text{is-color } x \supset x = B)$$

which, when joined with is-color W v is-color B, yields

$$\forall x(\text{is-color } x \supset x = W) \vee \forall x(\text{is-color } x \supset x = B).$$

This last sentence asserts that either W is the only color or B is the only color.

4.2.4 Conclusions

Circumscription is not a non-monotonic logic, but rather a form of non-monotonic reasoning augmenting ordinary first-order logic which can be used in other formalisms besides first-order logic. Circumscription does not affect irrelevant facts taken into account. If such facts do not contain the predicate symbol being circumscribed, then they will appear as unchanged conjuncts. Circumscription also may express heuristic situations better than axioms do.

Issues of proof are not discussed in detail, but the term of minimal entailment shows a notion of minimality which plays the same role as that of the fixed point in non-monotonic logic.

The main differences between circumscription and non-monotonic logic are the following. First, circumscription is concerned with minimal models, and non-monotonic logic with arbitrary models. Second, the reasoning of non-monotonic logic involves models directly, while the syntactic formulation of circumscription uses

4.3 Frame System

According to Bartlett [1932], psychological evidence shows that people use a large and well-coordinated body of knowledge from previous experiences to interpret new situations in their everyday cognitive activity. Representing knowledge about objects and events to specific situations is the purpose of Frames.

Frames were originally proposed by Marvin Minsky [1975] as a basis for understanding visual perception and natural-language dialogues, as a means of reasoning, and as a model for other complex behaviors. The basic idea of frames comes from the "schema" of Bartlett and the "paradigm" of Kuhn.

Minsky introduced the term "frame" to unify and denote a loose collection of related ideas about knowledge representation. Frames were put forward as a set of ideas for the design of a formal language for expressing knowledge and reasoning. Frame systems, then, are an alternative to semantic networks or predicate calculus, as a knowledge representation scheme.

There are two other interpretations for frames, however, namely metaphysical and heuristic representations. The "metaphysical" representation indicates that to use frames is to make certain kinds of assumptions about what

entities will exist in the world being described. In other words, to use frames is to assume that a specific kind of knowledge is to be represented by them. According to Minsky, visual perception may be facilitated by the storage of explicit two-dimensional view prototypes and rotational transformations among them. This expresses the idea of what sorts of things a program needs to know, rather than how those things can be represented.

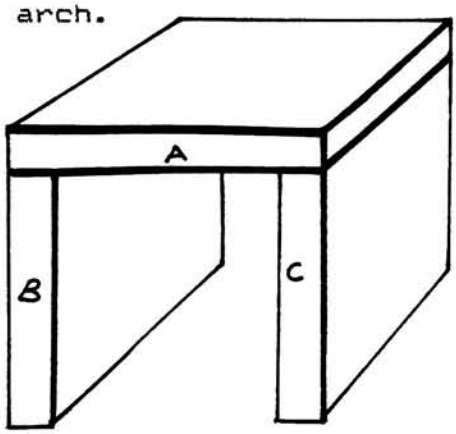
The "heuristic" or "implementation" interpretation, dictates that frames are a computational device for organizing stored representations in computer memory, and also perhaps a device for organizing the process of retrieval and inference that manipulate these stored representations. Reference can be made to the computational ease that pointers offer in a frame system to allow jumping from one frame to another, facilitating retrieval operations.

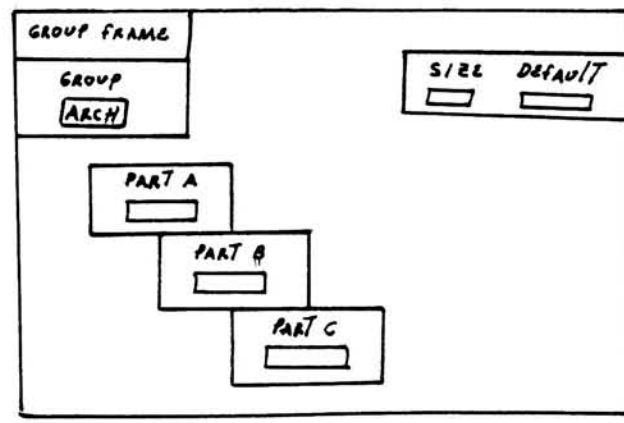
4.3.1 The Meaning of Frames

According to Minsky, when a new situation is encountered, a structure from memory is selected. This structure is what constitutes a frame, and is a remembered framework to be adapted to fit reality by changing details as necessary.

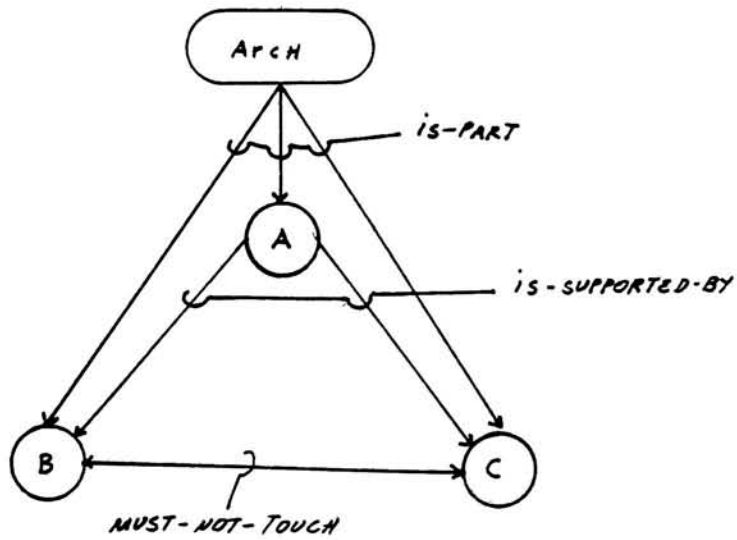
A frame is a data structure or an expression intended to represent objects or stereotypical situations. It consists of a collection of "slots" that describe aspects of the representation. The slots can be filled with other expressions or fillers which may themselves be frames, or simple names or identifiers which may be somehow associated with other frames. Slots cannot be filled by a slot-filler relation, however, because the trees formed by filling slots with frames recursively would be infinitely deep.

Associated with each slot are several kinds of information. Some of this information may be a set of conditions that must be met by any filler. Each slot may also be filled with a default value. Procedural information may also be associated to particular slots. That is, some of this information is about how to use the frame, some is about what is expected to happen, some is about what to do if expectations are not confirmed, etc. As an example consider the following simple representation of an arch.





A frame also may be thought of as a network of nodes and relations. The "top levels" of a frame are fixed and represent things that are always true about the supposed situation. The lower levels have many terminals called slots that must be filled by specific instances. Simple conditions are specified by markers that might require a terminal assignment to be a person, an object of sufficient value, or a pointer to a sub-frame of a certain type. For example, continuing with the previous case of an arch.



Frames provide a structure within which new data are interpreted in terms of concepts acquired through previous experience. The organization of this knowledge facilitates expectation-driven processing, looking for things that are expected based on the context in which it is

thought to be. The representational mechanism that makes possible this kind of reasoning is the slot, the place where knowledge fits within the larger context created by the frame. As another example consider.

House Frame:

Kind of: building

Number of rooms: an integer (default= 4)

Number of floors: an integer (default= 1)

Address: any country (default= U.S.)

Style of Construction: gothic, Spanish, etc.

John's House Frame:

Kind of: house

Number of rooms: 5

Number of floors: 2

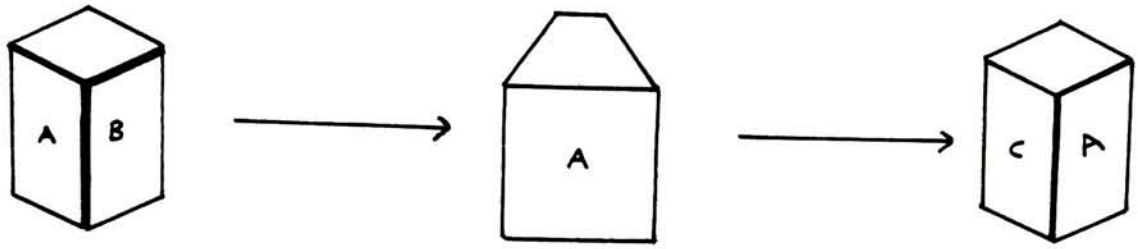
Address: Miami, Fl.

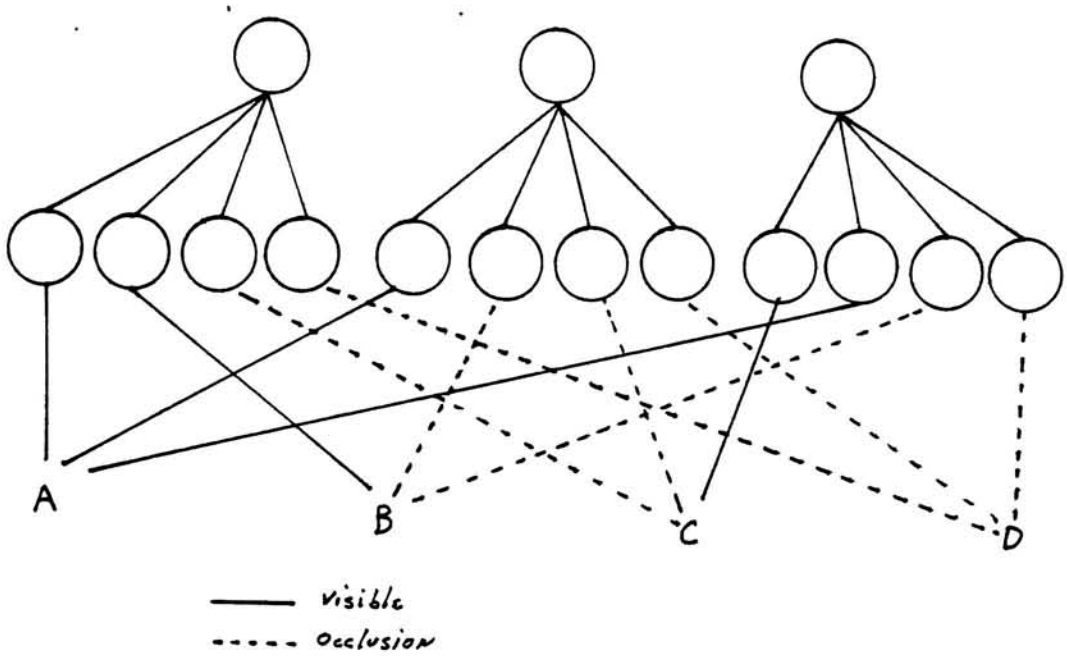
Style of construction: mediterranean

Collections of related frames are linked together into frame systems. The effects of important actions are mirrored by transformations between the frames of the system.

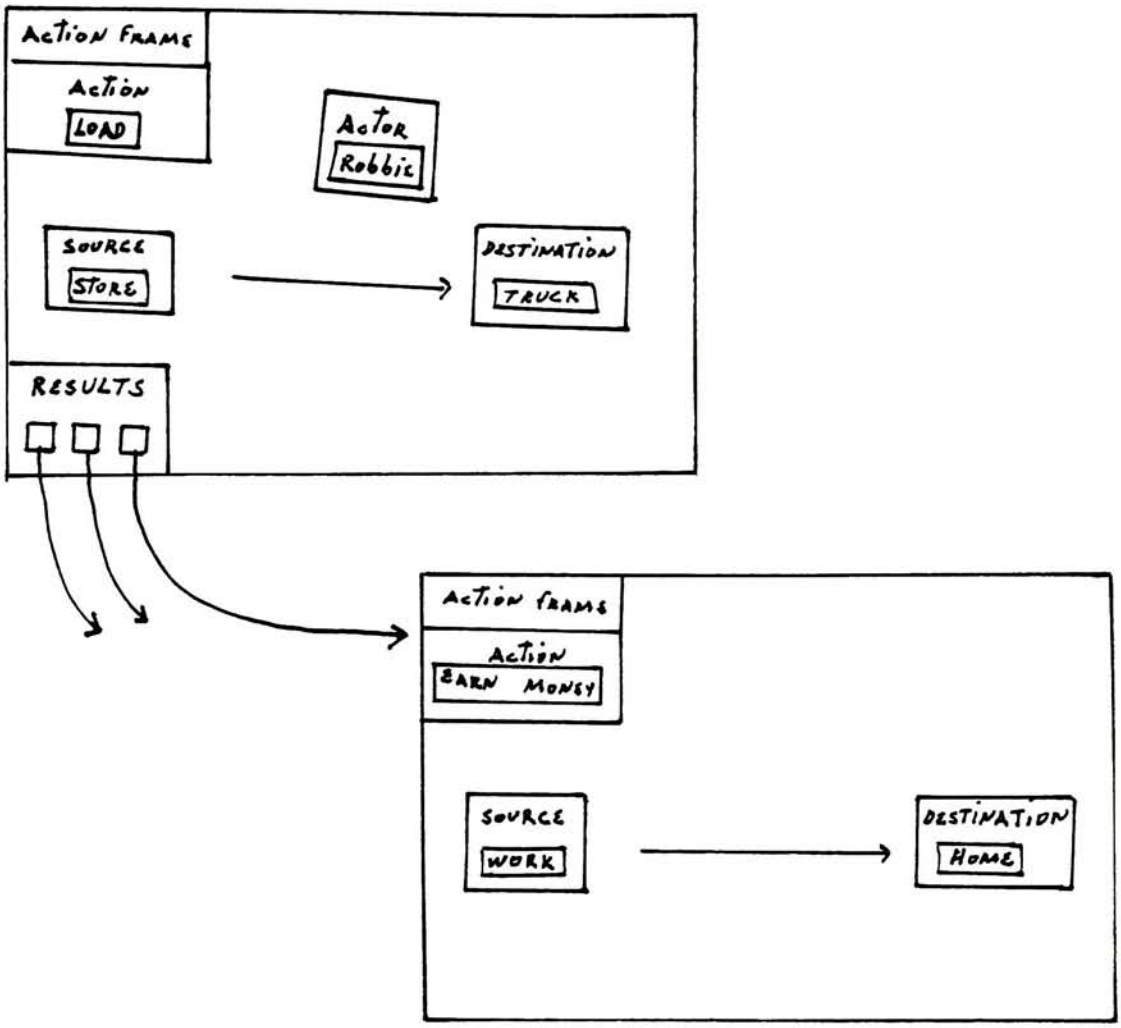
For visual scene analysis, the different frames of a system describe the scene from different points of view, and the transformations between one frame and another represent the effects of moving from place to place.

For example, consider the rotation of a cube.

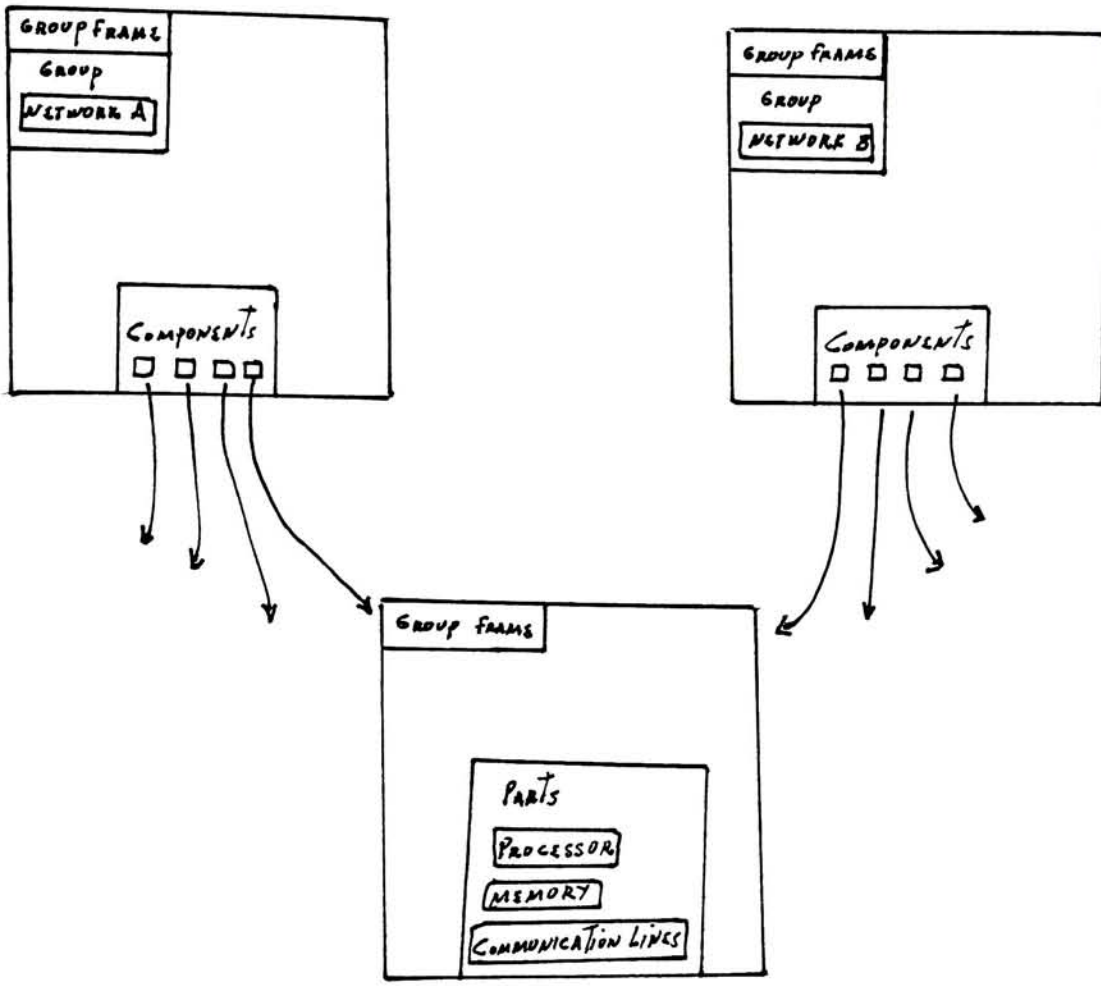




For non-visual kinds of frames, the transformations between frames can represent actions, cause-effect relations, or changes in conceptual points of view. For example



Since different frames of a system may share the same terminal slots, it is possible to coordinate information gathered from different points of view. For example consider the following frame system.



The power of the theory hinges on the inclusion of expectations and other kinds of presumptions. Frame slots are normally filled with default assignments which are attached loosely, so that they can be easily displaced by new items that better fit current situations. Thus, they also serve as variables or as special cases for reasoning.

Frame systems are linked by an information retrieval network. When a proposed frame cannot fit reality, this network provides a replacement frame. These inter-frame

structures make possible other ways to represent knowledge about facts, analogies, and other information useful in understanding and reasoning. For example, if a simple chair frame does not fit an object to be represented, frames of a reclining chair, arm chair, desk chair, etc. can be tried. In all cases, the several hypotheses that may be generated when there is a need to change frames should be weighed according to the cost involved in assuming them.

After a frame is proposed to represent a situation, a matching process tries to assign values to each frame's slots, that are consistent with the markers at each place. The matching place is controlled partially by information associated with the frame, and partially by knowledge about the system's current goals. The matching process is discussed in the following section.

4.3.2 Frame Inference

Rules of inference and rules of equivalence, in classical logic, indicate the steps to follow to prove the validity of an argument. However, there is no rule that says which of these rules to use and when to use them. Inference in frames is not as complicated as it is in logic, because a relatively large amount of information

may be represented by frames. There are also no rules to follow, and the inferences are quite evident.

One inference rule is instantiation. Given a frame representing a concept, generate an instance of the concept by filling in its slots.

Another kind of inference is 'criterial' inference, suggested by Minsky [1975] and realized explicitly in some applications of frames. If all the slots of a frame are filled, this rule enables the inference that an appropriate instance of the concept does indeed exist. For example, if an entity has four tires, engine, seats, and all the other slots of the frame called "car", then it must be a car. An example of the application of this rule can be found in perceptual reasoning. Suppose a program is used to recognize solids with occluded details and shadows. When some shapes are recognized, the program may infer through frame reasoning, which are solids.

A third form of frame reasoning, often called matching, has been proposed by Bobrow and Winograd [1977]. If an instance of a concept exists, matching might be used to determine whether it can be plausibly regarded as also being an instance of another concept. For example, is it possible to view John Smith as an accountant, where John Smith is an instance of "Man-frame", and "Accountant" is

another frame? This question can be rephrased as "Can an instance of the Accountant frame be found which matches John Smith?" The sense of matching here cannot mean a simple syntactic unification, but must rest, if possible, on some assumptions about the domain in which the frames in question express information.

Frame inference has the merit of putting frames, frame-instances and matching assumptions into a common language with a clear extensional semantics that makes it quite clear what all these structures mean. The usual inference rules are clearly correct and are sufficient to account for most of the required deductive properties of frames.

4.3.3 Other Representations

One concept behind frames, which seems to have arisen from the preceding section, is the idea of seeing one thing as though it were another, or of specifying an object by comparing it to a known prototype, noting the similarities and differences.

The simplest interpretation of this idea is the filling in of new details. Thus, to say John Smith is a man tells something about him, but to say he is a "philosopher," tells more. The "philosopher" frame would

presumably have slots which do not appear in the "man" frame, but it would also have a slot to be filled by the man instance for John Smith, or refer to him in some other way, thereby accessing all his slots.

Another more correct interpretation is that a frame represents a particular way of looking at an entity. For example, a man may also be a scientist, and neither of these is a further specification of the other because each frame can have slots not possessed by the other. Several properties may be true of a single entity, and that entity may be both a man and a scientist.

There is an apparent difficulty, however, in that a single thing may have apparently contradictory properties, when seen from different points of view. Thus, a man viewed as a businessman may be very adept with his business's finances, but may be a fool with his family's finances. The different points of view insulate the parts of the potential contradiction from one another.

There are several possible interpretations for this. One interpretation is related to the assertion of different properties in two frames; "skillful for business at work" and "skillful for business at home" are just different notions. Another interpretation is that the frames somehow encode an extra parameter; time or place, for example.

There is a third interpretation, however, which is to view frame systems as a metaphor or analogy, without asserting their truth. For example, a man may be looked at as a horse if the speed of his running is being considered. Such a caricature may be useful in reasoning without being taken to be absolutely true. What it really means to look at a man as a horse, is that certain properties or characteristics of a man are preserved under the mapping onto a horse which is defined by the analogy. Since horses run very fast, a man with "this" quality, under the mapping which defines the analogy, may be plausibly regarded as horse-like. The intention of such a caricature, then, is that some of the properties of the caricature will be transferred to the caricaturee. The analogy is correct, or plausible, when these transferred properties do, in fact, hold for the thing being caricatured, for instance, when the man runs fast, has long legs, etc.

In order to express caricatures in logic, there is a need to define a system for the translation of vocabulary. This seems to require some syntactic machinery which logic does not provide, namely the ability to substitute one relation symbol for another in an assertion. This "analogy mapping" was defined by R. Kling [1971] in the

following way: Let $\emptyset$ be the syntactic mapping "out" of the analogy. If $\lambda x \Psi(x)$ is the defining conjunction of the frame Horse-likeness, then

$$\text{Horse-like}(x) \quad == \Psi(x).$$

where Ψ may contain several existentially bound variables. It then may be said that Horse-like(John) is true only when $\emptyset(\Psi)$ holds for John. In other words, the asserted properties are actually true for John when the relation names are altered according to the syntactic mapping $\emptyset$. Therefore, a caricature frame needs to contain a specification of how its vocabulary should change to fit reality. With this modification, the rest of the reasoning involved is first-order.

4.3.4 Defaults

One aspect of frame reasoning which is often considered to lie outside of logic is the idea of a default value, a value which is taken to be a slot filler in the absence of explicit information to the contrary.

Defaults fall outside first-order reasoning in the sense that they cannot express the assumption of the default value as a simple first-order consequence of there being no contrary information.

As an example, consider a Summer-day frame, one of whose slots is Status, with possible values {sunny, rainy, cloudy}, and a default of sunny. That is, in the absence of contrary information, it is assumed that a summer day is sunny. If the day is rainy, though, the conclusion is that, contrary to what it was expected, the correct filler for the status slot is rainy. The state of knowledge has changed. The previous assumption was reasonable according to the state of knowledge at that time. Now, if * represents the state of knowledge, then Status(summer-day) = sunny was a reasonable inference from *; that is, * |- status(summer-day) = sunny. But when it is known that the day is rainy, there is a new state of knowledge *1, and *1 |- status(summer-day) = rainy.

In order for this to be deductively possible, it must be that *1 may be arrived at from * not only by adding new beliefs, but also by removing some old ones. Moreover, there is no contradiction between the earlier belief and the present one. There was no irrationality or madness, only misinformation.

As the example shows, default assumptions involve an implicit reference to the whole state of knowledge at the time the assumption was generated. Any event which changes the state of knowledge is liable, therefore, to upset these assumptions. If these references to knowledge

states are represented explicitly, then "default" reasoning can be expressed easily in logic.

Experience with manipulating collections of beliefs should dispel the feeling that all the ways new knowledge can affect previously held beliefs can be predicted. There is no theory for this process, but any mechanism, whether expressed in frames or otherwise, which makes strong assumptions on weak evidence, needs to have some method for "unpicking" these assumptions when things go wrong, or, equivalently, some method for controlling the propagation of inferences from assumptions.

To summarize, a close analysis of what defaults mean shows that they are connected with the idea of observations; they allow additions of fresh knowledge into a database. Their role in inference - the drawing of consequences of assumptions- is expressible in logic, but their interaction with observations requires that the role of the state of the system's own knowledge be made explicit. This requires not a new logic, but an unusual ontology, and some new primitive relations. It is necessary to be able to talk about the system itself, in its own language, and to involve assumptions about itself in its own process of reasoning.

4.3.5 Conclusions

Frames contain information about many aspects of the objects or situations they describe. This information can be used as though it had been explicitly observed. They also contain attributes that must be true of objects that will be used to fill individual slots.

Frames also can be viewed as complex semantic nets with the difference that they typically have a great deal of internal structure designed to make them useful in specific kinds of problem solving tasks.

The real force of the frame idea is not at the representational level at all, but rather at the implementation level, that is, a suggestion about how to organize large amounts of knowledge needed to perform cognitive tasks. This constitutes a recent attempt by AI researchers but much of this work is conjecture, and there are differences of opinion among them.

5.0 A Truth Maintenance System

To choose their actions, reasoning programs must be able to make assumptions and subsequently revise their beliefs when discoveries contradict these assumptions. The Truth Maintenance System (TMS) is a problem solving subsystem for performing these functions by recording and maintaining the reasons for program beliefs, and by recording knowledge about deductions. Such recorded reasons are useful in constructing explanations of program actions and in guiding the course of action of a problem solver.

TMS of Doyle [1970] is an implemented system that supports non-monotonic reasoning. It serves as a truth maintaining subsystem available to other reasoning or problem solving programs, thus its role is to maintain consistency among the statements generated by another program, not to generate inferences by itself. When an inconsistency is detected, it evokes its own reasoning mechanism, dependency-directed backtracking, to resolve the inconsistency by altering a minimal set of beliefs.

Formally, TMS has two basic responsabilities. First, to maintain a data base of proofs of formulas generated by an independent problem solver program. Its goal is to avoid the presence of both ~q and Mq in the data base

simultaneously. Second, to detect inconsistencies adding axioms to the theory in order to eliminate them.

The logic underlying TMS is explained in more detail in 4.1.1

5.1 Functional Description

TMS records and maintains arguments for potential program beliefs which are distinguished from currently held program beliefs. It works with two different data structures, nodes, which represent beliefs, and justifications, which represent reasons for beliefs.

A node is a statement or a rule, and is in one of two states which represents its "support status."

IN: If it has at least one valid justification

OUT: If it does not have any valid justifications. Attached to each node is a list of justifications, each of which represents one way of establishing the validity of that node.

Both IN and OUT nodes are kept in the system, IN nodes because they represent current beliefs, and OUT nodes, so that if the available information changes, all the reasoning that was required to create them will not have to be repeated.

The basic actions of TMS are

a) to create new nodes, to which the problem solving program can attach justifications, (statements of belief). TMS leaves this manipulation to the problem solving program.

b) to add or to retract new justifications for a node, which represents steps in an argument for the belief represented by the node. An argument step usually represents the application of some rule or procedure in the problem solving program.

c) to mark a node as a contradiction, which represents the inconsistency of any set of beliefs that enter into an argument for the node.

The problem solving program indicates the inconsistency of the beliefs represented by certain currently believed nodes. This is done by using these nodes in an argument for a new node, and then marking the new node as a contradiction. When this happens, dependency-directed backtracking analyzes the argument of the contradiction node to locate the assumptions occurring in the argument. It then makes a record of the inconsistency of this set of assumptions, and uses this record to change one of the

assumptions. After this, the contradiction node is no longer believed. An assumption is a node whose supporting justification has a nonempty list.

5.2 Justifications

A node may have several justifications, each of which represents a different reason for believing the node. The justifications comprise the node's justification set. A node, then, is believed if and only if at least one of its justifications is valid.

There are two kinds of justifications in TMS, reflecting two ways that the validity of a node can depend on the validity of others nodes.

Support-List justification: (SL (inlist) (outlist))
Conditional-Proof justification: (CP consequent (inny-pothesis) (outhypothesis))

SL-justifications are the most common. An SL-justification is valid if and only if each node in its inlist is IN (the set of beliefs), and each node in its outlist is OUT (the set of beliefs), that is, an SL-justification is a valid reason for belief if and only if each of the nodes in the inlist is believed and each of the nodes in the outlist is not believed. A CP-justification is valid if the consequent node is IN

whenever each node of the inhypothesis is IN and each node of the outhypothesis is OUT. CP-justifications represent hypothetical arguments and are more difficult to handle than SL-justifications. TMS deals with CP-justifications by attempting to convert them to equivalent SL-justifications. TMS can easily determine the validity of a CP-justification only when the justification's consequent and inhypotheses are IN and outhypotheses are OUT.

An SL-justification says that the justified node depends on each node in a set of other nodes and adds the dependencies of the referred nodes to the supporting lists. A CP-justification says that the node it justifies depends on the validity of a certain hypothetical argument and subtracts the dependencies of some nodes from the dependencies of others. SL-justifications with nonempty inlists and empty outlists represent normal deductions.

As an example consider

1) It is midnight (SL ( ) ( ))

2) The stars shine (SL (1) ( ))

A node like 1) with an empty inlist and outlist is referred to as a premise because it does not depend on the current belief, or lack of belief, in any other nodes. The inlist of the SL-justification of 2) contains node 1,

indicating that the chain of reasoning that led to the conclusion that node 2) is to be believed depends on a current belief in node 1.

The reasoning of TMS appears very similar to that of a predicate logic system except that it can handle the retraction of premises and make appropriate changes to the rest of the database. Now, if the outlist of an SL-justification is not empty, it can also handle default reasoning as is shown here

1) It is midnight (SL ( ) ( ))
2) The stars shine (SL (1) (3))
3) The sun shines

Node 2) is IN if node 1) is IN and if node 3) is OUT. If, at some future time, evidence appears that 3) is valid providing justification for it, TMS will make node 2) OUT since it no longer has a valid justification. Nodes such as 2) are called assumptions because they are IN on an SL-justification with a nonempty outlist.

TMS does not create justifications itself. The justification for node 2) comes from the domain knowledge in the problem solving program.

5.3 The Truth Maintenance Process

The truth maintenance process updates the current set of beliefs when the user adds to or subtracts from the justification-set of a node. Since retracting justifications does not present any real problems, only additions to the set of beliefs will be covered here.

The truth maintenance process starts when a new justification is added to a node. Minor updating is required if the justification is invalid or if it is valid but the node is already IN. If the justification is valid and the node is OUT, the node and its repercutions must be updated. TMS makes a list containing the justification and its repercutions, and marks each of these nodes to indicate that they have not been given well founded support.

TMS uses a set of "supporting nodes" to determine the support statuses of the nodes. An "affected consequence" is a consequence of a node which contains that node in its set of supporting nodes. A "repercution" is the transitive closure of the affected consequences of the node.

TMS then examines the justifications of these nodes to see if any are valid on the basis of unmarked nodes. If it finds any valid nodes, they are made IN. Nodes are made OUT if all their justifications are invalid.

The marked consequences of the nodes then are examined to see if they too can be given well founded support (non-circular proofs). Sometimes well founded support statuses are found for all nodes. At other times, some nodes remain marked due to circularities. A "consequence" is a node which mentions the node in one of the justifications in its justification set.

After the consequences have been examined, TMS initiates a constraint-relaxation process which assigns support statuses to the remaining nodes. Finally, TMS checks for contradictions and CP-justifications, performs dependency-directed backtracking and CP-justifications process if necessary, and signals the user program of the changes in support statuses of the nodes involved in truth maintenance.

There are two types of inconsistencies handled by TMS. The first type deals with both Mp and ~p being IN, and the second type deals with both p and ~p being IN.

In the first case, when a new justification is discovered for some formula which then invalidates some current assumptions, TMS must reexamine the current labeling to find a new labeling consonant with the enlarged set of justifications. This process takes on the appearance of a relaxation procedure for finding an

acceptable labeling, and then determining non-circular proofs for all valid formulas. The second type of inconsistency requires somewhat different process. TMS must traverse justifications, seeking the assumptions underlying the conflict formulas. To resolve the inconsistency of these assumptions, TMS converts the problem to one of the first type by producing a new justification for the denial of one of the assumptions in terms of the other assumptions. TMS cannot rule out an assumption Mp by deriving ~Mp, but instead it produces a derivation of ~p. This process is called dependency-directed backtracking.

In non-monotonic logic there is also another kind of inconsistency, that due to there being no fixed point at all. TMS can loop forever when presented a theory of this sort. Normally this does not happen.

5.4 Dependency-Directed Backtracking

When TMS makes a contradiction node IN, it invokes dependency-directed backtracking to find and remove at least one of the current assumptions in order to make the contradiction node OUT. The steps of this process are

a) Search the maximal assumptions: Trace through the foundations of the contradiction node C to find the set S= {A1,...An}, that contains an assumption A if and only if A

is in C's foundations and there is no other assumption B in the foundations of C such that A is in the foundations of B.

b) Summarize the cause of inconsistency: If no previous backtracking attempt on C discovered S to be the set of maximal assumptions, create a new node, NG, called 'no-good', to represent the inconsistency of S. If S was encountered earlier as a no-good set of a contradiction, use the previously created no-good node.

c) Select or reject a culprit: Select some Ai, the culprit, from S. Let D1,..Dk be the OUT nodes in the outlist of Ai's supporting-justifications. Select Dj from this set and justify it with

(SL (NG A1,...Ai-1,Ai+1,...An) (D1,...Dj-1,Dj+1...Dk))

This step recalls reduction ad absurdum if the D nodes are taken as denials of the selected assumption.

d) Repeat if necessary: If TMS finds other arguments so that the contradiction node C remains IN after the addition of the new justification from Dj, repeat the process. Finally, if the contradiction becomes OUT, then halt, or if no assumption can be found in C's foundations, notify the problem solving program of an unanalyzable contradiction and halt.

5.5 An Example

As an example consider a program scheduling a meeting, to be held at noon in either room 101 or 106

node 1: time(meeting) = 1200 (SL( )(2))

node 2: time(meeting) = 1200

node 3: room(meeting) = 101 (SL( )(4))

node 4: room(meeting) = 106

TMS makes nodes 1 and 3 IN and nodes 2 and 4 OUT. Supposing that new knowledge is acquired, and that it is not possible to hold the meeting at that time, in that room:

node 5: contradiction (SL(1,3)( ))

The dependency-directed backtracking system traces the foundations of node 5 to find two assumptions, nodes 1 and 3, both maximal:

node 6: no-good 1,3 (CP 5 (1,3)( ))

node 4: room(meeting) = 106 (SL(6,1)( ))

The backtracker justifies node 6 which means ~ (time(meeting) = 1200 ^ room(meeting)= 106). It arbitrarily selects node 3 as a culprit and justifies node 3's only OUT antecedent, node 4. Then TMS makes nodes 1, 4,

and 6 IN, and nodes 2, 3 and 5 OUT. Node 6 has a CP-justification equivalent to a premise SL-justification, since node 5 depends directly on the two assumptions, nodes 1 and 3, without any additional intervening nodes.

A further rule determines that room 106 will not be used, and creates another contradiction node to force a different choice of room:

node 7:	contradiction	(SL (4) ( ))
node 8:	no-good 1	(CP 7 (1) ( ))
node 2:	time(meeting) = 1200	(SL (8) ( ))

Tracing backwards from node 7 through nodes 4, 6 and 1, the backtracker finds that the contradiction depends on only one assumption, node 1. It creates the no-good node 8, justifies it with a CP-justification. The loss of belief in node 1 carried node 5 away too. TMS makes nodes 2, 3, 5 and 8 IN, and nodes 1, 4, 6 and 7 OUT.

5.6 Conclusions

TMS solves part of the belief problem, and provides a mechanism for making non-monotonic assumptions. The non-monotonic capabilities of TMS can be viewed as capabilities for dealing with incomplete knowledge.

TMS revises opinions rather than beliefs. Choosing what to "believe" in TMS, involves making judgements. A single new piece of information may lead to considerable changes in the set of opinions, where new beliefs change old ones only slightly.

TMS is a formal system of constraints among objects representing theorems of a first-order theory. The inference rules of the logic can be used to generate new constraints having the form "if x is believed, then y must be believed." At any moment, TMS will have a finite set of theorems and constraints. It then can implement the non-monotonic inference "if p is consistent, then q" since consistency is judged with respect to the current set of theorems, not with respect to everything provable from it. The addition of a new constraint may make a previous consistent judgement invalid, thus demanding readjustment of the entire set of beliefs. The goal in building TMS is to make this process consistent and computationally feasible. The algorithms for testing consistency are complex because they include non-monotonic rules as well as first-order theorems.

6.0 Conclusions

First-order logic lacks an explicit scheme to index into relevant knowledge. It is unable to handle changing or incomplete knoledge and has limitations in deductive inference. However, logic has a formal precision and a clear interpretability.

Non-monotonic logic and circumscription offer a better way to deal with inconsistencies caused by adquisition of new knowledge. The real problem in this system is the complexity found in the design and in the implementation of such systems.

None of the two previous non-monotonic reasoning methods deals with the problem of revision of belief. This problem is resolved in TMS, in which each statement has an associated set of justifications that are used to determine the current set of 'beliefs.' When it changes, these justifications are examined to update the data base of 'beliefs.'

The word frame has been applied to a variety of slot-and-filler representation structures, mostly following the ideas presented by M. Minsky. A frame is a general-purpose structure to represent things and stereotyped situations. A frame also can be viewed as a complex semantic net which has a great deal of internal structure

designed to make it useful in representing knowledge and problem-solving tasks. One of its outstandings characteristics is that it offers many relevant facts about the representation.

There are, however, other "knowledge-structures" such as 'scripts' which are not discussed here because they are special-purpose structures that describe a stereotyped sequence of events in a particular context. Other knowledge-structures is 'semantic nets' and 'conceptual dependency'.

All these structures are very useful because they provide a way to represent information and means or reasoning.

After reviewing these approaches there is no doubt that with them certain kinds of common sense reasoning are achievable, mainly when related with the generation of assumptions, cause-effect relations or movement representations.

Appendix

Minimization of Boolean Functions

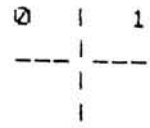
It is understood by minimization of a Boolean function, the process of finding the simpliest equivalent expression to the function, that is, the simpliest sum of products.

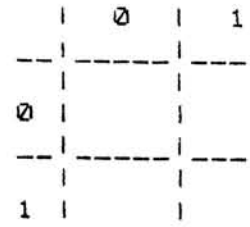
Simplification can be done algebraically utilizing the absorption laws and similar relations. But problems arise with the doubt of having not taken into account all possible ways of applying the simplification rules. To avoid this, two methods are discussed, which were designed by Karnaugh and Quine respectively.

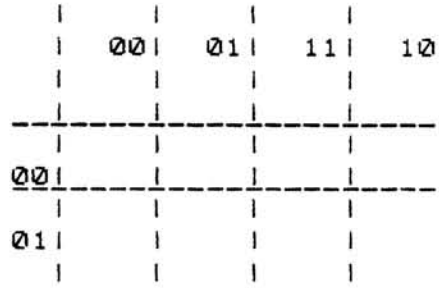
Karnaugh Method of Minimization

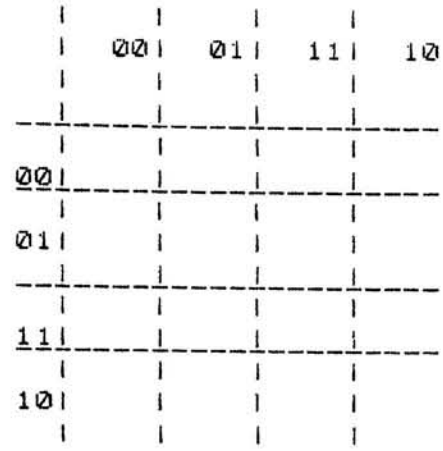
This is a graphic method which consists of representing each element of the function as an entry in the table or map. Simplification then, comes easily. This method works well with four variables or less.

It is assumed a number one to the occurrence of a variable and a zero otherwise. For example: xy' = 10; w'xy'= 0101, etc. Following the previous notation, the maps for one, two, three and four variables are


0
1


0
1
0
1


00
01
11
10
00
01



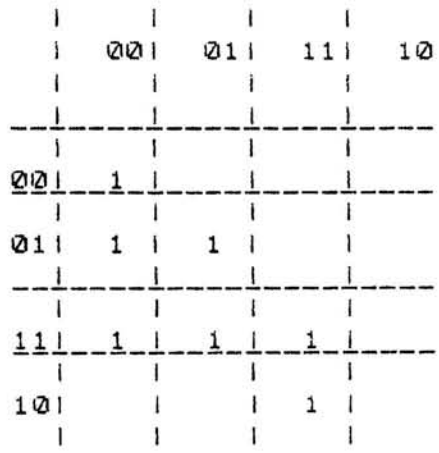
Each square contains either a 1 , if the term represented is required to be present, or a Ø (or blank) if it is required to be absent, or "X" if its presence or absence does not care.

The method consists in covering all 1's with the least number of rectangles. Adopting the previous notation for the maps, it becomes easier to identify contiguous areas.

Example:

F(w, x, y, z): w'x'y'z' + w'xy'z' + w'xy'z +

wxy'z' + wxy'z + wxyz + wx'yz



a)

W	X	Y	Z
Ø	Ø	Ø	Ø
Ø	1	Ø	Ø
Ø		Ø	Ø

w' y'z'

B)

W	X	Y	Z
1	1	1	1
1	Ø	1	1
1		1	1

w y z

C)

W	X	Y	Z
Ø	1	Ø	Ø
Ø	1	Ø	1
1	1	Ø	Ø
1	1	Ø	1
	1	Ø	

x y'

F(w, x, y, z): w'y'z' + wyz + xy'

Quine McCluskey Method of Minimization

The following method was originated by Willard van Orman Quine and modified by E. J. McCluskey.

The method has two different procedures, the first one is applied to get prime implicants and the second one to eliminate redundant prime implicants.

The first procedure is a simplification of products based on an application of the distributive law: Ex + Ex'= E(x+x')= E(1)= E, where E is a Boolean expression. Quine's method consists of expressing a function as a set of minterms or sum of products; finding a set of prime implicants (products that cannot be reduced), that is, beginning with the full disjunctive form (all elements of the function are connected by an "or" or by "+"), forming all possible combinations. The procedure is repeated until the prime implicants are reached. the result is the sum of the implicants.

McCluskey modified this method changing the Quine's algebraic description of terms for a numeric one, using the binary numbers. Thus, the process becomes easier.

Example:

F(w, x, y, z): w'x'y'z' + w'xy'z' + w'xy'z' +

wxy'z' + wxy'z + wxyz + wx'yz

The binary representation of these terms is:

0	0 0 0 0	w'x'y'z'
4	0 1 0 0	w'x y z
5	0 1 0 1	w'x y'z
12	1 1 0 0	w x y'z'
11	1 0 1 1	w x'y z
13	1 1 0 1	w x y'z
15	1 1 1 1	w x y z

which, at the same time, are grouped according to the number of 1's that they have.

Comparing these terms, the following combinations are produced:

Combination	Terms yielding this Combination
0 - 0 0	(0) (4)
0 1 0 -	(4) (5)
- 1 0 0	(4) (12)
- 1 0 1	(5) (13)
1 1 0 -	(12) (13)
	(13) (15)
	(11) (15)

Comparing adjacent groups again, a new prime implicant is produced: - 1 0 - Now the function becomes:

$$F(w,x,y,z): \underset{A}{w'y'x'} + \underset{B}{wxz} + \underset{C}{wyz} + \underset{D}{xy'}$$

The second part is a table that eliminates the redundant prime implicants. The table shows terms in disjunctive normal form versus prime implicants; a cross is placed in position (G,U), if the Gth term participates in the formation of the Uth prime implicant.

If a column contains a single cross, the prime implicant in whose row the cross occurs, is needed to represent that term. The prime implicant is marked and its row and column deleted. If the method exhausts all columns, the function is represented by the sum of prime implicants marked. However, if columns still remain, more prime implicants are marked, continuing with the process.

	0	4	5	11	12	13	15
A	X	X					
						X	X
				X			X
D		X	X		X	X	

The number 2^{n} means the participating elements per prime implicant, where n is the number of elements deleted.

7.0 Bibliography

Arnold, B. H., Logic And Boolean Algebras, Prentice-Hall, Englewood Cliffs, N.Y., 1962.

Carbonell, Jaime G., "Default Reasoning and Inheritance Mechanisms on Type Hierarchies", Comp. Sci. Dept., Carnegie-Mellon Univ.

Copy, Irving, Introduction to Logic, McMillan Publishing Co., N.Y., 1982.

Doyle, Jon, "A Glimpse of Truth Maintenance", A.I. Memo 461, M.I.T Press, Cambridge, Mass, 1978.

Doyle, Jon, "The Use of Dependency Relationships in the Control of Reasoning", AI working paper 133, M.I.T., Cambridge, Mass., 1976.

Doyle, Jon, "Truth Maintaining Systems for Problem Solving", M.I.T., A.I. Lab Technical Report 419, Cambridge, Mass.

Feigenbaum, Edward A., et al, The Handbook of Artificial Intelligence, Vols. I, III, William Kaufman Inc., New York, N.Y., 1982.

Fox, Mark S., "Reasoning with Incomplete Knowledge in a Resource Limited Environment: Integrating Reasoning and Knowledge Acquisition", Proceedings of the seventh International Joint Conference on A.I., Vol I, 1981.

Glorioso, Robert M. and Fernando C. Colon, Engineering Intelligent Systems, Digital Press, New York, N.Y., 1980.

Goodstein, R., Boolean Algebra, Pergamon Press, London, U.K., 1963.

Hilton, Alice Mary, Logic, Computing Machines, and Automation, Spartan Books, Washington, D.C., 1963.

Israel, David J. et al, "The Role of Logic in Knowledge Representation", Computer, Vol. 16, No. 10, Oct. 1983.

Kolata, Gina, "How Can Computers Get Common Sense? ", Science, Vol. 217, Sept. 24, 1982.

Korfhage, Robert R. , Logic and Algorithms, John Wiley & Sons Inc., New York, N.Y., 1966.

Lavesque, Hector J., "The Interaction with Incomplete Knowledge Bases: A Formal Treatment", Proceedings of the Seventh International Joint Conference on A.I., Vol. I, 1981.

Matteshish, Richard, Instrumental Reasoning and Systems Methodology, D. Reidel Publishing Co., Boston, Mass., 1978.

McCarthy John, "Addendum: Circumscription and Other Non-Monotonic Formalisms", Artificial Intelligence, Vol. 13, North-Holland Publishing Co., 1980.

McCarthy John, "Circumscription: A Form of Non-Monotonic Reasoning", Artificial Intelligence, Vol. 13, North-Holland Pulishing Co., 1980.

McCarthy John, "On the Model Theory of Knowledge", Memo AIM-312, Stanford A.I. Lab.,1978.

McCarthy John, "Programs with Common Sense", Semantic Information Processing, M.I.T. press, Cambridge, Mass., 1968.

McDermott Drew, and Jon Doyle,"An Introduction to Non-Monotonic Logic", Sixth International Joint Conference on A.I., Vol. I, 1979.

McDermott Drew, and Jon Doyle, "Non-Monotonic Logic I", Artificial Intelligence, Vol. 13, Norht-Holland Publishing Co., 1980.

McDermott Drew, "Non-monotonic Logic II: Non-Monotonic Modal Theories", Journal of the ACM, Vol. 29, No 1, 1982.

Minsky, Marvin, "A Framework for Representing Knowledge", ed. by P. H. Winston, McGraw-Hill, New York, N.Y., 1975.

Nilsson, Nils J., Principles of Artificial Intelligence, Tioga Publishing Co., Palo Alto, Cal., 1980.

Quine, W.V., Methods of Logic, Harvard Univ. Press, 4th edition, Cambridge, Mass., 1982.

Raphael, Bertram, The Thinking Computer, W. H. Freeman and Co., San Francisco, Ca., 1976.

Reiter R., "A logic for Default Reasoning", Artificial Intelligence, Vol.13, North-Holland Publishing Co., 1980.

Rich, Elaine, Artificial Intelligence, McGraw-Hill, New York, N.Y., 1983.

Rodenau, Sergiu, Boolean Functions and Equations, North-Holland Publishing,Co., Amsterdam, 1974.

Rosenberg, Steven, "Reasoning in Incomplete Domains", Sixth International Joint Conference on A.I., Vol I, 1979.

Sikorsky, R., Boolean Algebras, Spring-Verlag, New York, N.Y., 1969.

Wang, Hao, Logic, Computers and Sets, Chelsea Publishing Co., N.Y., 1970.

Weizenbaum, Joseph, Computer Power and Human Reason, W. H. Freeman and Co., San Francisco, Cal., 1976.

Whitehead & Russell, Principia Mathematica to * 56, Cambridge University Press, Cambridge, U.K., 1910-13

Winograd, Terry, "Extended Inference Modes in Reasoning by Computer Systems", Artificial Intelligence, Vol. 13, North Holland Publishing Co., 1980.

Winston, Patrick H., Artificial Intelligence, Adison-Wesley Pulishing Co., 1979.

Reference Bibliography

Aristotle, Methaphisics.

Aristotle, Nichomachaen Ethics.

Bartlett, F. C., Remembering, A Study in Experimental and Social sychology, Cambridge Univ. Press, Cambridge, 1932.

Boole G., An Investigation of the Laws of Thougth, Dover Publications, 1854.

Bobrow, D.G. and T. Winograd, "An Overview of KRL," Cognitive Science 1, 1977.

Cantor G., Contributions to the Founding of the theory of Transfinite Numbers, Open Court, London, 1915.

De Morgan, A. , A Formal Logic, 1847.

Frege, Grundgesetze der Arithmetik, 1893 Kegan Paul,Trench, Trubner, London.

Hamilton, W., Discussions on Philosophy, Literature, and Education, The Edinburgh Review, 1852.

Jevons, S., Principles of Science,London 1874.

Kant, I., Kritik der Reinen Vernunft (Critique of Pure Reason), 1781.

Kant,I., Kritik der Praktischen Vernunft (Critique of Practical Reason), 1788.

Kling, R. , "A Paradigm for Reasoning by Analogy," Artificial Intelligence, vol 2· 1971.

Leibnitz, G. V. von, Dissertio de Arte Combinatoria, 1666.

Mill, John S., A System of Logic, 1843.

Peano, G., Formulaire de Mathematiques, Five volumes, 1895-1908.

Pierce, C., Collected Papers, Reprinted by Harvard University Press in 1932.

Schroder, E., Abriss der Algebra der Logik, Leipzig, 1909.

Turing, A., M., "Computing Machinery and Intelligence", Mind, Vol. 59, 1950.

Venn, J., Boole's System of Logic, 1876.

Wang, H., Logic, Computers, and Sets. Chelsea Pub. Co., 1970.

Wittgenstein, L. von, Philosophical Investigation, 1953.