Artificial Intelligence.

ARTIFICIAL INTELLIGENCE:

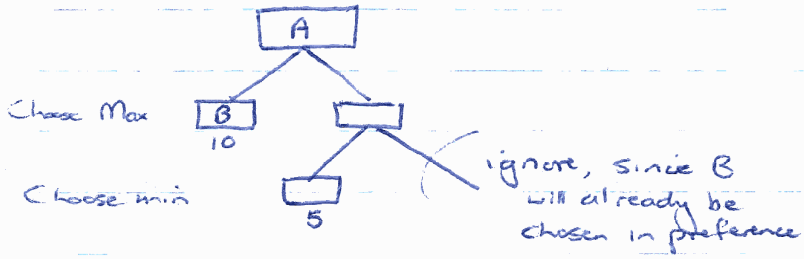
There are two main aspects to automatic problem solving: representation, and search. Both of these are affected by the problem (i.e. are problem dependent) - although we would like to have generality, research in specific problems has been far more fruitful.

In REPRESENTATION we are concerned with objects and their relations, (usually represented by lists), as well as the rules / operators for processing these (eg remove / add / compare lists). We will be more concerned with search methods:

1 SEARCH

There are many search methods available, eg:
'blind' search - { breadth first (queue nodes)
depth first (stack nodes)

'heuristic' (intelligent) search, using lookaheads & scoring.
eg minimax, α-β pruning



Topdown - goal to subgoal, is planning & analytic
Bottom up - initial assumptions to find conclusion, is proving & synthetic.

So, to find a path of moves from a given to a desired state:

- if there are a small number of paths, an EXHAUSTIVE SEARCH along every path of the tree can be used, either one at a time (depth first) or all at once (breadth first). This method is infallible.

- if there are many paths, a HEURISTIC SEARCH can evaluate each state in terms of estimated distance from the goal (how to design the evaluating function?).

- against an opponent, minimaxing evaluates states as best (or worst) for a particular player, considering what the opponent can make of them

- goal reduction: the GENERAL PROBLEM SOLVER sets up and/or tree with goal nodes, subgoal nodes, and TRANSFORM, REDUCE & APPLY edges.

2 THE GENERAL PROBLEM SOLVER

The GPS embodies two basic principles - means-end analysis and recursive problem-solving. The former is a technique for ensuring that an attempt is made to apply an operator only if there is some purpose to the application.

Suppose the possible attributes for two objects are defined. A difference between objects is, by definition, a discrepancy in their values on some attribute. An operator changes the values of attributes. Define an operator difference table by listing the operators and the differences which

they effect. This table is then used to guide subproblem selection. The following examples will illustrate:

2.1 The Monkey Problem.

A monkey is in a cage. Suspended over the centre of the cage out of the monkey's reach, is a bunch of bananas. There is a box in the corner. How can the monkey get the bananas?

Attributes: monkey's altitude, location
box's location
bananas altitude, location

Operators: things that the monkey can do

Operator Difference Table. (example)

	Operator			
Difference	Climb	Walk	Push	Reach
Monkey location		X	X	
Box location			X	
Banana location				X
Monkey Altitude	X			
Banana Altitude				X

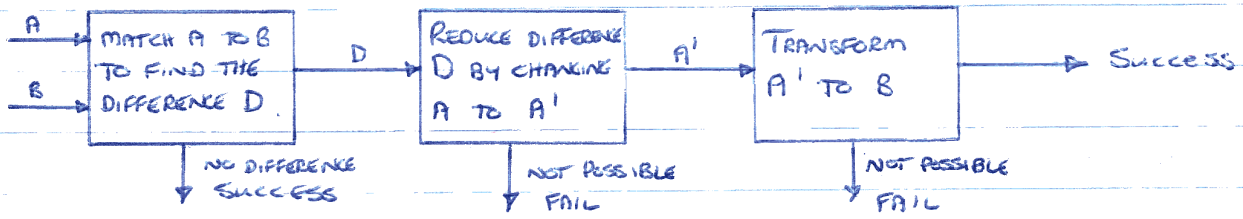
Note that the definition of the operator difference table is outside the GPS itself, since it is given to the program as part of the definition of the task environment.

Usually the problem is represented in a tree (although the monkey problem tree has only a single node).

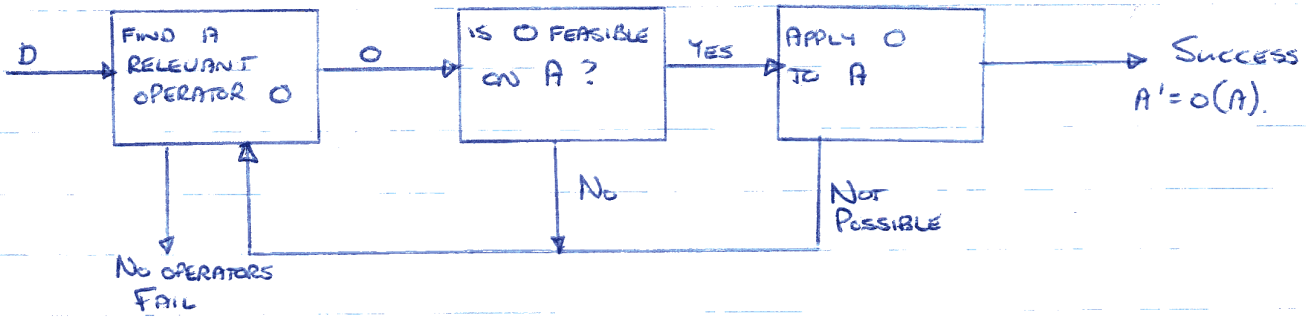
2.2 MEANS-END ANALYSIS.

This uses the difference between objects to guide the problem-solving process. The steps for the analysis of the abstract problem "transfer object A into object B" are:

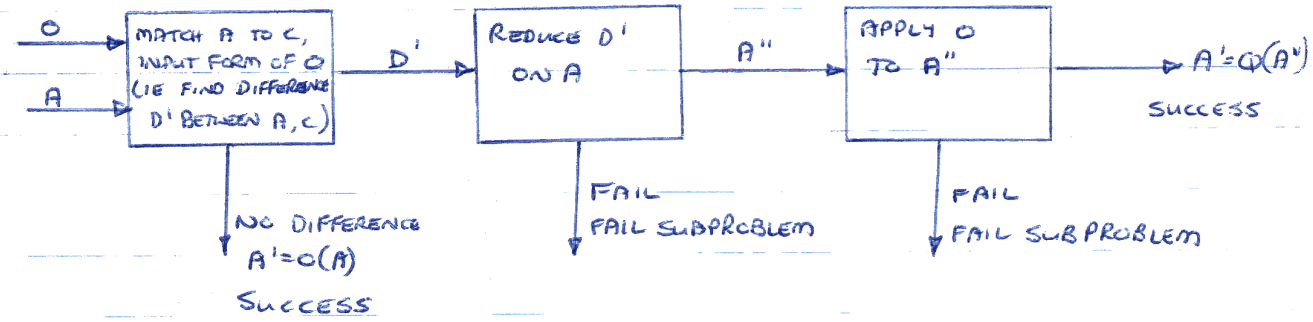
TRANSFORM A TO B.



REDUCE DIFFERENCE



APPLY



Problems can arise if "loops" occur, ie a problem becoming a subproblem of itself. Appropriate, though rather tedious programming arrangements can be made.

2.3 Psychological Implications (According to McDermott)

At what level is GPS a model of human thinking?

- at problem solution level?
- at electronic circuit level?
- at some intermediate level?

INFORMATION PROCESSING-LEVEL

at the level of ~~formulating~~ and ~~transforming~~ ~~information~~ (tree search).

Consider "traces" of human & computer processing

- no distinction between memory-searches and directory searches
- no meta-remarks or ~~expletives~~
- no 'bundling' of possibilities according to context
- a defined limit on foresight.

3. Computational Models for Language Processing.

Understanding something said, in terms of:
- making the right replies (Colby's PARRY)
- behaving appropriately (Winograd's SHRDLU)
- translating correctly (Wilks translator)
- answering questions (Charniak's story understander)
- paraphrasing (Schank's SCRIPTS)
- appearing to understand?

3.1 Colby's 'PARRY'

This program checks the input sentence against various patterns, and selects a response according to emotional variables, eg:

Input	Possible Outputs	Emotional Scale
I .... you	I .... you too	high fear ↑
	Why do you .... me?	
	I do not quite follow	
	Please change the subject	low

The advantages of PARRY are that it is robust (it hardly ever at a loss for words) and within its limited context it passes the TURING TEST (so what?). However, although it 'knows' <u>how</u> to converse, it does not know <u>what</u> it is saying, and thus offers no <u>model</u> of human knowledge

3.2 WINOGRAD'S SHRDLU.

Thus has a syntax module 'knowing' grammatical structures, a "semantics module" 'knowing' meanings of words, and a blocks module 'knowing' the state of the world.

Given an input command, it parses it and uses semantic specialists (on noun phrases, etc) to build up a possible semantic object, then checks for the actual object in the world. Thus "meanings are procedures", ie, they are either evaluations of possibility/actuality, or changes to the world.

The advantages of SHRDLU are that it relates to a world about which it has knowledge, and the isomorphism of input to action ensures 'understanding'.

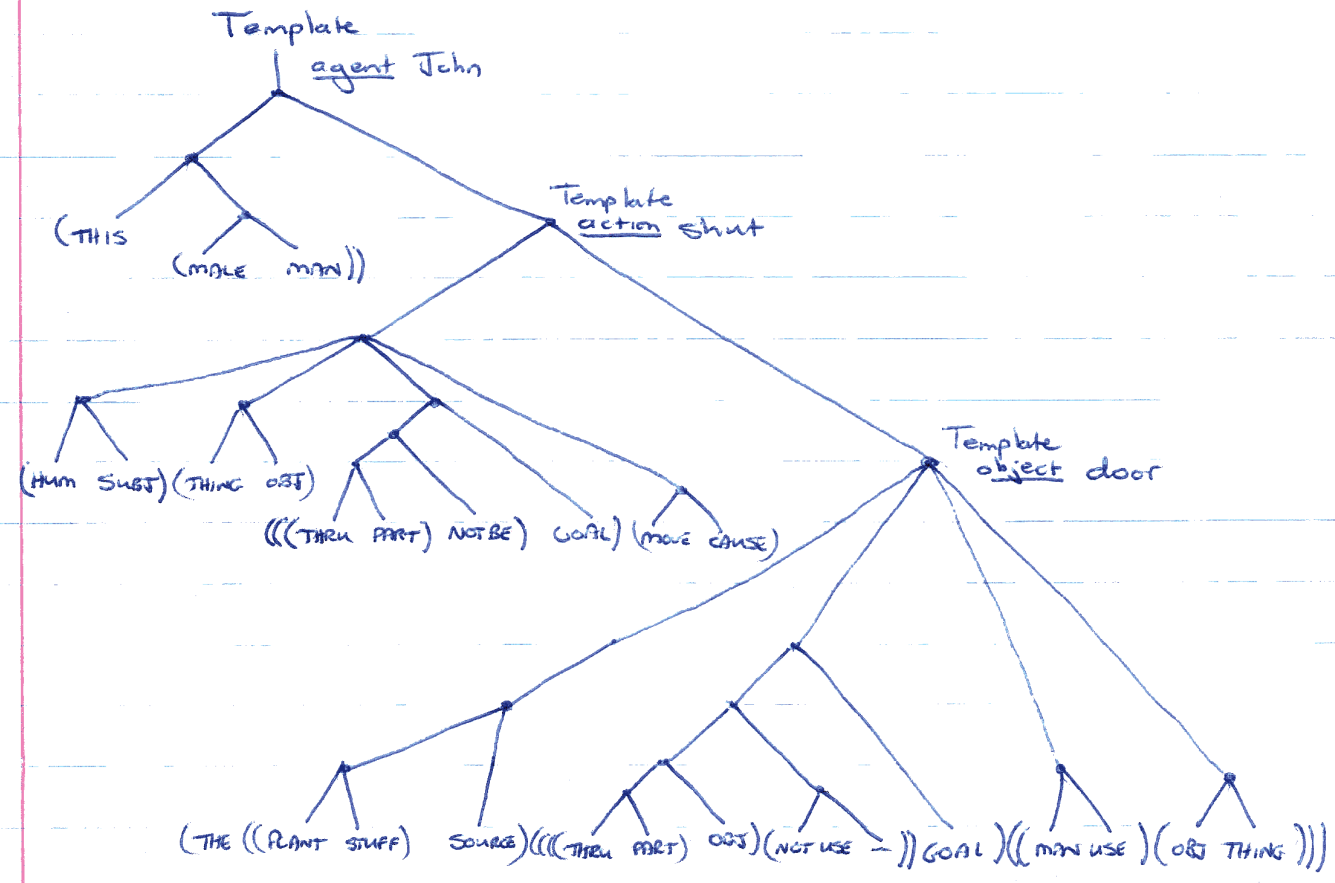
The disadvantages are that the world is small & difficult to expand, the objects are limited and already provided (ie no new knowledge), and we can question whether it always understands, or only when it can perform (ie establish isomorphism).

3.3 Wilks Translator.

The program uses

- fragmentation (split sentence into prepositions, subjunctions, etc)
- Case template matching (templates indicate the agent, action and object) to formula heads.
- preferential expansion
- paraplate matching
- basic mode pronoun resolution
- extraction
- common-sense rule matching.

Eg "John shut the door"



3.4 CHARNIAK'S STORY UNDERSTANDER.

The input is analysed grammatically and for word meanings, and an attempt is made to match input against 'proposition structures' in a database.

It is like Wilks (unlike Winograd) in its use of tentative inference, and like Winograd (unlike Wilks) in its restriction to a microworld.

It raises questions of the relation between:

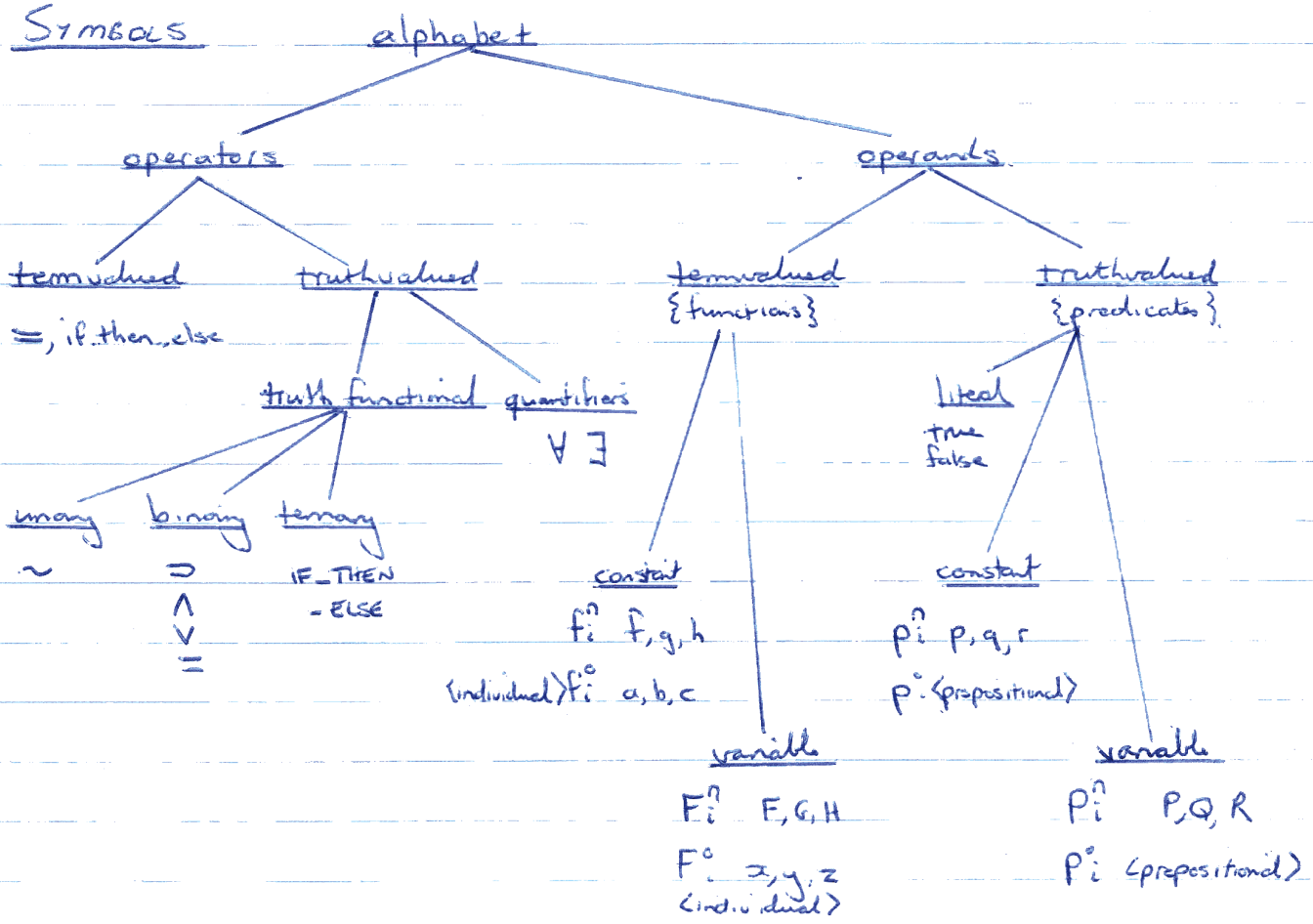
- knowledge of language
- knowledge of the world as a system
- knowledge of 'peculiarities' of a world

3.5 SCHANK'S SCRIPTS.

A script consists of a list of players, props & actions. Schank analyses verbs in terms of semantic primitives (12 in all).

4 THE PREDICATE CALCULUS. (2ND ORDER WITH EQUALITY).

4.1 SYMBOLS



Superscripts of n imply n-adic functions/predicates/~~operators~~

4.2 SYNTAX

individual constants; constant functions; variable functions; individual variables

$$\text{term} ::= f_i^0 \mid F_i^0 \mid f_i^n(\{t\}) \mid F_i^n(\{t\}) \mid \text{if } A \text{ then } t \text{ else } t$$

propositional constants; variable; constant predicate fns; variable fns

$$\text{atomic formula} ::= \text{true} \mid \text{false} \mid p_i^0 \mid P_i^0 \mid p_i^n(\{t\}) \mid P_i^n(\{t\}) \mid t = t$$

$$\text{wff} ::= \alpha \mid \sim A \mid A \supset A \mid A \wedge A \mid A \vee A \mid A \equiv A \mid \text{IF } A \text{ THEN } A \text{ ELSE } A \mid$$
$$\forall v_i\ A \mid \exists v_i\ A$$

If operands are restricted to:	we have:
propositional constants p_i^0	propositional calculus
+ propositional variable $P^0(pqr)$	<quantified>
individual constants f_i^0 abc	equality calculus
& individual variables F_i^0 xyz	
function constants f_i^n abc, fgh	first-order predicate
& predicate constants p_i^n pqr	calculus.
& individual variables F_i^0 xyz	(with equality)

4.3 SEMANTICS.

An interpretation I of a wff consists of:

- D - a non-empty set, the <u>domain</u> of the interpretation
- For every function constant & free function variable we assign some particular n-adic function $D^n \to D$
- For every predicate constant & free predicate variable we assign an n-adic predicate $D^n \to \{\text{true, false}\}$
- We also generally give interpretations to $\wedge, \vee, \equiv$, etc (implicit).

4.4 DEFNS:

- A wff is <u>valid</u> if it is true on every interpretation (ie no interpretations exist in which ~wff is true)
- A wff is <u>unsatisfiable</u> if it is false on every interpretation (ie ~wff is valid).

- A wff is valid iff its universal closure $\forall v_1 \forall v_2 \ldots A$ is valid
- A wff is unsatisfiable iff its existential closure $\exists v_1 \exists v_2 \ldots A$ is unsatisfiable

All occurences of a variable which is quantified within the scope of the quantifier are said to be bound by the quantifier.

4.5 GÖDEL: • The validity problem of the second-order predicate calculus is not even partially solveable.

CHURCH: • The validity problem of the first-order predicate calculus is unsolveable, but partially solveable.
• The validity problem of the propositional calculus is solveable.

(Validity problem: is there a general method to evaluate all of the ~~infinite~~ wffs in the calculus?)

4.6 Reminder: We are using:

n-ary constant functions f_i^n : f, g, h
individual constants f_i^0 : a, b, c

n-ary variable functions F_i^n : F, G, H
individual variables F_i^a : x, y, z

Predicate constants P_i^n : } p, q, r
Propositional constants P_i^0 : }

Predicate variables P_i^n : } P Q R
Propositional variables P_i^0 : }

We will be examining resolution, a partial decision procedure for unsatisfiability for the first order predicate calculus, applicable to wff's in clause form.

4.7 CLAUSE FORM.

In general, any wff in the predicate calculus can be put into clause form by applying a sequence of simple operations. ~~This process will be illustrated with an example:~~

~~$(\forall x)\{P(x) \Rightarrow \{(\forall y)\{P(y) \Rightarrow P(f(x,y))\} \wedge \sim(\forall y)\{Q(x,y) \Rightarrow P(y)\}\}\}$~~

A literal is an atomic formula or its negation (with no occurrences of the if-then-else operator)

A clause is a disjunction of one or more literals

(eg $P(f(x),a) \vee \sim q(y) \vee \sim p(a,y)$)

A wff is in clause form if it is of the form:

$$\underbrace{\forall x_{i_1} \forall x_{i_2} \ldots \forall x_{i_m}}_{\text{prefix of wff}} \underbrace{[C_1 \wedge C_2 \wedge \ldots \wedge C_k]}_{\text{matrix of wff}}$$

where each C_i is a clause.

Other types of normal form are:

PRENEX normal form: $Q v_1 Q v_2 \ldots Q v_n A$ where A contains no quantifiers

PRENEX - CONJUNCTIVE normal form: matrix in clause form.

PRENEX - DISJUNCTIVE normal form: matrix is a disjunction of conjunctions of literals.

CLAUSE FORM: closed, in prenex-conjunctive form, with a prefix consisting only of universal quantifiers

SKOLEM'S THEOREM: Every wff can be transformed into clause form in such a way that satisfiability is preserved.

METHOD OF TRANSFORMING TO CLAUSE FORM.

We shall use an example:

$$\forall x \{ p(x) \supset \exists z \{ \sim \forall y [q(x,y) \supset p(f(x,y))] \wedge \forall y [q(x,y) \supset p(x)]\}\}$$

① TAKE THE EXISTENTIAL CLOSURE, giving

$$\exists x_1 \forall x \{ p(x) \supset \exists z \{ \ldots \}\}$$

② ELIMINATE REDUNDANT QUANTIFIERS giving

$$\exists x_1 \forall x \{ p(x) \supset \{ \ldots \}\}$$

③ RENAME ANY VARIABLE THAT IS QUANTIFIED IN D MORE THAN ONCE, giving

$$\exists x_1 \forall x \{ p(x) \supset \{ \sim \forall y [q(x,y) \supset p(f(x,y))] \wedge \forall z [q(x,z) \supset p(x)]\}\}$$

④ ELIMINATE ALL CONNECTIVES OTHER THAN $\sim, \wedge, \vee$

(a) ELIMINATE THE if-then-else OPERATOR

Replace any atomic formula A containing one or more occurrences of a term if B then t' else t" by: IF B THEN A' ELSE A" where A' (A") is A with all occurrences of if B then t' else t" with t' (t")

eg A = if $(x=y)$ then $(p(x) \supset q(x))$ else $\sim p(x)$
becomes IF $(x=y)$ THEN $(p(x) \supset q(x))$ ELSE $\sim p(x)$

(b) ELIMINATE ALL CONNECTIVES EXCEPT $\wedge$ $\vee$ $\sim$

Replace: $A \supset B$ by $\sim A \vee B$

$A \equiv B$ by $(\sim A \vee B) \wedge (\sim B \vee A)$

IF A THEN B ELSE C by $(\sim A \vee B) \wedge (A \vee C)$

giving:

$$\exists x_1 \forall x \{\sim p(x) \vee \{\sim \forall y [\sim q(x,y) \vee p(f(x_1))] \wedge \forall z [\sim q(x,z) \vee p(x)]\}\}$$

(5) MOVE $\sim$ INWARD

Replace: $\sim \forall x A$ by: $\exists x \sim A$

$\sim \exists x A$ by: $\forall x \sim A$

$\sim (A \vee B)$ by: $\sim A \wedge \sim B$

$\sim (A \wedge B)$ by: $\sim A \vee \sim B$

$\sim \sim A$ by: A

until each occurrence of $\sim$ immediately precedes an atomic formula.

giving:

$$\exists x_1 \forall x \{\sim p(x) \vee \{\exists y [q(x,y) \wedge \sim p(f(x_1))] \wedge \forall z [\sim q(x,z) \vee p(x)]\}\}$$

(6) PUSH QUANTIFIERS TO THE RIGHT

Replace: $\exists(x)(A \vee B)$ by: $\begin{cases} A \vee \exists x B & \text{if } x \text{ is not free in } A \\ \exists x A \vee B & \text{if } x \text{ is not free in } B \end{cases}$

etc.

giving:

$$\exists x_1 \forall x \{\sim p(x) \vee \{[\exists y\, q(x,y) \wedge \sim p(f(x_1))] \wedge [\forall z \sim q(x,z) \vee p(x)]\}\}$$

⑦ Eliminate Existential Quantifiers

Pick out the leftmost wfp $\exists y\, B(y)$ and replace it with $B(f(x_{i1}, x_{i2}, \ldots, x_{in}))$

where $x_{i1} \ldots x_{in}$ are all the distinct free variables of $\exists y\, B(y)$ which are universally quantified to the left of $\exists y\, B(y)$

f is any n-ary function constant which does not already occur in the wff.

Repeat until all existential quantifiers have been removed

NB. If there are no x's, $\exists y\, B(y)$ is replaced by $B(a)$ where a is any individual constant which does not already occur in the wff.

We had: $\exists x_1 \forall x \{\sim p(x) \vee \{[\exists y\, q(x,y) \wedge \sim p(f(x_1))] \wedge \ldots$

giving:

$$\forall x \{\sim p(x) \vee \{\exists y\, q(x,y) \wedge \sim p(f(a))] \wedge [\forall z \sim q(x,z) \vee p(x)]\}\}$$

and then:

$$\forall x \{\sim p(x) \vee \{[q(x, g(x)) \wedge \sim p(f(a))] \wedge [\forall z \sim q(x,z) \vee p(x)]\}\}$$

⑧ Move (Universal) Quantifiers Left

$\forall x \forall z \{\ldots\}$

⑨ Distribute $\wedge$ over $\vee$ (creating prenex-conjunctive matrix)

Replace: $(A \wedge B) \vee C$ by: $(A \vee C) \wedge (B \vee C)$

$A \vee (B \wedge C)$ $\quad$ $(A \vee B) \wedge (A \vee C)$

giving:

$$\forall x \forall z \{[\sim p(x) \vee [q(x, g(x)) \wedge \sim p(f(a))] \wedge [\sim p(x) \vee \sim q(x,z) \vee p(x)]\}$$

and then:

$$\forall x \forall z \{[\sim p(x) \vee q(x, g(x))] \wedge [\sim p(x) \vee \sim p(f(a))] \wedge [\sim p(x) \vee \sim q(x,z) \vee p(x)]\}$$

(10) USE SIMPLIFICATION RULES WHICH PRESERVE SATISFIABILITY

FACTORING: Eliminate all duplicate occurrences of the same literal in a clause.

ELIMINATION OF TAUTOLOGIES: Eliminate any clause containing an atomic formula and its negation as literals.

The final wff is in clause form.

$$\forall x \forall z \{ [\sim p(x) \vee q(x, g(x))] \wedge [\sim p(x) \vee \sim p(f(a))] \}$$

and:

$$\forall x \{ [\sim p(x) \vee q(x, g(x))] \wedge [\sim p(x) \vee \sim p(f(a))] \}$$

as $(\forall x : \sim p(x)) \supset \sim p(f(a))$

we can drop the $\sim p(x)$ lit, giving

$$\forall x \{ [\sim p(x) \vee q(x, g(x))] \wedge \sim p(f(a)) \} \longrightarrow$$

4.8 HERBRAND'S PROCEDURES

By defn, a wff A is unsatisfiable iff it is false for all interpretations over all domains. Clearly it would help if we could have one special domain D such that A is unsatisfiable iff it is false on all interpretations in this one domain. In fact, for any wff, this HERBRAND UNIVERSE H_A does exist. It is the set of all terms that can be constructed from the individual constants a_i and the function constants f_i occurring in A.

eg. If A is the wff $\forall x [p(a) \vee q(b) \supset p(f(x))]$
then $H_A = \{a, b, f(a), f(b), f(f(a)), f(f(b)), \ldots\}$

If A is the wff $\forall x \forall y [p(f(x), y, g(x,y))]$
then $H_A = \{a, f(a), g(a,a), f(f(a)), g(a, f(a)), g(f(a), a) \ldots\}$

A ground instance of a clause is obtained by replacing all individual variables by some terms of H_A. Such clauses with no individual variables are called ground clauses.

HERBRAND'S THEOREM: A wff A in clause form is unsatisfiable iff there is an unsatisfiable finite conjunction of ground clauses.

This theorem suggests a procedure, called Herbrand's Procedure, for proving the unsatisfiability of wff's in clause form. For a given wff A in clause form, we can generate successively the ground instances G_i of the clauses of A and test successively whether their conjunction is unsatisfiable. By the theorem, if A is unsatisfiable, the procedure will detect it after a finite number of steps; otherwise the procedure may never terminate. One of the problems in implementing this procedure is how to test the unsatisfiability of a finite conjunction of ground clauses. We show two methods: DAVIS & PUTNAMS (1960) and the ground resolution method.

4.9 Davis & Putnam's Method

This method for determining whether a finite conjunction $S = G_1 \wedge G_2 \ldots \wedge G_N$ of ground clauses G_i is unsatisfiable consists essentially of applying repeatedly the rules listed below. At each step S is transfered to a new (pair of) conjunction(s) S' (and S'') such that S is unsatisfiable iff S' (and S'') is (are) unsatisfiable. The process must terminate as the number of distinct literals occurring in S is reduced in each step.

if G_i is a literal l then
  if $\exists j : G_j = \sim l$ then S is unsatisfiable
  else delete all G's in S containing l (including G_i) and then delete all occurences of $\sim l$ from the remaining G's, to get S'. If S' is empty, S is satisfiable.

else if l occurs in S then
  if $\sim l$ occurs in S then
    S can be put in the form

$$(A_1 \vee l) \wedge \ldots \wedge (A_{N_1} \vee l) \wedge$$
$$(B_1 \vee \sim l) \wedge \ldots \wedge (B_{N_2} \vee \sim l) \wedge$$
$$R_1 \wedge \ldots \wedge R_{N_3}.$$

    where A_i, B_i, R_i are ground clauses free of l and $\sim l$.
    Then $S' = A_1 \wedge \ldots \wedge A_{N_1} \wedge R_1 \wedge \ldots \wedge R_{N_3}$
    and $S'' = B_1 \wedge \ldots \wedge B_{N_2} \wedge R_1 \wedge \ldots \wedge R_{N_3}$
    S is unsatisfiable iff $(S' \vee S'')$ is unsatisfiable.
    ie both S' and S'' are unsatisfiable
  else delete all G's containing l to get S'. If S' is empty, S is satisfiable.

4.10 THE GROUND RESOLUTION METHOD.

The ground resolution method consists of a single inference rule, called the ground resolution rule, which is essentially an extension of the first half of Davis & Putnam's method:

"For any two ground clauses G_1 and G_2 containing a literal l and its negation $\sim l$ respectively, construct a new clause G_3, the resolvent of G_1 and G_2, by deleting all occurences of l and $\sim l$ in G_1 and G_2 respectively, and then letting G_3 be the disjunction of the resulting clauses G_1' and G_2' ie $G_3 = G_1' \vee G_2'$.
Eliminate duplicate occurences of literals in G_3".

SPECIAL CASE: If there are two one-literal clauses, a literal l and its complement $\sim l$, then their resolvent is the empty clause, denoted $\square$.

The key result is that: A finite conjunction S of ground clauses is unsatisfiable iff $\square$ can be derived from S by repeated application of the ground resolution rule.

Clearly, the process must always terminate because there are only finitely many distinct clauses (with no duplicate literals) that can be constructed from the literals of S. If the process terminates and $\square$ has not been derived then S is satisfiable

4.11 The Unification Algorithm

In 1965 Robinson introduced the Resolution Rule, a generalisation of the ground resolution rule which eliminates the need for ground clauses as it can be applied directly to any conjunction S of clauses.

Say we have a wff B in clause form:

$$\forall x \forall y \forall y_1 \forall z [p(x, f(x)) \wedge q(y, g(y)) \wedge (\sim p(a, y_1) \vee \sim q(y_1, z))]$$

The clauses of B are:

① $p(x, f(x))$

② $q(y, g(y))$

③ $\sim p(a, y_1) \vee \sim q(y_1, z)$

Consider ① & ③. We can obtain complementary literals if we replace x by a and y_1 by $f(a)$ to obtain:

①' : $p(a, f(a))$

③' : $\sim p(a, f(a)) \vee \sim q(f(a), z)$

We denote this substitution by $\{\langle x, a\rangle, \langle y_1, f(a)\rangle\}$ It is called a unifier because it unifies both $p(x, f(x))$ and $p(a, y_1)$ to yield the atomic formula $p(a, f(a))$.

If we resolve ①' & ③' we get ④ : $\sim q(f(a), z)$

Consider ② & ④. Apply the substitution $\{\langle y, f(a)\rangle, \langle z, g(f(a))\rangle\}$ to obtain : ②' : $q(f(a), g(f(a)))$

④' : $\sim q(f(a), g(f(a)))$

which can be resolved to obtain ⑤ : $\square$

Thus B is unsatisfiable.

In order to describe the general method, we need a few definitions.

A substitution is denoted by a finite set (possibly empty) of pairs of the form $\alpha = \{\langle v_1, t_1\rangle, \ldots, \langle v_n, t_n\rangle\}$ where the v_i's are distinct individual variables and each t_i is a term different from v_i.

For any literal L, $L' = L\alpha$ is the resulting literal after performing the substitution α, called an instance of L

The disagreement set of a disjunction of atf's $p(t_1^{(1)}, \ldots, t_n^{(1)}) \vee p(t_1^{(2)}, \ldots, t_n^{(2)}) \vee \ldots \vee p(t_1^{(N)}, \ldots, t_n^{(N)})$ (for short $A_1 \vee A_2 \vee \ldots \vee A_N$) is a set of subterms $\{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}\}$ obtained as follows

We regard each of the atf's A_i as a string of symbols and detect the first symbol position in which not all the A_i's have the same symbol. Then, for each i we let $\tau^{(i)}$ be the subterm of A_i that begins with this symbol position.

---

eg the disagreement set of the disjunction:

$p(x, g(f(y,z), x), y) \vee p(x, g(a, b), b) \vee p(x, g(g(h(x), a), y), h(x))$.

is the set of subterms $\{f(y,z), a, g(h(x), a)\}$

A substitution α is called a unifier for a disjunction of atf's $A_1 \vee A_2 \vee \ldots \vee A_N$ if $A_1\alpha = A_2\alpha = \ldots = A_N\alpha$

If there is a unifier for a disjunction, the disjunction is unifiable

A unifier β for a disjunction of atf's $A_1 \vee A_2 \vee \ldots \vee A_N$ is a most general unifier if for each unifier α for the disjunction, $A_i\alpha$ is an instance of $A_i\beta$, all i

The atf $B = A_1\beta = A_2\beta = \ldots = A_N\beta$ is unique except for alphabetic variants, and is called a factor of the disjunction.

Examples • $p(a,x) \vee p(b,y)$ and $p(a,f(x)) \vee p(x,f(b))$ are not unifiable

• $\{\langle y,a\rangle, \langle x,b\rangle, \langle z,f(b)\rangle\}$ is a unifier for $p(a,f(x)) \vee p(y,z)$, and $\{\langle y,a\rangle, \langle z,f(x)\rangle\}$ is a most general unifier

• $\{\langle z,a\rangle, \langle x,h(a,g(y))\rangle, \langle w,g(y)\rangle\}$ is a most general unifier for $p(a,x,f(g(y))) \vee p(z,h(z,w),f(w))$

The most basic part of the (general) resolution method is the application of the unification algorithm, which finds a most general unifier β for a given unifiable disjunction of atomic formulae $p(t_1^{(1)},\ldots,t_n^{(1)}) \vee \ldots \vee p(t_1^{(N)}\ldots t_n^{(N)})$ denoted $A_1 \vee A_2 \vee \ldots \vee A_N$ and reports a failure when the disjunction is not unifiable.

THE UNIFICATION ALGORITHM

We start with the empty substitution α_0.

Loop: **if** α_k makes all the atf's identical

then $\beta = \alpha_k$; **stop**

else determine disagreement set D_k of $A_1\alpha_k \vee \ldots \vee A_N\alpha_k$

locate in D_k some variable v_k and some term t_k such that t_k does not contain v_k (if none, fail)

$\alpha_{k+1} = \alpha_k$ with all occurrences of v_k replaced by t_k + $\langle v_k, t_k\rangle$.

The algorithm must terminate because each α_k, $k \geq 0$, contains substitutions for k distinct individual variables $V_1, V_2, \ldots, V_k$ and there are only finitely many distinct variables in $A_1 \vee A_2 \vee \ldots \vee A_N$.

Example: $p(a, x, F(g(y))) \vee p(z, h(z,w), f(w))$

$A_1 \quad \vee \quad A_2$

$\alpha_0 = \{\}$, $D_0 = \{a, z\}$

$\alpha_1 = \{\langle z, a\rangle\}$ $\quad A_1\alpha_1 \vee A_2\alpha_1 = p(a, x, F(g(y))) \vee p(a, h(a,w), f(w))$

$D_1 = \{x, h(a,w)\}$

$\alpha_2 = \{\langle z, a\rangle, \langle x, h(a,w)\rangle\}$

$A_1\alpha_2 \vee A_2\alpha_2 = p(a, h(a,w), f(g(y))) \vee p(a, h(a,w), f(w))$

$D_2 = \{g(y), w\}$

$\alpha_3 = \{\langle z, a\rangle, \langle x, h(a, g(\quad))\rangle, \langle w, g(y)\rangle\}$

Since $A_1\alpha_3 = A_2\alpha_3 = p(a, h(a, g(y)), f(g(y)))$, $\beta = \alpha_3$ is a most general unifier.

4.12 The Resolution Rule

This method is based on repeated application of the Resolution Rule, a very powerful inference rule. We start by applying the resolution rule to a pair of clauses C_1 and C_2 from S to obtain a (general) resolvent C_3 of C_1 and C_2. We then add the new clause C_3 to S and apply the rule again until $\square$ is derived.

Robinson's Theorem: A given wff S in clause form is unsatisfiable iff $\square$ can be derived eventually by repeated application of the resolution rule.

Let C_1 and C_2 be two clauses such that no individual variable is common to both C_1 and C_2.
A subdisjunction of C_1 is a disjunction of some (or all) of the atf's of C_1.

Suppose that $p(t_1^{(1)}, \ldots, t_n^{(1)}) \vee \ldots \vee p(t_1^{(N)}, \ldots, t_n^{(N)})$ is a subdisjunction of C_1 and $\sim p(\tau_1^{(1)}, \ldots, \tau_n^{(1)}) \vee \ldots \vee \sim p(\tau_1^{(M)}, \ldots, \tau_n^{(M)})$ is a subdisjunction of C_2, and that a most general unifier β exists for the disjunction of the two subdisjunctions. Let $p(r_1, \ldots, r_n)$ be the factor of this disjunction. Then we can construct a new clause C_3, the <u>resolvent</u> of C_1 and C_2, by taking $C_1\beta \vee C_2\beta$ after eliminating all occurrences of $p(r_1, \ldots, r_n)$ in $C_1\beta$ and $\sim p(r_1, \ldots, r_n)$ in $C_2\beta$.

The requirement that there are no common variables to C_1 and C_2 is not a real restriction because any set of clauses $C_1(x), C_2(x), \ldots, C_k(x)$ can be replaced by an equivalent set of clauses $C_1(x_1), C_2(x_2), \ldots, C_k(x_k)$, since the wff's

A: $\forall x [C_1(x) \wedge C_2(x) \wedge \ldots \wedge C_k(x)]$

B: $\forall x_1 \forall x_2 \ldots \forall x_k [C_1(x_1) \wedge C_2(x_2) \wedge \ldots \wedge C_k(x_k)]$

are equivalent.

Example: Is $\exists x \exists y \forall z \{[p(x,y) \supset p(y,z) \vee p(z,z)] \wedge [(p(x,y) \wedge q(x,y)) \supset (q(x,z) \wedge q(z,z))]\}$ valid?

First negate and convert to clause form:

$\forall x \forall y \{p(x,y) \wedge (\sim p(y, f(x,y)) \vee \sim p(f(x,y), f(x,y)) \vee q(x,y)) \wedge (\sim p(y, f(x,y)) \vee \sim p(f(x,y), f(x,y)) \vee \sim q(x, f(x,y)) \vee \sim q(f(x,y), f(x,y))\}$

We then rename the variables in the remaining clauses, getting

① $p(x_1, y_1)$

② $\sim p(y_2, f(x_2, y_2)) \vee \sim p(f(x_2, y_2), f(x_2, y_2)) \vee q(x_2, y_2)$

③ $\sim p(y_3, f(x_3, y_3)) \vee \sim p(f(x_3, y_3), f(x_3, y_3)) \vee \sim q(x_3, f(x_3, y_3)) \vee \sim q(f(x_3, y_3), f(x_3, y_3))$

By clauses ① & ② with $\{\langle x_1, y_2\rangle, \langle y_1, f(x_2, y_2)\rangle\}$ we obtain:

④ $\sim p(f(x_2, y_2), f(x_2, y_2)) \vee q(x_2, y_2)$

Unifying ① & ④ and resolving (using $\{\langle x_1, f(x_2, y_2)\rangle, \langle y_1, f(x_2, y_2)\rangle\}$)

⑤ $q(x_2, y_2)$

By clauses ③ & ⑤ with $\{\langle x_2, f(x_3, y_3)\rangle, \langle y_2, f(x_3, y_3)\rangle\}$ we get:

⑥ $\sim p(y_3, f(x_3, y_3)) \vee \sim p(f(x_3, y_3), f(x_3, y_3)) \vee \sim q(x_3, f(x_3, y_3))$

By ⑤ & ⑥ with $\{\langle x_2, x_3\rangle, \langle y_2, f(x_3, y_3)\rangle\}$ we get

⑦ $\sim p(y_3, f(x_3, y_3)) \vee \sim p(f(x_3, y_3), f(x_3, y_3))$

By ① & ⑦ with $\{\langle x_1, f(x_3, y_3)\rangle, \langle y_1, f(x_3, y_3)\rangle\}$ we get

⑧ $\sim p(y_3, f(x_3, y_3))$

By ① & ⑧ with $\{\langle x_1, y_3\rangle, \langle y_1, f(x_3, y_3)\rangle\}$ we get □

This implies that $\sim A$ is unsatisfiable, so A is valid. ⇒