Resolution Strategies

1. The **Breadth-first** strategy - All of the first-level resolvents are computed first, then the second-level resolvents, and so on, where an ith-level resolvent is one whose deepest parent is an (i-1)th-level resolvent.

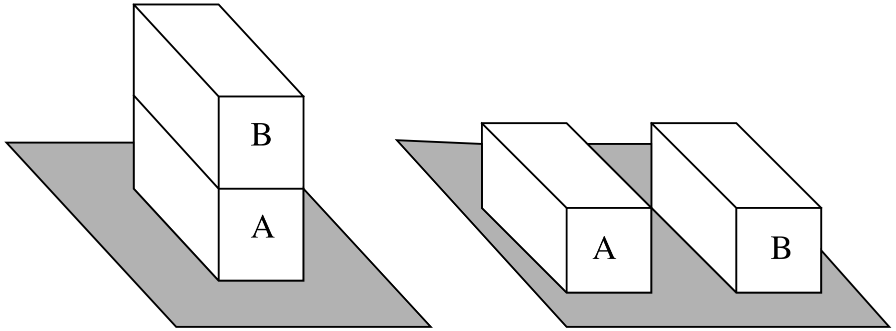
2. The **Unit-preference** strategy - Perform resolutions where one of the sentences is a single literal (also known as a **unit clause**).

3. The **Set-of-support** strategy - It starts by identifying a set of sentences called the **set of support**. For example, select at least one parent of each resolvent from among the clauses resulting from the negation of the query or from their desendants.

4. The **linear-input form** strategy - Every resolution combines one of the input sentences (from the KB or the query) with some other sentence.

Situation Calculus

- Represent change in first-order logic.



Initial Situation: Final Situation:

$On(B, A) \wedge On(A, Table)$ $On(B, Table)$

Use **situation variables** in representing a sequence of situations:

$On(B, A, S) \wedge On(A, Table, S)$ $\exists s_f\, [On(B, Table, s_f)]$

Introduce an operator, **Putontable**, which changes situations by putting a block on the table:

$$\forall s \forall x [\neg On(x, Table, s) \Rightarrow On(x, Table, Putontable(x,s))]$$

Need to know also something is not on the table if it is on something that is not a table:

$$\forall s \forall y \forall z [On(y,z,s) \wedge \neg Equal(z, Table) \Rightarrow \neg On(y, Table, s)]$$

Situation Calculus (Cont'd)

Initial axioms:

On(B, A, S)
On(A, Table, S)
$\forall s\ \forall x[\neg On(x,Table,s) \Rightarrow On(x,Table,Putontable(x,s))]$
$\forall s \forall y \forall z[On(y,z,s)\ \Lambda \neg Equal(z,Table) \Rightarrow \neg On(y,Table,s)]$

The theorem to be proved:

$\exists\ s_f\ [On(B, Table, s_f)]$

Negating the theorem:

$\forall\ s_f\ [\neg On(B, Table, s_f)]$

Conversion to CNF:

On(B, A, S) (1)

On(A, Table, S) (2)

$On(x,Table, s_3)$ V $On(x,Table,Putontable(x, s_3))$ (3)

$\neg On(y,z, s_4)$ V Equal(z,Table) V $\neg On(y,Table, s_4)$ (4)

¬Equal(B, A) (5)

¬Equal(B,Table) (6)

¬Equal(A,Table) (7)

$\neg On(B, Table, s_f)$ (8)

Situation Calculus (Cont'd)

(3)
$On(x, Table, s_3)$ V
On(x, Table, Putontable(x, s_3))

(8)
¬On(B, Table, s_f)

(9)
On(B, Table, s_9)

(4)
¬On(y, z, s_4) V
Equal(z, Table) V
$\neg On(y, Table, s_4)$

(10)
$\neg On(B, w, s_{10})$ V
Equal(w, Table)

(7)
¬Equal(A,Table)

(11)
¬On(B, A, s_{11})

(1)
On(B, A, S)

(12)
Nil

Situation Calculus (Cont'd)

(3)
On(x, Table, s_3) ∨
On(x, Table,
Putontable(x, s_3))

(8)
¬On(B, Table, s_f) ∨
Answer(s_f)

(9)
On(B, Table, s_9) ∨
Answer(Putontable(B, s_9))

(4)
¬On(y, z, s_4) ∨
Equal(z, Table) ∨
¬On(y, Table, s_4)

(10)
¬On(B, w, s_{10}) ∨
Equal(w, Table) ∨
Answer(Putontable(B, s_{10}))

(7)
¬Equal(A,Table)

(11)
¬On(B, A, s_{11}) ∨
Answer(Putontable(B, s_{11}))

(1)
On(B, A, S)

(12)
Answer(Putontable(B, S))

The Frame Problem

A need to describe the parts of the world that stay the same after certain operations were performed.

Suppose that the problem to be solved is similar to the previous one, but requiring both block B and block A to be on the table in the final situation. That is:

$$\exists\, s_f\, [\mathrm{On}(B, \mathrm{Table}, s_f) \wedge \mathrm{On}(A, \mathrm{Table}, s_f)]$$

Negating the theorem gives:

$$\neg\mathrm{On}(B, \mathrm{Table}, s_f)] \vee \neg\mathrm{On}(A, \mathrm{Table}, s_f) \qquad (8)$$

The Frame problem (Cont'd)

(3)
On(x, Table, S_3) V
On(x, Table,
Putontable(x, S_3))

(8)
¬On(B, Table, S_f) V
¬On(A, Table, S_f)

(4)
¬On(y, z, S_4) V
Equal(z, Table) V
¬On(y, Table, S_4)

(9)
On(B, Table, S_9) V
¬On(A,Table,Putontable(B, S_9))

(7)
¬Equal(A,Table)

(10)
¬On(B, w, S_{10}) V
Equal(w, Table) V
¬On(A,Table,Putontable(B, S_{10}))

(1)
On(B, A, S)

(11)
¬On(B, A, S_{11}) V
¬On(A,Table,Putontable(B, S_{11}))

(2)
On(A, Table, S)

(12)
¬On(A,Table,Putontable(B, S))

S is a constant (the initial situation), NO substitution is possible!

Frame Axiom

The Putontable axiom says what happens to the object being moved, but it does not say what happens to other objects. We know block A stays on the table after block B is put on the table, but that piece of knowledge is not in the logic! The system wouldn't know which predicate's values are unchanged in going from one situation to another.

Frame axiom: A statement about how predicates survive operations. **A frame axiom tells how On survives Putontable:**

$$\forall s\ \forall x\ \forall y\ \forall z\ [\ On(x, y, s)\ \Lambda\ \neg Equal(x, z)\ \Rightarrow\ On(x, y, Putontable(z, s))\ ]$$

It means: if x is on y before the Putontable operation, then x remains on y afterward, as long as x was not the object put on the table. In CNF:

¬On(p, q, S_0) V Equal(p, r) V On(p, q, Putontable(r, S_0))

Resolving it with (12): ¬On(A,Table,Putontable(B, S))

¬On(A, Table, S) V Equal(A, B)

Resolving (13) with (5) and the result with (2), a Nil is obtained. Thus completed the proof.