32002: AI (Unification Algorithm, Theorem Proving using Resolution) **Spring 2014**

Lecture 11: January 28, 2014

Lecturer: K.R. Chowdhary *: Professor of CS (GF)*

Disagreement Set. If A is any set of well-formed expressions, we call the set D the disagreement set of A, whenever D is the set of all well-formed subexpressions of the well-formed expressions in A, which begin at the first symbol position at which not all well-formed expressions in A have the same symbol.

Example 11.1 *Find out the disagreement set for given set of wffs.*

Solution. Let the string is, $A = \{P(x, h(x, y), y),\ P(x, k(y), y),\ P(x, a, b)\}$, having three predicate expressions. The disagreement set for A is,

$$D = \{h(x, y), k(y), a\}. \tag{11.1}$$

For A to be a finite nonempty set of well-formed expressions for which the substitution algorithm (1) terminates with "return σ_A", the substitution σ_A available as output of the unification algorithm is called the most general unifier (mgu) of A, and A is said to be most generally unifiable.

Algorithm 1 Algorithm-Unification(Input: A, Output: σ_A)

```
1: Set σ_0 = ϵ, k = 0
2: while true do
3:    if Aσ_k is a singleton then
4:       Set σ_A = σ_k
5:       terminate
6:    end if
7:    Let U_k be the earliest and V_k be the next earliest, in the disagreement set D_k of Aσ_k (see 11.1)
8:    if U_k is a variable, and does not occur in V_k then
9:       set σ_{k+1} = σ_k{V_k/U_k},
10:      k = k + 1
11:   else
12:      (A is not unifiable)
13:      exit.
14:   end if
15: end while
```

Through manually running the algorithm 1 for the disagreement set 11.1, stepwise computation for σ_k is as follows:

$$\sigma_{k+1} = \sigma_k\{k(y)/h(x, y)\}$$
$$\Rightarrow \sigma_1 = \{k(y)/h(x, y)\}$$

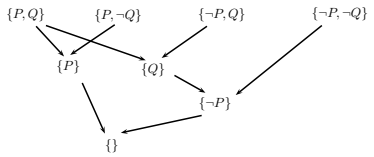

Figure 11.1: DAG for Theorem Proving.

which, in the next iteration becomes,

$$\begin{aligned} \sigma_2 &= \sigma_1\{a/k(y)\} \\ \Rightarrow \sigma_2 &= \{k(y)/h(x,y)\}\{a/k(y)\} \end{aligned}$$

The same process is repeated for the disagreement set of 3rd argument in A, which results to substitution set as $\{b/y\}$. For obtaining the unifier σ_k, the necessary relation required between U_k and V_k is, U_k has to be a variable, and V_k can be a constant, variable, function, or predicate. U_k may even be a predicate or function with variable.

11.1 Resolution Principle

The Resolution principle is mathematical oriented rather than human oriented. It is quite powerful both in the psychological sense that it condones single inferences which are often beyond the ability of human to grasp, and in theoretical sense that it alone, as sole inference principle, forms a complete system of FOPL. Because of only one inference type, it allows to avoid the combinatorial obstacles to efficiency.

The resolution method is a procedure for checking whether a set of clauses, Γ, is *unsatisfiable*. The resolution methods consist in building a certain kind of labeled *DAG* (Directed Acyclic Graph) whose leaves are labeled with clauses in Γ and whose interior nodes are labeled according to the resolution rule.

Given two clauses $C = A \cup \{P\}$ and $C' = B \cup \{\neg P\}$ (where P is a propositional letter, $P \notin A$ and $\neg P \notin B$), the resolvent of C and C' is the clause $R = A \cup B$ obtained by canceling out P and $\neg P$. A resolution DAG for Γ is a DAG whose leaves are labeled with clauses from Γ and such that every interior node n has exactly two predecessors, n_1 and n_2 so that n is labeled with the resolvent of the clauses labeling n_1 and n_2. A resolution refutation for Γ is a resolution DAG with a single root whose label is the empty clause.

Example 5.1 A resolution refutation for the set of clauses

$$\Gamma = \{\{P, Q\}, \{P, \neg Q\}, \{\neg P, Q\}, \{\neg P, \neg Q\}\}.$$

□

is shown in figure 11.1.

11.2 Theorem Proving Formalism

It is a syntactic inference procedure, when applied to clauses, determines, if the satisfied set is unsatisfiable. Proof is similar to proof by *contradiction* and deduce [] (i.e., null). If for example, we have set of clauses

(axioms) $C_1, C_2, \ldots, C_n$, and we want to deduce D, i.e., D is logical consequence of of $C_1, C_2, \ldots, C_n$. For this we add $\neg D$ to the set $\{C_1, C_2, \ldots, C_n\}$, then we show that set is unsatisfiable by deducing contradiction.

Let $C_1 = \neg P \vee Q$, and $C_2 = \neg Q \vee R$, then following can be deduced through resolution,

$$\frac{P \Rightarrow Q, Q \Rightarrow R}{P \Rightarrow R} \tag{11.2}$$

equivalently,

$$\frac{(\neg P \vee Q), (\neg Q \vee R)}{\therefore (\neg P \vee R)} \tag{11.3}$$

It can be easily verified that $(\neg P \vee Q) \wedge (\neg Q \vee R) \models (\neg P \vee R)$, hence $(\neg P \vee Q) \wedge (\neg Q \vee R) \Rightarrow (\neg P \vee R)$ is a valid statement. Thus, $\neg P \vee R$ is inference or the resolvent. Arriving to a proof by above is called proof by *refutation.*

11.3 Proof by Resolution

To prove a theorem, one obvious strategy is to search forward from the axioms, using sound rules of inference. Hence, we try to prove a theorem by refutation. It requires to show that negation of a theorem cannot be True. The steps for a proof by resolution are:

1. Assume that negation of the theorem is True.
2. Try to show that axioms and assumed negation of theorem, together are True, which cannot be True.
3. Conclude that above leads to contradiction.
4. Conclude that theorem is True because its negation cannot be True.

Algorithm 2 Algorithm-Resolve(Input: α(the theorem to be proved), β(the set of axioms))

```
1: β = β ∪ {¬α}
2: Γ = clausal form of β
3: while there is a resolvable pair of clauses C_i, C_j ∈ Γ do
4:    C = resolve(C_i, C_j)
5:    Γ = Γ ∪ {C}
6:    if C = NIL then
7:       return "Theorem α is true"
8:    end if
9: end while
10: Report that theorem is False
```

Example 5.2. Following axioms are about the observed block relationship shown in figure 11.2.

$$on(cylinder, box)$$
$$on(box, table)$$

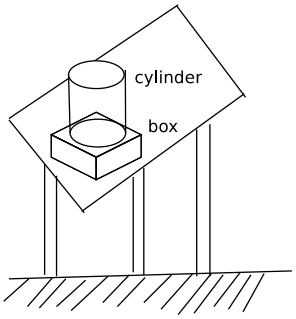

Figure 11.2: Objects on Table.

These are already in the clausal form, hence need not to be converted to the clausal form.

It is required to show that object *cylinder* is above table, i.e., $above(cylinder, table)$, given the the following:

- $\forall x \forall y[on(x, y) \rightarrow above(x, y)]$, and
- $\forall x \forall y \forall z[above(x, y) \wedge above(y, z) \rightarrow above(x, z)]$.

After we have gone through the procedure for conversion to clausal form, the above axioms get transformed to:

- $\neg on(u, v) \vee above(u, v)$
- $\neg above(x, y) \vee \neg above(y, z) \vee above(x, z)$

The expression to be proved is "above(cylinder, table)"; its negation is $\neg above(cylinder, table)$. Let us list all the clauses systematically.

(1) $\neg on(u, v) \vee above(u, v)$

(2) $\neg above(x, y) \vee \neg above(y, z) \vee above(x, z)$

(3) $on(cylinder, box)$

(4) $on(box, table)$

(5) $\neg above(cylinder, table)$

Now, we manually run the algorithm 2 on the clauses (1)-(5), as well as those which would created new, to unify them according to unification algorithm 1, until we reach to a null resolvent.

We resolve clauses (2) and (5) and bind x to '*cylinder*'and z to '*table*'. Applying the resolution, we get resolvent (6). Unifier for this is $\{cylinder/x, table/z\}$.

(2) $\neg above(cylinder, y) \vee \neg above(y, table) \vee above(cylinder, table)$

(5) $\neg above(cylinder, table)$

(6) $\neg above(cylinder, y) \vee \neg above(y, table)$

Now, we resolve (1) with (6), instantiating u with y and instantiating (called binding) v with '*table*', we get (7). Unifier for this is $\{y/u, table/v\}$.

(1) $\neg on(y, table) \vee above(y, table)$

(6) $\neg above(cylinder, y) \vee \neg above(y, table)$

(7) $\neg on(y, table) \vee \neg above(cylinder, y)$

We use (1) again with (7) with u instantiated to $cylinder$ and v replaced y. Unifier for this is $\{cylinder/u, y/v\}$.

(1) $\neg on(cylinder, y) \vee above(cylinder, y)$

(7) $\neg on(y, table) \vee \neg above(cylinder, y)$

(8) $\neg on(cylinder, y) \vee \neg on(y, table)$

Next use (3) and (8), instantiating y to box, with unifier $\{box/y\}$. a set with exactly one element

(3) $on(cylinder, box)$

(8) $\neg on(cylinder, box) \vee \neg on(box, table)$

(9) $\neg on(box, table)$

Finally, the (4) and (9) are resolved to get NIL:

(4) $on(box, table)$

(9) $\neg on(box, table)$

(10) NIL

Since we have arrived at the contradiction, so that negation of the theorem: $\neg above(cylinder, table)$ must be False. Hence the theorem $above(cylinder, table)$ must be True. □

References

[1] Chowdhary K.R. (2020) Logic and Reasoning Patterns. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_3