Logic for AI

Master 1 IFI

Andrea G. B. Tettamanzi

Nice Sophia Antipolis University

Computer Science Department

andrea.tettamanzi@unice.fr

Session 4

Unification and Resolution

Agenda

- Substitution
- Unification
- Resolution for Predicate Logic

Introduction

- In the first session, we introduced resolution in propositional logic
- We shall now extend it to (first-order/Herbrand) predicate logic
- The most important part of applying the resolution principle is finding a literal in a clause that is complementary to a literal in another clause
- For clauses containing no variables, this is very simple
- However, for clauses containing variables, it is more complicated

Motivating Example

Clause 1: $P(x) \vee Q(x)$

Clause 2: $\neg P(f(x)) \vee R(x)$

$\Downarrow$

Clause 1': $P(f(a)) \vee Q(f(a))$

Clause 2': $\neg P(f(a)) \vee R(a)$

$\Rightarrow$ $Q(f(a)) \vee R(a)$

Clause 1*: $P(f(x)) \vee Q(f(x))$

Clause 2*: $\neg P(f(x)) \vee R(x)$

$\Rightarrow$ $Q(f(x)) \vee R(x)$

Substitution

- A **substitution** is a finite set of the form

$$\{t_1/v_1, \ldots, t_n/v_n\}$$

  where
  - Every v_i is a variable
  - Every t_i is a term different from v_i
  - All variables v_i are different
- When t_1, …, t_n are ground terms, the substitution is called a **ground substitution**.
- We denote by φθ the application of substitution θ to sentence φ

Composition of Substitutions

Let

$$\theta = \{t_1/x_1, \ldots, t_n/x_n\}$$

$$\lambda = \{u_1/y_1, \ldots, u_m/y_m\}$$

Then

$$\theta \circ \lambda = \{t_1\lambda/x_1, \ldots, t_n\lambda/x_n, u_1/y_1, \ldots, u_m/y_m\}$$

by deleting any element $t_j\lambda/x_j$ such that $t_j\lambda = x_j$

and any element u_i/y_i such that $y_i \in \{x_1, \ldots, x_n\}$

In other words: $\phi[\theta \circ \lambda] = (\phi\theta)\lambda$

Example

Let

$$\theta = \{f(y)/x, z/y\}$$

$$\lambda = \{a/x, b/y, y/z\}$$

Then

$$\begin{aligned} \theta \circ \lambda &= \{f(b)/x, y/y, a/x, b/y, y/z\} \\ &= \{f(b)/x, y/z\} \end{aligned}$$

Monoid of Substitutions

- The composition of substitutions is associative

$$(\theta \circ \lambda) \circ \mu = \theta \circ (\lambda \circ \mu)$$

- The empty substitution is both a left and right neutral element

$$\epsilon \circ \theta = \theta \circ \epsilon = \theta$$

Unifier

- A substitution θ is called a **unifier** for a set $\{\varphi_1, \ldots, \varphi_n\}$ if and only if $\varphi_1\theta = \ldots = \varphi_n\theta$
- The set $\{\varphi_1, \ldots, \varphi_n\}$ is said to be **unifiable** if there exists a unifier for it

- A unifier σ for a set $\{\varphi_1, \ldots, \varphi_n\}$ of sentences is a **most general unifier** (MGU) if and only if, for each unifier θ for the set, there exists a substitution λ such that

$$\theta = \sigma \circ \lambda$$

Disagreement Set

The **disagreement set** of a nonempty set W of sentences is obtained by:

- Locating the first symbol (counting from the left) at which not all the sentences in W have exactly the same symbol
- Extracting from each expression in W the subexpression that begins with the symbol occupying that position

The set of these respective subexpressions is the disagreement set of W.

Example: W = { P(x, f(y, z)), P(x, a), P(x, g(h(k(x)))) }

Disagreement set: { f(y, z), a, g(h(k(x))) }.

Unification Algorithm

1) Set $k = 0$, $W_k = W$, and $\sigma_k = \varepsilon$
2) If W_k is a singleton, then **STOP**: σ_k is a MGU for W
   Else find the disagreement set D_k of W_k
3) If there exist elements v_k and t_k in D_k such that v_k is a variable that does not occur in t_k, then continue to Step 4.
   Else **STOP**: W is not unifiable
4) Let
$$\sigma_{k+1} = \sigma_k \circ \{t_k/v_k\}$$
$$W_{k+1} = W_k\{t_k/v_k\}$$
5) Set $k = k + 1$ and go back to Step 2

Unification Theorem

If W is a finite nonempty unifiable set of expressions, then the unification algorithm will always terminate at Step 2 and the last σ_k is a MGU for W

Proof (sketch):

- Since W is unifiable, we let θ be any unifier for W
- We show by induction on k that there is a substitution λ_k such that

$$\theta = \sigma_k \circ \lambda_k$$

Factor

- If two or more literals (with the same sign) of a clause C have a MGU σ, then Cσ is called a **factor** of C
- If Cσ is a unit clause, it is called a **unit factor** of C

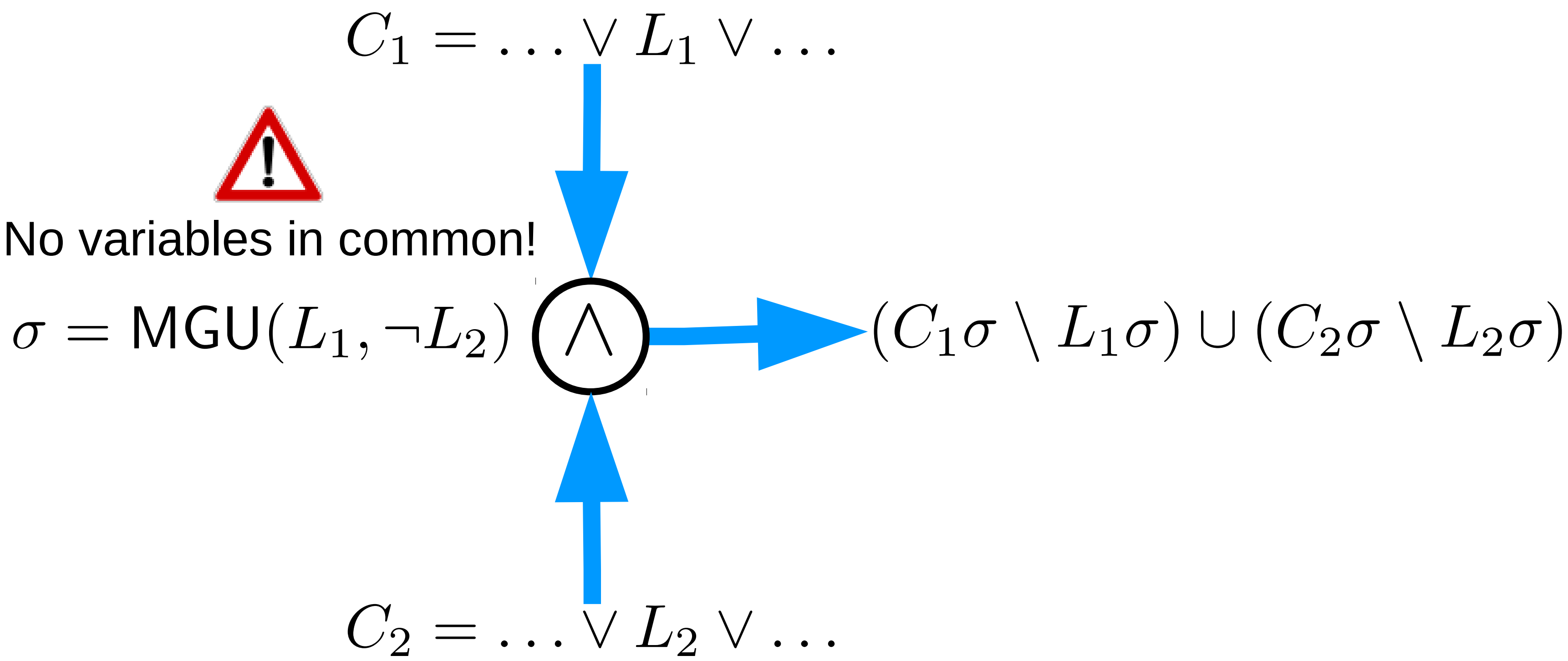
Example:

$$C = \underline{P(x) \vee P(f(y))} \vee \neg Q(x)$$

$$\sigma = \{f(y)/x\}$$

$$C\sigma = P(f(y)) \vee \neg Q(x)$$

Binary Resolvent



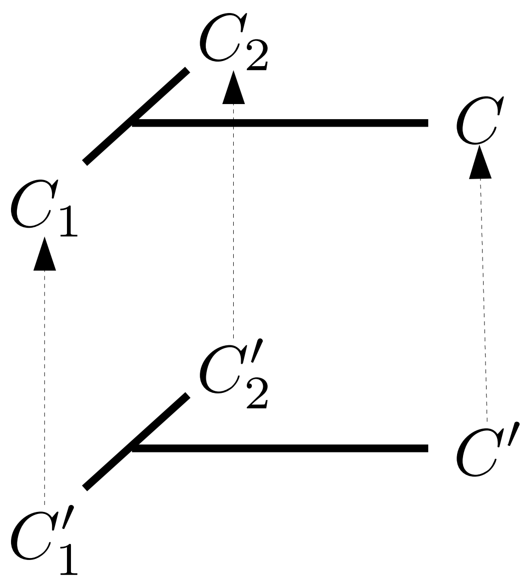
Resolvent

A resolvent of (parent) clauses C_1 and C_2 is one of the following binary resolvents:

- A binary resolvent of C_1 and C_2
- A binary resolvent of C_1 and a factor of C_2
- A binary resolvent of a factor of C_1 and C_2
- A binary resolvent of a factor of C_1 and a factor of C_2

Lifting Lemma

If C_1' and C_2' are instances of C_1 and C_2, respectively, and if C' is a resolvent of C_1' and C_2', then there exists a resolvent C of C_1 and C_2 such that C' is an instance of C.



Completeness of Resolution

A set S of clauses is unsatisfiable if and only if there is a deduction of the empty clause F from S.

Proof (sketch):

- [⇒]: Suppose S is unsatisfiable; let T a complete semantic tree for S; T has a finite closed semantic tree T'. Use structural induction on T' together with the Lifting Lemma to show that there is a deduction of the empty clause from S
- [⇐]: Suppose there is a deduction of F. Let R_1, …, R_k be the resolvents in the deduction. Assume S is satisfiable. Then, there is a model M of S. If M |= C_1 and C_2, it also |= any resolvent; then M |= R_1, …, R_k; then M |= F, which is impossible!

Thank you for your attention