

## 8 General first order representation

### 8.1 First order language

#### Propositional core

- **constants**  $A, B, C, D$
- **predicates**  $on(\cdot, \cdot)$ 
  - associated arity, e.g., arity of  $on$  is 2
- **atomic ground sentence** is equivalent to atomic proposition; it is created by applying predicate to constants; e.g.,  $on(A, B)$
- **true and false symbols**  $\top, \perp$
- **compound ground sentence** built using operations:  $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ ; e.g.,  $on(A, B) \wedge \neg on(B, C)$

This is equivalent to propositional logic. For example, instead of writing  $on(A, B)$ , we used the notation  $AonB$ .

#### Extension: propositional schema

Instead of repeating the same rules for  $AonB$ ,  $BonC$ , etc., we can use variables and the predicate  $on(x, y)$ .

- **variables**  $x, y, z$
- **atomic formula** apply predicate to constants and variables; e.g.,  $on(A, x), on(A, B), on(x, y)$
- **compound formula** use operators of propositional calculus; e.g.,  $on(A, X) \wedge \neg on(x, B)$
- **special equality predicate**  $=$   
(Note that instead of writing  $= (x, B)$ , write  $x = B$ )
- **quantifiers**  $\forall$  and  $\exists$ . If  $\varphi(\dots x \dots)$  is a formula, then  $\forall x \varphi(\dots x \dots)$  and  $\exists x \varphi(\dots x \dots)$  are formulas.

In such a formula,  $x$  is called a **bound** variable, and variables that are not bound are **free**.

- **open formula:** has free variables.
- **closed formula:** no free variables (all variables bound). A closed formula is also called a **sentence**.

Note: Sentences have truth values (open formulas do not).

This defines the basic first order language. Note that the first order language strictly extends the representational capacity of propositional logic.

### Further extension: functions

- **function symbols**  $f(\cdot, \cdot)$
- **term** composition of functions, constants, and variables; e.g.

$$f(g(x), h(A, B))$$

- **ground term** is a term with no variables. Ground terms can represent complex objects like strings, trees, expressions.
- **open term** contains variables (a term does not contain quantifiers)

## 8.2 Formal inference rules

Operate on sentences. For example:

$\forall$  elimination:

$$\frac{\forall x \varphi(x)}{\varphi(t)} \quad \text{for any ground term } t$$

$\exists$  elimination:

$$\frac{\exists x \varphi(x)}{\varphi(A')} \quad \text{for a new constant } A'$$

$\exists$  introduction:

$$\frac{\varphi(t)}{\exists x \varphi(x)} \quad t \text{ is a ground term}$$

### 8.3 A simple formal inference system: Resolution

Rules: resolution, substitution, simplification

Assume facts (sentences) represented in *clausal form*:

#### Clausal form

$$\forall x_1 \dots \forall x_n \quad \neg p_1(t_1) \vee \dots \vee \neg p_k(t_k) \vee q_1(t_{k+1}) \vee \dots \vee q_\ell(t_{k+\ell})$$

where the  $t_i$  are terms that have all variables  $x_j$  bound by a  $\forall$

**Conjunctive normal form** Conjunction of clauses

$$\forall x_1 \dots \forall x_n \quad c_1 \wedge c_2 \wedge \dots \wedge c_m$$

where each  $c_i$  is in clausal form

Note: any sentence can be converted into conjunctive normal form:

- eliminate implications ( $\rightarrow$ ) and equivalences ( $\leftrightarrow$ )
- move negations inward (DeMorgan's laws)
- standardize variables apart
- move  $\forall$  and  $\exists$  to left (in order)
- remove  $\exists$  by "Skolemization":  
Go from outside in, and for each  $\forall x_1 \forall x_2 \dots \forall x_n \exists y$ , eliminate the  $\exists y$  by substituting  $y$  with  $f'(x_1, x_2, \dots, x_n)$ , where  $f'$  is a new function symbol (or constant symbol if  $n = 0$ )
- distribute  $\wedge$  over  $\vee$
- flatten, e.g.,  $(p \vee q) \vee r$  becomes  $(p \vee q \vee r)$
- drop  $\forall x$  and assume all variables are universal

(The general conversion procedure can be found in Russell and Norvig, P.296-297.)

**Inference rules****Specialization (substitution):**

$$\frac{\varphi}{[\varphi]_{x/t}}$$

replace  $x$  by term  $t$

**Resolution:**

$$\frac{\alpha \vee \neg p(\underline{v}) \quad \beta \vee p(\underline{v})}{\alpha \vee \beta}$$

**Simplification:**

$$\frac{\alpha \vee \neg p \vee \neg p}{\alpha \vee \neg p} \quad \frac{\alpha \vee p \vee p}{\alpha \vee p} \quad \frac{\alpha \vee \neg p \vee p}{\top}$$

(same as for propositional logic)

**Example**

- |  |   |
|--|---|
| 1: $\neg on(x, y) \vee \neg on(y, z) \vee \neg on(z, x)$ | given                                   |
| 2: $on(A, A) \vee on(B, B)$                              | given                                   |
| 3: $\neg on(w, w) \vee \neg on(w, w) \vee \neg(w, w)$    | apply substitution $x/w, y/w, z/w$ on 1 |
| 4: $\neg on(w, w)$                                       | apply simplification on 3               |
| 5: $\neg on(A, A)$                                       | substitution $w/A$ on 4                 |
| 6: $on(B, B)$  | resolution on 2 and 5                   |
| 7: $\neg on(B, B)$                                       | substitution $w/B$ on 4                 |
| 8: $\perp$   | resolution on 6 and 7                   |
- Contradiction!

## 8.4 Unification

Unification is matching formulas by substitution (specialization)

### Example

- |  |                         |
|--|-------------------------|
| 1: $\forall x P(A, x)$                             | given                   |
| 2: $\forall y \forall z \neg P(y, z) \vee P(z, y)$ | given                   |
| 3: $\forall z P(A, z)$                             | substitution $x/z$ on 1 |
| 4: $\forall z \neg P(A, z) \vee P(z, A)$           | substitution $y/A$ on 2 |
- Two sub-formulas are unified, and we can apply resolution:
- 5:  $\forall z P(z, A)$

It is useful to have procedure  $\text{UNIF}(\varphi, \chi)$  which returns either:

- a substitution (binding list)  $s = \{x_1/y_1, \dots, x_n/y_n\}$  such that  $[\varphi]_s = [\chi]_s$ , or
- fail, if none exists

$s$  is a unifier of  $\varphi$  and  $\chi$

### Example

$$\begin{aligned}
 \text{UNIF}(\text{on}(A, x), \text{on}(A, B)) &= x/B \\
 \text{UNIF}(\text{on}(A, x), \text{on}(y, B)) &= x/B, y/A \\
 \text{UNIF}(\text{on}(A, x), \text{on}(y, f(y))) &= y/A, x/f(A) \\
 \text{UNIF}(\text{on}(x, y), \text{on}(y, f(y))) &= \text{fail} \\
 \text{UNIF}(\text{on}(A, x), \text{on}(x, B)) &= \text{fail} \quad (\text{but could standardize apart}) \\
 \text{UNIF}(\text{on}(x, f(x)), \text{on}(g(y), y)) &= \text{fail}
 \end{aligned}$$

## 8.5 Most general unifier

There can be more than one substitution; e.g.

$$\text{UNIF}(\text{on}(A, x), \text{on}(y, z))$$

could return  $\{y/A, x/z\}$ , or  $\{y/A, z/x\}$ , or  $\{y/A, x/B, z/B\}$ , or  $\{y/A, x/A, z/A\}$ , etc.

We are interested in the *most general unifier*:

$$\text{MGU}(\text{on}(A, x), \text{on}(y, z)) = \{y/A, x/z\}$$

Most General Unifier:

- makes least commitments
- exists and is unique up to variable renaming (if the terms are unifiable)
- can be turned into any other unifier by applying additional substitutions

With MGU, we can merge substitution and resolution into one step:

### Resolution with unification

$$\frac{\alpha \vee \neg p(\underline{v}) \quad \beta \vee p(\underline{u})}{[\alpha \vee \beta]_{\text{MGU}(p(\underline{v}), p(\underline{u}))}}$$

### Unification Algorithm

- Robinson 1965: exponential algorithm
- Huet 1976: almost linear time algorithm  $O(n\alpha(n))$
- Paterson and Wegman 1976: improved Huet's algorithm to linear time

A few more examples:

$$\text{UNIF}(f(x, g(y, y), x), f(z, z, g(w, f(t)))) = ?$$

Answer:  $x/g(f(t), f(t)) \quad z/g(f(t), f(t)) \quad y/f(t) \quad w/f(t)$

$$\text{UNIF}(f(x, h(A, t), g(x)), f(g(y), h(z, y), t)) = ?$$

Answer: fail

## Robinson's algorithm

---

**Algorithm 1** UNIFY( $t_1, t_2$ )

---

```
1: current substitution  $\leftarrow$  empty substitution
2: if  $t_1$  or  $t_2$  is a variable then
3:   if  $t_1 = t_2$  then
4:     return (true, empty substitution)
5:   end if
6:   let  $t_1$  be variable
7:   if  $t_1$  occurs in  $t_2$  then
8:     return false
9:   end if
10:  return (true,  $t_1/t_2$ )
11: else
12:  if terms  $t_1$  and  $t_2$  cannot be matched then
13:    return false
14:  end if
15:   $t_1 = f(a_1, \dots, a_n), t_2 = f(b_1, \dots, b_n)$ 
16:  for all pairs  $(a_i, b_i)$  ( $i = 1 \dots n$ ) do
17:    make current substitution on  $a_i$  and  $b_i$  and call UNIFY on them
18:    if UNIFY successful then
19:      apply returned substitution to current substitution
20:    else
21:      return false
22:    end if
23:  end for
24: end if
25: return (true, current substitution)
```

---

## Readings

Russell and Norvig: Chapters 8 and 9

Genesereth and Nilsson: Chapter 4

Burris: 3.12

Knight, Kevin: Unification: a multidisciplinary survey, *ACM computing surveys*, March 1989