

Logic Programming

Cezary Kaliszyk **Georg Moser**

Institute of Computer Science @ UIBK

Winter 2015



Outline

Outline of the Lecture

Monotone Logic Programs

introduction, **basic constructs**, **unification**, database and recursive programming, termination

Incomplete Data Structures and Constraints

incomplete data structures, definite clause grammars, constraint logic programming, answer set programming

Full Prolog

semantics, correctness proofs, meta-logical predicates, cuts nondeterministic programming, efficient programs, complexity

Summary of Last Lecture

Definition

- **goals** (aka formulas) are constants or compound terms
- goals are typically non-ground

Definitions

- a **clause** or **rule** is an universally quantified logical formula of the form

$$A :- B_1, B_2, \dots, B_n.$$
 where A and the B_i 's are goals
- A is called the **head** of the clause; the B_i 's are called the **body**
- a rule of the form $A :-$ is called a **fact**; we write facts simply A .

Definition

a **logic program** is a finite set of clauses

Basic Constructs

Example

```

child_of(joseph_I , leopold_I).
child_of(karl_VI , leopold_I).
child_of(maria_theresia , karl_VI).
child_of(joseph_II , maria_theresia).
child_of(joseph_II , franz_I).
child_of(leopold_II , maria_theresia).
child_of(leopold_II , franz_I).
child_of(maria_antoinette , maria_theresia).
child_of(franz_II , leopold_II).

male(franz_I).          female(maria_theresia).
male(franz_II).        female(marie_antoinette).
male(joseph_I).
male(joseph_II).
male(karl_VI).
male(leopold_I).
male(leopold_II).

husband_wife(franz_I , maria_theresia).
  
```

Review of Basic Constructs

Definitions

- a **fact** describes a relation (predicate) between terms

```
child_of(joseph_II , maria_theresia).
```

 which reads “Joseph II is the child of Maria Theresa.”
- child_of is the name of the relation
- the **arity** denotes the number of arguments
- predicates are also denoted as child_of/2
- fact that do not contain variables are **ground**

Fact

the order of the arguments is essential, hence it is important to choose meaningful names for predicates

Choosing Names

1 describe the arguments

```
typ1_typ2_typ3_typ4 (Arg1 , Arg2 , Arg3 , Arg4)
```

2 refine the name

```
person_person (X,Y).           % too coarse
child_person (Child , Person)  % better
child_parent (Child , Parent)  % perfect
```

3 indicate the relation

```
child_ofparent (Child , Parent) % preposition
expression_improvedprogram (Exp , IExp) % participle
expr_improved (Exp , IExp)
consists_of (X,Y)                % verb
```

4 abbreviations

```
country_/8
```

Definition

- a **query** tests whether a relation holds

```
:- child_of(joseph_II , maria_theresia).
```
- queries are equivalent to **use cases**, as they are checked whenever the program is safed (in GUPU) or compiled (in general)

Why does a Query fail?

- 1 the query doesn't follow from the data represented in the program; the negation of the query does not necessarily hold
- 2 the program is a complete representation; the negation of the query does hold

Fact

Horn logic cannot distinguish between these options

Definition

a **negative query** verifies that the goal fails

```
:- child_of(joseph_II , friedrich_II).
```

NB: occurring variables are **existentially** quantified

Definition

a **general query** with variables provide answer substitutions

```
:- child_of(Child , maria_theresia).
:- child_of(_Child , maria_theresia).
:- child_of(Child , Child).
```

Definition

a **complex query** combines several goals and typically make use of **shared variables**

```
:- child_of(joseph_II , Mum), female(Mum).
```

Definition

- a **rule** consists of a head and a body, separated by “:-”

```
mother_of(Mum, Child) :-
  child_of(Child, Mum),
  female(Mum).
```

- a rule is **recursive**, if the body contains the predicate in the head

Definitions

- we distinguish between the **set of solutions** of a query and the **sequence of solutions**
- the sequence may contain redundant solutions
- redundant solutions may be due to existential variables

Example

```
% recursive rule
married_with(Husband, Wife) :-
  husband_wife(Husband, Wife).
married_with(PersonA, PersonB) :-
  married_with(PersonB, PersonA).
```

```
% non-recursive rule
married_with(Husband, Wife) :-
  husband_wife(Husband, Wife).
married_with(Wife, Husband) :-
  husband_wife(Husband, Wife).
```

Example

```
:- ancestor_of(X,X).
```

```
ancestor_of(Ancessor, Predecessor) :-
  child_of(Predecessor, Ancessor).
ancestor_of(Ancessor, Predecessor) :-
  child_of(Person, Ancessor),
  ancestor_of(Person, Predecessor).
```

Example

```
:- ancestor_of_2(X,X).
```

```
ancestor_of_2(Ancessor, Predecessor) :-
  child_of(Predecessor, Ancessor).
ancestor_of_2(Ancessor, Predecessor) :-
  ancestor_of_2(Person, Predecessor),
  child_of(Person, Ancessor).
```

Definition

composition of substitutions

$$\theta = \{X_1 \mapsto t_1, \dots, X_n \mapsto t_n\}$$

and

$$\sigma = \{Y_1 \mapsto s_1, \dots, Y_k \mapsto s_k\}$$

is substitution

$$\theta\sigma = \{X_1 \mapsto t_1\sigma, \dots, X_n \mapsto t_n\sigma\} \cup \{Y_i \mapsto s_i \mid Y_i \notin \{X_1, \dots, X_n\}\}$$

Example

$$\theta = \{X \mapsto g(Y, Z), Y \mapsto a\} \quad \theta\sigma = \{X \mapsto g(Y, f(X)), Y \mapsto a, Z \mapsto f(X)\}$$

$$\sigma = \{X \mapsto f(Y), Z \mapsto f(X)\} \quad \sigma\theta = \{X \mapsto f(a), Z \mapsto f(g(Y, Z)), Y \mapsto a\}$$

Definition

- substitution θ is **at least as general** as substitution σ if $\exists \mu \theta \mu = \sigma$
- **unifier** of set S of terms is substitution θ such that $\forall s, t \in S \ s\theta = t\theta$
- **most general unifier (mgu)** is at least as general as any other unifier

Example

terms $f(X, g(Y), X)$ and $f(Z, g(U), h(U))$ are unifiable:

$$\{X \mapsto h(a), Y \mapsto a, Z \mapsto h(a), U \mapsto a\}$$

$$\{X \mapsto h(U), Y \mapsto U, Z \mapsto h(U)\}$$

$$\{X \mapsto h(g(U)), Y \mapsto g(U), Z \mapsto h(g(U)), U \mapsto g(U)\}$$

mgu

Theorem

- *unifiable terms have mgu*
- \exists *algorithm to compute mgu*

Definition

- sequence $E = u_1 \stackrel{?}{=} v_1, \dots, u_n \stackrel{?}{=} v_n$ is called an **equality problem**
- if $E = X_1 \stackrel{?}{=} v_1, \dots, X_n \stackrel{?}{=} v_n$, with X_i pairwise distinct and $X_i \notin \text{Var}(v_j)$ for all i, j , then E is in **solved form**
- let $E = X_1 \stackrel{?}{=} v_1, \dots, X_n \stackrel{?}{=} v_n$ be a equality problem in solved form E induces substitution $\sigma_E = \{X_1 \mapsto v_1, \dots, X_n \mapsto v_n\}$

Unification Algorithm

$$u \stackrel{?}{=} u, E \Rightarrow E$$

$$f(s_1, \dots, s_n) \stackrel{?}{=} f(t_1, \dots, t_n), E \Rightarrow s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n, E$$

$$f(s_1, \dots, s_n) \stackrel{?}{=} g(t_1, \dots, t_n), E \Rightarrow \perp \quad f \neq g$$

$$X \stackrel{?}{=} t, E \Rightarrow X \stackrel{?}{=} t, E\{X \mapsto t\} \quad X \in \text{Var}(E), X \notin \text{Var}(t)$$

$$X \stackrel{?}{=} t, E \Rightarrow \perp \quad X \neq t, X \in \text{Var}(t)$$

$$t \stackrel{?}{=} X, E \Rightarrow X \stackrel{?}{=} t, E \quad t \notin \mathcal{V}$$

Theorem

- 1 *equality problems E is unifiable iff the unification algorithm stops with a solved form*
- 2 *if $E \Rightarrow^* E'$ such that E' is a solved form, then $\sigma_{E'}$ is mgu of E*

Example

$$f(X, g(Y), X) \stackrel{?}{=} f(Z, g(U), h(U)) \Rightarrow X \stackrel{?}{=} Z, g(Y) \stackrel{?}{=} g(U), X \stackrel{?}{=} h(U)$$

$$\Rightarrow X \stackrel{?}{=} Z, g(Y) \stackrel{?}{=} g(U), Z \stackrel{?}{=} h(U)$$

$$\Rightarrow X \stackrel{?}{=} Z, Y \stackrel{?}{=} U, Z \stackrel{?}{=} h(U)$$

$$\Rightarrow X \stackrel{?}{=} h(U), Y \stackrel{?}{=} U, Z \stackrel{?}{=} h(U) \quad \text{mgu}$$