

# Logic Programming

Cezary Kaliszyk    **Georg Moser**

Institute of Computer Science @ UIBK

Winter 2015



# Summary of Last Lecture

## Answer Set Programming

- novel approach to modelling and solving search and optimisation problems
- $\neg$  programming, but a specification language
- $\neg$  Turing complete
- purely declarative
- restricted to finite models

## Example ((part of) 8-queens problem)

```
:- not (1 = count(Y : queen(X,Y))), row(X)
```

- expresses that exactly one queen appears in every row and column
- is read as a rule: “if  $X$  is a row,  $1 = \text{count}(Y : \text{queen}(X,Y))$  holds”

## Grounders and Solvers



## Comparison

### Prolog

- proof search
- Turing complete
- control
- efficiency

### ASP

- model search
- finite domain
- specification language
- generality

# 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

# 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

# Theory of Logic Programs

# Theory of Logic Programs

## Definitions

- goal clause

$$\leftarrow B_1, \dots, B_n$$

consists of sequence  $B_1, \dots, B_n$  of goals

# Theory of Logic Programs

## Definitions

- goal clause

$$\leftarrow B_1, \dots, B_n$$

consists of sequence  $B_1, \dots, B_n$  of goals

- **empty** goal clause  $\leftarrow$  is denoted by  $\square$



# Theory of Logic Programs

## Definitions

- goal clause

$$\leftarrow B_1, \dots, B_n$$

consists of sequence  $B_1, \dots, B_n$  of goals

- empty goal clause  $\leftarrow$  is denoted by  $\square$
- **resolvent** of goal clause  $\leftarrow B_1, \dots, B_i, \dots, B_m$  and rule  $A \leftarrow A_1, \dots, A_n$  is goal clause

$$\leftarrow B_1\theta, \dots, B_{i-1}\theta, A_1\theta, \dots, A_n\theta, B_{i+1}\theta, \dots, B_m\theta$$

provided  $B_i$  (**selected goal**) and  $A$  unify with **most general unifier**  $\theta$

NB: see week 2 for the most general unifier

# Selective Linear Definite Clause Resolution

## Definitions

- SLD-derivation of logic program  $P$  and goal clause  $G$  consists of

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutions

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutionssuch that
  - $G_0 = G$

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutions

such that

- $G_0 = G$
- $G_{i+1}$  is resolvent of  $G_i$  and  $C_i$  with mgu  $\theta_i$

# Selective Linear Definite Clause Resolution

## Definitions

- **SLD-derivation** of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutions

such that

- $G_0 = G$
- $G_{i+1}$  is resolvent of  $G_i$  and  $C_i$  with mgu  $\theta_i$
- $C_i$  has no variables in common with  $G, C_0, \dots, C_{i-1}$



# Selective Linear Definite Clause Resolution

## Definitions

- SLD-derivation of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutionssuch that
  - $G_0 = G$
  - $G_{i+1}$  is resolvent of  $G_i$  and  $C_i$  with mgu  $\theta_i$
  - $C_i$  has no variables in common with  $G, C_0, \dots, C_{i-1}$
- **SLD refutation** is finite SLD derivation ending in  $\square$

# Selective Linear Definite Clause Resolution

## Definitions

- SLD-derivation of logic program  $P$  and goal clause  $G$  consists of
  - 1 maximal sequence  $G_0, G_1, G_2, \dots$  of goal clauses
  - 2 sequence  $C_0, C_1, C_2, \dots$  of variants of rules in  $P$
  - 3 sequence  $\theta_0, \theta_1, \theta_2, \dots$  of substitutions

such that

- $G_0 = G$
- $G_{i+1}$  is resolvent of  $G_i$  and  $C_i$  with mgu  $\theta_i$
- $C_i$  has no variables in common with  $G, C_0, \dots, C_{i-1}$
- SLD refutation is finite SLD derivation ending in  $\square$
- **computed answer substitution** of SLD refutation of  $P$  and  $G$  with substitutions  $\theta_0, \theta_1, \dots, \theta_m$  is restriction of  $\theta_0\theta_1 \cdots \theta_m$  to variables in  $G$

## Example

```
plus(0,X,X).
```

```
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

```
times(0,X,0).
```

```
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
```

```
:- times(X,X,Y)
```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
θ0:

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)
C1: times(0,X1,0).

```



## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)
C1: times(0,X1,0).
θ1: X0 ↦ 0, X1 ↦ s(0), U0 ↦ 0

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
      C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
      θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)
      C1: times(0,X1,0).
      θ1: X0 ↦ 0, X1 ↦ s(0), U0 ↦ 0
G2: :- plus(0,s(0),Y)
      C2: plus(0,X2,X2).
      θ2: X2 ↦ s(0), Y ↦ s(0)

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
    C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
    θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)
    C1: times(0,X1,0).
    θ1: X0 ↦ 0, X1 ↦ s(0), U0 ↦ 0
G2: :- plus(0,s(0),Y)
    C2: plus(0,X2,X2).
    θ2: X2 ↦ s(0), Y ↦ s(0)
G3: □

```

## Example

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U), plus(U,Y,Z).
:- times(X,X,Y)

```

## SLD-refutation

```

G0: :- times(X,X,Y)
      C0: times(s(X0),Y0,Z0) :- times(X0,Y0,U0), plus(U0,Y0,Z0)
      θ0: X ↦ s(X0), Y0 ↦ s(X0), Z0 ↦ Y
G1: :- times(X0,s(X0),U0), plus(U0,s(X0),Y)
      C1: times(0,X1,0).
      θ1: X0 ↦ 0, X1 ↦ s(0), U0 ↦ 0
G2: :- plus(0,s(0),Y)
      C2: plus(0,X2,X2).
      θ2: X2 ↦ s(0), Y ↦ s(0)
G3: □          computed answer substitution: X ↦ s(0), Y ↦ s(0)

```

## Definition

- a **selection function** selects the next goal  $G$  in goal clause, where resolution is attempted
- Prolog's selection function proceeds left to right

## Definition

- a **selection function** selects the next goal  $G$  in goal clause, where resolution is attempted
- Prolog's selection function proceeds left to right

## Theorem

$\forall$  logic programs  $P$  and goal clause  $G$

$\forall$  computed answer substitutions  $\theta$

## Definition

- a **selection function** selects the next goal  $G$  in goal clause, where resolution is attempted
- Prolog's selection function proceeds left to right

## Theorem

$\forall$  logic programs  $P$  and goal clause  $G$

$\forall$  computed answer substitutions  $\theta$

$\forall$  **selection functions**  $\mathcal{S}$

$\exists$  computed answer substitution  $\theta'$  using  $\mathcal{S}$

## Definition

- a **selection function** selects the next goal  $G$  in goal clause, where resolution is attempted
- Prolog's selection function proceeds left to right

## Theorem

$\forall$  logic programs  $P$  and goal clause  $G$

$\forall$  computed answer substitutions  $\theta$

$\forall$  **selection functions**  $S$

$\exists$  computed answer substitution  $\theta'$  using  $S$

such that  $\theta'$  is at least as general as  $\theta$  (with respect to variables in  $G$ )



# Search or SLD Trees

## Definition

a **search tree** (aka **SLD tree**) of a goal  $G$  is a tree  $T$  such that

- the root of  $T$  is labelled with  $G$
- the nodes of  $T$  are labelled with conjunctions of goals, where one goal is selected (wrt a selection function)

# Search or SLD Trees

## Definition

a **search tree** (aka **SLD tree**) of a goal  $G$  is a tree  $T$  such that

- the root of  $T$  is labelled with  $G$
- the nodes of  $T$  are labelled with conjunctions of goals, where one goal is selected (wrt a selection function)
- $\exists$  edge from node  $N$  for each clause, whose head unifies with the selected goal
- edges are labelled with (partial) answer substitutions

# Search or SLD Trees

## Definition

a **search tree** (aka **SLD tree**) of a goal  $G$  is a tree  $T$  such that

- the root of  $T$  is labelled with  $G$
- the nodes of  $T$  are labelled with conjunctions of goals, where one goal is selected (wrt a selection function)
- $\exists$  edge from node  $N$  for each clause, whose head unifies with the selected goal
- edges are labelled with (partial) answer substitutions
- leaves are **success nodes**, if the empty goal (denoted by  $\square$ ) has been reached or **failure nodes** otherwise

# Search or SLD Trees

## Definition

a **search tree** (aka **SLD tree**) of a goal  $G$  is a tree  $T$  such that

- the root of  $T$  is labelled with  $G$
- the nodes of  $T$  are labelled with conjunctions of goals, where one goal is selected (wrt a selection function)
- $\exists$  edge from node  $N$  for each clause, whose head unifies with the selected goal
- edges are labelled with (partial) answer substitutions
- leaves are **success nodes**, if the empty goal (denoted by  $\square$ ) has been reached or **failure nodes** otherwise

## Remark

a search tree captures all SLD derivations wrt a given selection function

## Example (cont'd)

```
plus(0,X,X).
```

```
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

```
times(0,X,0).
```

```
times(s(X),Y,Z) :- times(X,Y,U),
                  plus(U,Y,Z).
```

□

```
plus(0,s(0),s(0))
```

 $U_3 \mapsto 0$ 

```
times(0,s(0),U_3), plus(U_3,s(0),s(0))
```

```
times(s(0),s(0),s(0))
```

 $X_0 \mapsto 0, Y \mapsto 0$ 

```
plus(X_0,Y,0), times(s(X_0),s(X_0),s(0))
```

```
times(0,0,s(0))
```

 $X \mapsto 0, Y \mapsto s(0)$ 
 $X \mapsto s(X_0)$ 

```
plus(X,Y,s(0)), times(X,X,s(0))
```



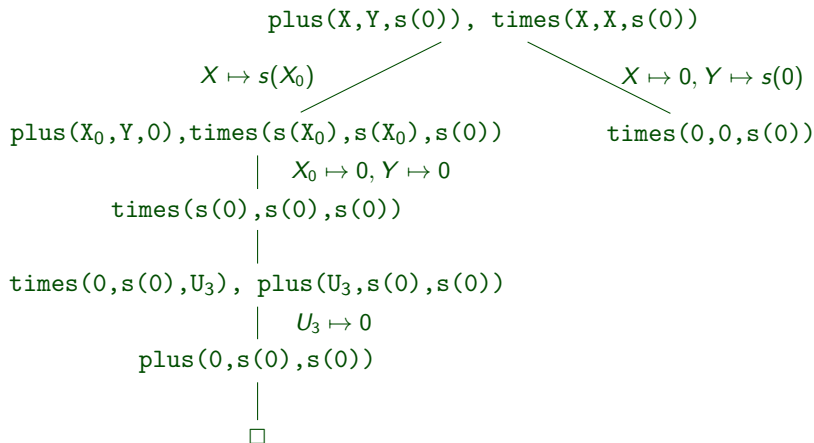
## Example (cont'd)

```
plus(0,X,X).
```

```
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).
```

```
times(0,X,0).
```

```
times(s(X),Y,Z) :- times(X,Y,U),
                  plus(U,Y,Z).
```



NB: search trees are a tree representation of SLD-derivations

# Proof Trees

## Definitions

- a **proof tree** for a program  $P$  and a goal  $G$  is a tree, whose nodes are goals and whose edges represent reduction of goals
- the root is the query  $G$



# Proof Trees

## Definitions

- a **proof tree** for a program  $P$  and a goal  $G$  is a tree, whose nodes are goals and whose edges represent reduction of goals
- the root is the query  $G$
- the edges are labelled with (partial) answer substitutions

# Proof Trees

## Definitions

- a **proof tree** for a program  $P$  and a goal  $G$  is a tree, whose nodes are goals and whose edges represent reduction of goals
- the root is the query  $G$
- the edges are labelled with (partial) answer substitutions
- a proof tree for a conjunction of goals  $G_1, \dots, G_n$  is the set of all proof trees for  $G_i$

# Proof Trees

## Definitions

- a **proof tree** for a program  $P$  and a goal  $G$  is a tree, whose nodes are goals and whose edges represent reduction of goals
- the root is the query  $G$
- the edges are labelled with (partial) answer substitutions
- a proof tree for a conjunction of goals  $G_1, \dots, G_n$  is the set of all proof trees for  $G_i$

## Remark

a proof tree is a different representation of **one successful** solution represented by a search tree

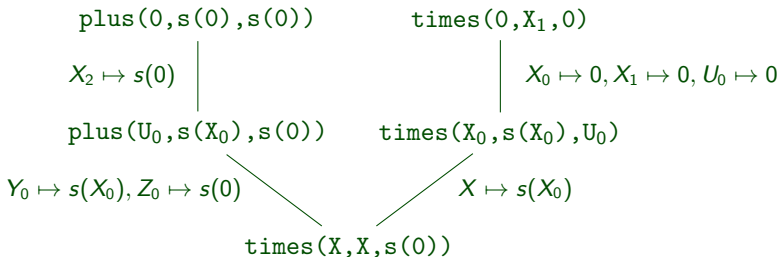
## Example (cont'd)

```

plus(0,X,X).
plus(s(X),Y,s(Z)) :- plus(X,Y,Z).

times(0,X,0).
times(s(X),Y,Z) :- times(X,Y,U),
                  plus(U,Y,Z).

```



# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program

# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program
- the **Herbrand base** is the set of all ground goals formed from predicates in  $P$  and terms in the Herbrand universe

# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program
- the **Herbrand base** is the set of all ground goals formed from predicates in  $P$  and terms in the Herbrand universe
- an **interpretation** is a subset of the Herbrand base

# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program
- the **Herbrand base** is the set of all ground goals formed from predicates in  $P$  and terms in the Herbrand universe
- an **interpretation** is a subset of the Herbrand base
- an interpretation  $I$  is a **model** if it is closed under rules:

$$\forall \text{ rules } A : -B_1, \dots, B_n: \quad \text{if } B_1, \dots, B_n \in I, \text{ then } A \in I$$



# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program
- the **Herbrand base** is the set of all ground goals formed from predicates in  $P$  and terms in the Herbrand universe
- an **interpretation** is a subset of the Herbrand base
- an interpretation  $I$  is a **model** if it is closed under rules:  
$$\forall \text{ rules } A : -B_1, \dots, B_n: \text{ if } B_1, \dots, B_n \in I, \text{ then } A \in I$$
- the **minimal** model of  $P$  is the intersection of all models

# Semantics

## Definitions

- the **Herbrand universe** for a program  $P$  is the set of all closed terms built from constants and function symbols appearing in the program
- the **Herbrand base** is the set of all ground goals formed from predicates in  $P$  and terms in the Herbrand universe
- an **interpretation** is a subset of the Herbrand base
- an interpretation  $I$  is a **model** if it is closed under rules:  
$$\forall \text{ rules } A : -B_1, \dots, B_n: \text{ if } B_1, \dots, B_n \in I, \text{ then } A \in I$$
- the **minimal** model of  $P$  is the intersection of all models

## Theorem

*the minimal model is unique*

# Declarative, Operational, and Denotational Semantics

## Definition

- the **declarative** semantics of  $P$  (aka its **meaning**) is the minimal model of  $P$
- we also say that the **meaning** of a logic program  $P$ , is the set of (ground unit) goals deducible from  $P$

# Declarative, Operational, and Denotational Semantics

## Definition

- the **declarative** semantics of  $P$  (aka its **meaning**) is the minimal model of  $P$
- we also say that the **meaning** of a logic program  $P$ , is the set of (ground unit) goals deducible from  $P$

## Definitions

the **operational** semantics describes the meaning of a program procedurally

# Declarative, Operational, and Denotational Semantics

## Definition

- the **declarative** semantics of  $P$  (aka its **meaning**) is the minimal model of  $P$
- we also say that the **meaning** of a logic program  $P$ , is the set of (ground unit) goals deducible from  $P$

## Definitions

the **operational** semantics describes the meaning of a program procedurally

## Definition

the **denotational** semantics assign meanings to programs based on associating with the program a function over the domain computed by the program

# Correctness and Completeness

## Definition

the **intended meaning** of a logic program is a set of ground facts

# Correctness and Completeness

## Definition

the **intended meaning** of a logic program is a set of ground facts

## Definition

a program  $P$  is called

- **correct** with respect to the intended meaning  $M$ , if the meaning of  $P$  is a subset of  $M$
- **complete** if the intended meaning  $M$  is a subset of the meaning of  $P$