Chapter 3

Procedural Interpretation

Outline

- Defining programs formally
- Introducing the computation method SLD-resolution
- Discussing various choices and their impact

Atoms, and Term Bases

- $TU_{F,V}$ term universe (V Variables, F function symbols)
- $\prod$ ranked alphabet of predicate symbols

Term base $TB_{\prod,F,V}$ (over $\prod$, F, and V) is smallest set A of atoms with

1. $p \in A$, if $p \in \prod^{(0)}$
2. $p(t_1, ..., t_n) \in A$, if $p \in \prod^{(n)}$ with $n \geq 1$ and $t_1, ..., t_n \in U_{F,V}$

Queries and Programs

- query :⇔ finite sequence $B_1, \ldots, B_n$ of atoms
- empty query :⇔ empty sequence (denoted by □) of atoms
- $H \leftarrow \underline{\underline{B}}$ (definite) clause :⇔ H atom ("head of clause"), $\underline{\underline{B}}$ query ("body of clause")
- $H \leftarrow$ □ unit clause (also called: fact; standard notation: $H \leftarrow$)
- (definite) program :⇔ finite set of clauses

Intuitive Meaning of Clauses and Queries

A clause $H \leftarrow B_1, ..., B_n$ can be understood as the formula

$$\forall x_1, ..., x_k (B_1 \wedge ... \wedge B_n \rightarrow H)$$

where $x_1, ..., x_k$ are the variables occurring in $H \leftarrow B_1, ..., B_n$.

(Thus a unit clause $H \leftarrow$ encodes $\forall x_1, ..., x_k\ H$)

A query $A_1, ..., A_n$ can be understood as the formula

$$\exists x_1, ..., x_k (A_1 \wedge ... \wedge A_n)$$

where $x_1, ..., x_k$ are the variables occurring in $A_1, ..., A_n$.

(Thus the empty query $\square$ is equivalent to *true*)

Negated Queries and Definite Goals

Be careful:

$$\neg \exists x_1, \ldots, x_k (A_1 \wedge \ldots \wedge A_n) \qquad \text{(negated query)}$$
$$\Leftrightarrow \forall x_1, \ldots, x_k \neg (A_1 \wedge \ldots \wedge A_n)$$
$$\Leftrightarrow \forall x_1, \ldots, x_k \; \mathit{false} \vee \neg (A_1 \wedge \ldots \wedge A_n)$$
$$\Leftrightarrow \forall x_1, \ldots, x_k \; \mathit{false} \leftarrow (A_1 \wedge \ldots \wedge A_n) \qquad \text{(constraint in the sense of CLP)}$$

What is Being Computed?

- A program *P* can be interpreted as a set of axioms.
- A query *Q* can be interpreted as the request for finding an instance $Q\theta$ which is a logical consequence of *P*.
- A successful derivation provides such a θ. In this way, the derivation is a proof of $Q\theta$.

To be continued in Chapter 4: Declarative Interpretation

How Do We Compute?

- A computation is a sequence of derivation steps.
- In each step:

  1. an atom A is selected in the current query and a program clause $H \leftarrow \underline{\underline{B}}$ is chosen.
  2. If A and H are unifiable, then A is replaced by $\underline{\underline{B}}$ and an MGU of A and H is applied to the resulting query.

- The computation is successful if it ends with the empty query.
- The resulting answer substitution θ is obtained by combining the MGUs of each step.

An SLD-Derivation Step (No Variables)

SLD = Selection rule driven Linear resolution for Definite clauses

Consider

- a program P
- a query $\underline{\underline{A}}, B, \underline{\underline{C}}$
- a clause $B \leftarrow \underline{\underline{B}} \in P$

- B is the selected atom
- The resulting query $\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}}$ is called the SLD resolvent
- Notation: $\underline{\underline{A}}, B, \underline{\underline{C}} \Longrightarrow \underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}}$

Example Ground Program and Query

```
happy :- sun, holidays.
happy :- snow, holidays.
snow  :- cold, precipitation.
cold  :- winter.
precipitation :- holidays.
winter.
holidays.
```

```
| ?- happy.
```

An SLD-Derivation Step (General Case)

Consider

- a program P
- a query $\underline{\underline{A}}, B, \underline{\underline{C}}$
- a clause $c \in P$
- a variant $H \leftarrow \underline{\underline{B}}$ of c variable disjoint with the query
- an MGU θ of B and H

SLD-resolvent of $\underline{\underline{A}}, B, \underline{\underline{C}}$ and c wrt. B with MGU θ :$\Leftrightarrow$ $(\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}})\theta$

SLD-derivation step :$\Leftrightarrow$ $\underline{\underline{A}}, B, \underline{\underline{C}} \overset{\theta}{\underset{c}{\Longrightarrow}} (\underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}})\theta$

input clause :$\Leftrightarrow$ variant $H \leftarrow \underline{\underline{B}}$

We say: “clause c applicable to atom B”

Example Program and Query

```
add(X,0,X).
add(X,s(Y),s(Z)) :- add(X,Y,Z).

mul(X,0,0).
mul(X,s(Y),Z) :- mul(X,Y,U), add(X,U,Z).


| ?- mul(s(s(0)),s(s(0)),V).


| ?- mul(V,W,s(s(0))).
```

The 4 Steps of Resolving Query and Clause

1. Selection: Select an atom in the query.

2. Renaming: Rename (if necessary) the clause.

3. Instantiation: Instantiate query and clause by an MGU of the selected atom and the head of the clause.

4. Replacement: Replace the instance of the selected atom by the instance of the body of the clause.

SLD-Derivations

A maximal sequence of SLD-derivation steps

$$Q_0 \overset{\theta_1}{\underset{c_1}{\Longrightarrow}} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\underset{c_{n+1}}{\Longrightarrow}} Q_{n+1} \ldots$$

is an SLD-derivation of $P \cup \{Q_0\}$

:⇔

- Q_0, ..., Q_{n+1}, ... are queries, each empty or with one atom selected in it;
- θ_1, ..., θ_{n+1}, ... are substitutions;
- c_1, ..., c_{n+1}, ... are clauses of P;
- for every SLD-derivation step, standardization apart holds.

Standardization Apart

The input clause is variable disjoint from the initial query and from the substitutions and input clauses used at earlier steps.

Formally:

$$Var(c'_i) \cap \left(Var(Q_0) \cup \bigcup_{j=1}^{i-1} (Var(\theta_j) \cup Var(c'_j)) \right) = \emptyset$$

for $i \geq 1$, where c'_i is the input clause used in the i-th SLD-derivation step $Q_{i-1} \overset{\theta_i}{\underset{c_i}{\Longrightarrow}} Q_i$

Result of a Derivation

Let $\xi = Q_0 \stackrel{\theta_1}{\Longrightarrow} Q_1 \ldots \stackrel{\theta_n}{\Longrightarrow} Q_n$ be a finite SLD-derivation.

- ξ successful :⇔ $Q_n = \square$
- ξ failed :⇔ $Q_n \neq \square$ and no clause is applicable to selected atom of Q_n

Let ξ be successful.

- computed answer substitution (CAS) of Q_0 (w.r.t. ξ) :⇔ $(\theta_1 \cdots \theta_n) \mid_{Var(Q_0)}$
- computed instance of Q_0 :⇔ $Q_0\theta_1 \cdots \theta_n$

Choices

In each SLD-derivation step the following four choices are made.

1. Choice of the renaming
2. Choice of the MGU
3. Choice of the selected atom in the query
4. Choice of the program clause

How do they influence the result?

Resultants: What is Proved After a Step?

resultant associated with $Q \stackrel{\theta}{\Longrightarrow} Q_1$:⇔ implication $Q\theta \leftarrow Q_1$

Consider

- a program P
- a resultant $R = Q \leftarrow \underline{\underline{A}}, B, \underline{\underline{C}}$
- a clause c
- a variant $H \leftarrow \underline{\underline{B}}$ of c variable disjoint with R
- an MGU θ of B and H

SLD-resolvent of resultant R and c w.r.t. B with MGU θ :⇔ $(Q \leftarrow \underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}})\theta$

SLD-resultant step :⇔ $Q \leftarrow \underline{\underline{A}}, B, \underline{\underline{C}} \stackrel{\theta}{\underset{c}{\Longrightarrow}} (Q \leftarrow \underline{\underline{A}}, \underline{\underline{B}}, \underline{\underline{C}})\theta$

Resultants and SLD-Derivations

Consider an SLD-derivation

$$\xi = Q_0 \overset{\theta_1}{\underset{c_1}{\Longrightarrow}} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\underset{c_{n+1}}{\Longrightarrow}} Q_{n+1} \ldots$$

For $i \geq 0$

$$R_i :\Leftrightarrow Q_0\theta_1 \cdots \theta_i \leftarrow Q_i$$

is called the resultant of level i of ξ.

The resultant R_i describes what is “proved” after i derivation steps; in particular:

- $R_0 : Q_0 \leftarrow Q_0$
- $R_n : Q_0\theta_1 \cdots \theta_n$ if $Q_n = \square$ (because $\square \triangleq$ “true”)

Propagation (I)

The selected atom of a resultant $Q \leftarrow Q_i$ is defined as the atom selected in Q_i.

Lemma 3.12

Suppose that $R \stackrel{\theta}{\underset{c}{\Longrightarrow}} R_1$ and $R' \stackrel{\theta'}{\underset{c}{\Longrightarrow}} R'_1$ are two SLD-resultant steps such that

- R is an instance of R',
- in R and R' atoms in the same positions are selected.

Then R_1 is an instance of R'_1.

Proof: see [Apt97, page 55]

Propagation (II)

Corollary 3.13

Suppose that $Q \stackrel{\theta}{\underset{c}{\Longrightarrow}} Q_1$ and $Q' \stackrel{\theta'}{\underset{c}{\Longrightarrow}} Q'_1$ are two SLD-derivation steps such that

- Q is an instance of Q',
- in Q and Q' atoms in the same positions are selected.

Then Q_1 is an instance of Q'_1.

Similar SLD-Derivations

Consider two (initial fragments of) SLD-derivations

$$\xi = Q_0 \overset{\theta_1}{\underset{c_1}{\Longrightarrow}} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\underset{c_{n+1}}{\Longrightarrow}} Q_{n+1} \cdots$$

and

$$\xi' = Q'_0 \overset{\theta'_1}{\underset{c_1}{\Longrightarrow}} Q'_1 \ldots Q'_n \overset{\theta'_{n+1}}{\underset{c_{n+1}}{\Longrightarrow}} Q'_{n+1} \cdots$$

ξ and ξ' are similar

$:\Leftrightarrow$

- length (ξ) = length (ξ'),
- Q_0 and Q'_0 are variants,
- in Q_i and Q'_i atoms in the same positions are selected ($i \in [0, \ldots, n]$)

A Theorem on Variants

Theorem 3.18

Consider two similar SLD-derivations ξ, ξ'. Then for every $i \geq 0$, the resultants R_i and R'_i of level i of ξ and ξ', respectively, are variants of each other.

Proof.

Base Case ($i = 0$): $R_0 = Q_0 \leftarrow Q_0$ $\quad R'_0 = Q'_0 \leftarrow Q'_0$

Induction Case ($i \rightarrow i + 1$): $R_i \overset{\theta_{i+1}}{\underset{c_{i+1}}{\Longrightarrow}} R_{i+1}$ $\quad R'_i \overset{\theta'_{i+1}}{\underset{c_{i+1}}{\Longrightarrow}} R'_{i+1}$

R_i variant of R'_i

implies R_i instance of R'_i and vice versa

implies R_{i+1} instance of R'_{i+1} and vice versa (Lemma 3.12)

implies R_{i+1} variant of R'_{i+1}

Answer Substitutions of Similar Derivations

Corollary 3.19

Consider two similar successful SLD-derivations of Q_0 with CAS θ and η. Then $Q_0\theta$ and $Q_0\eta$ are variants of each other.

Proof. By Theorem 3.18 applied to the final resultants $Q_0\theta \leftarrow \square$ and $Q_0\eta \leftarrow \square$ of these SLD-derivations.

This shows that choice 1 (choice of a renaming) and choice 2 (choice of an MGU) have no influence – modulo renaming – on the statement proved by a successful SLD-derivation.

Selecting Atoms in Queries

Let *INIT* be the set of *all* initial fragments of *all* possible SLD-derivations in which the last query in non-empty.

A selection rule is a function which for every $\xi^{<} \in$ *INIT* yields an occurrence of an atom in the last query of $\xi^{<}$.

An SLD-derivation ξ is via a selection rule $\mathcal{R}$ if for every initial fragment $\xi^{<}$ of ξ ending with a non-empty query Q, $\mathcal{R}(\xi^{<})$ is the selected atom of Q.

PROLOG employs the simple selection rule "Select the leftmost atom".

Switching Lemma

Lemma 3.32

Consider an SLD-derivation $$\xi = Q_0 \overset{\theta_1}{\underset{c_1}{\Longrightarrow}} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\underset{c_{n+1}}{\Longrightarrow}} Q_{n+1} \overset{\theta_{n+2}}{\underset{c_{n+2}}{\Longrightarrow}} Q_{n+2} \ldots$$

where

- Q_n includes two atoms A_1 and A_2
- A_1 is the selected atom of Q_n
- $A_2\theta_{n+1}$ is the selected atom of Q_{n+1}

Then for some Q'_{n+1}, θ'_{n+1}, and θ'_{n+2} $$\xi' = Q_0 \overset{\theta_1}{\underset{c_1}{\Longrightarrow}} Q_1 \ldots Q_n \overset{\theta'_{n+1}}{\underset{c_{n+2}}{\Longrightarrow}} Q'_{n+1} \overset{\theta'_{n+2}}{\underset{c_{n+1}}{\Longrightarrow}} Q_{n+2} \ldots$$

where

- A_2 is the selected atom of Q_n
- $A_1\theta'_{n+1}$ is the selected atom of Q'_{n+1}
- $\theta'_{n+1}\theta'_{n+2} = \theta_{n+1}\theta_{n+2}$

Proof: see [Apt97, page 65]

Independence of Selection Rule

Theorem 3.33

Let ξ be a successful SLD-derivation of $P \cup \{Q_0\}$. Then for every selection rule $\mathcal{R}$ there exists a successful SLD-derivation ξ' of $P \cup \{Q_0\}$ via $\mathcal{R}$ such that

- CAS of Q_0 (w.r.t. ξ) = CAS of Q_0 (w.r.t. ξ'),
- ξ and ξ' are of the same length.

This shows that choice 3 (choice of a selected atom) has no influence in case of successful queries.

Objectives

- Defining programs formally
- Introducing the computation method SLD-resolution
- Discussing various choices and their impact

Proof Sketch of Theorem 3.33

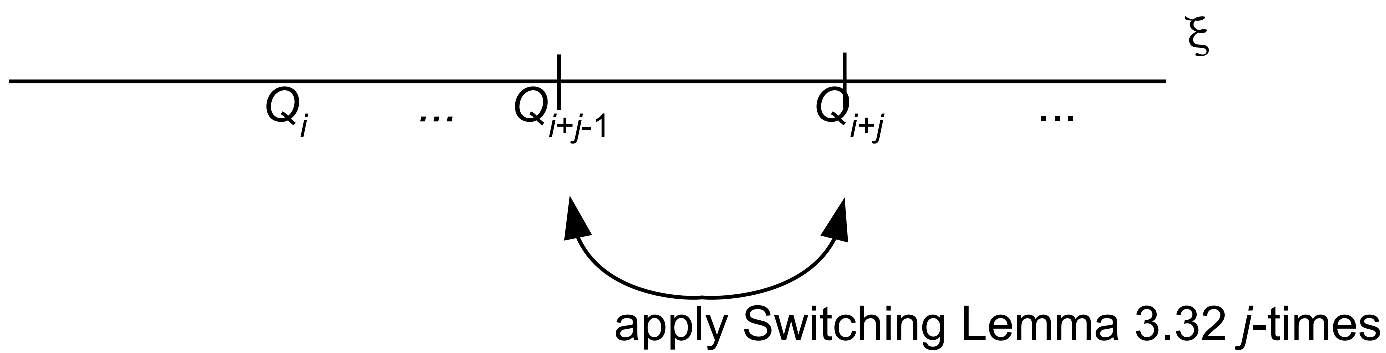
$$\xi = Q_0 \stackrel{\theta_1}{\Longrightarrow} \ldots \stackrel{\theta_n}{\Longrightarrow} Q_n = \square$$

Induction on *i*:

assume "ξ is via $\mathcal{R}$ up to Q_{i-1}"

$\mathcal{R}$ selects A in Q_i

$A\theta_{i+1} \ldots \theta_{i+j}$ is selected atom of Q_{i+j} in ξ for some $j > 1$ (ξ successful !)



SLD-Trees Visualize Search Space

SLD-tree for $P \cup \{Q_0\}$ via selection rule $\mathcal{R}$

$:\Leftrightarrow$

- the branches are SLD-derivations of $P \cup \{Q_0\}$ via $\mathcal{R}$
- every node Q with selected atom A has exactly one descendant for every clause c of P with is applicable to A.
  This descendant is a resolvent of Q and c w.r.t. A.

SLD-tree successful $:\Leftrightarrow$ tree contains the node $\Box$

SLD-tree finitely failed $:\Leftrightarrow$ tree is finite and not successful

SLD-tree via “leftmost selection rule” corresponds to Prolog's search space

Variant Independence

Selection rule $\mathcal{R}$ variant independent

$:\Leftrightarrow$

in all initial fragments of SLD-derivations which are similar (c.f. Slide 22), $\mathcal{R}$ chooses the atom in the same position in the last query.

- Selection rule “select leftmost atom” is variant independent
- Selection rule “select leftmost atom if query contains variable *x*, otherwise select rightmost atom” is variant dependent

The Branch Theorem

Theorem 3.38

Consider an SLD-tree $\mathcal{T}$ for $P \cup \{Q_0\}$ via a variant independent selection rule $\mathcal{R}$. Then every SLD-derivation of $P \cup \{Q_0\}$ via $\mathcal{R}$ is similar to a branch in $\mathcal{T}$.

This shows that choice 4 (choice of a program clause) has no influence on the search space as a whole.