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)

- $\Pi$ ranked alphabet of predicate symbols

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

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

- **query** :⇔ finite sequence $B_1, ..., B_n$ of atoms

- empty query :⇔ empty sequence (denoted by □) of atoms

- $H ← B$ (definite) clause :⇔ $H$ atom (“head of clause”), $B$ query (“body of clause”)

- $H ← □$ unit clause (also called: fact; standard notation: $H ←$)

- (definite) program :⇔ finite set of clauses
Intuitive Meaning of Clauses and Queries

A clause \( H \leftarrow B_1, \ldots, B_n \) can be understood as the formula

\[
\forall x_1, \ldots, x_k (B_1 \land \ldots \land B_n \rightarrow H)
\]

where \( x_1, \ldots, x_k \) are the variables occurring in \( H \leftarrow B_1, \ldots, B_n \).
(Thus a unit clause \( H \leftarrow \) encodes \( \forall x_1, \ldots, x_k H \))

A query \( A_1, \ldots, A_n \) can be understood as the formula

\[
\exists x_1, \ldots, x_k (A_1 \land \ldots \land A_n)
\]

where \( x_1, \ldots, x_k \) are the variables occurring in \( A_1, \ldots, A_n \).
(Thus the empty query \( \Box \) is equivalent to \( true \))
Negated Queries and Definite Goals

Be careful:

\[ \neg \exists x_1, \ldots, x_k (A_1 \land \ldots \land A_n) \]  
\[ \iff \forall x_1, \ldots, x_k \neg (A_1 \land \ldots \land A_n) \]  
\[ \iff \forall x_1, \ldots, x_k false \lor \neg (A_1 \land \ldots \land A_n) \]  
\[ \iff \forall x_1, \ldots, x_k \text{false} \leftarrow (A_1 \land \ldots \land A_n) \]  
\[ \quad \text{(constraint in the sense of CLP)} \]
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 $\theta$. 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 $H \leftarrow B$ is chosen.
  2. If $A$ and $H$ are unifiable, then $A$ is replaced by $B$ 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 $\theta$ 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 \( A, B, C \)
- a clause \( B \leftarrow B \in P \)

- \( B \) is the selected atom
- The resulting query \( A, B, C \) is called the SLD resolvent
- Notation: \( A, B, C \Rightarrow A, B, C \)
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 \( A, B, C \)
- a clause \( c \in P \)
- a variant \( H \leftarrow B \) of \( c \) variable disjoint with the query
- an \( \text{MGU} \ \theta \) of \( B \) and \( H \)

SLD-resolvent of \( A, B, C \) and \( c \) wrt. \( B \) with \( \text{MGU} \ \theta : \iff (A, B, C) \theta \)

SLD-derivation step : \( \iff A, B, C \xrightarrow{c} (A, B, C) \theta \)

input clause : \( \iff \) variant \( H \leftarrow B \)

We say: “clause \( c \) applicable to atom \( B \)”
Example Program and Query

\[
\begin{align*}
\text{add}(X, 0, X). \\
\text{add}(X, s(Y), s(Z)) & :\ - \text{add}(X, Y, Z). \\
\text{mul}(X, 0, 0). \\
\text{mul}(X, s(Y), Z) & :\ - \text{mul}(X, Y, U), \ \text{add}(X, U, Z).
\end{align*}
\]

\[
\begin{align*}
| \ ?- \ & \text{mul}(s(s(0)), s(s(0)), V). \\
| \ ?- \ & \text{mul}(V, W, s(s(0))).
\end{align*}
\]
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 \textit{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

\[
Q_0 \overset{\theta_1}{\Rightarrow} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\Rightarrow} Q_{n+1} \ldots
\]

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

\( \iff \)

- \( Q_0, \ldots, Q_{n+1}, \ldots \) are queries, each empty or with one atom selected in it;
- \( \theta_1, \ldots, \theta_{n+1}, \ldots \) are substitutions;
- \( c_1, \ldots, c_{n+1}, \ldots \) 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:

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

for \( i \geq 1 \), where \( c'_i \) is the input clause used in the \( i \)-th SLD-derivation step \( Q_{i-1} \overset{\theta_i}{\longrightarrow} Q_i \).
Result of a Derivation

Let $\xi = Q_0 \implies Q_1 \implies \cdots \implies Q_n$ be a finite SLD-derivation.

- $\xi$ successful $\iff Q_n = \square$
- $\xi$ failed $\iff Q_n \neq \square$ and no clause is applicable to selected atom of $Q_n$

Let $\xi$ be successful.

- computed answer substitution (CAS) of $Q_0$ (w.r.t. $\xi$) $\iff (\theta_1 \cdots \theta_n) \mid \text{Var}(Q_0)$
- computed instance of $Q_0$ $\iff Q_0 \theta_1 \cdots \theta_n$
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?

\[ \text{resultant associated with } Q \implies Q_1 : \iff \text{implication } Q^\theta \leftarrow Q_1 \]

Consider
- a program \( P \)
- a resultant \( R = Q \leftarrow A, B, C \)
- a clause \( c \)
- a variant \( H \leftarrow B \) of \( c \) variable disjoint with \( R \)
- an \text{MGU} \( \theta \) of \( B \) and \( H \)

SLD-resolvent of resultant \( R \) and \( c \) w.r.t. \( B \) with \text{MGU} \( \theta \) : \iff (Q \leftarrow A, B, C)^\theta

SLD-resultant step : \iff Q \leftarrow A, B, C \xrightarrow{c} (Q \leftarrow A, B, C)^\theta
Resultants and SLD-Derivations

Consider an SLD-derivation

\[ \xi = \ll[Q_0 \rightarrow Q_1 \ldots Q_n \rightarrow Q_{n+1} \ldots] \]

For \( i \geq 0 \)

\[ R_i : \iff Q_0 \theta_1 \ldots \theta_i \leftarrow Q_i \]

is called the resultant of level \( i \) of \( \xi \).

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

- \( R_0 : Q_0 \leftarrow Q_0 \)
- \( R_n : Q_0 \theta_1 \ldots \theta_n \) if \( Q_n = \Box \) (because \( \Box \equiv \text{“true”} \))
The selected atom of a resultant $Q \leftarrow Q_i$ is defined as the atom selected in $Q_i$.

**Lemma 3.12**
Suppose that $R \xrightarrow{\theta} R_1$ and $R' \xrightarrow{\theta'} R'_1$ 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]
Corollary 3.13

Suppose that $Q \xrightarrow{c^0} Q_1$ and $Q' \xrightarrow{c^{0'}} Q'_1$ 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

\[ \xi = Q_0 \overset{\theta_1}{\Longrightarrow} Q_1 \ldots Q_n \overset{\theta_{n+1}}{\Longrightarrow} Q_{n+1} \ldots \]

and

\[ \xi' = Q'_0 \overset{\theta'_1}{\Longrightarrow} Q'_1 \ldots Q'_n \overset{\theta'_{n+1}}{\Longrightarrow} Q'_{n+1} \ldots \]

\( \xi \) and \( \xi' \) are similar

\[ :\Leftrightarrow \]

- \( \text{length}(\xi) = \text{length}(\xi') \),
- \( Q_0 \) 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 $\xi$, $\xi'$. Then for every $i \geq 0$, the resultants $R_i$ and $R'_i$ of level $i$ of $\xi$ and $\xi'$, 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 ($i \rightarrow i + 1$): $R_i \overset{\theta_{i+1}}{\Longrightarrow} R_{i+1}$ \quad $R'_i \overset{\theta'_{i+1}}{\Longrightarrow} R'_{i+1}$

$R_i$ 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 $R'_{i+1}$
Corollary 3.19

Consider two similar successful SLD-derivations of $Q_0$ with \textit{cas} $\theta$ and $\eta$. Then $Q_0\theta$ and $Q_0\eta$ are variants of each other.

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

This shows that choice 1 (choice of a renaming) and choice 2 (choice of an \textit{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 is 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 $\xi$ is via a selection rule $\mathcal{R}$ if for every initial fragment $\xi^<$ of $\xi$ 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 \( \xi = Q_0 \rightarrow_{c_1} Q_1 \rightarrow_{c_2} \ldots \rightarrow_{c_{n+1}} Q_{n+1} \rightarrow_{c_{n+2}} Q_{n+2} \ldots \)

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 \( Q'_{n+1} \), \( \theta'_{n+1} \), and \( \theta'_{n+2} \)

\( \xi' = Q_0 \rightarrow_{c_1} Q_1 \rightarrow_{c_2} \ldots \rightarrow_{c_{n+2}} Q'_{n+1} \rightarrow_{c_{n+1}} Q'_{n+2} \ldots \)

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

- $\text{CAS of } Q_0 \text{ (w.r.t. } \xi) = \text{CAS of } Q_0 \text{ (w.r.t. } \xi')$,
- $\xi$ and $\xi'$ 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

Induction on $i$:
assume “$\xi$ 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 $\xi$ for some $j > 1$ ($\xi$ successful !)

\[
\xi = Q_0 \Rightarrow \ldots \Rightarrow Q_n = \square
\]

apply Switching Lemma 3.32 $j$-times
SLD-Trees Visualize Search Space

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

$\iff$

- 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 $\iff$ tree contains the node $\square$

SLD-tree finitely failed $\iff$ 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:

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

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

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