A small Prolog$^-$-program

- natural numbers in Prolog$^-$:

```
nat(0).                 % (nat1)
nat(s(X)) :- nat(X).    % (nat2)
```

- an *SLD-derivation*:

```
?- nat(s(s(0))).        % Is 2 a natural number?
?- nat(s(0)).           % by (nat2)
?- nat(0).              % by (nat2)
?-.                     % by (nat1)
```

- *SLD-refutation*: SLD-derivation with final goal `?-.`

The context-free syntax of Prolog^{-}

$\langle \mathit{prog} \rangle$	$::= \langle \mathit{pred} \rangle \;\hat{\{}\; \langle \mathit{pred} \rangle \;\hat{\}}$	
$\langle \mathit{pred} \rangle$	$::= \langle \mathit{clause} \rangle \;\hat{\{}\; \langle \mathit{clause} \rangle \;\hat{\}}$	(*predicate definitions*)
$\langle \mathit{clause} \rangle$	$::= \langle \mathit{lit} \rangle \;\texttt{:-}\; \langle \mathit{lit} \rangle \;\hat{\{}\; \texttt{,}\; \langle \mathit{lit} \rangle \;\hat{\}}\; \texttt{.}$	(*rules*)
	$\hat{\mid}\; \langle \mathit{lit} \rangle \;\texttt{.}$	(*facts*)
$\langle \mathit{lit} \rangle$	$::= \langle \mathit{predid} \rangle \;\hat{\mid}\; \langle \mathit{predid} \rangle \;\texttt{(}\; \langle \mathit{pat} \rangle \;\hat{\{}\; \texttt{,}\; \langle \mathit{pat} \rangle \;\hat{\}}\; \texttt{)}$	(*(positive) literals*)
$\langle \mathit{pat} \rangle$	$::= \langle \mathit{varid} \rangle \;\hat{\mid}\; \langle \mathit{conid} \rangle \;\hat{\mid}\; \langle \mathit{conid} \rangle \;\texttt{(}\; \langle \mathit{pat} \rangle \;\hat{\{}\; \texttt{,}\; \langle \mathit{pat} \rangle \;\hat{\}}\; \texttt{)}$	(*patterns*)
$\langle \mathit{conid} \rangle$	$::= \hat{(}\texttt{a} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{z}\hat{)}\; \hat{\{}\texttt{a} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{z}\hat{\}}$	(*constructors*)
$\langle \mathit{predid} \rangle$	$::= \hat{(}\texttt{a} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{z}\hat{)}\; \hat{\{}\texttt{a} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{z}\hat{\}}$	(*predicate symbols*)
$\langle \mathit{varid} \rangle$	$::= \hat{(}\texttt{A} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{Z}\hat{)}\; \hat{\{}\texttt{A} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{Z} \;\hat{\mid}\; \texttt{a} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{z} \;\hat{\mid}\; \texttt{0} \;\hat{\mid}\; \ldots \;\hat{\mid}\; \texttt{9}\hat{\}}$	(*variables*)
$\langle \mathit{goal} \rangle$	$::= \texttt{?-}\; \langle \mathit{lit} \rangle \;\hat{\{}\; \texttt{,}\; \langle \mathit{lit} \rangle \;\hat{\}}\; \texttt{.}$	(*goals*)
	$\hat{\mid}\; \texttt{?-.}$	(*empty goal*)

Example: Summation (I)

- addition in Prolog$^-$:

```
nat(0).                                  % (nat1)
nat(s(X)) :- nat(X).                     % (nat2)

sum(0, X, X)        :- nat(X).           % (sum1)
sum(s(X), Y, s(Z)) :- sum(X, Y, Z).      % (sum2)
```

- Is $2 + 5 = 7$?

```
?- sum(<2>, <5>, <7>).                   % <n> = s(...s(0)...) with n times s
?- sum(<1>, <5>, <6>).                   % by (sum2)
?- sum(<0>, <5>, <5>).                   % by (sum2)
?- nat(<5>).                             % by (sum1)
?- nat(<4>).                             % by (nat2)
?- ….
?- nat(<0>).                             % by (nat2)
?-.                                      % by (nat1)
```

Example: Summation (II)

- addition in Prolog^{-}:

```
nat(0).                              % (nat1)
nat(s(X)) :- nat(X).                 % (nat2)

sum(0, X, X)        :- nat(X).       % (sum1)
sum(s(X), Y, s(Z)) :- sum(X, Y, Z).  % (sum2)
```

- Solve $x + 1 = 3$.

```
            ?- sum(X , <1>, <3>).
{X =s(X1)}  ?- sum(X1, <1>, <2>).    % by (sum2)
{X1=s(X2)}  ?- sum(X2, <1>, <1>).    % by (sum2)
{X2=0}      ?- nat(<1>).             % by (sum1)
            ?- nat(<0>).             % by (nat2)
            ?-.                      % by (nat1)
```

- `X = s(X1) = s(s(X2)) = s(s(0))`

Example: Computations without success

- addition in Prolog$^-$:

```
nat(0).                                 % (nat1)
nat(s(X)) :- nat(X).                    % (nat2)

sum(0, X, X)       :- nat(X).           % (sum1)
sum(s(X), Y, s(Z)) :- sum(X, Y, Z).     % (sum2)
```

- Solve $x + 1 = 3$.

```
            ?- sum(X , <1>, <3>).
{X =s(X1)}  ?- sum(X1, <1>, <2>).      % by (sum2)
{X1=s(X2)}  ?- sum(X2, <1>, <1>).      % by (sum2)
{X2=s(X3)}  ?- sum(X3, <1>, <0>).      % by (sum2)
```

Example: Lists

- lists in Prolog$^-$:

```
list([]).
list([X|Xs]) :- list(Xs).
```

- abbreviate `[A|[B|[C|[]]]]` by `[A, B, C]`

Example: Sum of Lists

- sum of lists in Prolog^{-}:

```
listsum([]    , 0).
listsum([X|Xs], Z) :- listsum(Xs, Y), sum(X, Y, Z).

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

- sum of [<2>, <3>, <1>]:

```
                 ?-  listsum([<2>, <3>, <1>], U).
                 ?-  listsum([<3>, <1>], Y), sum(<2>, Y, U).
                 ?-  listsum([<1>], V), sum(<3>, V, Y), sum(<2>, Y, U).
                 ?-  listsum([], T), sum(<1>, T, V), sum(<3>, V, Y), sum(<2>, Y, U).
{T =0}           ?-  sum(<1>, 0, V), sum(<3>, V, Y), sum(<2>, Y, U).
{V =s(V1)}       ?-  sum(0, 0, V1), sum(<3>, s(V1), Y), sum(<2>, Y, U).
{V1=0}           ?-  sum(<3>, s(0), Y), sum(<2>, Y, U).
{Y =s(s(s(Y1)))} ?-* sum(0, s(0), Y1), sum(<2>, s(s(s(Y1))), U).
{Y1=s(0)}        ?-  sum(<2>, <4>, U).
{U =s(s(U1))}    ?-* sum(0, <4>, U1).
{U1=<4>}         ?-.
```

- `U = s(s(U1)) = s(s(<4>)) = <6>`

Example: Propositional logic (Aussagenlogik)

- terms made of variables, conjunction (^), disjunction (v), and negation (~)
- satisfiable (`sat`) and invalid (`inv`) formulae

```
sat(true).
sat(X ^ Y) :- sat(X), sat(Y).
sat(X v Y) :- sat(X).
sat(X v Y) :- sat(Y).
sat(~ X)   :- inv(X).
```

```
inv(X ^ Y) :- inv(X).
inv(X ^ Y) :- inv(Y).
inv(X v Y) :- inv(X), inv(Y).
inv(~ X)   :- sat(X).
```

- Is `U ^ ~(~ U v W)` satisfiable?

```
             ?- sat(U ^ ~(~ U v W)).
             ?- sat(U), sat(~(~ U v W)).
{U=true}     ?- sat(~(~ true v W)).
             ?- inv(~ true v W).
             ?- inv(~ true), inv(W).
             ?- sat(true), inv(W).
             ?- inv(W).
{W=~ Z}      ?- sat(Z).
(Z=true}     ?-.
```

- `U = true` and `W = ~ Z = ~ true` satisfies `U ^ ~(~ U v W)`

SLD-Resolution (Selective Linear Definite clause resolution)

Let

- P be a Prolog$^-$ program and
- $G = (\texttt{?-}\ L_1, ..., L_n\ \texttt{.})$ with $n \geq 1$ be a goal of P.

If

- there is an $i \in [n]$,
- there is a variant $C = (M_0\ \texttt{:-}\ M_1, ..., M_m\ \texttt{.})$ of a rule of P (i.e. C is constructed by renaming variables) such that G and C have no variables in common, and
- σ is the most general unifier of L_i and M_0,

then

$$G' = (\texttt{?-}\ \tilde{\sigma}(L_1), ..., \tilde{\sigma}(L_{i-1}), \tilde{\sigma}(M_1), ..., \tilde{\sigma}(M_m), \tilde{\sigma}(L_{i+1}), ..., \tilde{\sigma}(L_n)\ \texttt{.})$$

is called a *resolvent of G and C with σ*. The literal L_i is called the *selected literal* in G.

SLD-Derivation and SLD-Refutation

Let

- P be a Prolog^{-} program and
- G be a goal for P.

An *SLD-derivation of* (P, G) is a (possibly infinite) sequence $G_0, G_1, G_2, G_3, \ldots$ of goals for P, such that

- $G_0 = G$,
- there is a sequence $C_1, C_2, C_3, \ldots$ of variants of rules of P,
- there is a sequence $\sigma_1, \sigma_2, \sigma_3, \ldots$ of most general unifiers, and
- G_{i+1} is a resolvent of G_i and C_{i+1} with σ_{i+1} for every $i \in \mathbb{N}$.

An *SLD-refutation of* (P, G) is a finite SLD-derivation $G_0, G_1, G_2, \ldots, G_n$ of (P, G) with $n \in \mathbb{N}$ such that $G_n = (\texttt{?-.})$.

Notation of SLD-refutations:

$$\begin{array}{cc} & G_0 \\ \{\sigma_1\} & G_1 \\ \vdots & \vdots \\ \{\sigma_n\} & G_n \end{array}$$

Computed Answer

Let

- P be a Prolog^{-} program,
- G be a goal for P, and
- $\begin{pmatrix} & G_0 \\ \{\sigma_1\} & G_1 \\ \vdots & \vdots \\ \{\sigma_n\} & G_n \end{pmatrix}$ be an SLD-refutation of (P, G).

The substitution σ which results from restricting the composition $\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_n$ to the variables in G is called a *computed answer for* (P, G).

Relationship between clauses of Prolog$^-$ and formulas of first-order predicate logic

	Prolog$^-$	**formula**
rule	`L0 :- L1, ..., Ln.`	$\forall X(L_0 \leftarrow L_1 \wedge \ldots \wedge L_n)$
goal	`?- L1, ..., Ln.`	$\forall X(\neg L_1 \vee \ldots \vee \neg L_n)$
empty goal	`?-.`	$\left(\bigvee_{L \in \emptyset} L \leftarrow \bigwedge_{L \in \emptyset} L\right) \iff (\text{false} \leftarrow \text{true})$

where $\forall X$ represents the universal quantification of all variables that occur in the Horn-clause.

$$\forall X(L_0 \leftarrow L_1 \wedge \ldots \wedge L_n) \iff \forall X(L_0 \vee \neg(L_1 \wedge \ldots \wedge L_n)) \qquad \text{(since } a \leftarrow b \iff a \vee \neg b\text{)}$$

$$\iff \forall X(\underbrace{L_0 \vee \neg L_1 \vee \ldots \vee \neg L_n}_{\text{Horn clause}}) \qquad \text{(by DeMorgan's law)}$$