A more realistic Example:

$$
\begin{array}{lcl}
\mathsf{app}(X, Y, Z) & \leftarrow & X = [\,], \; Y = Z \\
\mathsf{app}(X, Y, Z) & \leftarrow & X = [H|X'], \; Z = [H|Z'], \; \mathsf{app}(X', Y, Z') \\
 & \leftarrow & \mathsf{app}(X, [Y, c], [a, b, Z])
\end{array}
$$

Remark:

$[\,]$	$==$	the atom empty list
$[H\|Z]$	$==$	binary constructor application
$[a, b, Z]$	$==$	Abbreviation for: $[a\|[b\|[Z\|[\,]]]]$

Accordingly, a program p is constructed as follows:

$$
\begin{array}{lcl}
t & ::= & a \mid X \mid \_ \mid f(t_1, \ldots, t_n) \\
g & ::= & p(t_1, \ldots, t_k) \mid X = t \\
c & ::= & p(X_1, \ldots, X_k) \leftarrow g_1, \ldots, g_r \\
q & ::= & \leftarrow g_1, \ldots, g_r \\
p & ::= & c_1 \ldots c_m q
\end{array}
$$

- A term t either is an atom, a (possibly anonymous) variable or a constructor application.
- A goal g either is a literal, i.e., a predicate call, or a unification.
- A clause c consists of a head $p(X_1, \ldots, X_k)$ together with body consisting of a sequence of goals.
- A program consists of a sequence of clauses together with a sequence of goals as query.

Procedural View of PuP-Programs:

literal	$==$	procedure call
predicate	$==$	procedure
definition	$==$	body
term	$==$	value
unification	$==$	basic computation step
binding of variables	$==$	side effect

Warning: Predicate calls ...

- do not return results!
- modify the caller solely through side effects :-)
- may fail. Then, the following definition is tried $\Longrightarrow$ backtracking

Inefficiencies:

Backtracking: • The matching alternative must be searched for $\Longrightarrow$ Indexing

- Since a successful call may still fail later, the stack can only be cleared if there are no pending alternatives.

Unification: • The translation possibly must switch between build and check several times.

- In case of unification with a variable, an Occur Check must be performed.

Type Checking: • Since Prolog is untyped, it must be checked at run-time whether or not a term is of the desired form.

- Otherwise, ugly errors could show up.

Some Optimizations:

- Replacing last calls with jumps;
- Compile-time type inference;
- Identification of deterministic predicates ...

Example:

$$
\begin{array}{rcl}
\mathsf{app}(X, Y, Z) & \leftarrow & X = [\,], \ Y = Z \\
\mathsf{app}(X, Y, Z) & \leftarrow & X = [H|X'], \ Z = [H|Z'], \ \mathsf{app}(X', Y, Z') \\
 & \leftarrow & \mathsf{app}([a, b], [Y, c], Z)
\end{array}
$$

Observation:

- In PuP, functions must be simulated through predicates.
- These then have designated input- and output parameters.
- Input parameters are those which are instantiated with a variable-free term whenever the predicate is called.

  These are also called ground.
- In the example, the first parameter of app is an input parameter.
- Unification with such a parameter can be implemented as pattern matching !
- Then we see that app in fact is deterministic !!!

5.1 Groundness Analysis

A variable X is called ground w.r.t. a program execution π starting program entry and entering a program point v, if X is bound to a variable-free term.

Goal:

- Find all variables which are ground whenever a particular program point is reached !
- Find all arguments of a predicate which are ground whenever the predicate is called !

Idea:

- Describe groundness by values from $\mathbb{B}$:

  $1 \quad == \quad$ variable-free term;

  $0 \quad == \quad$ term which contains variables.

- A set of variable assignments is described by Boolean functions :-)

  $X \leftrightarrow Y \quad == \quad X$ is ground iff Y is ground.

  $X \wedge Y \quad == \quad X$ and Y are ground.

Idea (cont.):

- The constant function 0 denotes an unreachable program point.
- Occurring sets of variable assignments are closed under substitution.

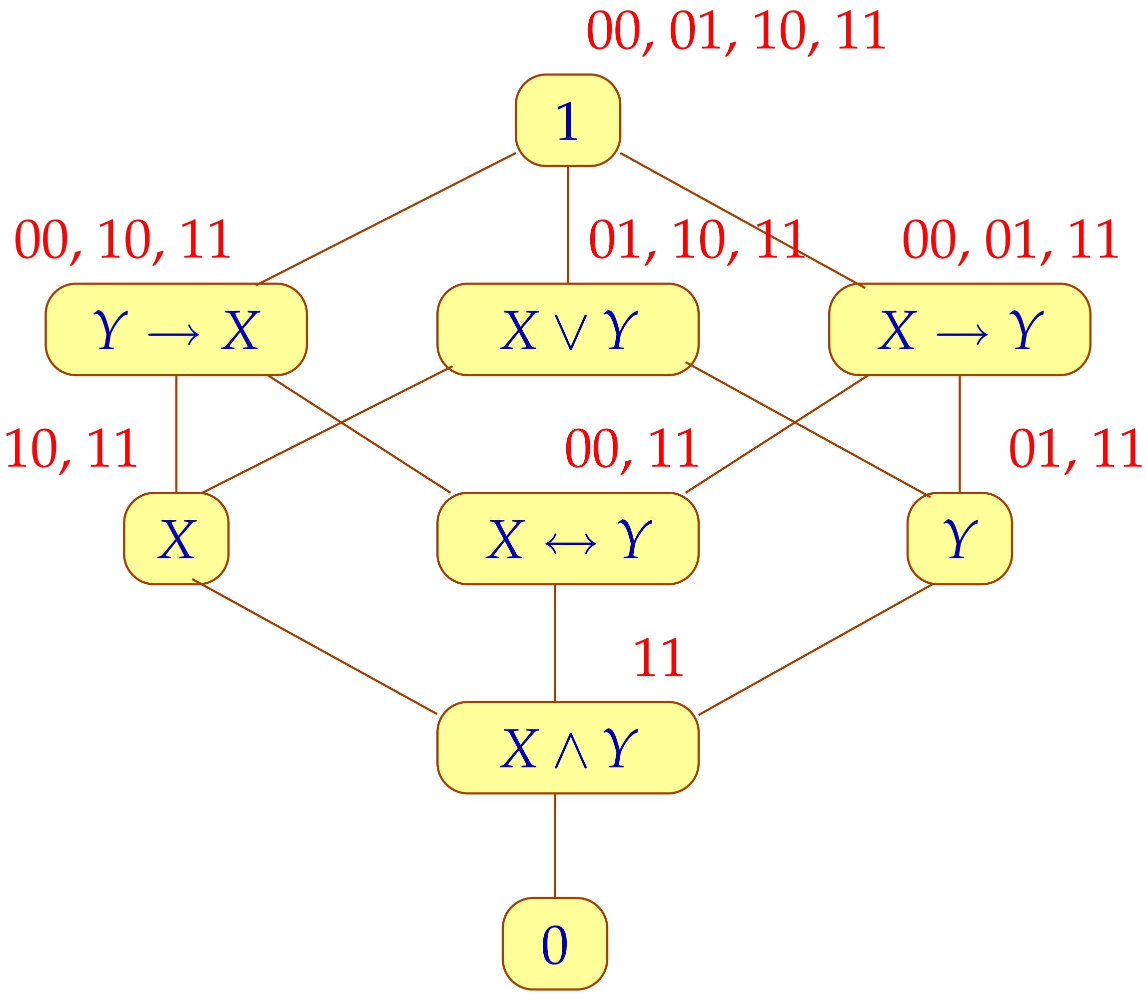
  This means that for every occurring function $\phi \neq 0$,

  $$\phi(1, \ldots, 1) = 1$$

  These functions are called positive.
- The set of all positive functions is called Pos.

  Ordering: $\phi_1 \sqsubseteq \phi_2$ if $\phi_1 \Rightarrow \phi_2$.
- In particular, the least element is 0 :-)

Example:


00, 01, 10, 11
1
00, 10, 11
01, 10, 11
00, 01, 11
$Y \rightarrow X$
$X \vee Y$
$X \rightarrow Y$
10, 11
00, 11
01, 11
X
$X \leftrightarrow Y$
Y
11
$X \wedge Y$
0

Remarks:

- Not all positive functions are monotonic !!!
- For k variables, there are $2^{2^k-1}+1$ many functions.
- The height of the complete lattice is 2^k.
- We construct an interprocedural analysis which for every predicate p determines a (monotonic) transformation

$$[\![p]\!]^\sharp \; : \; \mathsf{Pos} \to \mathsf{Pos}$$

- For every clause, $\quad p(X_1, \ldots, X_k) \Leftarrow g_1, \ldots, g_n \quad$ we obtain the constraint:

$$[\![p]\!]^\sharp \, \psi \quad \sqsupseteq \quad \exists \, X_{k+1}, \ldots, X_m. \, [\![g_n]\!]^\sharp \, (\ldots ([\![g_1]\!]^\sharp \, \psi) \ldots)$$

// $\quad m$ number of clause variables

Abstract Unification:

$$[\![X = t]\!]^\sharp \psi \quad = \quad \psi \wedge (X \leftrightarrow X_1 \wedge \ldots \wedge X_r)$$

$$\text{if} \quad \mathit{Vars}(t) = \{X_1, \ldots, X_r\}.$$

Abstract Literal:

$$[\![q(s_1, \ldots, s_k)]\!]^\sharp \psi \quad = \quad \mathsf{combine}^\sharp_{s_1,\ldots,s_k}(\psi, [\![q]\!]^\sharp\,(\mathsf{enter}^\sharp_{s_1,\ldots,s_k}\psi))$$

// analogous to procedure call !!

Thereby:

$$\mathsf{enter}^\sharp_{s_1,\ldots,s_k}\psi \quad = \quad \mathsf{ren}\,(\exists\, X_1,\ldots,X_m.\ [\![\bar{X}_1 = s_1,\ldots,\bar{X}_k = s_k]\!]^\sharp\psi)$$

$$\mathsf{combine}^\sharp_{s_1,\ldots,s_k}(\psi,\psi_1) \quad = \quad \exists\,\bar{X}_1,\ldots,\bar{X}_r.\ \psi \wedge [\![\bar{X}_1 = s_1,\ldots,\bar{X}_k = s_k]\!]^\sharp(\overline{\mathsf{ren}}\,\psi_1)$$

where

$$\exists\, X.\,\phi \quad = \quad \phi[0/X] \vee \phi[1/X]$$

$$\mathsf{ren}\,\phi \quad = \quad \phi[X_1/\bar{X}_1,\ldots,X_k/\bar{X}_k]$$

$$\overline{\mathsf{ren}}\,\phi \quad = \quad \phi[\bar{X}_1/X_1,\ldots,\bar{X}_r/X_r]$$

Example:

$$\mathsf{app}(X,Y,Z) \quad\leftarrow\quad X = [\,],\; Y = Z$$
$$\mathsf{app}(X,Y,Z) \quad\leftarrow\quad X = [H|X'],\; Z = [H|Z'],\; \mathsf{app}(X',Y,Z')$$

Then

$$\begin{array}{lll} [\![\mathsf{app}]\!]^\sharp(X) & \sqsupseteq & X \wedge (Y \leftrightarrow Z) \\ [\![\mathsf{app}]\!]^\sharp(X) & \sqsupseteq & \mathbf{let}\ \psi = X \wedge H \wedge X' \wedge (Z \leftrightarrow Z') \\ & & \mathbf{in}\ \exists\, H, X', Z'.\ \mathsf{combine}^\sharp_{\dots}(\psi, [\![\mathsf{app}]\!]^\sharp(\mathsf{enter}^\sharp_{\dots}(\psi))) \end{array}$$

where for $\quad \psi = X \wedge H \wedge X' \wedge (Z \leftrightarrow Z')$:

$$\begin{array}{lll} \mathsf{enter}^\sharp_{\dots}(\psi) & = & X \\ \mathsf{combine}^\sharp_{\dots}(\psi, X \wedge (Y \leftrightarrow Z)) & = & (X \wedge H \wedge X' \wedge (Z \leftrightarrow Z') \wedge (Y \leftrightarrow Z') \end{array}$$