Constraint Logic Programming
User Defined Constraints

*User defined constraints* or *rules* allow the programmer to reuse constraints.

\[
\text{parallel_resistors}(V, I, R1, R2) : - \\
V = I1 \times R1 , V = I2 \times R2, I = I1 + I2.
\]

A *User defined constraint* is of the form \( p(t_1, \ldots, t_n) \) where \( p \) is an \( n \)-ary *predicate* and \( t_1, \ldots, t_n \) are expressions from the constraint domain.
Definitions

*literal*: A primitive constraint or a user defined constraint

*goal*: A sequence of literals.

A *rule*: is of the form $H :\!:- B$ where

1. $H$ (the *head* of the rule) is a *user-defined constraint*; and

2. $B$ (the *body* is a *goal*).
A CLP Program: Definition

A (constraint logic) program is a sequence of rules.

The definition of a predicate \( p \) in a program \( P \) is the sequence of rules appearing in \( P \) which have a head involving \( p \).
Variants

A *syntactic object* is a constraint, user-defined constraint, rule or goal.

The result of applying a *renaming* $\rho$ to a syntactic object $o$ written $\rho(o)$ is the expression obtained by replacing each variable $x$ in $o$ by $\rho(x)$.

A syntactic object $o$ is a variant of a syntactic object $o'$ if there is a renaming $\rho$ such that $\rho(o) \equiv o'$.

Renaming $\{x \mapsto a, y \mapsto b\}$, renames $f(x, g(x, y), m)$ to $f(a, g(a, b), m)$. 
Rewriting

Let $G$ be a goal of the form

$$L_1, \ldots, L_{i-1}, L_i, L_{i+1}, \ldots, L_m$$

where $L_i$ is the user defined constraint $p(t_1, \ldots, t_n)$ and let $R$ be a rule of the form

$$p(s_1, \ldots, s_n) : \neg B$$

A rewriting of $G$ at $L_i$ by $R$ using $\rho$ is the goal

$$L_1, \ldots, L_{i-1},
\quad t_1 = \rho(s_1), \ldots, t_n = \rho(s_n), \rho(B),
\quad L_{i+1}, \ldots, L_m$$

where $\rho$ is a renaming chosen so that the variables in $\rho(R)$ do not appear in $G$
Examples

% factorial

fac(0,1). % (R1)
fac(N, N*F) :-
    N >= 1, fac(N-1, F). % (R2)

% voltage divider

voltage_divider(V,I, R1, R2, VD, ID) :-
    V1 = I * R1,
    VD = I2 * R2,
    V = V1 + VD,
    I = I2 + ID.
Evaluation

A state is a pair \( \langle G|C \rangle \) where \( G \) is a goal and \( C \) is a constraint. \( C \) is called the constraint store.

A derivation step from \( \langle G_1|C_1 \rangle \) to \( \langle G_2|C_2 \rangle \), written

\[
\langle G_1|C_1 \rangle \Rightarrow \langle G_2|C_2 \rangle
\]

is defined as follows:
Let $G_1$ be the sequence of literals $L_1, \ldots, L_m$ there are two cases:

1. $L_1$ is a primitive constraint. Then $C_2$ is $C \land L_1$ and, if $solv(C_2) = false$, $G_2$ is the empty goal otherwise $G_2$ is $L_2, \ldots, L_m$.

2. $L_1$ is a user-defined constraint. Then $C_2$ is $C_1$ and $G_2$ is a rewriting of $G_1$ at $L_1$ by some rule $R$ in the program using a renaming $\rho$ such that the variables in $\rho(R)$ are different from those of $C_1$ and $G_1$. If there is no rule defining the predicate of $L_1$ then $C_2$ is $false$ and $G_2$ is the empty goal.