The CLP Operational Model

Adapted from the model presented in Kim Marriott and Peter Stuckey, *Programming with Constraints: An Introduction*
Terminology

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.

A *literal* is either a primitive constraint or a user-defined constraint.

A *goal* is a sequence of literals $L_1, \ldots, L_m$. If $m = 0$ the goal is *empty* and is written $\square$.

A *rule* $R$ is of the form $A :- B$ where $A$ is a user-defined constraint and $B$ is a goal. $A$ is the *head* of $R$ and $B$ is the body.

A *fact* is a rule with the empty goal as its body: $A :- \square$ and is simply written as $A$.

A *program* is a sequence of rules.
Rewritings

Let goal $G$ be of the form

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

where $L_i$ is a user-defined constraint $p(t_1, \ldots, t_n)$.

Let rule $R$ be of the form $p(t_1, \ldots, t_n) :- B$.

Let $\rho$ be a renaming (i.e. a mapping that replaces variables with new ones).

A rewriting of $G$ at $L_i$ by $R$ using $\rho$ is the new goal formed from $G$ by replacing $L_i$ with

$t_1 = \rho(s_1), \ldots, t_n = \rho(s_n), \rho(B)$

where $\rho$ is chosen so that the variables in $\rho(R)$ do not appear in $G$. 
Derivation Steps

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

A derivation step from \( \langle G_1 \mid C_1 \rangle \) to \( \langle G_2 \mid C_2 \rangle \) is written:

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

It is defined as follows.

Let \( G_1 \) be the sequence of literals \( L_1, L_2, \ldots, L_m \).

Case 1: \( L_1 \) is a primitive constraint. Then \( C_2 \) is \( C_1 \land L_1 \). If the constraint solver determines that \( C_2 \) is unsatisfiable, then \( G_2 \) is the empty goal; otherwise \( G_2 \) is \( L_2, \ldots, L_m \).

Case 2: \( L_1 \) is a user-defined constraint. Let \( L_1 \) have the form \( p(t_1, \ldots, t_n) \). Select a rule \( R \) in program whose head is a literal \( p(s_1, \ldots, s_n) \). Then \( C_2 \) is \( C_1 \) and \( G_2 \) is found by a rewriting of \( G \) at \( L_1 \) using \( R \). If there is no rule to use for the rewriting, then \( C_2 \) is false and \( G_2 \) is the empty goal.
**Derivations**

A *derivation* for a goal $G$ is a sequence of derivation steps starting with $\langle G \mid true \rangle$.

A derivation can continue until the goal becomes empty. A derivation that can no longer continue can be either successful or failed.

A derivation is successful if the last state is $\langle \Box \mid C_n \rangle$, and the constraint solver doesn’t determine that constraint $C_n$ is unsatisfiable. The constraint that is the result of simplifying $C_n$ with respect to the variables in $G$ is an *answer* to $G$.

A derivation is failed if the last state is $\langle \Box \mid C_n \rangle$, and the constraint solver determines that constraint $C_n$ is unsatisfiable.
Example Derivation

CLP(\(\mathcal{R}\)) program:

\[
\text{cf}(C,F) :- \quad \text{F}=1.8\times C+32. \quad / * \text{ rule R1 } */
\]
\[
\text{double}(X,Y) := \quad \text{Y}=2\times X. \quad / * \text{ rule R2 } */
\]

Consider the goal \(\text{cf}(A,B), \quad \text{double}(A,200)\).

\[\langle \text{cf}(A, B), \text{double}(A, 200) | \text{true} \rangle\]

\[
\Rightarrow
\]

using R1:

\[\langle A = C, B = F, F = 1.8 \times C + 32, \quad \text{double}(A, 200) | \text{true} \rangle\]
\[
\Rightarrow \\
\langle B = F, F = 1.8 \times C + 32, double(A, 200) \mid A = C \rangle
\]

\[
\Rightarrow \\
\langle F = 1.8 \times C + 32, double(A, 200) \mid A = C, B = F \rangle
\]

\[
\Rightarrow \\
\langle double(A, 200) \mid A = C, B = F, F = 1.8 \times C + 32 \rangle
\]

\[
\Rightarrow \\
\text{using R2:} \\
\langle A = X, 200 = Y, Y = 2 \times X \mid A = C, B = F, F = 1.8 \times C + 32 \rangle
\]
\[ \langle 200 = Y, Y = 2 \times X \mid A = C, B = F, F = 1.8 \times C + 32, A = X \rangle \]

\[ \Rightarrow \]

\[ \langle Y = 2 \times X \mid A = C, B = F, F = 1.8 \times C + 32, A = X, 200 = Y \rangle, \]

\[ \Rightarrow \]

\[ \langle \Box \mid A = C, B = F, F = 1.8 \times C + 32, A = X, 200 = Y, Y = 2 \times X \rangle \]

Simplifying with respect to the variables in \( G_0 \) (namely \( A, B \)) we get the answer \( A = 100, B = 212 \).

This is a successful derivation.
Derivation Trees

There may be more than one derivation for a goal. The derivation tree contains all the derivations for a given goal $G$. The CLP system will in effect incrementally construct the derivation tree as it searches for an answer.

Definition: a *derivation tree* for a goal $G$ and program $P$ is a tree with states as nodes. The root is $\langle G \mid \text{true} \rangle$. The children of each state $\langle G_i \mid C_i \rangle$ are the states that can be reached in a single derivation step. A state with two or more children is a *choicepoint*.

CLP(R) evaluates a goal by performing a depth-first, left-to-right traversal of the goal’s derivation tree. Whenever a success state is encountered the system returns the corresponding answer. The user can accept the answer, or reject it (so that traversal continues).
Simplified Derivation Trees

We can produce a simplified derivation tree by omitting uninteresting steps and simplifying the resulting constraints. A simplified derivation for a goal $G$ includes the first and last states in the full derivation, and every state for which the first literal in the goal is a user-defined constraint.

Simplified derivation tree for the previous example:

\[
\langle cf(A, B), \text{double}(A, 200) \mid true \rangle \Rightarrow \langle \text{double}(A, 200) \mid B = 1.8 \times A + 32 \rangle \Rightarrow \langle \Box \mid A = 100, B = 212 \rangle
\]