The Essence of Contraint Logic Programming

Hassan Aït-Kaci
ILOG, Inc.

November 3, 2006

In 1987, at the height of research interest in Logic Programming (LP), Jaffar and Lassez, proposed a novel Logic Programming *scheme* called *Constraint Logic Programming* (CLP) [2]. The idea was to generalize the operational and denotational semantics of LP by dissociating the relational level—pertaining to resolving definite clauses made up of relational atoms—and the data level pertaining to the nature of the arguments of these relational atoms (*e.g.*, for Prolog, first-order Herbrand terms). Thus, for example, in Prolog seen as a CLP language, clauses such as:

```
append([],L,L).
append([H|T],L,[H|R]) :- append(T,L,R).
```

are construed as:

```
append(X1,X2,X3) :- true
                  | X1 = [], X2 = L, X3 = L.
append(X1,X2,X3) :- append(X4,X5,X6)
                  | X1 = [H|T], X2 = L, X3 = [H|R],
                    X4 = T, X5 = L, X6 = R.
```

The Höhfeld-Smolka Scheme Höhfeld and Smolka [1] proposed a refinement of the Jaffar-Lassez's scheme [2] both more general and simpler than what was originally proposed in that it abstracts away the syntax of constraint formulae and relaxes some technical demands on the constraint language—in particular, the somewhat baffling "solution-compactness" requirement[1] [2].

The Höhfeld-Smolka constraint logic programming scheme requires a set $\mathfrak{R}$ of *relational symbols* (or, predicate symbols) and a *constraint language* $\mathcal{L}$. It needs very few assumptions about the language $\mathcal{L}$, which must only be characterized by:

- $\mathcal{V}$, a countably infinite set of *variables* (denoted as capitalized $X, Y, \ldots$);
- Φ, a set of *formulae* (denoted $\phi, \phi', \ldots$) called *constraints*;
- a function *Var*: $\Phi \mapsto \mathcal{V}$, which assigns to every constraint ϕ the set *Var*(ϕ) of *variables constrained by* ϕ;

[1] "Compactness" in logic is the property stating that if a formula is provable, then it is provable in finitely many steps.

- a family of "admissible" *interpretations* $\mathfrak{A}$ over some domain $D^{\mathfrak{A}}$;
- the set $Val(\mathfrak{A})$ of ($\mathfrak{A}$-)*valuations*, *i.e.*, total functions, $\alpha : \mathcal{V} \mapsto D^{\mathfrak{A}}$.

Thus, $\mathcal{L}$ is not restricted to any specific syntax, *a priori*. Furthermore, nothing is presumed about any specific method for proving whether a constraint holds in a given interpretation $\mathfrak{A}$ under a given valuation α. Instead, we simply assume given, for each admissible interpretation $\mathfrak{A}$, a function $[\![\_\_]\!]^{\mathfrak{A}} : \Phi \mapsto \mathbf{2}^{(Val(\mathfrak{A}))}$ which assigns to a constraint $\phi \in \Phi$ the set $[\![\phi]\!]^{\mathfrak{A}}$ of valuations which we call the *solutions* of ϕ under $\mathfrak{A}$.

Generally, and in our specific case, the constrained variables of a constraint ϕ will correspond to its free variables, and α is a solution of ϕ under the interpretation $\mathfrak{A}$ if and only if ϕ holds true in $\mathfrak{A}$ once its free variables are given values α. As usual, we shall denote this as "$\mathfrak{A}, \alpha \models \phi$."

Then, given $\mathfrak{R}$, the set of relational symbols (denoted $r, r_1, \ldots$), and $\mathcal{L}$ as above, the language $\mathfrak{R}(\mathcal{L})$ of *relational clauses* extends the constraint language $\mathcal{L}$ as follows. The syntax of $\mathfrak{R}(\mathcal{L})$ is defined by:

- the same countably infinite set $\mathcal{V}$ of *variables*;
- the set $\mathfrak{R}(\Phi)$ of formulae ρ from $\mathfrak{R}(\mathcal{L})$ which includes:
  - all $\mathcal{L}$-constraints, *i.e.*, all formulae ϕ in Φ;
  - all relational atoms $r(X_1, \ldots, X_n)$, where $X_1, \ldots, X_n \in \mathcal{V}$, mutually distinct;

  and is closed under the logical connectives $\&$ (conjunction) and $\rightarrow$ (implication); *i.e.*,
  - $\rho_1 \;\&\; \rho_2 \in \mathfrak{R}(\Phi)$ if $\rho_1, \rho_2 \in \mathfrak{R}(\Phi)$;
  - $\rho_1 \rightarrow \rho_2 \;\; \in \mathfrak{R}(\Phi)$ if $\rho_1, \rho_2 \in \mathfrak{R}(\Phi)$;
- the function $Var : \mathfrak{R}(\Phi) \mapsto \mathcal{V}$ extending the one on Φ in order to assign to every formula ρ the set *Var*(ρ) of the *variables constrained by* ρ:
  - $Var(r(X_1, \ldots, X_n)) = \{X_1, \ldots, X_n\}$;
  - $Var(\rho_1 \;\&\; \rho_2) = Var(\rho_1) \;\cup\; Var(\rho_2)$;
  - $Var(\rho_1 \rightarrow \rho_2) = Var(\rho_1) \;\cup\; Var(\rho_2)$;
- the family of "admissible" *interpretations* $\mathfrak{A}$ over some domain $D^{\mathfrak{A}}$ such that $\mathfrak{A}$ extends an admissible interpretation $\mathfrak{A}_0$ of $\mathcal{L}$, over the domain $D^{\mathfrak{A}} = D^{\mathfrak{A}_0}$ by adding relations $r^{\mathfrak{A}} \subseteq D^{\mathfrak{A}} \times \ldots \times D^{\mathfrak{A}}$ for each $r \in \mathfrak{R}$;
- the same set $Val(\mathfrak{A})$ of *valuations* $\alpha : \mathcal{V} \mapsto D^{\mathfrak{A}}$.

Again, for each interpretation $\mathfrak{A}$ admissible for $\mathfrak{R}(\mathcal{L})$, the function $[\![\_\_]\!]^{\mathfrak{A}} : \mathfrak{R}(\Phi) \mapsto \mathbf{2}^{(Val(\mathfrak{A}))}$ assigns to a formula $\rho \in \mathfrak{R}(\Phi)$ the set $[\![\phi]\!]^{\mathfrak{A}}$ of valuations which we call the *solutions* of ρ under $\mathfrak{A}$. It is defined to extend the interpretation of constraint formulae in $\Phi \subseteq \mathfrak{R}(\Phi)$ inductively as follows:

- $[\![r(X_1,\ldots,X_n)]\!]^{\mathfrak{A}} = \{\alpha \mid \langle\alpha(X_1),\ldots,\alpha(X_n)\rangle \in r^{\mathfrak{A}}\}$;
- $[\![\phi_1 \ \& \ \phi_2]\!]^{\mathfrak{A}} = [\![\phi_1]\!]^{\mathfrak{A}} \ \cap \ [\![\phi_2]\!]^{\mathfrak{A}}$;
- $[\![\phi_1 \rightarrow \phi_2]\!]^{\mathfrak{A}} = (\,Val(\mathfrak{A}) - [\![\phi_1]\!]^{\mathfrak{A}}) \ \cup \ [\![\phi_2]\!]^{\mathfrak{A}}$.

Note that an $\mathcal{L}$-interpretation $\mathfrak{A}_0$ corresponds to an $\mathfrak{R}(\mathcal{L})$-interpretation $\mathfrak{A}$, namely where $r^{\mathfrak{A}_0} = \emptyset$ for every $r \in \mathfrak{R}$.

As in Prolog, we shall limit ourselves to *definite relational clauses* in $\mathfrak{R}(\mathcal{L})$ that we shall write in the form:

$$r(\vec{X}) \ \leftarrow r_1(\vec{X}_1) \ \& \ \ldots \ \& \ r_m(\vec{X}_m) \ \& \ \phi,$$

$(0 \leq m)$, making its constituents more conspicuous and also to be closer to 'standard' Logic Programming notation, where:

- $r(\vec{X}), r_1(\vec{X}_1), \ldots, r_m(\vec{X}_m)$ are relational atoms in $\mathfrak{R}(\mathcal{L})$; and,
- ϕ is a constraint formula in $\mathcal{L}$.

Given a set $\mathfrak{C}$ of definite $\mathfrak{R}(\mathcal{L})$-clauses, a *model* of $\mathfrak{C}$ is an $\mathfrak{R}(\mathcal{L})$-interpretation such that every valuation $\alpha : \mathcal{V} \mapsto D^{\mathfrak{M}}$ is a solution of every formula ρ in $\mathfrak{C}$, *i.e.*, $[\![\rho]\!]^{\mathfrak{M}} = Val(\mathfrak{M})$. It is a fact established in [1] that any $\mathcal{L}$-interpretation $\mathfrak{A}$ can be extended to a *minimal model* $\mathfrak{M}$ of $\mathfrak{C}$. Here, minimality means that the added relational structure extending $\mathfrak{A}$ is minimal in the sense that if $\mathfrak{M}'$ is another model of $\mathfrak{C}$, then $r^{\mathfrak{M}} \subseteq r^{\mathfrak{M}'}$ $(\subseteq D^{\mathfrak{A}} \times \ldots \times D^{\mathfrak{A}})$ for all $r \in \mathfrak{R}$.

Also established in [1], is a fixed-point construction. The minimal model $\mathfrak{M}$ of $\mathfrak{C}$ extending the $\mathcal{L}$-interpretation $\mathfrak{A}$ can be constructed as the limit $\mathfrak{M} = \bigcup_{i\geq 0} \mathfrak{A}_i$ of a sequence of $\mathfrak{R}(\mathcal{L})$-interpretations $\mathfrak{A}_i$ as follows. For all $r \in \mathfrak{R}$ we set:

$$\begin{aligned} r^{\mathfrak{A}_0} &= \emptyset; \\ r^{\mathfrak{A}_{i+1}} &= \{\langle\alpha(x_1),\ldots,\alpha(x_n)\rangle \mid \alpha \in [\![\rho]\!]^{\mathfrak{A}_i} \ ; \ r(x_1,\ldots,x_n) \leftarrow \rho \in \mathfrak{C}\}; \\ r^{\mathfrak{M}} &= \textstyle\bigcup_{i\geq 0} r_i^{\mathfrak{A}}. \end{aligned}$$

A *resolvent* is a formula of the form $\rho \ \ [\!] \ \ \phi$, where ρ is a possibly empty conjunction of relational atoms $r(X_1,\ldots,X_n)$ (its *relational part*) and ϕ is a possibly empty conjunction of $\mathcal{L}$-constraints (its *constraint part*). The symbol $[\!]$ is in fact just the symbol $\&$ in disguise. It is simply used to emphasize which part is which. (As usual, an empty conjunction is assimilated to *true*, the formula which takes all arbitrary valuations as solution.)

Finally, the Höhfeld-Smolka scheme defines constrained *resolution* as a reduction rule on resolvents which gives a sound and complete interpreter for *programs* consisting of a set $\mathfrak{C}$ of definite $\mathfrak{R}(\mathcal{L})$-clauses. The reduction of a *resolvent* R of the form:

- $B_1 \ \& \ \ldots \ \& \ r(X_1,\ldots,X_n) \ \& \ \ldots B_k \ \ [\!] \ \ \phi$

by the (renamed) program clause:

- $r(X_1,\ldots,X_n) \leftarrow A_1 \ \& \ \ldots \ \& \ A_m \ \& \ \phi'$

is the new resolvent R' of the form:

- B_1 & $\ldots$ & A_1 & $\ldots$ & A_m & $\ldots B_k$ $\|$ ϕ & ϕ'.

The soundness of this rule is clear: under every interpretation $\mathfrak{A}$ and every valuation such that R holds, then so does R', *i.e.*, $[\![R']\!]^{\mathfrak{A}} \subseteq [\![R]\!]^{\mathfrak{A}}$. It is also not difficult to prove its completeness: if $\mathfrak{M}$ is a minimal model of $\mathfrak{C}$, and $\alpha \in [\![R]\!]^{\mathfrak{M}}$ is a solution of the formula R in $\mathfrak{M}$, then there exists a sequence of reductions of (the $\mathfrak{R}(\mathcal{L})$-formula) R to an $\mathcal{L}$-constraint ϕ such that $\alpha \in [\![\phi]\!]^{\mathfrak{M}}$.

References

[1] Markus Höhfeld and Gert Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Stuttgart, Germany, October 1988.

[2] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, W. Germany, January 1987.