Constraint Solving Tecniques

Notes by Catuscia Palamidessi

The issue of implementing CLP languages mostly reduces to how to implement the constraint part and how to integrate it with the usual logic inference engine.

Let us recall some of the alternatives to simplify constraints without solving them completely. For example, information on single variables could be propagated instead of testing the global consistency of the constraints. Let us show a simple example. Suppose that T_i, $i : 1, .., 4$ are variables ranging on the finite domain $\{1, ..., 4\}$, that correspond to four tasks with order of execution expressed by constraints like $before(T_i, T_j)$, where $i \neq j$. Let us consider the following set of constraints:

$before(T_1, T_2)$.
$before(T_2, T_4)$.
$before(T_1, T_3)$.
$before(T_3, T_4)$.
$notequal(T_2, T_3)$.

Starting from these constraints and the information on the domain, (i.e. $T_i \in \{1, 2, 3, 4\}$) we can use a local consistency test between pairs of variables in order to find restrictions to their ranges. For example, from $T_1 \in \{1, 2, 3, 4\}$, $T_2 \in \{1, 2, 3, 4\}$ and $before(T_1, T_2)$ we can infer that $T_1 \in \{1, 2, 3\}$ and $T_2 \in \{2, 3, 4\}$. Essentially, the values in the domain of a given variable are removed if inconsistent w.r.t. to the values of another one (arc consistency). If we propagate the domains restriction of the variables T_1 and T_2 to the other variables and then we apply arc consistency test as much as possible, we obtain the following restrictions on the domains of the variables:

$$T_1 \in \{1, 2\}, T_2 \in \{2, 3\}, T_3 \in \{2, 3\}, T_4 \in \{3, 4\}$$

Using this technique we have restricted the ranges of the constraint variables, but we have still not computed a solution. Here the idea is to choose values for some variables from the corresponding ranges, in order to get a solution, which is called *labeling*. Thus in the example if we set $T_1 = 1$, using the constraint $before$ we can infer $T_i = i, i : 2, ..., 4$.

For finite domains with integers, equations and disequations, there are other consistency tests such as the *maxima and minima criterion*, in which at every step, using information on maximum and minimum values of variables, we restrict their range values.

In simple examples like the previous one, solving directly the constraints seems better than using propagation through constraints. On the other hand, when using hundreds of variables, propagation is much more convenient as it requires only local tests on constraints, even if it can produce inconsistent configurations.

In general a CLP program has a standard schema:

- Declaration of problem variables with their domains (in case of finite domains their ranges);
- Setting of the constraints;
- Labeling of variables.

Using a method such as propagation (which is intrinsically deterministic), the only part that could introduce nondeterminism is the labeling of variables. However, in many cases, after propagation the search space results so small that only few choices are possible.