6.2 CLP in Prolog

Freitag, 3. Juli 2015 08:30

In Prolog, one first has to say which constraint theory CT should be used.

CLP-libraries come in modules (some of them are included in Prolog-distributions).

use_module/1 is a predicate to import modules

To import the library with the constraint theory CT_{FD}, the Prolog program must contain the directive:

```
:- use_module(library(clpfd)).
```

Problem: Constraint solver for CT is needed to check satisfiability of constraints in each computation step (and to simplify constraints).

$\Rightarrow$ should be automatic + efficient

But: most constraint theories are undecidable or only have very time-consuming decision procedures (CT_{FD} is undecidable).

Solution: Instead of checking

$$CT \cup \{\forall X = X, true\} \models \exists \, CO$$

this question is only "approximated".
This is efficient, but not always correct
(i.e., there might be conjunctions of constraints CO where Prolog falsely detects their satisfiability).

To approximate satisfiability in CT_{FD}, one typically uses <u>path-consistency</u>.

<u>Def. 6.2.1</u> (Path Consistency)
Let $CO = \varphi_1 \wedge \ldots \wedge \varphi_m$ be a conjunction of constraints with $\varphi_i \in At(\Sigma_{FD}, \Delta_{FD}, \mathcal{V})$. Let $X_1, \ldots, X_n$ be the variables in CO and let $D_1, \ldots, D_n$ be subsets of $\mathbb{Z}$. $D_1, \ldots, D_n$ are <u>admissible domains</u> for $X_1, \ldots, X_n$ w.r.t. CO iff for all φ_i and all variables X_j the following holds:

for all $a_j \in D_j$ there exist $a_1 \in D_1, \ldots, a_{j-1} \in D_{j-1}, a_{j+1} \in D_{j+1}, \ldots, a_n \in D_n$

such that $CT_{FD} \models \varphi_i[X_1/a_1, \ldots, X_n/a_n]$.

CO is <u>path-consistent</u> iff there are admissible domains

$D_1, \dots, D_n$ that are all not empty.

Problem: Satisfiability of the constraints separately:
if φ_1 and φ_2 are both satisfiable,
then $\varphi_1 \wedge \varphi_2$ still does not need to be satisfiable.

Automated checking of Path-Consistency:

1. Let $D_1 = \mathbb{Z}, \dots, D_n = \mathbb{Z}$.
2. Process the constraints φ after each other. For each φ:
3. Process the variables after each other: For X_j:
   Reduce its domain D_j to those values where φ can be made true if the other variables can only take values from their domains.
4. The whole process is repeated until the constraints do not change any more.

Ex 622 Let CO be

$$X_1 \ \#> 5 \ \wedge \ X_1 \ \#< X_2 \wedge \ X_2 \ \#< 9$$

- Beginning: $D_1 = \mathbb{Z}$, $D_2 = \mathbb{Z}$
- Consider $X_1 \ \#> 5 \Rightarrow D_1 = \{6, 7, 8, \dots\}$, $D_2 = \mathbb{Z}$
- Consider $X_1 \ \#< X_2 \Rightarrow$

$D_1 = \{6,7,8,\ldots\}$, $D_2 = \mathbb{Z}$ For every $a_1 \in D_1$ there exists $a_2 \in D_2$ such that $a_1 \# < a_2$.

$D_1 = \{6,7,\ldots\}$, $D_2 = \{7,8,\ldots\}$ For every $a_2 \in D_2$ there exists $a_1 \in D_1$ such that $a_1 \# < a_2$.

- Consider $X_2 \# < 9 \Rightarrow D_1 = \{6,7,\ldots\}$, $D_2 = \{7,8\}$
- Consider $X_1 \# > 5 \Rightarrow D_1 = \{6,7,\ldots\}$, $D_2 = \{7,8\}$
- Consider $X_1 \# < X_2 \Rightarrow D_1 = \{6,7\}$, $D_2 = \{7,8\}$

$\vdots$

Now nothing changes anymore $\Rightarrow$
CO is path-consistent ($D_1 \neq \emptyset$, $D_2 \neq \emptyset$).
Simplify$_C$(CO) = X1 in 6..7, X1 #< X2, X2 in 7..8

The constraint signature contains more symbols:
in, ...

CT_{FD} "should" be used for finite domains, but this is not enforced:

?- X1 in 6..sup, X1 #< X2, X2 in inf..8

$$\text{sup} \;\hat{=}\; \infty \qquad \text{inf} \;\hat{=}\; -\infty$$

CT_{FD} has a predicate "label" which enforces that all solutions are enumerated:

?- X1 #> 5, X1 #< X2, X2 #< 9, label([X1, X2]).

list of variables for which answer substitutions should be computed

X1 = 6, X2 = 7 ;
X1 = 6, X2 = 8 ;
X1 = 7, X2 = 8 ;
false

label can only be used if all the variables have finite domains:

?- X1 #> 5, X1 #< X2, label([X1, X2]).

prog. error

Ex 6.2.3 Incorrectness of Path Consistency

?- X1 #> X2, X1 #=< X2.

$X_1 \#> X_2 \wedge X_1 \#=< X_2$ is path-consistent, but unsatisfiable.

$D_1 = \mathbb{Z}$ $D_2 = \mathbb{Z}$ are admissible domains.

- for every $a_1 \in D_1$, there exists $a_2 \in D_2$ such that
  $a_1 \;\cancel{\#>}\; a_2$
- for every $a_1 \in D_1$, there exists $a_2 \in D_2$ such that
  $a_1 \;\cancel{\#=<}\; a_2$
- for every $a_2 \in D_2$, ...

Ex 6.2.4 n-queens problem

- chess board of size $n \times n$
- place n queens on the board that cannot beat each other

	1	2	3	4
1			X	
2	X			
3				X
4		X		

- Represent the positions of queens by a list $[X_1, \dots, X_n]$ where X_i is the row for the queen of column i. (e.g. [2,4,1,3]).
- ?- queens(4,L) will compute solution L for chess-board of size 4×4.
- "L ins 1..N" means "X in 1..N" for

every element X of L

- all_different is pre-defined
- first three literals: L is a permutation of {1, .., N}