Programmiersprachen

Alexandru Berlea

Institut für Informatik
TU München

Wintersemester 2006/2007

Constraints über endliche (ganzzahlige) Wertebereiche (FD-Constraints)

Constraints:

$$C \quad ::= \quad \text{true} \mid \text{false} \mid C \wedge C \mid X \text{ in } n..m \mid$$

$$X \text{ in } [k_1, \ldots, k_l] \mid X + Y = Z \mid X \; \mathcal{R} \; Y$$

mit $n, m, k_1, \ldots, k_l \in \mathbb{N}$

$\mathcal{R} \in \{<, \leq, =, >, \geq, \neq\}$

X, Y, Z Variablen oder ganze Zahlen

Beispiel

FD-Constraints in SWI-Prolog:

```
?- use_module(library('clp/bounds')).
Yes

?- X in 1..2, Y in 3..5, Z #= X+Y.
X = _G137389{1..2}
Y = _G137395{3..5}
Z = _G137401{4..7}
```

Constraint Satisfaction Probleme (CSPs)

- Ein CSP ist definiert durch:
  - Eine Menge von Variablen $X_1 \in D_1, \ldots, X_n \in D_n$, mit $D_1, \ldots, D_n$ endlichen Wertebereichen
  - Eine Menge von Constraints, die von den Variablen zu erfüllen sind.
- Eine Lösung eines CSP ist eine Variablenbelegung, die die Constraints erfüllt.

Lösung eines CSP mit CP(FD)

Der Ansatz zur Lösung eines CSP mit Constraints über endliche Bereiche besteht typischerweise aus drei Komponenten:

1. Deklaration der Wertebereiche der Variablen.
2. Aufsetzen der Constraints
3. Suche einer Lösung (Backtracking, Branch-and-Bound)

Beispiel: CSP Problem

- Finde eine Belegung, so dass die Folgende Berechnung (zur Basis 10) wahr ist!

$$\begin{array}{cccccc} & & S & E & N & D \\ & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

- Suchraum hat die Größe 10^8. Exhaustive Suche ist praktisch nicht einsetzbar.

- Ein Mensch würde das Problem lösen, indem er Constraints dynamisch ableitet. Z.B. muss M gleich 1 sein. Es folgt, dass S gleich 9 oder 8 ist. U.s.w...

Beispiel: Send more money

1. Ansatz (SWI-Prolog):

```
:- use_module(library('clp/bounds')).

solve(S,E,N,D,M,O,R,Y) :-
        [S,E,N,D,M,O,R,Y] in 0..9,
        S#>0, M#>0,
        all_different([S,E,N,D,M,O,R,Y]),
                        1000*S + 100*E + 10*N + D
        +               1000*M + 100*O + 10*R + E
        #= 10000*M + 1000*O + 100*N + 10*E + Y.
```

```
?- solve(S,E,N,D,M,O,R,Y).
M = 1, O = 0, S = 9, E in 2..8, N in 5..8,
D in 2..8, R in 2..8, Y in 2..8
```

Suche

- Ein Constraint-Löser schließt Werte aus, die nicht zu einer Lösung beitragen können:

```
X in 1..3, Y in 4..6, Y #= 2*X.
X in 2..3 Y in 4..6
```

- I.A. muss man noch eine Suche über diese Werte durchführen, um eventuelle eindeutige Lösungen zu finden.
  Man braucht ein Prädikat, welches Variablen alle Werte aus ihren Wertebereichen nach einer vorgegebenen Strategie zuordnet.

```
X in 1..3, Y in 4..6, Y #= 2*X, label([X,Y]).
X = 2 Y = 4 ;
X = 3 Y = 6 ;
No
```

Beispiel: Send more money

2. Lösung mit Suche:

```
solve1(S,E,N,D,M,O,R,Y) :-
      [S,E,N,D,M,O,R,Y] in 0..9,
      S#>0, M#>0,
      all_different([S,E,N,D,M,O,R,Y]),
                     1000*S + 100*E + 10*N + D
      +              1000*M + 100*O + 10*R + E
      #= 10000*M + 1000*O + 100*N + 10*E + Y,
      label([S,E,N,D,M,O,R,Y]).
```

```
?- solve1(S,E,N,D,M,O,R,Y).
S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2
```