Constraint Programming

Constraint Programming: Extending the SAT language

- We've seen usefulness of SAT and MAX-SAT
  - Candidate solutions are assignments
  - Clauses are a bunch of competing constraints on assignments
- Constraint programming offers a richer language:
  - convenient
    - Don't have to express each constraint as a disjunction of literals
    - Encodings closer to how you think about problem
  - maybe more efficient
    - Fewer constraints: saves on storage, indexing, and propagation
    - Special handling for particular types of constraints
  - maybe more general
    - Leads toward generalizations, e.g., real-valued variables

ECLiPSe

(= ECLiPSe Constraint Logic Programming System)

- One of many constraint programming software packages
- Free for academic use
- Nice constraint language
- Several solver libraries
- Extensible – you can define your own new constraint types and new solvers

Integer constraints

What happens if you don't say this?

- X :: [2,4,6,8,10..20] % X has one of these vals
- X #= Y % # for a constraint
- X #< Y % less than
- X #\= 3 % inequality
- X + Y #= Z % arithmetic
- X*Y + Z^2 #= 70
- ordered([A,B,C,D])
- alldifferent([A,B,C,D])
- sum([A,B,C,D], E)
- minlist([A,B,C,D], C)
- minlist([A,B,C,D], 3)
- occurrences(…)

Which of these are syntactic sugar?

Global constraints

Real-number constraints

- X :: 1.0 .. Inf % X has a real value in this range
- X $= Y % $ for a constraint on real numbers
- X $< Y % less than
- X $\= 3 % inequality
- X + Y $= Z % arithmetic
- X*Y + Z^2 $= 70
- ordered([A,B,C,D])
- alldifferent([A,B,C,D])
- sum([A,B,C,D], E)
- minlist([A,B,C,D], C)
- minlist([A,B,C,D], 3)
- occurrences(…)

How about numeric precision?
(How do we check if $= is satisfied?)

Interval arithmetic …

Logical operators

- A #= B or A #= C
- A #= B and neg A #= C
- Cost #= (A #= B) + (A #= C)
  - Cost has value 0, 1, or 2
  - If we know A,B,C, we have information about Cost … and vice-versa!
  - Another constraint might say Cost #< 1.

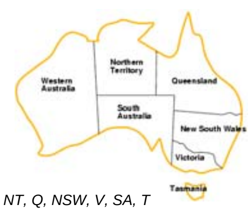
Set constraints

- Variables whose values are sets (rather than integers or reals)
- Constrain A to be a subset of B
- Constrain intersection of A, B to have size 2
- Etc.

Constraint Logic Programming

- ECLiPSe is an extension of Prolog
  - actually a full-fledged language with recursion, etc.
- So a typical ECLiPSe program does the encoding as well as the solving. Advantages?
  - don't have to read/write millions of constraints
  - don't have to store millions of constraints at once (generate new constrained variables during search, eliminate them during backtracking)
  - easier to hide constraint solving inside a subroutine
  - less overhead for small problems
- But for simplicity, we'll just worry about the "little language" of constraints.
  - You can do the encoding yourself in Perl.

Example: Map-Coloring



- Variables *WA, NT, Q, NSW, V, SA, T*
- Domains D_i = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or (WA,NT) in {(red,green),(red,blue),(green,red), (green,blue),(blue,red),(blue,green)}

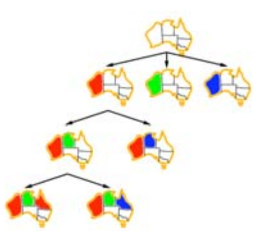
slide thanks to Tuomas Sandholm

Example: Map-Coloring

- Solutions are complete and consistent assignments
- e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

slide thanks to Tuomas Sandholm

We'll talk about solvers next week



slide thanks to Tuomas Sandholm

Varieties of CSPs

- Discrete variables
  - finite domains:
    - *n* variables, domain size *d* → $O(d^n)$ complete assignments
    - e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)
  - infinite domains:
    - integers, strings, etc.
    - e.g., job scheduling, variables are start/end days for each job
    - need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
- Continuous variables
  - e.g., start/end times for Hubble Space Telescope observations
  - linear constraints solvable in polynomial time by Linear Programming

slide thanks to Tuomas Sandholm

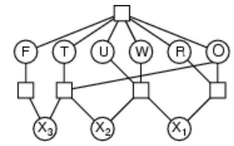
Varieties of constraints

- Unary constraints involve a single variable,
  - e.g., SA ≠ green in the map coloring example
- Binary constraints involve pairs of variables,
  - e.g., SA ≠ WA in the map coloring example
- Higher-order constraints involve 3 or more variables,
  - e.g., cryptarithmetic column constraints (next slide)

slide adapted from Tuomas Sandholm

Example: Cryptarithmetic

```
  T W O
+ T W O
F O U R
```



- Variables: *F T U W R O* X_1 X_2 X_3
- Domains: {0,1,2,3,4,5,6,7,8,9}
- Constraints: *Alldiff (F,T,U,W,R,O)*
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$

slide thanks to Tuomas Sandholm

More examples

- At the ECLiPSe website:
- http://eclipseclp.org/examples/
- Let's play with these in a running copy of ECLiPSe!