DM841

Discrete Optimization

Lecture 3

Introduction to MiniZinc

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Constraint Languages

Resume

- Modelling in CP
  - Examples: graph labelling with consecutive numbers, cryptoarithmetic
- Overview on Constraint Programming
  - modelling
  - search = backtracking + branching
  - propagate (inference) + filtering

Constraint Programming:
representation (modeling language) + reasoning (search + propagation)

Outline

1. Constraint Languages

Constraint Programming Systems

(modeling)
Expressive language stream
+
(efficient solvers)
Algorithm stream

CP systems typically include

- general purpose algorithms for constraint propagation (arc consistency on finite domains)
- built-in constraint propagation for various constraints (eg, linear, Boolean, global constraints)
- built-in for constructing various forms of search

Logic Programming

Logic programming is the use of mathematical logic for computer programming.

First-order logic is used as a purely declarative representation language, and a theorem-prover or model-generator is used as the problem-solver.

Logic programming supports the notion of logical variables

- Syntax – Language
  - Alphabet
  - Well-formed Expressions
    E.g., 4X + 3Y = 10; 2X - Y = 0
- Semantics – Meaning
  - Interpretation
  - Logical Consequence
- Calculi – Derivation
  - Inference Rule
  - Transition System

Example

Rules are written in the form of clauses:

```
H :- B1, ..., Bn.
```

and are read declaratively as logical implications:

H if B1 and ... and Bn.

```
fallible(X) :- human(X).
```

As a clause in a logic program, it can be used both as a procedure to test whether X is fallible by testing whether X is human, and as a procedure to find an X which is fallible by finding an X which is human.

The clause

```
human(socrates).
```

can be used both as a procedure to show that `socrates` is `human`, and as a procedure to find an X that is human by "assigning" `socrates` to X.

Logic Programming

Example: Prolog

> *A logic program is a set of axioms, or rules, defining relationships between objects.*
>
> *A computation of a logic program is a deduction of consequences of the program.*
>
> *A program defines a set of consequences, which is its meaning.*

Sterling and Shapiro: The Art of Prolog, Page 1.

To deal with the other constraints one has to add other constraint solvers to the language. This led to Constraint Logic Programming

Programming Languages

- imperative programming
- declarative programming:
  - functional programming
  - logical programming (interpreting computation as logical deduction)

Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow.

Declarative programming for solving constrained combinatorial (optimization) problems means expressing the properties of solutions that must be found by "the solver".

Declarative Programming

activity of programming (according to R. Kowalski):

$$\text{Algorithm} = \text{Logic} + \text{Control}.$$

Logic: what must be done is defined
Control: how a solution must be found

- Imperative programming: the programmer must provide both logic and control
- Logic programming: the programmer is only required to provide a logical specification. Everything related to control is relegated to the abstract machine: deduction rule (resolution). Logic deduction idea that can be traced back to K. Gödel and J. Herbrand in the 1930s.

the resulting programs are really surprising in their brevity, simplicity and clarity

PROgramming in LOGic

It is desired to arrange three 1s, three 2s, . . . , three 9s in a list (which therefore will consist of 27 numbers) such that, for each $i \in [1, 9]$, there are exactly i numbers between two successive occurrences (in the list) numbered i.

Example:

1,2,1,8,2,4,6,2 OK

1,2,1,8,2,4,2,6 NO (only a single number between the last two occurrences of 2).

```
sol(Ls) :- 1
  list_of_27(Ls),
  sublist([9,_,_,_,_,_,_,_,_,9,_,_,_,_,_,_,_,_,9], Ls),
  sublist([8,_,_,_,_,_,_,_,8,_,_,_,_,_,_,_,8], Ls),
  sublist([7,_,_,_,_,_,_,7,_,_,_,_,_,_,7], Ls),
  sublist([6,_,_,_,_,_,6,_,_,_,_,_,6], Ls),
  sublist([5,_,_,_,_,5,_,_,_,_,5], Ls),
  sublist([4,_,_,_,4,_,_,_,4], Ls),
  sublist([3,_,_,3,_,_,3], Ls),
  sublist([2,_,2,_,2], Ls),
  sublist([1,1], Ls).

list_of_27(Ls):-
  Ls = [_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_].
```

`:-` means definitons; `,` means logical and

Prolog Approach

- Prolog II till Prolog IV [Colmerauer, 1990]
- CHIP V5 [Dincbas, 1988] http://www.cosytec.com (commercial)
- CLP [Van Hentenryck, 1989]
- Ciao Prolog (Free, GPL)
- GNU Prolog (Free, GPL)
- SICStus Prolog
- ECLiPSe [Wallace, Novello, Schimpf, 1997] http://eclipse-clp.org/ (Open Source)
- Mozart programming system based on Oz language (incorporates concurrent constraint programming) http://www.mozart-oz.org/ [Smolka, 1995]

Other Approaches

Libraries:
Constraints are modeled as objects and are manipulated by means of special methods provided by the given class.

- CHOCO (free) http://choco.sourceforge.net/
- Kaolog (commercial) http://www.koalog.com/php/index.php
- ILOG CP Optimizer `www.cpoptimizer.ilog.com` (ILOG, commercial)
- Gecode (free) `www.gecode.org`
  C++, Programming interfaces Java and MiniZinc
- G12 Project
  `http://www.nicta.com.au/research/projects/constraint_programming_platform`
- Google OR-Tools

Other Approaches

Modelling languages:

- OPL [Van Hentenryck, 1999] ILOG CP Optimizer `www.cpoptimizer.ilog.com` (ILOG, commercial)
- MiniZinc [Monash University with support from OPTIMA] (open source, works for various systems, ECLiPSe, Geocode)
- Comet
- AMPL

- Catalogue of Constraint Programming Tools:
  http://openjvm.jvmhost.net/CPSolvers/

CP Languages

Greater expressive power than mathematical programming

- constraints involving disjunction can be represented directly
- constraints can be encapsulated (as predicates) and used in the definition of further constrains

However, CP models can often be translated into MIP model by

- eliminating disjunctions in favor of auxiliary Boolean variables
- unfolding predicates into their definitions

CP Languages

- Fundamental difference to LP
  - language has structure (global constraints)
  - different solvers support different constraints
- In its infancy
- Key questions:
  - what level of abstraction?
    - solving approach independent: LP, CP, ...?
    - how to map to different systems?
  - Modeling is very difficult for CP
    - requires lots of knowledge and tinkering

Constraint Programming

- Declarative modeling methodology:
  - Describe properties of a feasible solution.
  - Convey the structure of the problem as explicitly as possible.
  - Express substructures of the problem.
  - Give solvers as much information as possible.
- Computational paradigm:
  - Use constraints to reduce the domain of each variable.
  - Remove from domains some / the values that cannot appear in any solution.

Summary

- Model your problem via Constraint Satisfaction Problem
- Declare Constraints + Program Search
- Constraint Propagation
- Languages

Resume

- Modelling in MILP and CP
  - First example: graph labelling with consecutive numbers
  - Second example: Cryptarithmetic (or verbal arithmetic or cryptarithm): Send More Money
- Overview on constraint programming:
  representation (modeling language) + reasoning (search + propagation)
  - search = backtracking + branching
  - propagate, filtering, pruning
  - level of consistency (arc/generalized + value/bound/domain)