Problem Modeling: An Illustration

Eric Monfroy

September 9, 2019

Overview

- What is modeling?
- Converting problems modeled with CSP set constraints into SAT instances

"History" of modeling in constraint programming

- "Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it." Eugene Freuder.
- Constraint programming paradigme: One has to focus on WHAT not on HOW.

$\Rightarrow$ The user models her/his problem, the "solver" solves it.

- solver: mechanisms/algorithms for computing solutions of a problem
- solution: a set of values for variables that satisfies the problem, optimizes a criterion, ...

A tentative definition

- No "official" nor "formal" definition (to my knowledge)
- modeling: translating some verbal problem statements into a formal (possibly mathematical) language (possibly understandable by a solver)
    - possibly mathematical: arithmetic equations, matrices, . . .
    - possibly understandable by a solver: that can be the input of a solver

How to model

- notion of variables
- notion of types of variables
- notion of relations (constraints) between variables

A simple example

The n-queen problem: place n chess queens on an $n \times n$ chessboard so that no two queens threaten each other.

- a verbal problem statement
- a description of the problem
- some implicit knowledge:
    - chessboard?
    - a queen can move?
    - possible moves of a queen?
    - threaten or "attack"?
    - ...

n-queen: a model

- Variables: $x_{i,j}, \quad i, j \in [1..n]$
  - 1, if there is a queen in row i, column j
  - 0, otherwise
- Domains (possible values): $\{0, 1\}$
- Constraints (requirements):
  - one and only one queen per row:
    $\forall i \in [1..n], \; \sum_{j=1}^{n} x_{i,j} = 1$
  - one and only one queen per column:
    $\forall j \in [1..n], \; \sum_{i=1}^{n} x_{i,j} = 1$
  - 0 or 1 queen per diagonal (4.n -6 diagonals)
    $x_{2,1} + x_{1,2} \leqslant 1$
    ...
- Note: "$= 1$" can be replaced by "$\leqslant 1$" adding
  $\sum_{i=1}^{n}(\sum_{j=1}^{n} x_{i,j}) = n$

n-queen: a second model

- Variables: $x_i, \quad i \in [1..n]$
  Rows of the chessboard
- Domains (possible values): $\{1, \ldots, n\}$
  column index of each queen placed
- Constraints (requirements):
  - one and only one queen per column:
    $\forall i, j \in [1..n], i \neq j, \quad x_i \neq x_j$
  - one and only one queen per row:
    already in the formulation of variables and domains
  - not 2 queens on a diagonal
    $\forall i, j \in [1..n], i \neq j, \quad x_i - x_j \neq i - j$ and $x_i - x_j \neq j - i$

4-queen: a third model

- Variables: $x_{i,j}, \quad i,j \in [1..4]$
  - true, if there is a queen in row i, column j
  - false, otherwise
- Domains (possible values): $\{true, false\}$
- Constraints (requirements):
  - one and only one queen per row (first row only):
    $x_{1,1} \vee x_{1,2} \vee x_{1,3} \vee x_{1,4}$
    $\wedge(\neg x_{1,1} \vee \neg x_{1,2}) \wedge (\neg x_{1,1} \vee \neg x_{1,3}) \wedge (\neg x_{1,1} \vee \neg x_{1,4})$
    $\wedge(\neg x_{1,2} \vee \neg x_{1,3}) \wedge (\neg x_{1,2} \vee \neg x_{1,4}) \wedge (\neg x_{1,3} \vee \neg x_{1,4})$
  - one and only one queen per column:
    similar, inverting i and j
  - diagonals (the main diagonal only)
    $(\neg x_{1,1} \vee \neg x_{2,2}) \wedge (\neg x_{1,1} \vee \neg x_{3,3}) \wedge (\neg x_{1,1} \vee \neg x_{4,4})$
    $\wedge(\neg x_{2,2} \vee \neg x_{3,3}) \wedge (\neg x_{2,2} \vee \neg x_{4,4}) \wedge (\neg x_{3,3} \vee \neg x_{4,4})$

n-queen: a 4th model

- Variables: $x_i, \quad i \in [1..n]$
  Rows of the chessboard
- Domains (possible values): $\{1, \ldots, n\}$
  column index of each queen placed
- Constraints (requirements):
  - one and only one queen per column:
    $alldifferent(\{x_1, \ldots, x_n\})$
  - one and only one queen per row:
    already in the formulation of variables and domains
  - not 2 queens on a diagonal
    $alldifferent(\{x_i + i | i \in [1..n]\})$
    $alldifferent(\{x_i - i | i \in [1..n]\})$

n-queen: a 5th model

- Variables: $R_i, C_i, \quad i \in [1..n]$
  Rows, Columns, and Cells of the chessboard
- Domains (possible values): $\{q_1, \ldots, q_n\}$
- Constraints (requirements):
  - one and only one queen per row:
    $\forall i \in [1..n], \quad R_i = \{q_i\}$
  - one and only one queen per column:
    $\forall i \in [1..n], \quad |C_i| = 1$ and $|\bigcup_{i=1}^{n} C_i| = n$
  - not 2 queens on a diagonal
    $|(R_1 \cap C_2) \cup (R_2 \cap C_1)| \leqslant 1$
    ...

n-queen: a 6th model

no, I'm tired,
and I feel you too!

Another simple example

Cryptarithmetic: a mathematical equation of letters, each one representing a number.

$$\begin{array}{r} \text{SEND} \\ +\text{MORE} \\ \hline =\text{MONEY} \end{array}$$

send+more a first model

- Variables: $S, E, N, D, M, O, E, N, Y$
  Letters of the equation
- Domains (possible values): $\{0, \ldots, 9\}$
  the numbers
- Constraints (requirements):
  - first digits are not 0:
    $S \neq 0,\ M \neq 0$
  - the operation:
    $S*1000+E*100+N*10+D+M*1000+O*100+R*10+E$
    $=$
    $M*10000+O*1000+N*100+E*10+Y$

send+more a first model

- Variables: $S, E, N, D, M, O, E, N, Y, C_1, C_2, C_3, C_4$
  Letters of the equation
- Domains (possible values):
  $S, E, N, D, M, O, E, N, Y \in \{0, \ldots, 9\}$ and
  $C_1, C_2, C_3, C_4 \in \{0, \ldots, 1\}$
  the numbers
- Constraints (requirements):
  - first digits are not 0:
    $S \neq 0,\ M \neq 0$
  - the operation, column after column, with carries:

$$
\begin{aligned}
&D + E = 10 * C_1 + Y \\
&N + R + C_1 = 10 * C_2 + E \\
&E + O + C_2 = 10 * C_3 + N \\
&S + M + C_3 = 10 * C_4 + O \\
&C_4 = M
\end{aligned}
$$

Models

- as many models as wanted
- based or not based on the same type of variables
- based or not based on the same type of constraints
- made or not for a specific solver (model 1 for LP, model 2 for FD CP, model 3 for SAT solver, model 4 for minizinc, model 5 for set constraints, . . .)

Motivations

THERE IS NOT A UNIQUE MODEL

- the user is generally more comfortable with a type of modeling
- the user does not have a solver of each type
- for a given problem, the models are not solved as efficiently
- some problems are easier modeled with a union of sub-problems, each one with a different type of modeling
- some models are more readable for some users or for some problems

Thus

- Modeling is **crucial**
- It is important to let the user decide the type of modeling
- BUT: efficiency (speed, quality of solutions, ...) is also important

Ideas

- Model transformation
  - improving a model
  - using more efficiently solved constraints
  - ...
- Model conversion
  - e.g., from CSP into SAT, from CP into ILP:
  - to adapt a model to a given solver
  - to let the user models as wanted
  - BUT the target depends on the problem and the solvers at disposal
- Robust models
  - change a model into an equivalent but more robust one (less sensitive to changes, errors, ...)
- Generic models
- ...

Examples of Modeling Realizations

Modelers:

- ECLiPSe
- Minizinc
- Essence
- MCSP
- (CP systems)
- ...

Modeling formats (languages):

- AMPL
- XCSP3
- FlatZinc
- ...

Models vs. Instances

Not easy to define:

- model + data = instance
- a model may represent a set of problems (ex: n-queens)
- an instance represents a problem (n-queen with n=4: 4-queen)
- an instance is "flat" while a model may contain "iterations"