Knowledge Representation and Reasoning
Introduction to Constraint Programming

Antoni Ligęza

ligeza@agh.edu.pl

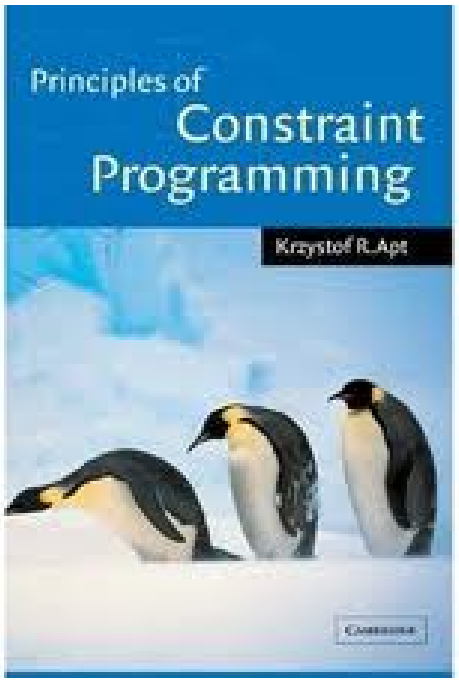
AGH University of Science and Technology
Kraków, Poland

Knowledge Representation and Reasoning
AGH Kraków
2017

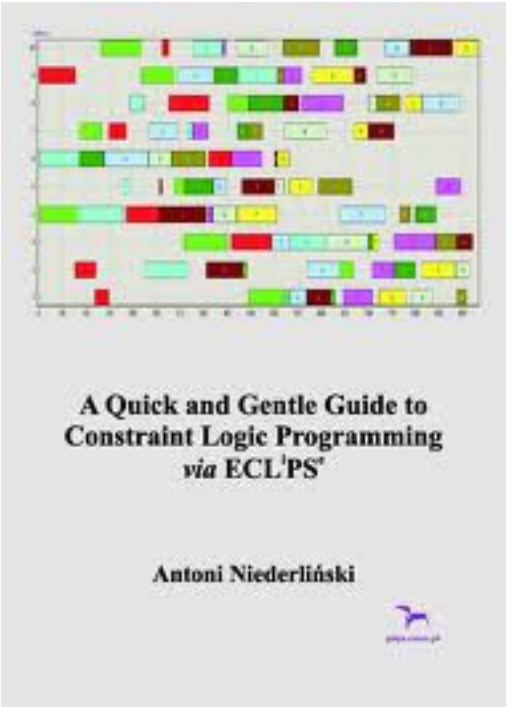
Outline

Presentation Outline

Some Books

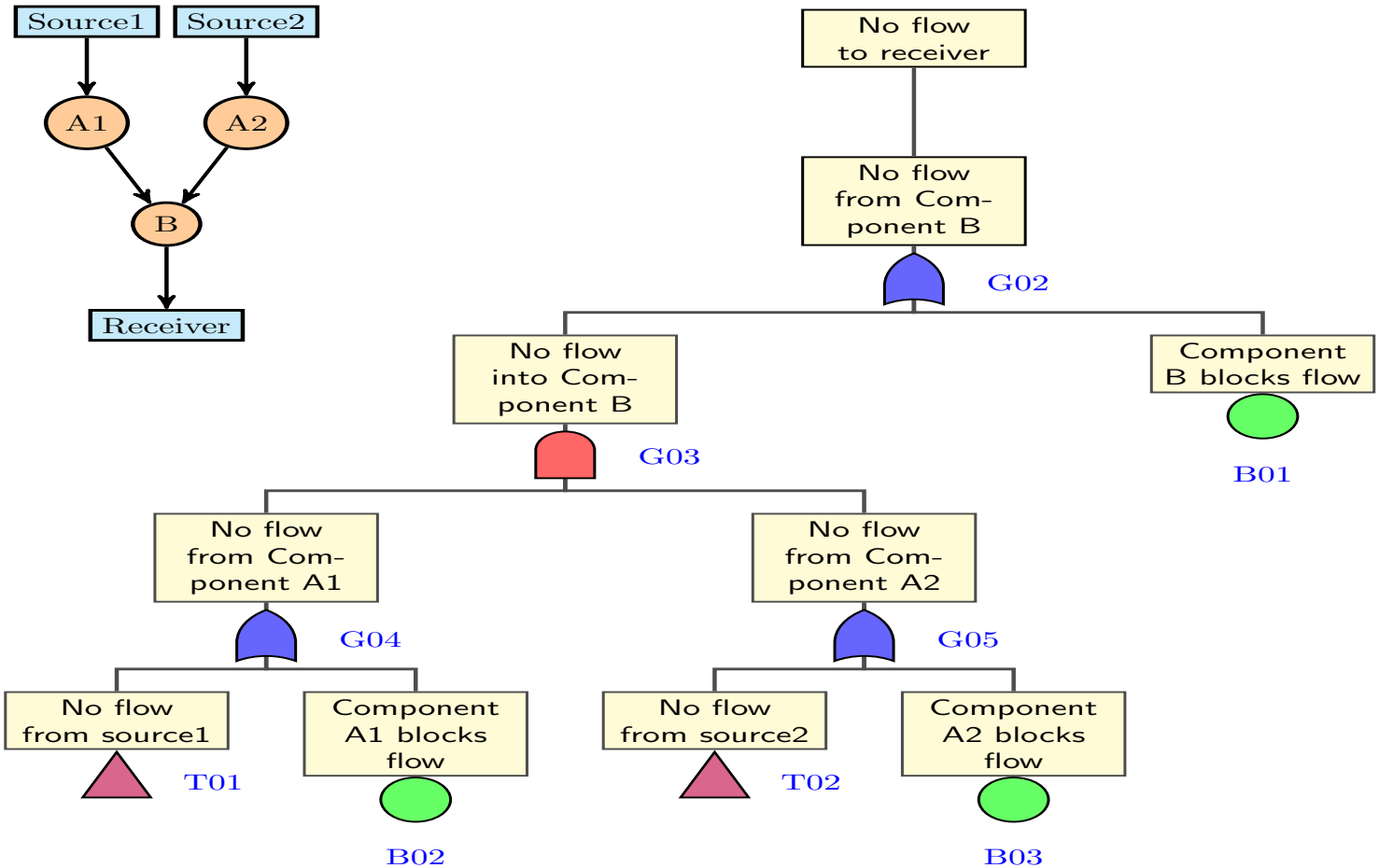

Principles of
Constraint
Programming
Krzysztof R. Apt


Handbook of Constraint Programming


A Quick and Gentle Guide to
Constraint Logic Programming
via ECLiPSe
Antoni Niederliński

Presentation Outline

An Example: System + Logic Model for Diagnostic Reasoning

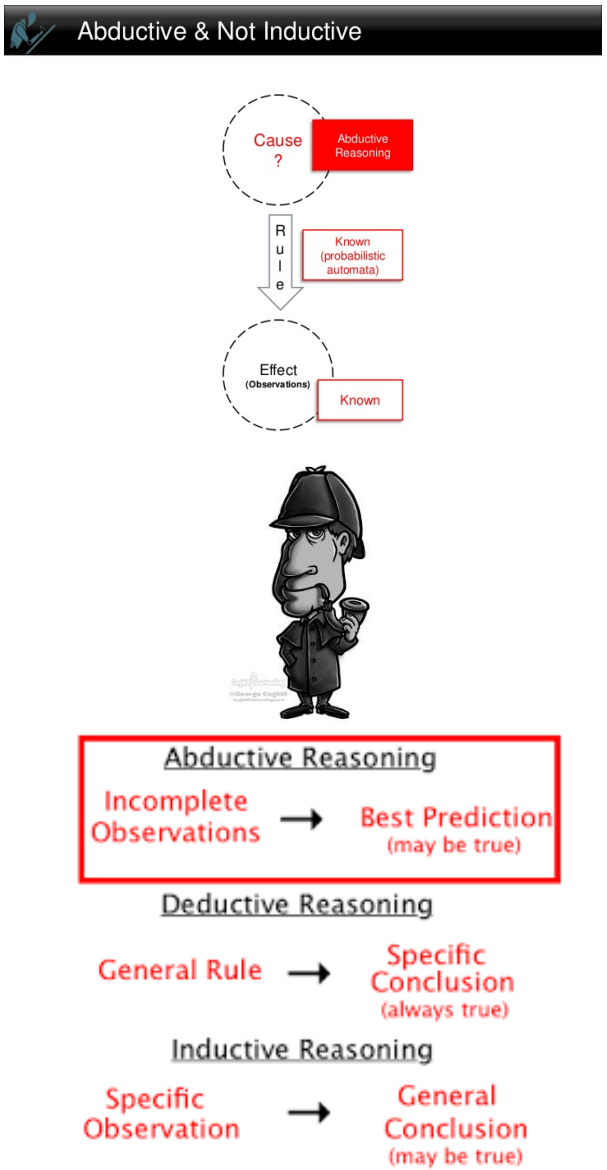


Presentation Outline

Abduction and Constraints: Introduction

Some loosely provocative questions and statements...

- abduction: what, why and where — what for?
- abduction: a method of logical inference (but invalid!),
- abduction vs. deduction,
- abduction: primary method used by Sherlock Holmes!
- abduction: inevitable ambiguity (potential/admissible solutions; many of them),
- abduction: more constraints — better abduction,
- abduction + constraints + SAT (minimal models).



Abduction and Constraints: Introduction

Abduction

- Abduction — principal way of problem solving — generation of hypotheses,
- Abduction — performed with backtracking search,
- Abduction — produces numerous, admissible solutions

Abduction: Logical model

$$\frac{\alpha \Longrightarrow \beta, \beta}{\alpha}$$

$$HYP^{+} \cup HYP^{-} \cup KB \models OBS^{+} \cup OBS^{-}$$

$$HYP^{+} \cup HYP^{-} \cup KB \cup OBS^{+} \cup OBS^{-} \not\models \bot$$

An intuitive example: find explanations for *wet_street*

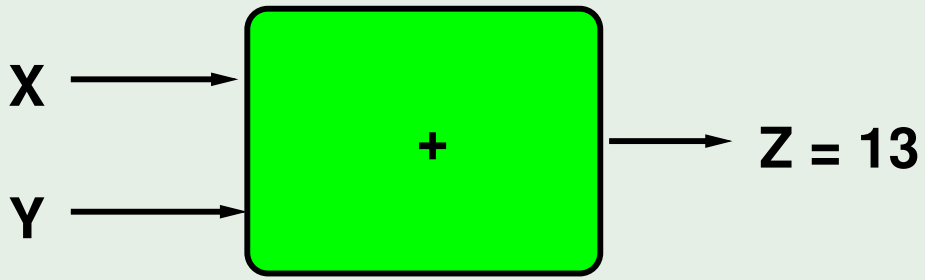
- *rain* $\longrightarrow$ *water*
- *sprinkler* $\longrightarrow$ *water*
- *snow* $\wedge$ *temperature* $\longrightarrow$ *water*
- *water* $\longrightarrow$ *wet_street*,
- *cleaning* $\longrightarrow$ *wet_street*
- *oil* $\longrightarrow$ *wet_street*

Presentation Outline

The role of constraints in abduction

Abductive problem without constraints

- X, Y, Z - variables, $X, Y \in \{0, 1, 2, \ldots, 9\}$, $Z \in \{0, 1, 2, \ldots, 18\}$
- system: $Z = X + Y$



- Observed: $Z = 13$
- Possible explanations:
  - $(X = 4$ and $Y = 9)$,
  - $(X = 5$ and $Y = 8)$,
  - ... ,
  - $(X = 9$ and $Y = 4)$.
- 6 admissible solutions.

The role of constraints in abduction

Abductive problem with constraints

- X, Y, Z - variables, $X, Y, Z \in \{0, 1, 2, \ldots, 9\}$,

$$Z = X + Y$$

- **Constraint:**

$$Y < X - 3$$

- Observed: $Z = 13$
- Possible explanations: $(X = 9$ and $Y = 4)$,
- 1 admissible solution.

Conclusion

- CONSTRAINTS can refine results of ABDUCTION; less models generated,
- propagation of CONSTRAINTS can reduce computational effort,
- ABDUCTION + CONSTRAINTS = CONSTRUCTIVE ABDUCTION

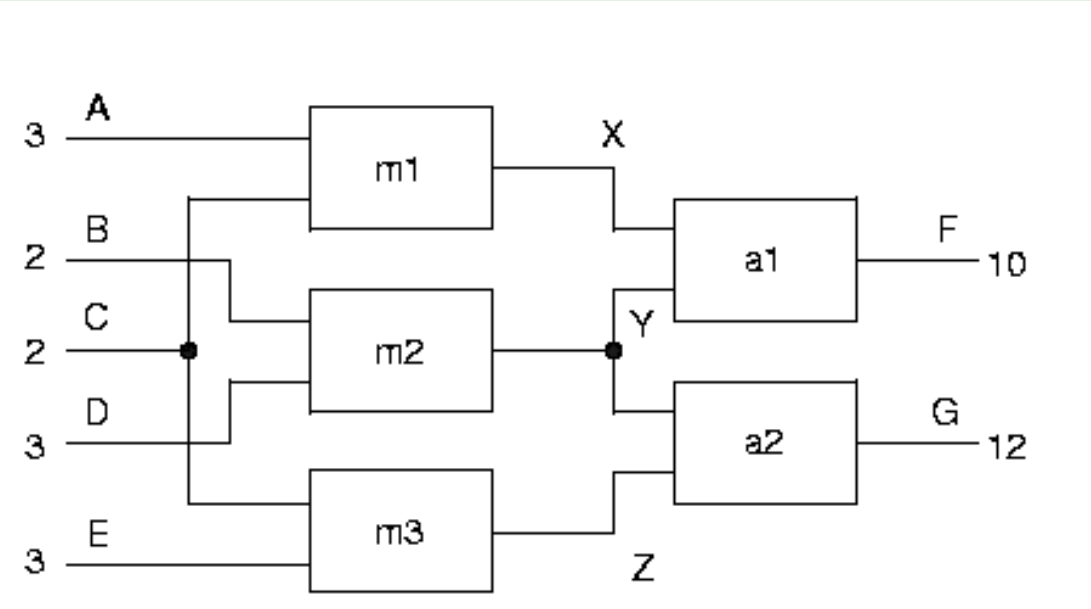
Presentation Outline

The example problem I

The Paradigm to be Explored Further on

ABDUCTION $\simeq$ DIAGNOSIS

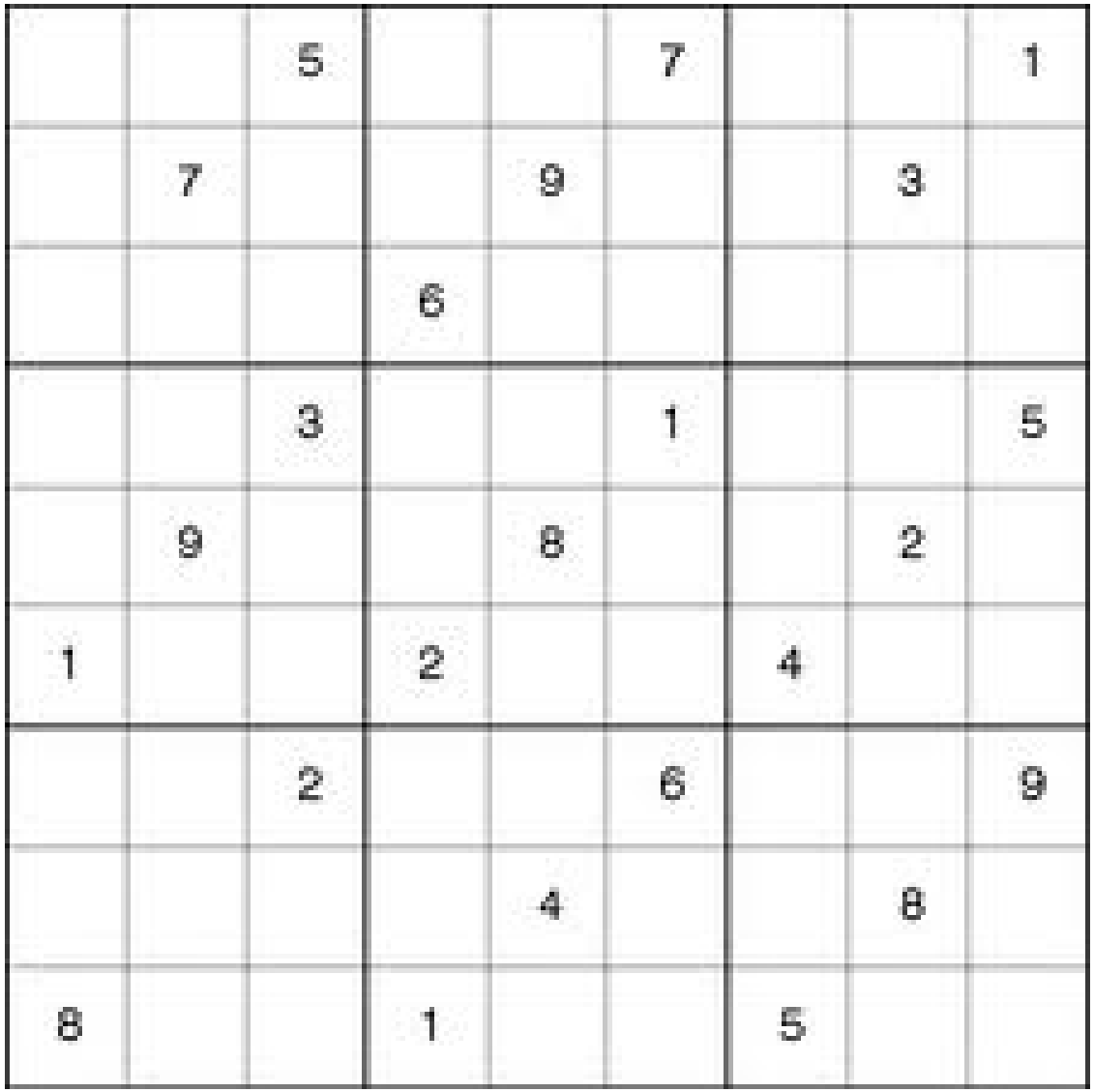
The Multiplier-Adder System




```
[...]
A #= 3,
B #= 2,
C #= 2,
D #= 3,
E #= 3,
F #= 10,
G #=12,
[...]
```

Presentation Outline

Constraint Satisfaction Problems



		5			7			1
	7			9			3	
			6					
		3			1			5
	9			8			2	
1			2			4		
		2			6			9
				4			8	
8			1			5		

Figure: Sudoku: An example Constraint Satisfaction Problem

Constraint Satisfaction Problems

Figure: An example Constraint Satisfaction Problem

Constraint Satisfaction Problems

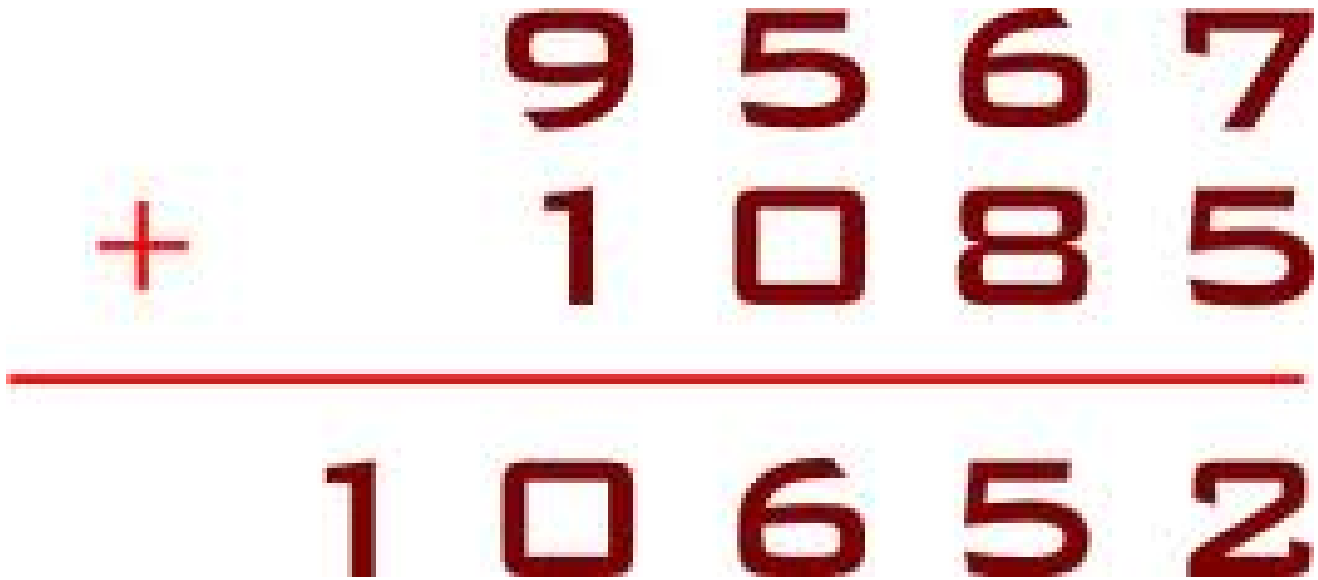


Figure: The unique solution of the example Constraint Satisfaction Problem

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Constraint Satisfaction Problems

- Constraint (Logic) Programming:
  - simple, transparent statement; practical problem,
  - zero, one, or many solutions,
  - problem: combinatorial explosion.
- Decision factors in CP/CLP:
  - variable — which variable too choose,
  - value — which value to choose,
  - propagation — how to propagate constraints,
  - heuristics — what heuristics can be used.
- CP/CLP basic solution paradigm:
  - select a variable,
  - select a value,
  - propagate constraints,
  - if conflict — backtrack; if unique — return solution; else — loop.

Presentation Outline

Constraint Satisfaction Problem

CSP statement

- $X = \{X_1, X_2, \ldots, X_k\}$ — a set of variables,
- $D = \{D_1, D_2, \ldots, D_k\}$ — their domains,
- $C = \{(S_i, R_i) : i = 1, 2, \ldots, n\}$ — constraints,
  - S_i — scope — a selection of variables,
  - R_i — relation defined over Cartesian Product of domains appropriate for the scope variables,

CSP solution

A solution to CSP given by (X, D, C) is any assignment of values to variables of X of the form

$$\{X_1 = d_1, X_2 = d_2, \ldots, X_k = d_k\},$$

such that $d_i \in D_i$, and for any constraint in $(S_i, R_i) \in C$, R_i is satisfied by the appropriate projection of the solution vector $(d_1, d_2, \ldots, d_k)$ over variables of S_i.

CP/CSP — Characteristics

CP vs. OPT

- CSP: first solution counts,
- OPT:best solution counts.

Binary vs. finite domains; SAT

- SAT: binary domains (0 or 1; true or false),
- CSP: finite discrete domains.

CSP: Problems

- large number of variables,
- large domains,
- numerous constraints,
- different types of constraints,
- unpredictable, irregular, hard to trace.

Presentation Outline

Example: SEND+MORE=MONEY

Figure: An example Constraint Satisfaction Problem

- 8 variables: `S, E, N, D, M, O, R, Y`,
- 10-values in each domain,
- alldifferent(S, E, N, D, M, O, R, Y),
- basic search-space size: 10^8,
- reduced search-space size:
  $9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 = 362880$

A simple CLP code I

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

solve(Vars):- Vars=[S,E,N,D,M,O,R,Y],
              sendmoremoney([S,E,N,D,M,O,R,Y]),
              label(Vars).
```

Pretty good results! I

```
?- time(sendmoremoney(V)).
% 6,758 inferences, 0.00 CPU in 0.00 seconds (0% CPU, Infinite Lips
V = [9, _G11470, _G11473, _G11476, 1, 0, _G11485, _G11488],
_G11470 in 4..7,
all_different([_G11470, _G11473, _G11476, _G11485, _G11488, 0, 1, 9
1000*9+91*_G11470+ -90*_G11473+_G11476+ -9000*1+ -900*0+10*_G11485+
_G11473 in 5..8,
_G11476 in 2..8,
_G11485 in 2..8,
_G11488 in 2..8.

?- time(solve(V)).
% 10,088 inferences, 0.01 CPU in 0.00 seconds (308% CPU, 1008800 Li
V = [9, 5, 6, 7, 1, 0, 8, 2].
```

MiniZinc Coding I

```
include "alldifferent.mzn";

var 1..9: S;
var 0..9: E;
var 0..9: N;
var 0..9: D;
var 1..9: M;
var 0..9: O;
var 0..9: R;
var 0..9: Y;

constraint           1000 * S + 100 * E + 10 * N + D
                   + 1000 * M + 100 * O + 10 * R + E
       = 10000 * M + 1000 * O + 100 * N + 10 * E + Y;

constraint alldifferent([S,E,N,D,M,O,R,Y]);
```

MiniZinc Coding II

```
solve satisfy;

output ["   ",show(S),show(E),show(N),show(D),"\n",
        "+  ",show(M),show(O),show(R),show(E),"\n",
        "= ",show(M),show(O),show(N),show(E),show(Y),"\n"];
```