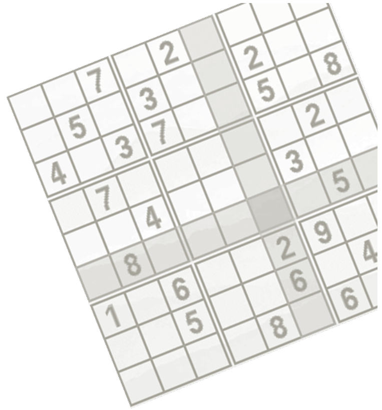


Constraint Programming

Roman Barták
Department of Theoretical Computer Science and Mathematical Logic

- **Logic-based puzzle,** whose goal is to enter digits 1-9 in cells of 9×9 table in such a way, that no digit appears twice or more in every row, column, and 3×3 sub-grid.

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

A bit of history

1979: first published in New York

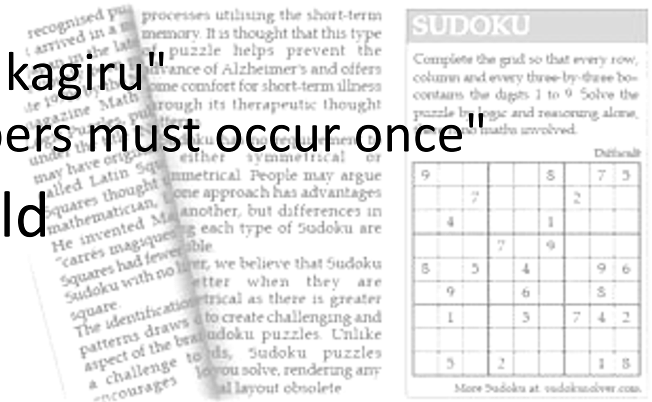
under the name „Number Place“

1986: became popular in Japan

Sudoku – from Japanes "**Su**dji wa **doku**shin ni kagiru"

"the numbers must be single" or "the numbers must occur once"

2005: became popular in the western world



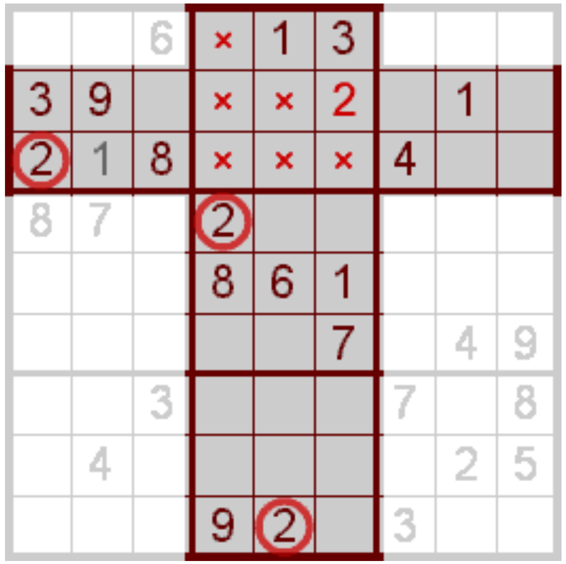
How to find out which digit to fill in?

- Use information that each digit appears exactly once in each row and column.

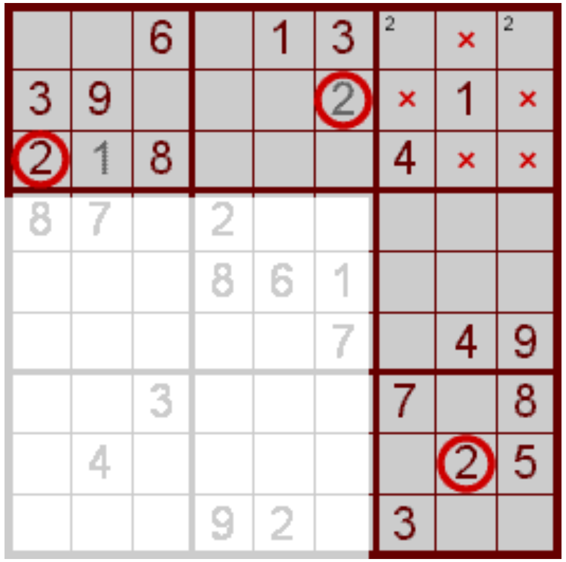
What if this is not enough?

- Look at columns

  or combine information from rows and columns

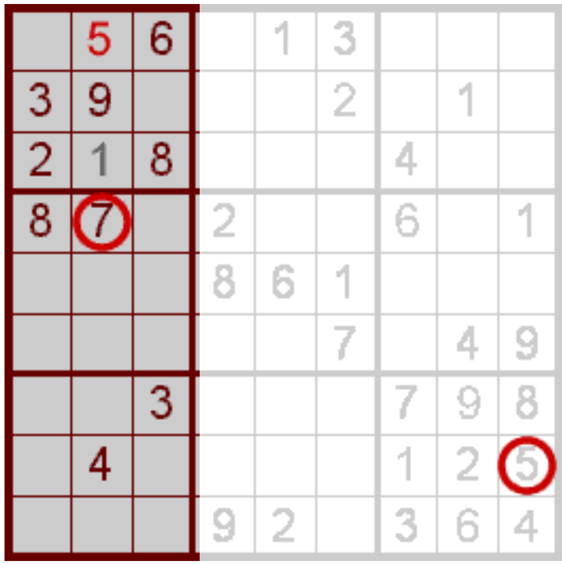


Sudoku – One More Step

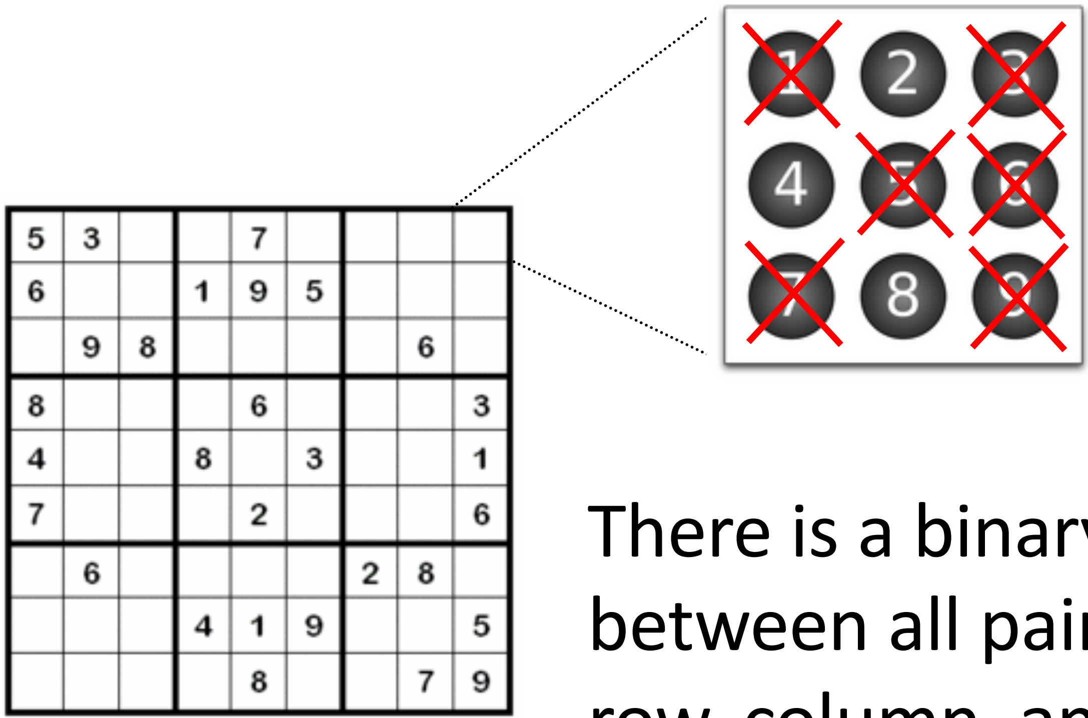


- If neither rows and columns provide enough information, we can note allowed digits in each cell.

- The position of a digit cand be infereed from positions of other digits and resrictions of Sudoku that each digit appears one in a column (row, sub-grid)



Sudoku in General



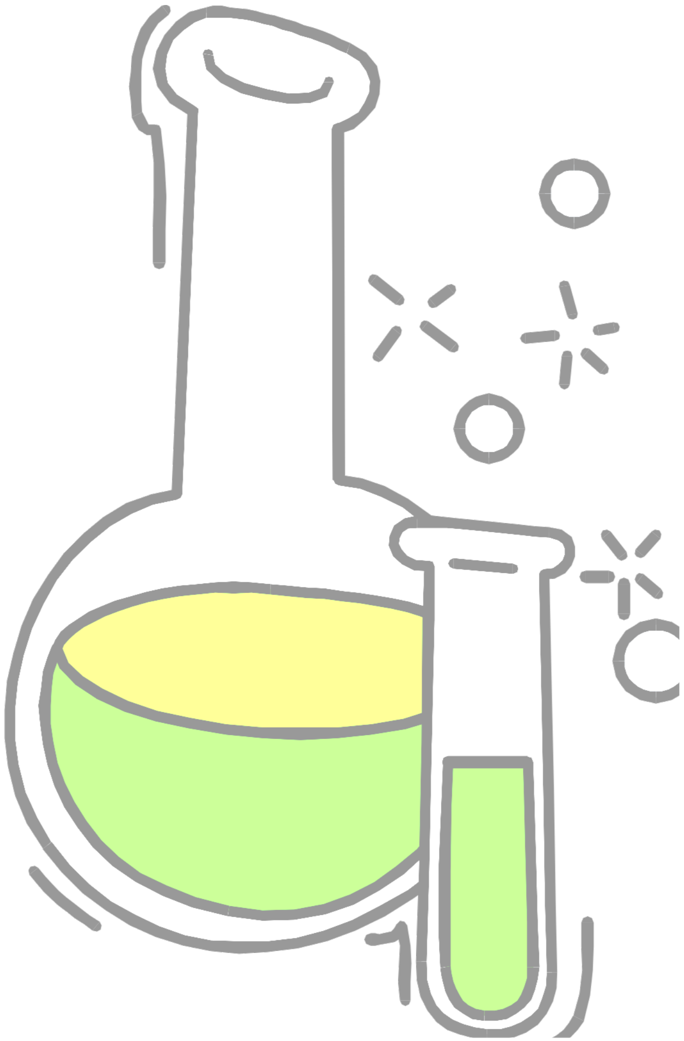
We can see every cell as a **variable** with possible values from **domain** {1,…,9}.

There is a binary inequality **constraint** between all pairs of variables in every row, column, and sub-grid.

Such formulation of the problem is called a **constraint satisfaction problem.**

Constraint Satisfaction Algorithms

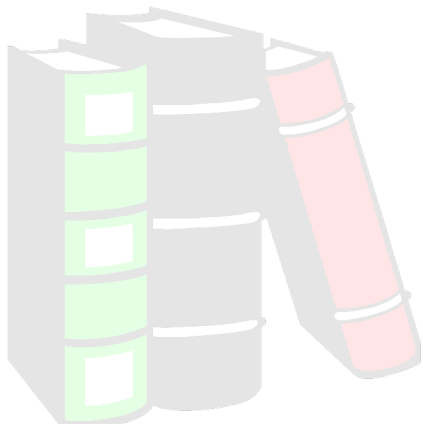
- Local search techniques
  - HC, MC, RW, Tabu, GSAT, Genet
- Search algorithms
  - GT, BT, BJ, BM, DB, LDS
- Consistency techniques
  - NC, AC, DAC, PC, DPC, RPC, SC
- Consistency techniques in search
  - FC, PLA, LA
- Constraint Optimisation
  - B&B
- Over-constrained problems
  - PCSP, ProbCSP, FuzzyCSP, VCSP, SCSP, constraint hierarchies

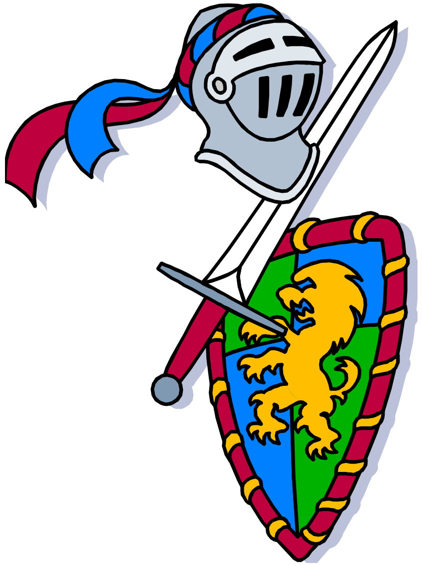


Constraint Modelling

- Tips and tricks, Constraint Logic Programming

- **Books**
  - P. Van Hentenryck: **Constraint Satisfaction in Logic Programming**, MIT Press, 1989
  - E. Tsang: **Foundations of Constraint Satisfaction**, Academic Press, 1993
  - K. Marriott, P.J. Stuckey: **Programming with Constraints: An Introduction**, MIT Press, 1998
  - R. Dechter: **Constraint Processing**, Morgan Kaufmann, 2003
  - **Handbook of Constraint Programming**, Elsevier, 2006
- **Journals**
  - **Constraints**, An International Journal. Springer Verlag
  - **Constraint Programming Letters**, free electronic journal
- **On-line resources**
  - **Course Web** (transparencies)
    http://ktiml.mff.cuni.cz/~bartak/podminky/
  - **On-line Guide to Constraint Programming** (tutorial)
    http://ktiml.mff.cuni.cz/~bartak/constraints/
  - **Constraints Archive** (archive and links)
    http://4c.ucc.ie/web/archive/index.jsp
  - **Constraint Programming online** (community web)
    http://www.cp-online.org/

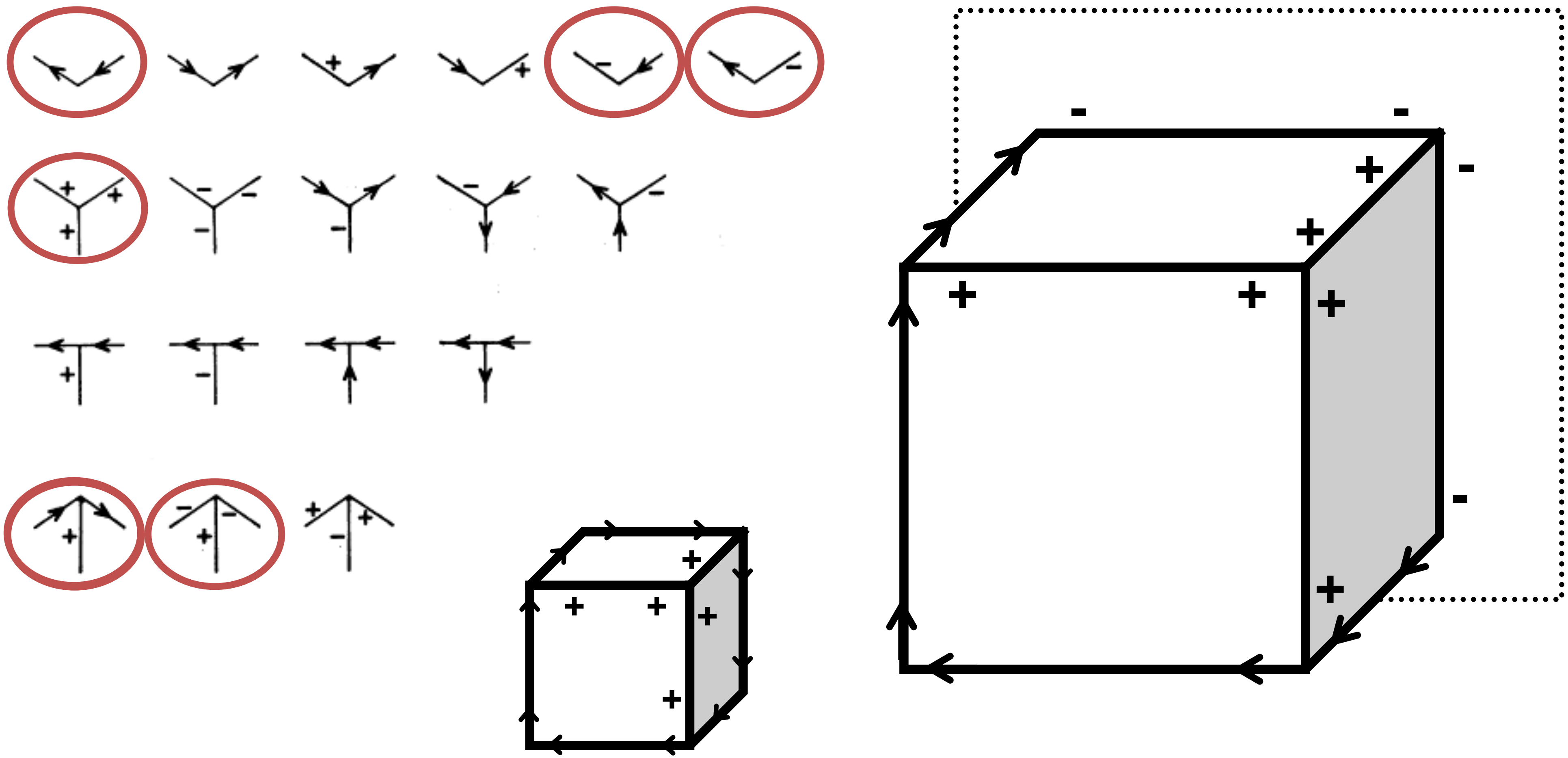




- **Artificial Intelligence**
  - Scene labelling (Waltz 1975)
  - How to help the search algorithm?
- **Interactive Graphics**
  - Sketchpad (Sutherland 1963)
  - ThingLab (Borning 1981)
- **Logic Programming**
  - unification $\rightarrow$ constraint solving (Gallaire 1985, Jaffar, Lassez 1987)
- **Operations Research and Discrete Mathematics**
  - NP-hard combinatorial problems

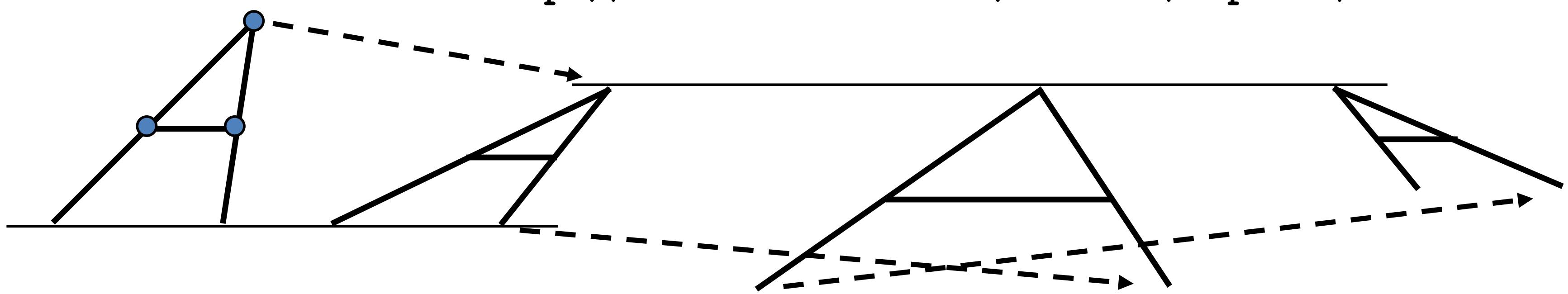
inferring 3D meaning of lines in a 2D drawing

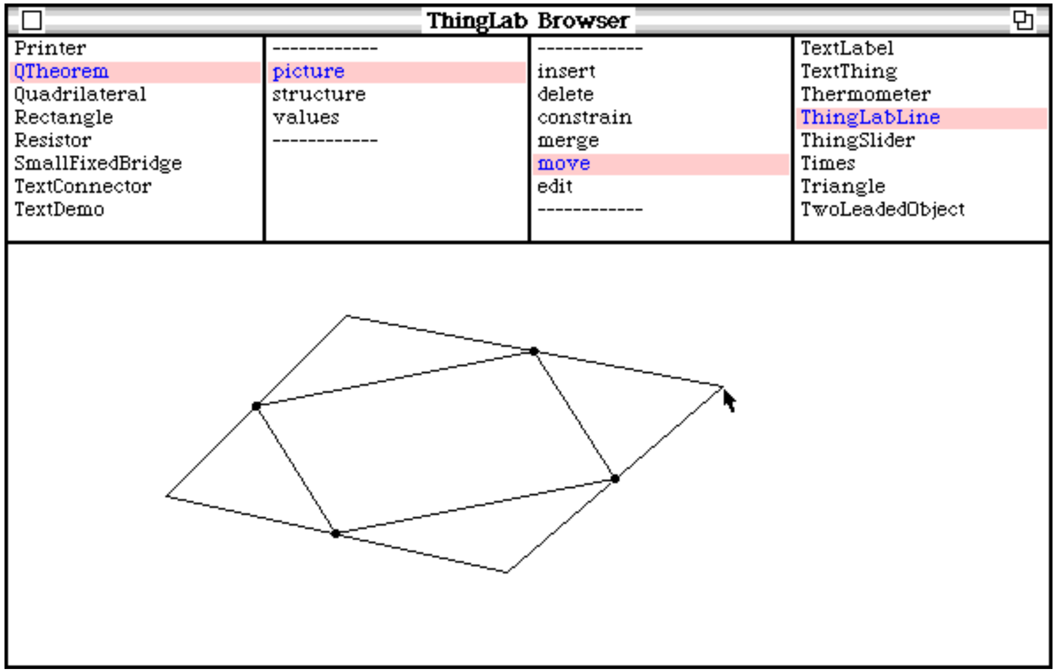
- convex (+), concave (-) and border (←) edges
- we are looking for a physically feasible interpretation

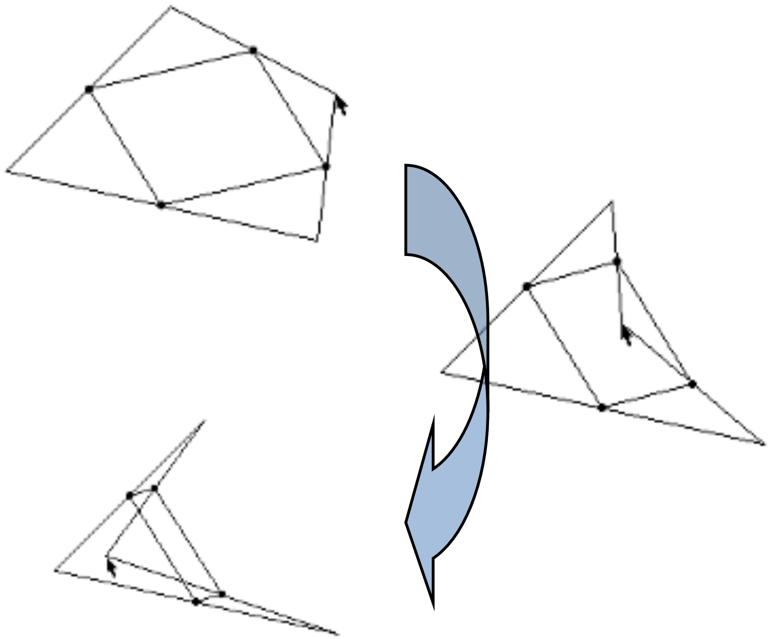


manipulating graphical objects described via constraints

http://ktiml.mff.cuni.cz/~bartak/diploma/downloads.html

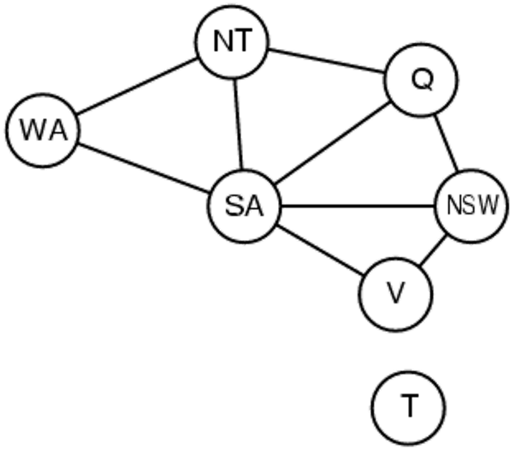






http://www.cs.washington.edu/research/constraints/

Assign colours (red, blue, green) to states, such that neighbours have different colours.



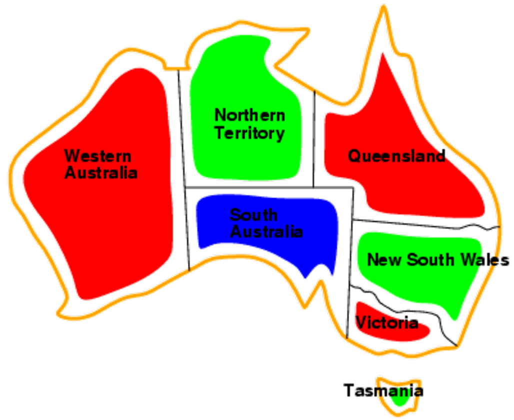
CSP Model

- variables: {WA, NT, Q, NSW, V, SA, T}
- domains: {r, b, g}
- constraints: WA ≠ NT, WA ≠ SA etc.

Can be described as a **constraint network** (nodes=variables, edges=constraints)

Solution

WA = r, NT = g, Q = r, NSW = g,
V = r, SA = b, T = g



Assign digits 0,…,9 to letters S,E,N,D,M,O,R,Y in such a way that:

- SEND + MORE = MONEY
- different letters are assigned to different digits
- S and M are different from 0

Model 1:

```
E,N,D,O,R,Y in 0..9, S,M in 1..9
            1000*S + 100*E + 10*N + D
+           1000*M + 100*O + 10*R + E
= 10000*M + 1000*O + 100*N + 10*E + Y
```

Model 2:

using „carry“ 0-1 variables

```
E,N,D,O,R,Y in 0..9, S,M in 1..9, P1,P2,P3 in 0..1
   D+E = 10*P1+Y
P1+N+R = 10*P2+E
P2+E+O = 10*P3+N
P3+S+M = 10*M +O
```

```
all_different(S,E,N,D,M,O,R,Y)
```

N Queens Problem

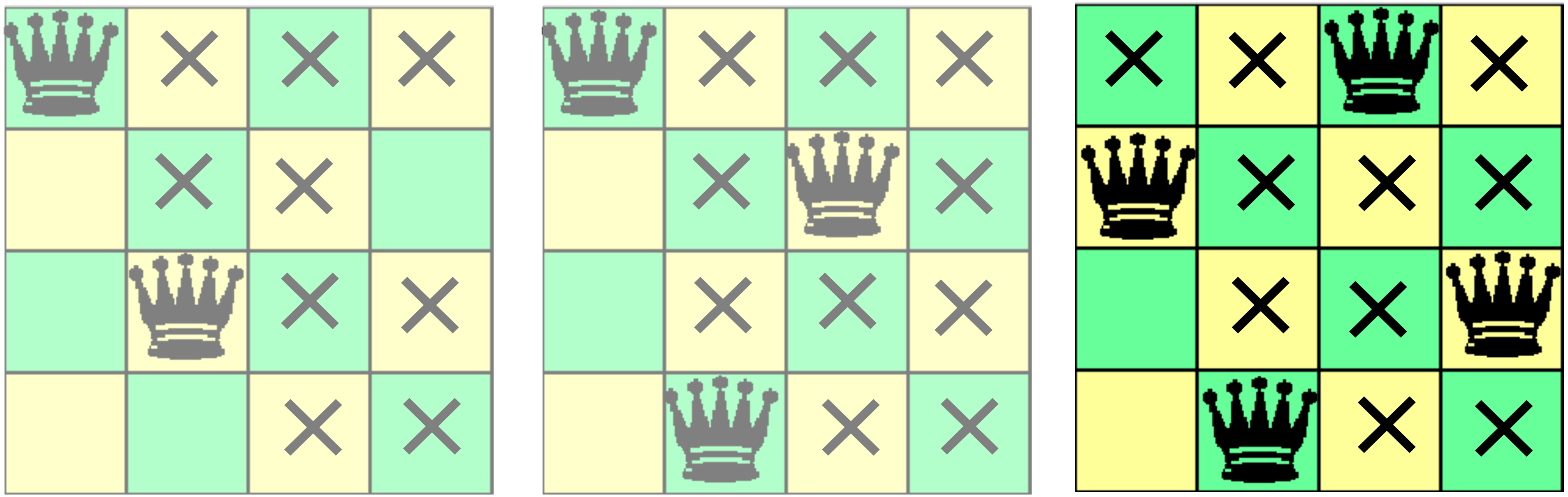
allocate N queens to a chess board of size N×N in a such way that no two queens attack each other

the core decision: each queen is located in its own column

variables: N variables r(i) with the domain {1,…,N}

constraints: no two queens attack each other

$$\forall i \neq j \quad r(i) \neq r(j) \wedge |i-j| \neq |r(i)-r(j)|$$

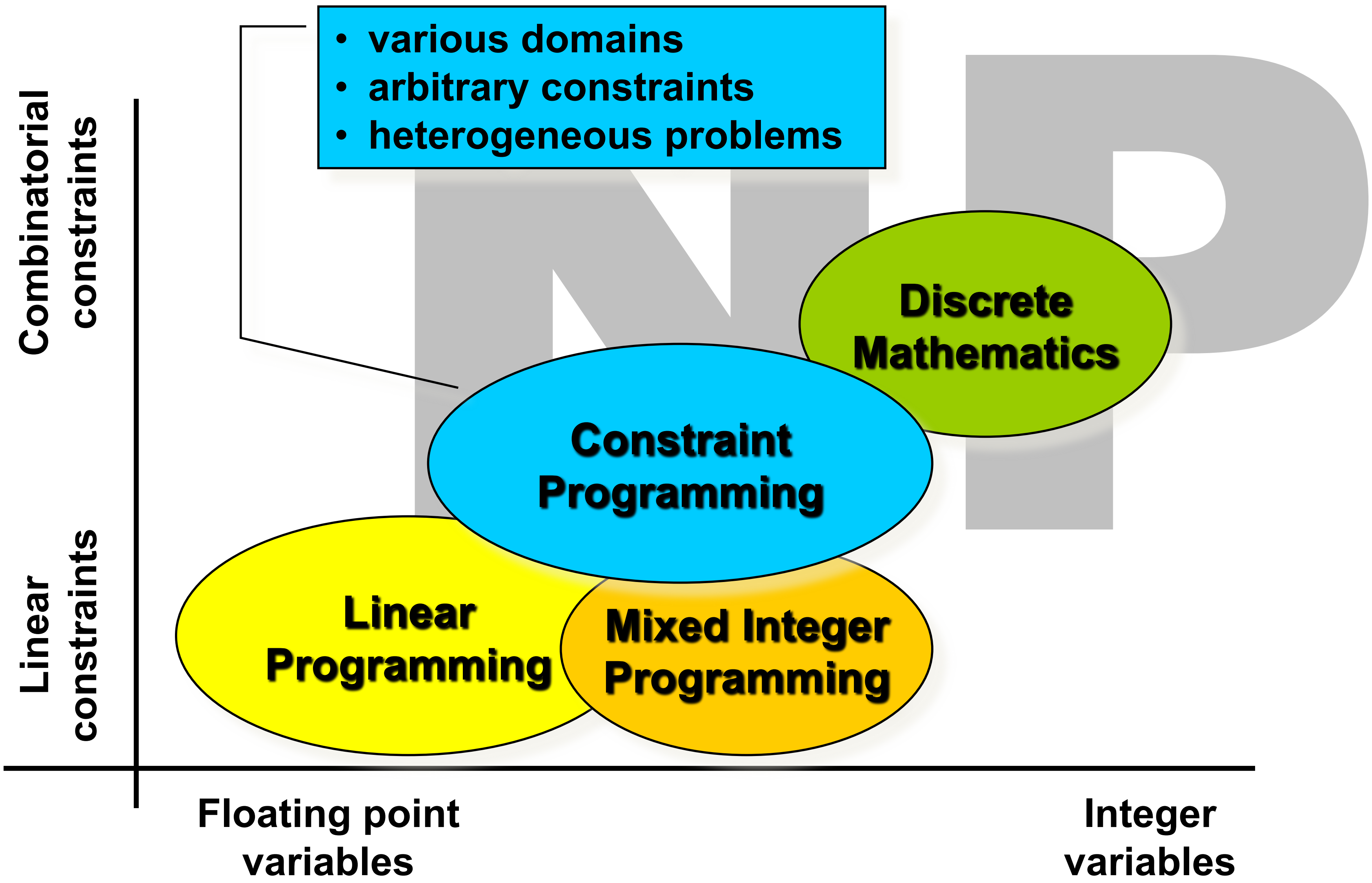


Bioinformatics

- DNA sequencing (Celera Genomics)
- deciding the 3D structure of proteins from the sequence of amino acids

Planning and Scheduling

- automated planning of spacecraft activities (Deep Space 1)
- manufacturing scheduling


• various domains
• arbitrary constraints
• heterogeneous problems
NP
Combinatorial constraints
Linear constraints
Discrete Mathematics
Constraint Programming
Linear Programming
Mixed Integer Programming
Floating point variables
Integer variables

Constraint Satisfaction Problem (CSP) consists of:

- a finite set of **variables**
  - describe attributes of the solution
    for example a location of a queen in the chess board
- **domains** – finite sets of possible values for variables
  - describe options that we need to decide
    for example, rows for queens
  - sometimes, there is a common super domain for all the variables and individual variables�much domains are defined via unary constraints
- a finite set of **constraints**
  - constraint is a relation over a subset of variables
    for example locationA $\neq$ locationB
  - constraint can be defined in extension (a set of compatible value tuples) or using a formula (see above)

A feasible solution of a constraint satisfaction problem is a complete consistent assignment of values to variables.

- **complete** = each variable has assigned a value
- **consistent** = all constraints are satisfied

Sometimes we may look for all the feasible solutions or for the number of feasible solutions.

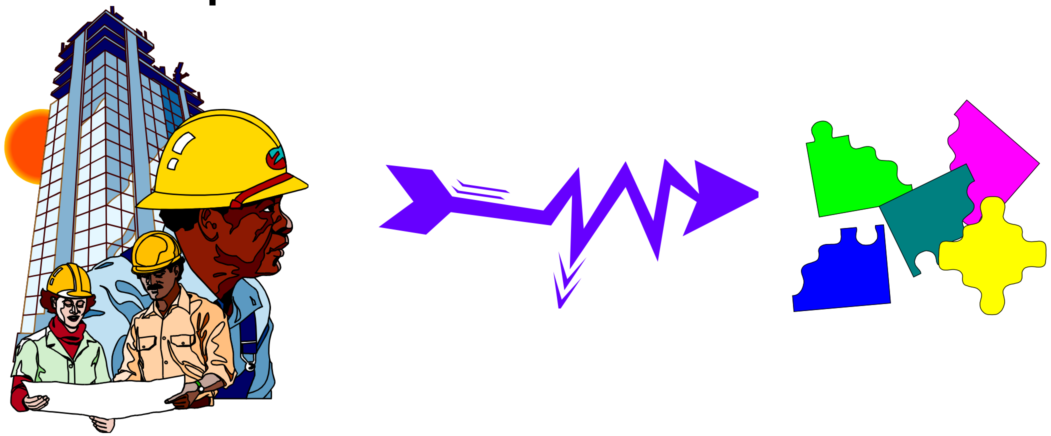
An optimal solution of a constraint satisfaction problem is a feasible solution that minimizes/maximizes a value of some objective function.

- **objective function** = a function mapping feasible solutions to real numbers

The Core Topics

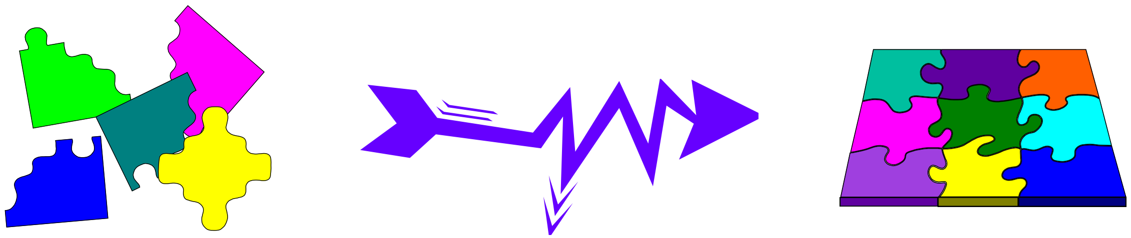
Problem Modelling

How to describe a problem as a constraint satisfaction problem?



Solving Techniques

How to find values for the variables satisfying all the constraints?



Properties of Constraints

- express **partial information**
  - X is greater than 3, but the exact value of X is not given
- provide a **local view** of the problem
  - connect only a few variables (not all of them)
- can be **heterogeneous**
  - domains can be different (numbers, strings etc.)
- are **non-directional** (functions)
  - $X = Y+2$ can be used to compute both X and Y
- are **declarative**
  - do not determine the procedure for satisfaction
- are **additive**
  - the order of constraints is not important, their conjunction is crucial
- are **rarely independent**
  - share variables

Advantages of CP

- **close to real-life problems**
  - we all use constraints when formulating problems
  - many real world features can be captured as constraints
- **declarative manner**
  - focus on problem description rather than on problem solving
- **co-operative problem solving**
  - a uniform framework for integration of various solving approaches
  - simple (search) and sophisticated (inference) techniques
- **semantic foundations**
  - clean and elegant modelling languages
  - roots in logic programming
- **applications**
  - not just academic exercise but already used to solve real-life problems

Limitations of CP

- **efficiency**
  - combinatorial explosion
  - many problems are in the NP-complete class
- **hard-to-predict behaviour**
  - the efficiency is not known until the model is tried on real data
- **model stability**
  - new data = new problem
- **too local**
  - through the individual constraints, the complete problem is not "visible" (can be solved via global constraints)
  - distributed computations
- **weak co-operation of solvers**
  - integrating various solving techniques is hard, usually done via shared variables only

Representation of constraints:

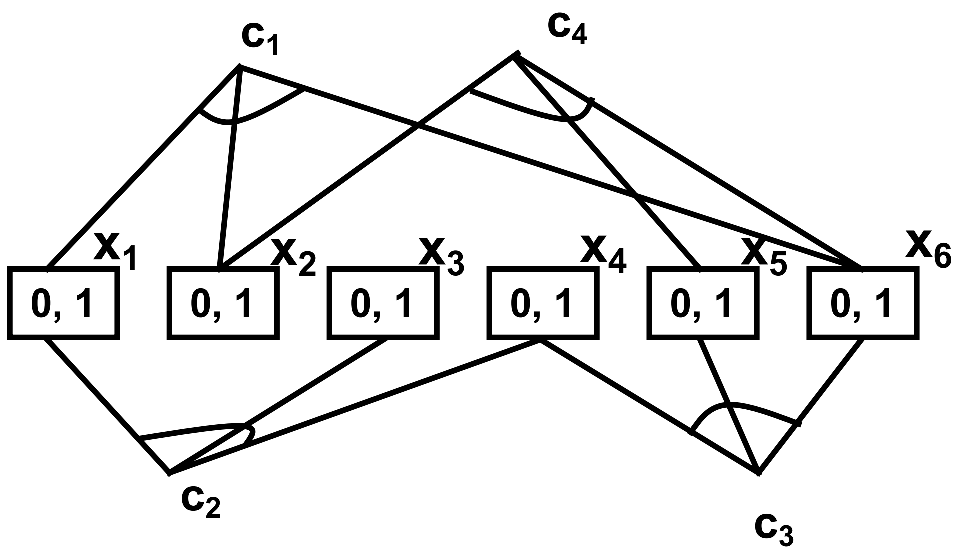
- intentional (algebraic/logic formulae)
- in extension (a set of compatible value tuples, 0-1 matrix)

Representation of a CSP as a (hyper)graph

- nodes = variables
- (hyper)egdes = constraints

Example:

- variables $x_1,\ldots,x_6$ with domain {0,1}
- c_1: $x_1+x_2+x_6=1$
- c_2: $x_1-x_3+x_4=1$
- c_3: $x_4+x_5-x_6>0$
- c_4: $x_2+x_5-x_6=0$



The world is not binary ...

but it can be transformed to a binary one!

Binary CSP

CSP + all the constraints are binary

Note: unary constraints can be easily encoded in the domain of a variable

Equivalence of CSPs

Two constraint satisfaction problems are equivalent if they have the same sets of solutions.

Extended Equivalence of CSPs

Problem solutions can be syntactically transformed between the problems.

Can any CSP be transformed to an (extended) equivalent binary CSP?

The world is not binary ...

but it can be transformed to a binary one!

Binary CSP

CSP + all the constraints are binary

Note

va

Equi

Tw

sa

Exte

Pr

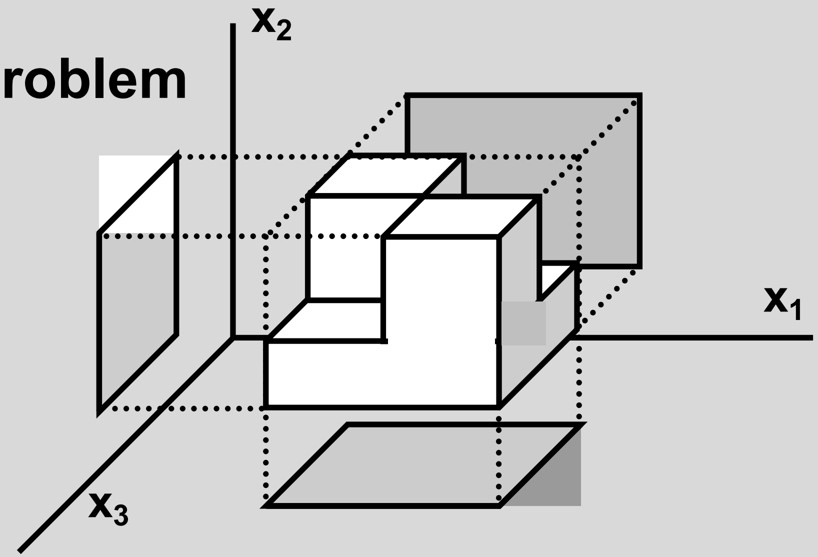
pr

Can

b

Projection *(Montanary 1974)*:

- **Straightforward, but does not give an equivalent problem**
- **Bounds consistency**
  - **better efficiency**
  - **weaker domain filterin**



Swapping variables and constraints.

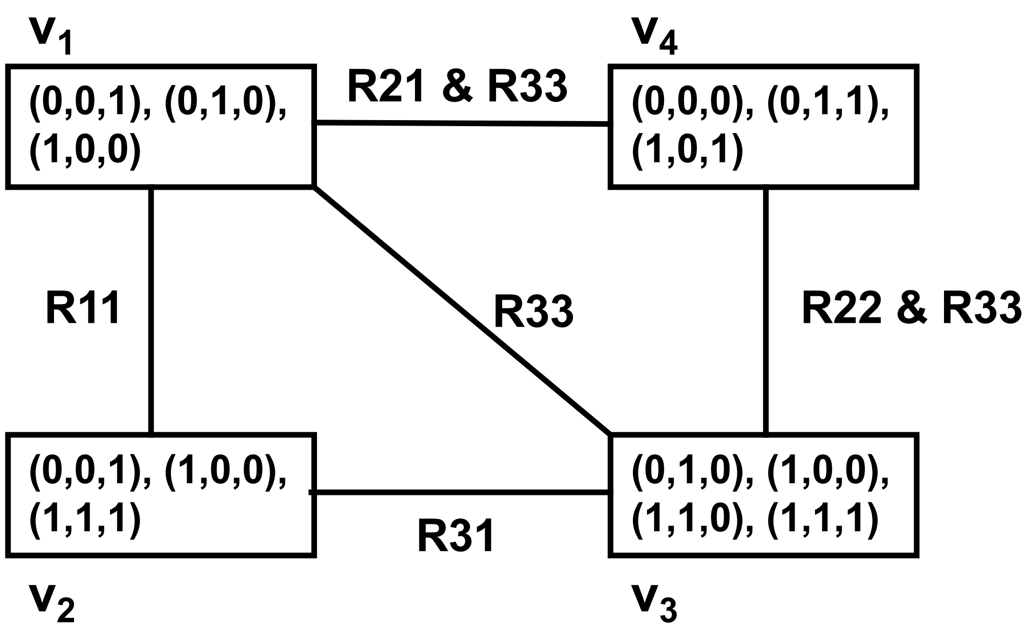
- k- ary constraint c is converted to a **dual variable** v_c with the domain consisting of compatible tuples

- for each pair of constraints c a c' sharing some variables there is a **binary constraint** between v_c a $v_{c'}$ restricting the dual variables to tuples in which the original shared variables take the same value

Example:

- variables $x_1,\dots,x_6$ with domain {0,1}

- c_1: $x_1+x_2+x_6=1$
- c_2: $x_1-x_3+x_4=1$
- c_3: $x_4+x_5-x_6>0$
- c_4: $x_2+x_5-x_6=0$

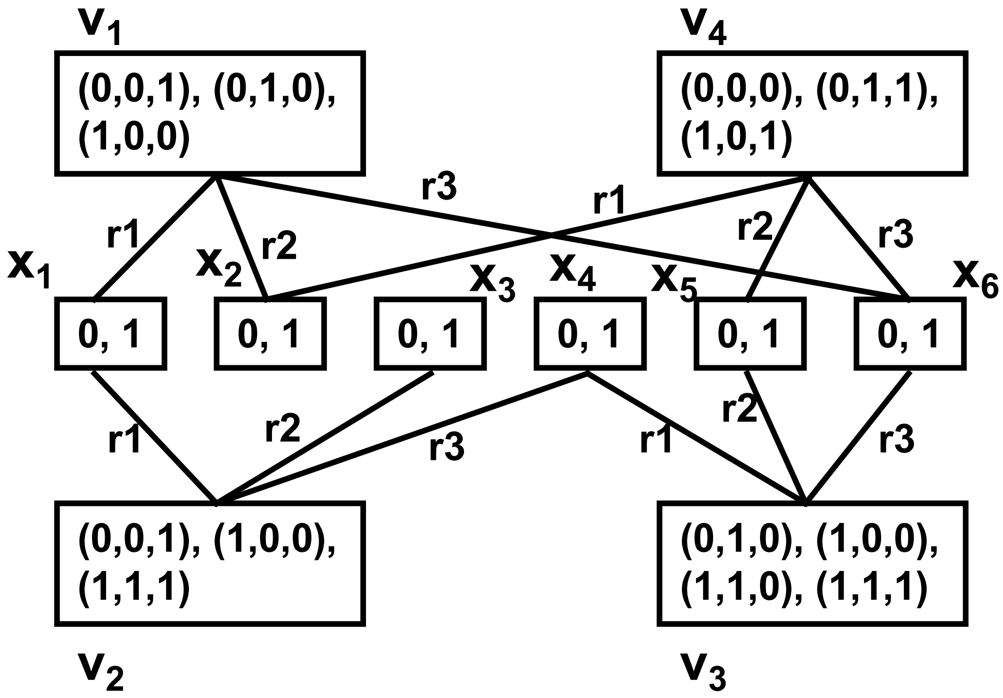


New dual variables for (non-binary) constraints.

- k- ary constraint c is translated to a **dual variable** v_c with the domain consisting of compatible tuples
- for each variable x in the constraint c there is a constraint between x a v_c restricting tuples of dual variable to be compatible with x

Example:

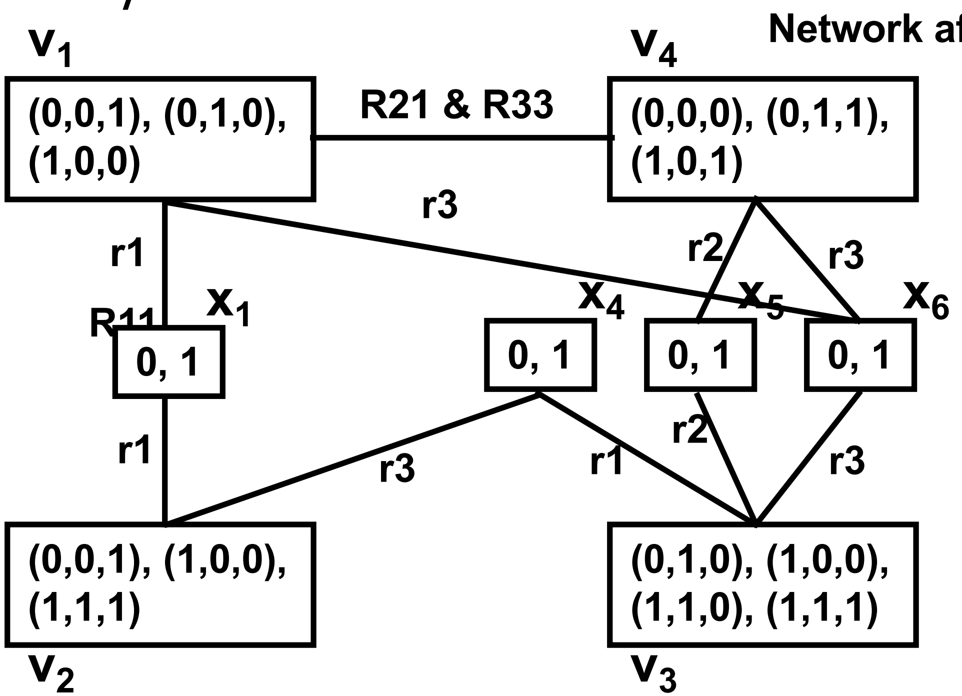
- variables $x_1,\ldots,x_6$ with domain {0,1}
- c_1: $x_1+x_2+x_6=1$
- c_2: $x_1-x_3+x_4=1$
- c_3: $x_4+x_5-x_6>0$
- c_4: $x_2+x_5-x_6=0$



A hidden variable encoding can be transformed to a dual encoding:

- Paths of length 2 between any pair of dual variables are substituted by a binary constraint that combines both relations over the path (r1 and r1 form R11); beware of edges shared between more paths!
- If the original variable becomes isolated (or is connected to a single constraint), then remove the variable.

Example:



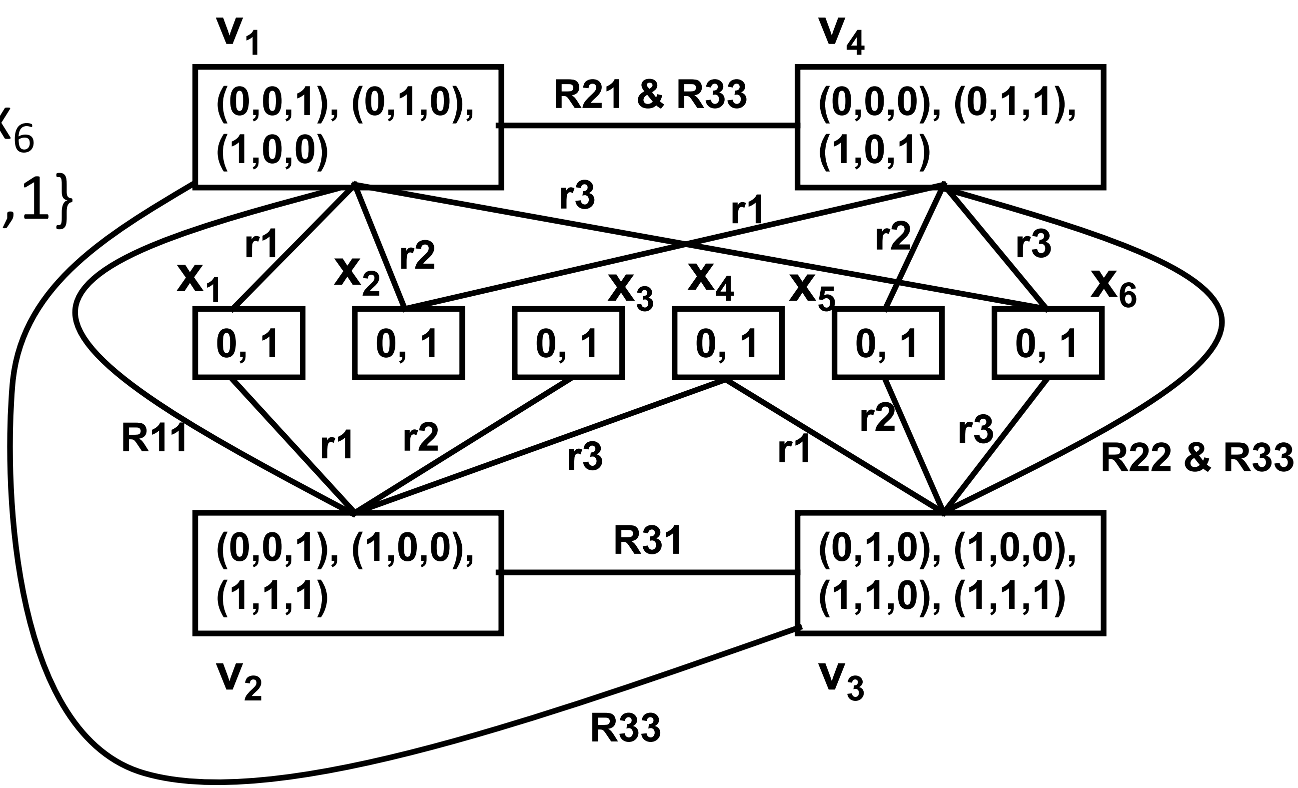
In each transformation step we obtain an equivalent CSP
⇒ **„hybrid“ encoding**

The transformation can also be done in the reverse direction.

Hidden variable encoding can be extended by the dual encoding.

Example:

- Variables $x_1,\ldots,x_6$ with domain {0,1}
- c_1: $x_1+x_2+x_6=1$
- c_2: $x_1-x_3+x_4=1$
- c_3: $x_4+x_5-x_6>0$
- c_4: $x_2+x_5-x_6=0$



Final Notes on Binarisation

- **Why do we do binarisation?**
  - a unified form of a CSP
  - many solving approaches are formulated for binary CSPs
  - tradition (historical reasons)

- **Which encoding is better?**
  - hard to say ;-)
  - dual encoding:
    better propagation but constraints in extension
  - hidden variable encoding:
    keeps original variables but weaker propagation

- **Binary vs non-binary constraints**
  - more complex propagation algorithms for non-binary constraints
  - exploiting semantics of constraints for more efficient and stronger domain filtering

Department of Theoretical Computer Science and Mathematical Logic
bartak@ktiml.mff.cuni.cz