Constraint Programming (CP)

eVITA Winter School 2009

Optimization

Tomas Eric Nordlander

SINTEF

Outline

- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Commercial Application
- Pointers
- Summary

Constraint Programming (CP)

- **The basic idea** of Constraint programming is to solve problems by stating the constraints involved—the solution should satisfy all the constraints.
  - It is an alternative approach to functional programming, that combines reasoning and computing techniques over constraints.

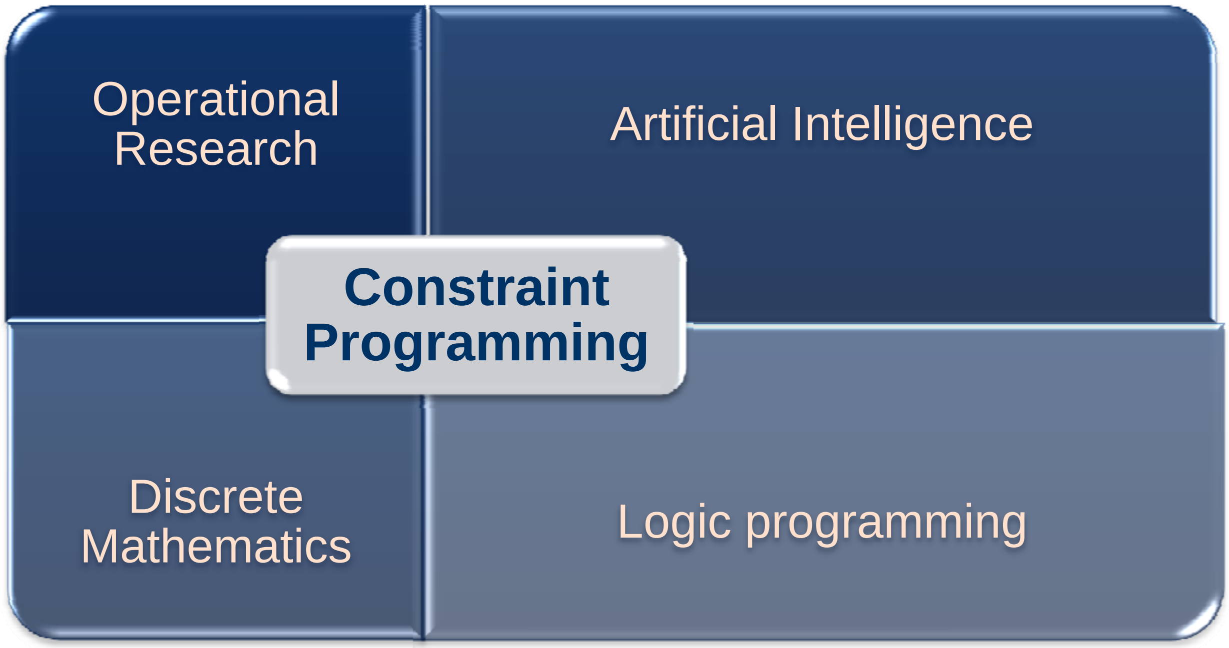
- **Constraint satisfaction**
  - Constraint satisfaction deals mainly with finite domains
  - Combinatorial solving methods

- **Constraint solving**
  - Constraint satisfaction deals mainly with infinite domains
  - Solving methods based more on mathematical techniques (Example: Taylor series)

A Brief History of Constraint Programming

- **Sixties**
  - Sketchpad, also known as Robot Draftsman (Sutherland in, 1963).
- **Seventies**
  - The concept of Constraint Satisfaction Problem (CSP) was developed (Montanari 1974).
  - The scene labelling problem, (Waltz, 1975).
  - Experimental languages .
- **Eighties**
  - Constraint logic programming.
  - Constraint Programming.
- **Nineties**
  - Successfully tackled industrial personnel, production and transportation scheduling, as well as design problems.
- **The last and the upcoming years**
  - Constraint Programming one of the basic technologies for constructing the planning systems.
  - Research Focus: Constraint Acquisition, Model Maintenance, Ease of Use, Explanation, Dynamic Constraints, Hybrid techniques, Uncertainty, etc..

CP Interdisciplinary Nature



Outline

- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Commercial Application
- Pointers
- Summary

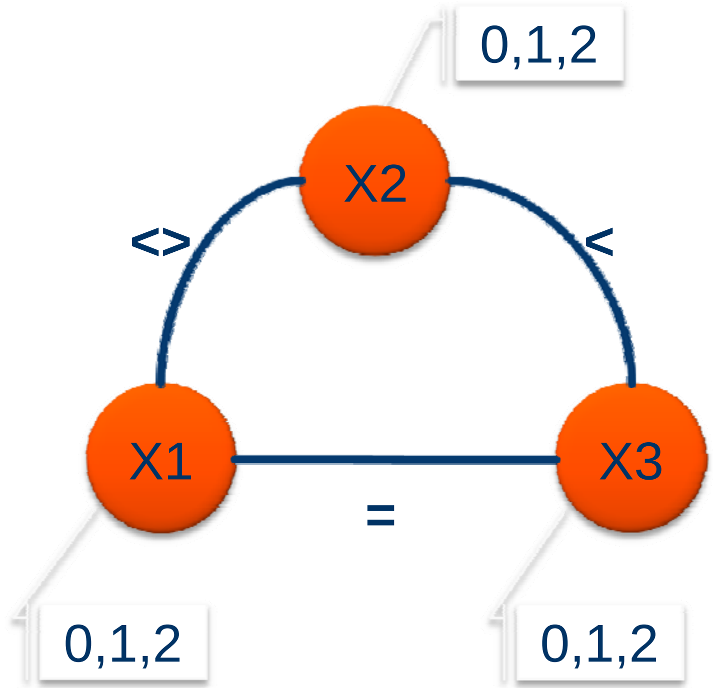
What components are there?

- Model
  - Constraint Satisfaction Problem (CSP)
- Search Algorithms
- Consistency Algorithms
- Heuristics

- Solving a CSP achieve one of the following goals:
  - demonstrate there is no solution;
  - find any solution;
  - find all solutions;
  - find an optimal, or at least a good, solution given some objective evaluation function.

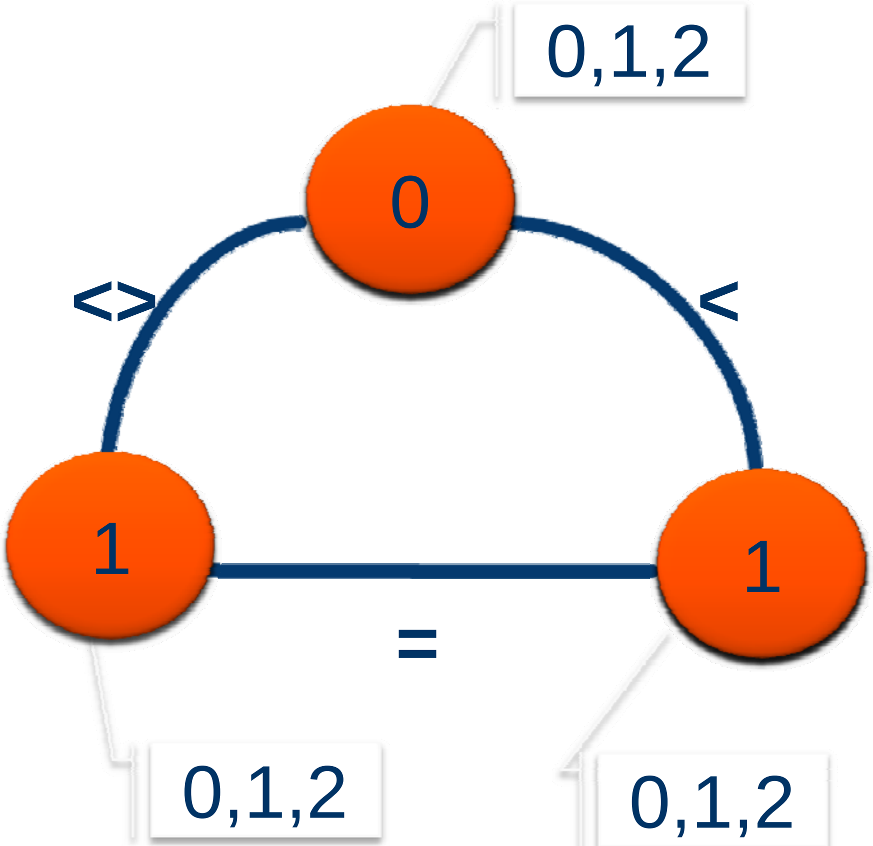
CSP Model

- **Constraint Satisfaction Problem (CSP)**
  - **Definition**
    - a set of <u>variables</u> X={X1,..., Xn},
    - for each variable Xi, a finite set Di of possible values (its <u>domain</u>), and
    - a set of <u>constraints</u> C, where each constraint C<j> is composed of a scope vars(C<j>) of the variables that participate in that constraint and a relation rel(C<j>) ⊆ Dj1 × Dj2 × …× Djt, that specifies the values that variables in the scope can be assigned simultaneous

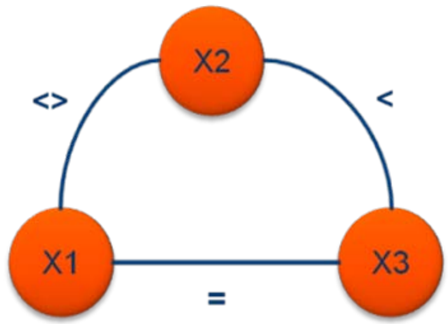


CSP: Search

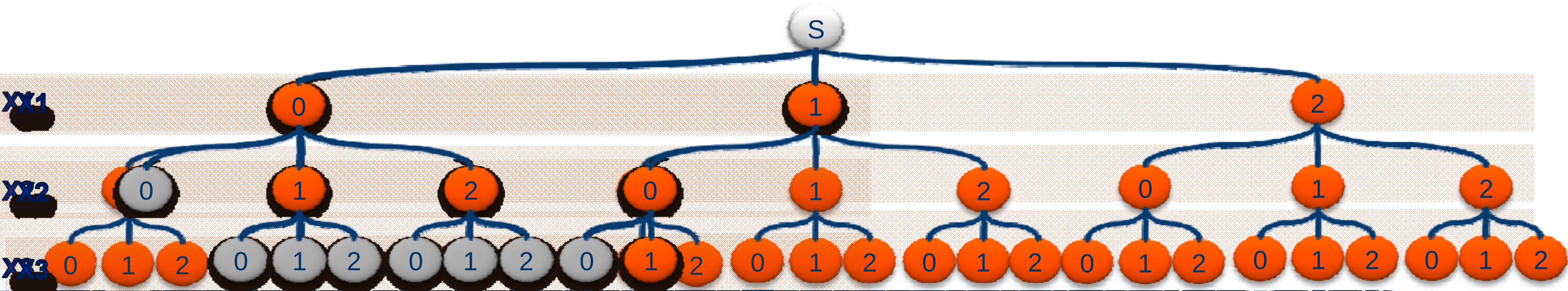
- Simple Backtrack (BT)
- Heuristic
  - Variable ordering X1…X3
  - Value ordering 0…2



CSP: Search

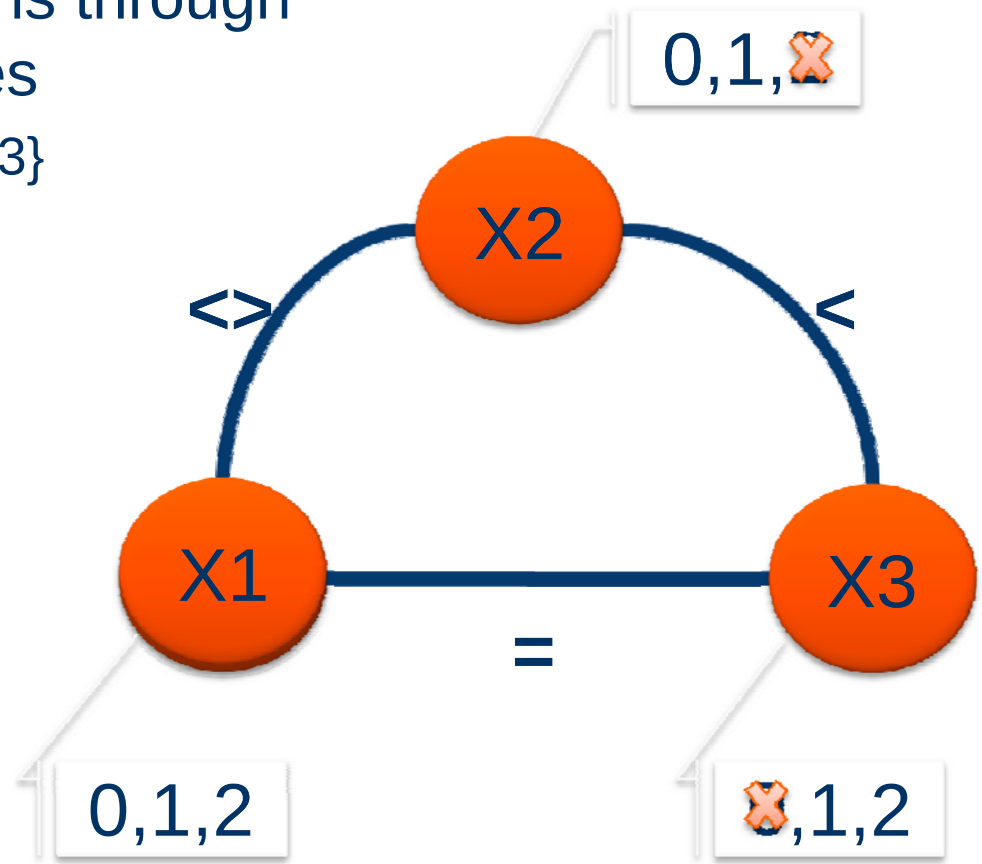


- Simple Backtrack (BT)
- Heuristic
  - Variable ordering X1…X3
  - Value ordering 0…2



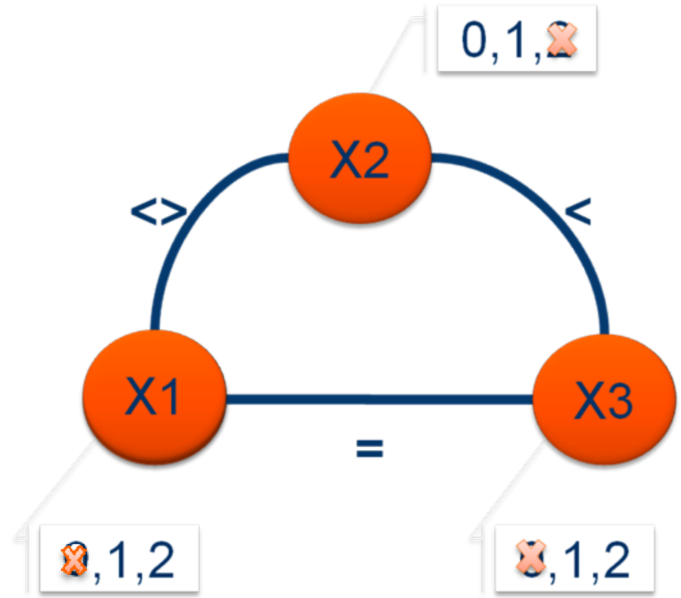
CSP: Constraint Propagation

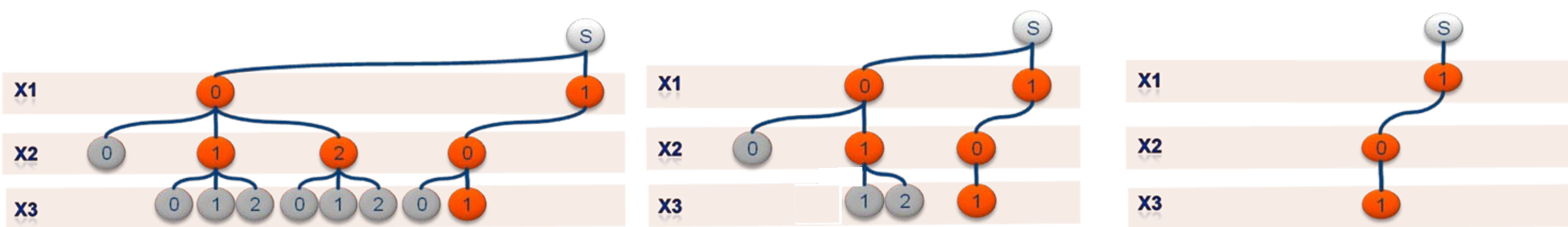
- Constraint propagations through consistency techniques
  - Arc Consistency {X2-X3}



CSP: Constraint Propagation

- Example of Arc Consistency
  - Arc Consistency {X2-X3}
  - Arc Consistency {X1-X3}
  - Arc Consistency {X2-X1}





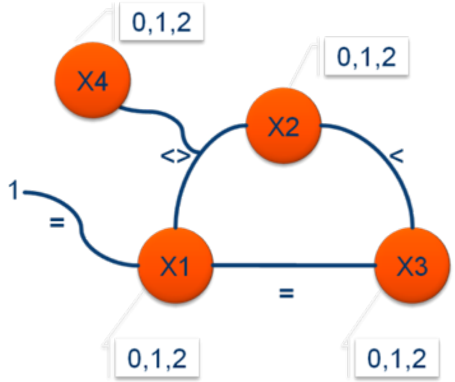
CSP: seems fairly limited?

- CSP and solving methods are much richer then previous example showed, in particular when it comes to:
  - Domain
  - Constraints
  - Search
  - Consistency techniques

CSP: Domain and Constraints

- Domain
  - **finite** (but also continues)
  - **Integer**
  - Reals
  - Boolean (SAT)
  - String
  - Combinations of above

- Constraints
  - linear (but also nonlinear)
  - Unary
  - **Binary**
  - Higher arity
  - Global Constraints



CSP: Search & Consistency tech.

General algorithms

- Generate and Test
- Simple Backtracking
- Intelligent Backtracking

Constraint propagations

- Node* Consistency
- Arc** Consistency
- Path Consistency

Algorithms using consistency checking

- Forward Checking (FC)
- Partial Look Ahead (PLA)
- Full Look (FL)
- **Maintaining Arc Consistency (MAC)**

**Node = Variable*
***Arc = Constraint*

CSP: Modelling

- Critical for success
- Very Easy and very hard
- Often Iterative process
  - Open CSP
  - Dynamic CSP

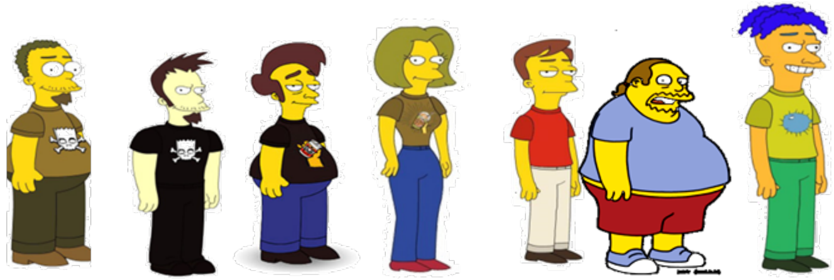
- Trick includes
  - Aux. constraints
    - Redundant to avoid trashing
    - Remove some solution to break symmetry
    - Specialized constraints
  - Aux. Variables
  - Etc.

Outline

- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Application areas
- Pointers
- Summary

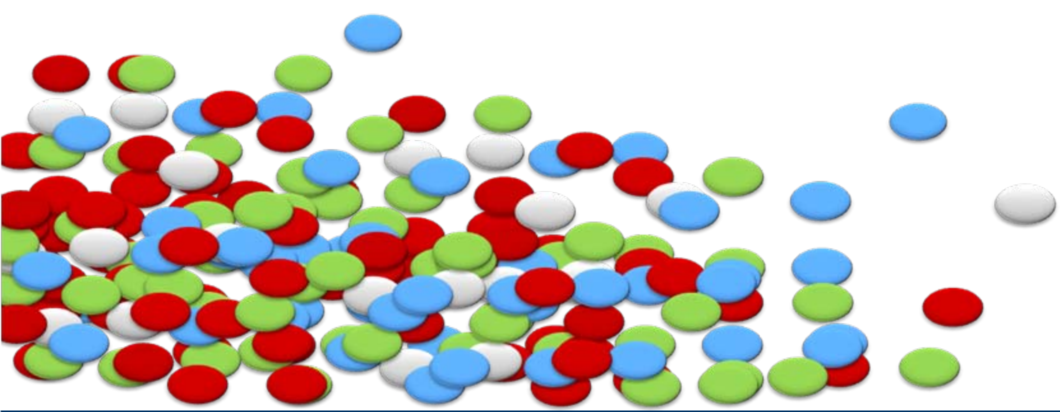
CSP Examples

- Graph Colouring
- Scheduling



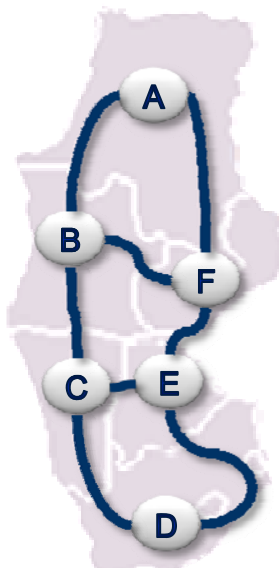
Graph Coloring

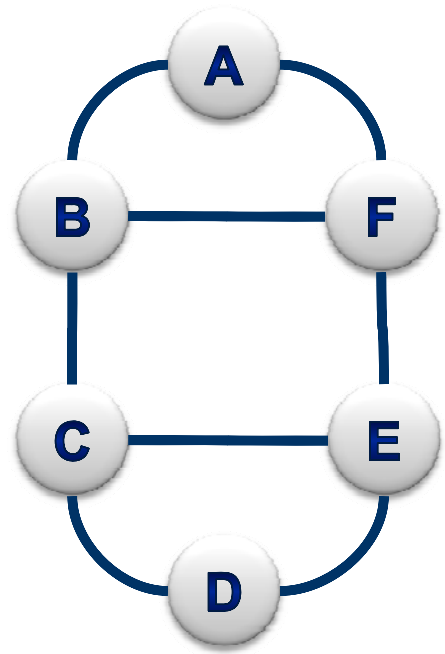
- The graph colouring problem involves assigning colours to vertices in a graph such that adjacent vertices have distinct colours.
- This problem relates to problem such as scheduling, register allocation, optimization.



Graph Coloring

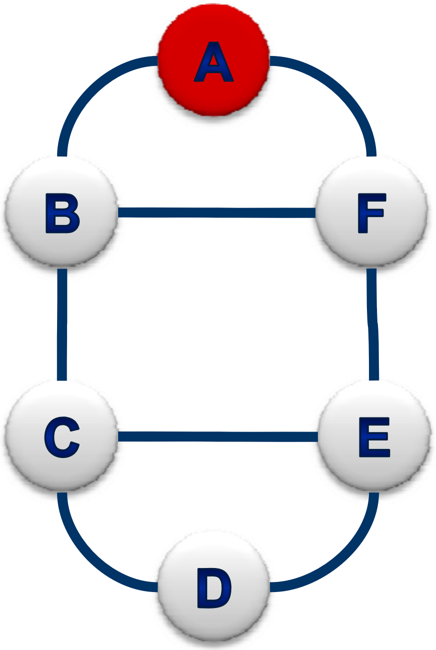
- Variable order: A-F
- Value order: Red, Green, Blue.





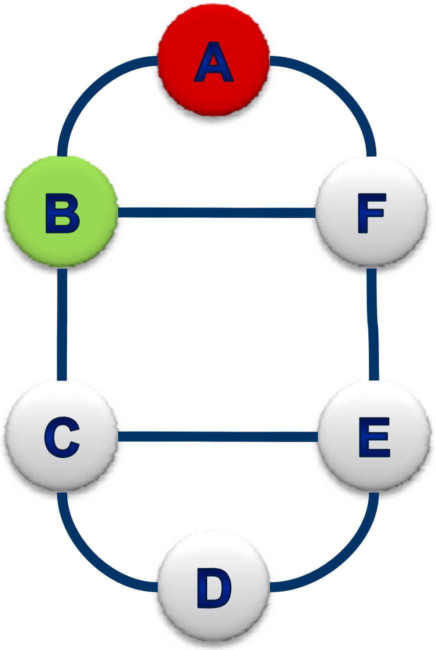
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



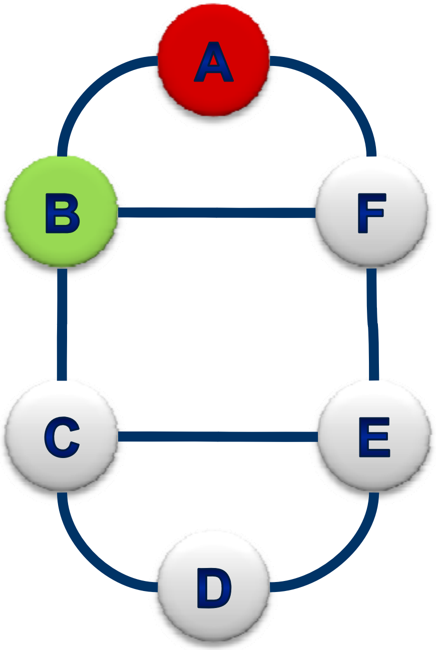
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



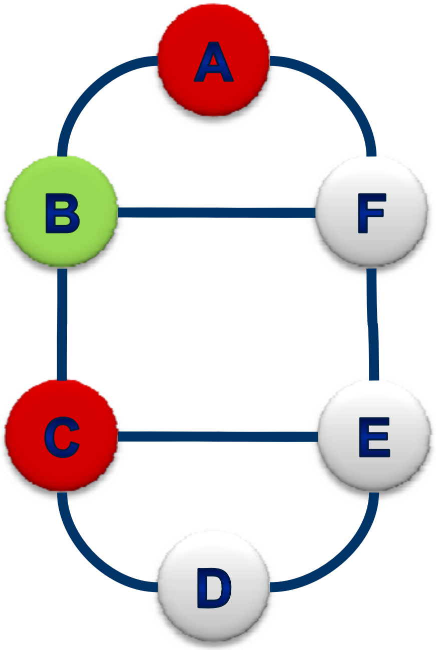
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



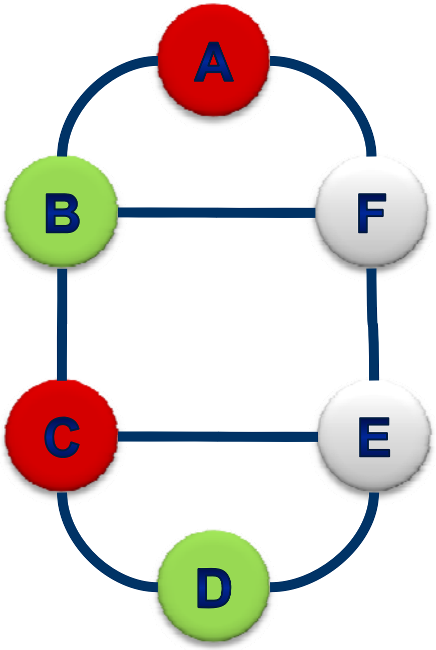
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



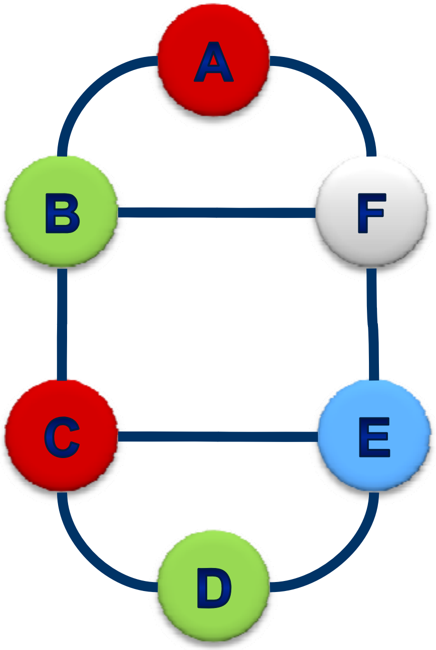
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



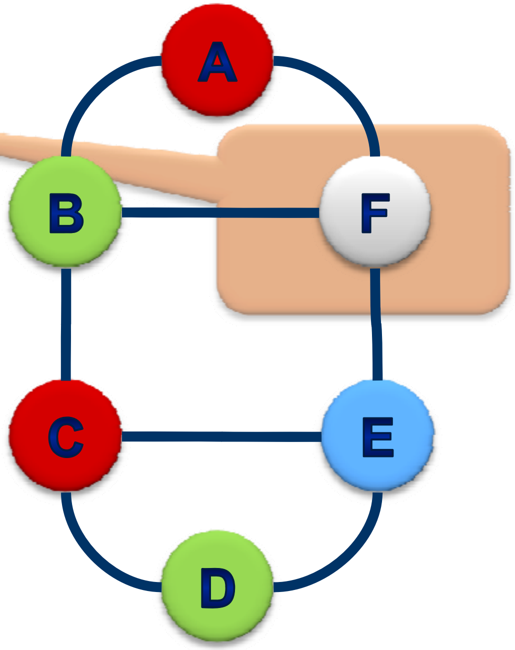
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



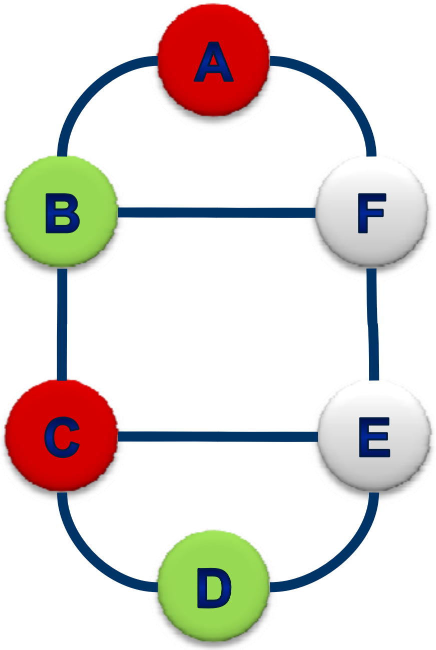
Graph Coloring

We reach a end node without being able to generate a solution…so we need to backtrack



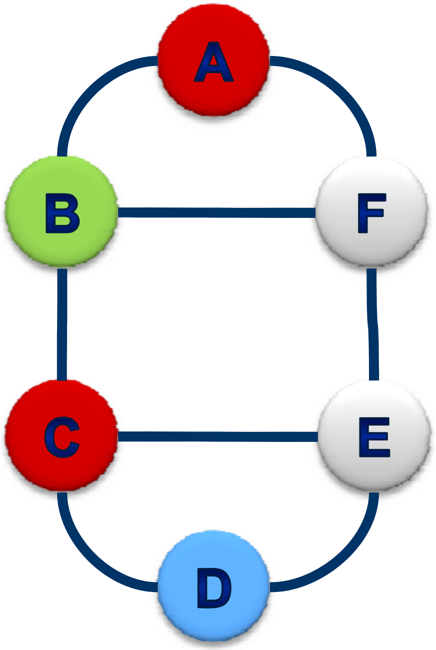
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



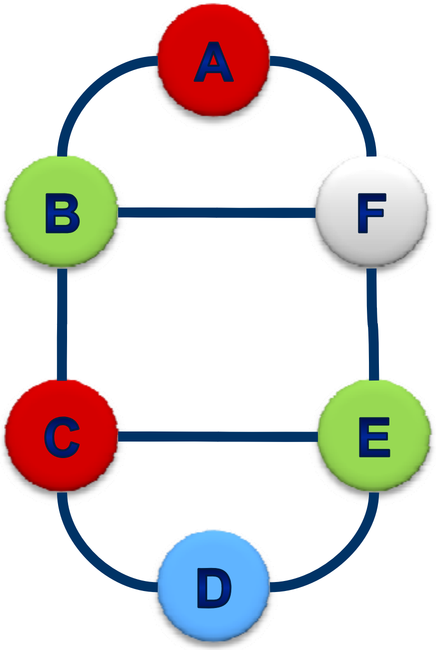
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.



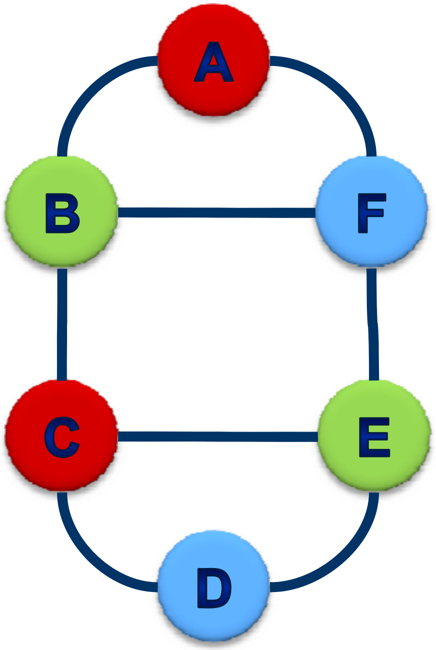
Graph Coloring

- Variable order: A-F
- Value order: Red, Green, Blue.

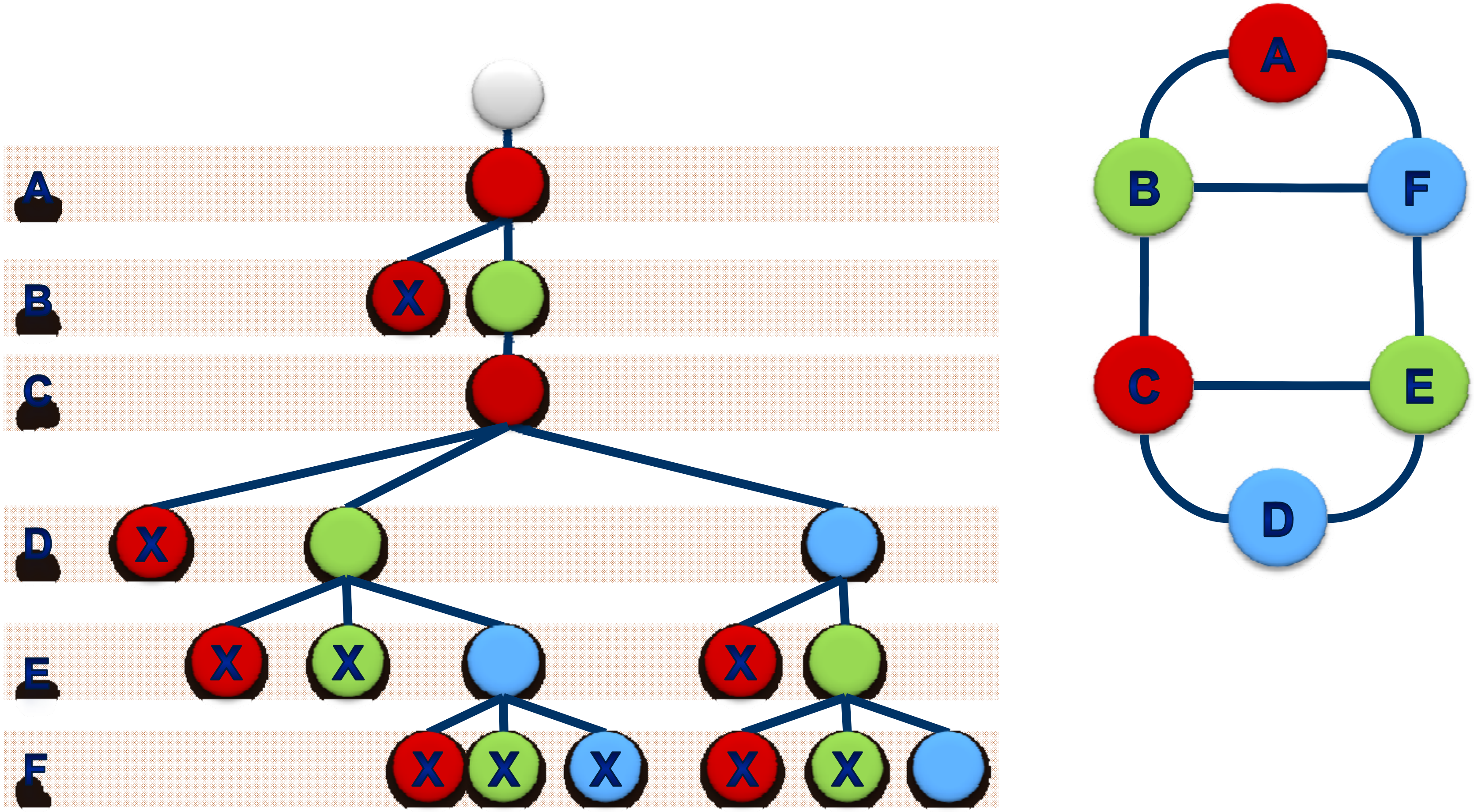


Graph Coloring

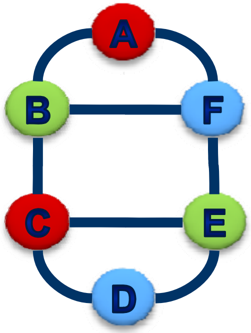
- Variable order: A-F
- Value order: Red, Green, Blue.

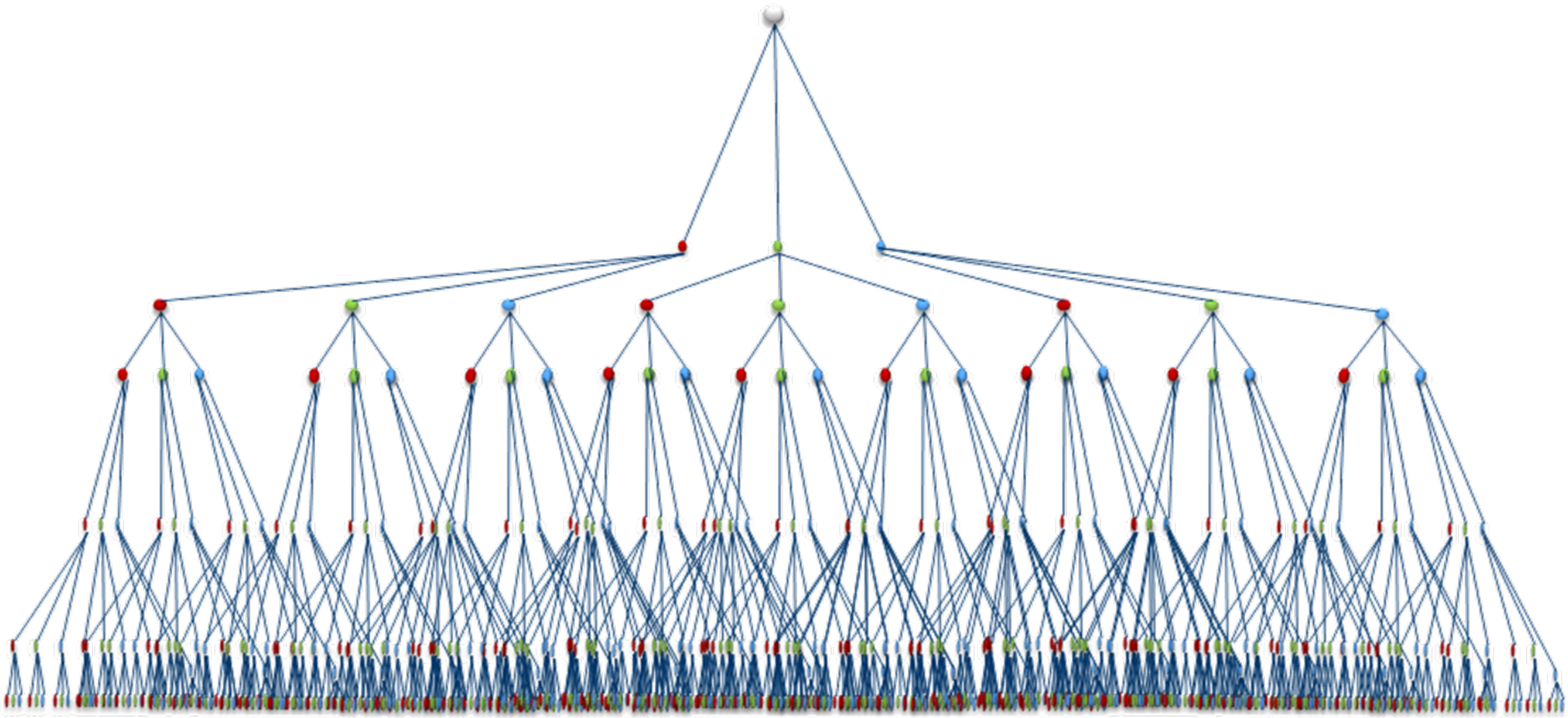


Graph Coloring

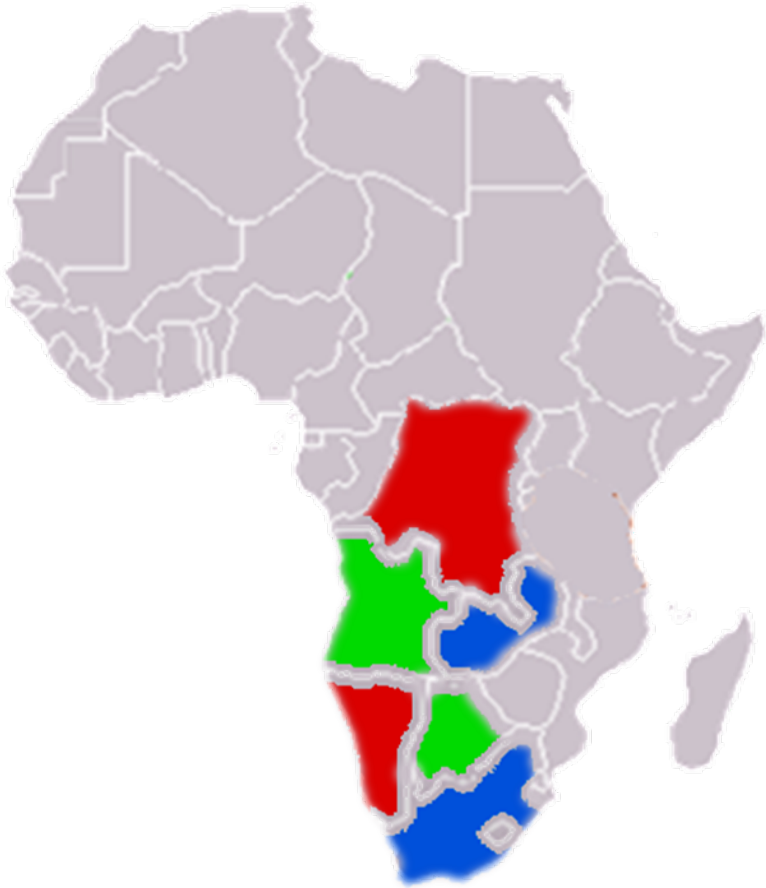


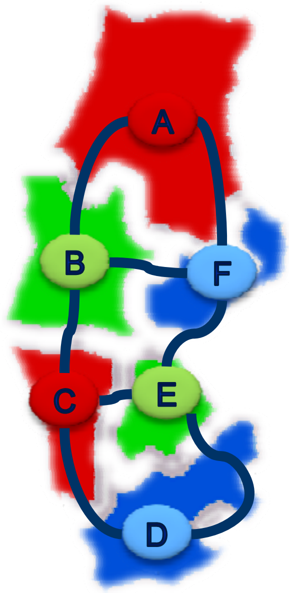
Graph Coloring

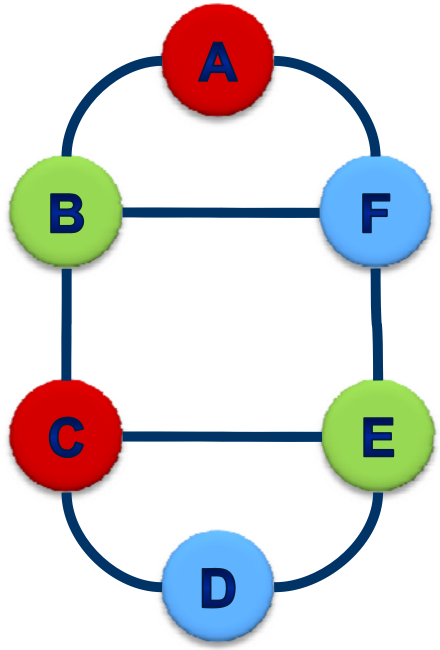




Graph Coloring







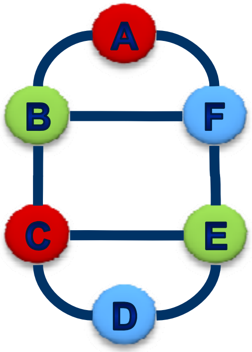
Graph Colouring (SICStus Prolog)

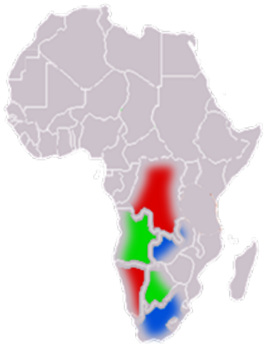
```
:- use_module(library(clpfd)).
solve_AFRICA(A,B,C,D,E,F):-

domain([A,B,C,D,E,F], 1, 3),    % Variables & their domain size
                                 % colour 1=RED, 2=GREEN or 3=BLUE
A #\= B,                         % Constraints
A #\= F,
B #\= F,
B #\= C,
F #\= E,
C #\= E,
C #\= D,
E #\= D,

labeling([],[A,B,C,D,E,F]).      % assign values to the Variables
```

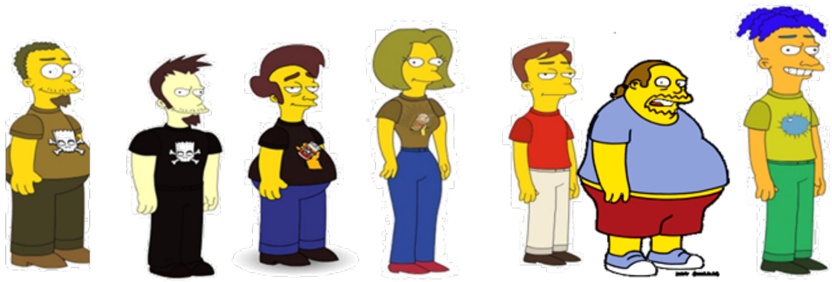
```
| ?- solve_AFRICA (A,B,C,D,E,F).
A = 1,
B = 2,
C = 1,
D = 3,
E = 2,
F = 3?
yes
| ?-
```





CSP example: Scheduling

- We have 7 patients that need different surgeries.
- Our 4 Operations rooms are open 24/7
- We have 13 people in our medical staff, each surgery demands one or more from the staff.

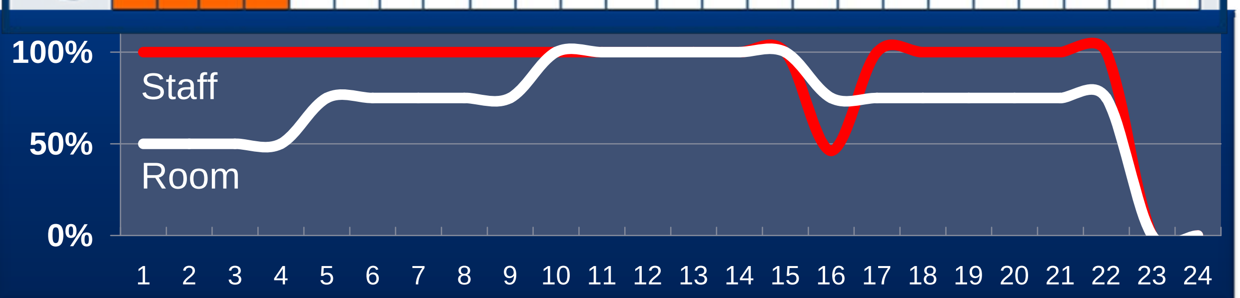


Exp. Duration (h)	16	6	13	7	5	18	4
Resource demand (number of Staff)	2	9	3	7	10	1	11
Starting time	?	?	?	?	?	?	?

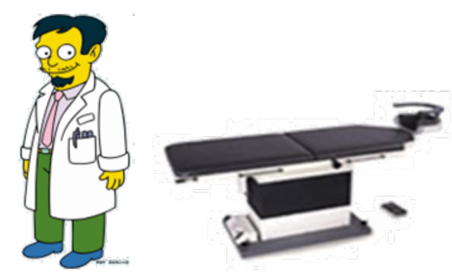
- Give us the optimal plan (starting time for the surgical task) to minimize the total end time?

A optimal solution!

Exp. Duration (h)	16	6	13	7	5	18	4
Resource demand (number of Staff)	2	9	3	7	10	1	11
Starting time	?	?	?	?	?	?	?



- An optimal schedule, all surgeries are conducted within 23 hours.
- Utilisation of the 13 staff and the 4 rooms



CSP example: Scheduling

```
:- use_module(library(clpfd)).
:- use_module(library(lists), [append/3]).

schedule(Ss, Rs, End) :-
   length(Ss, 7),
   Ds = [16,6,13,7,5,18,4],
   Rs = [2,9,3,7,10,1,11],
   domain(Ss,1,30),
   domain([End],1,50),
   after(Ss, Ds, End),
   cumulative(Ss, Ds, Rs, 13),
   append(Ss, [End], Vars),
   labeling([minimize(End)], Vars).

after([], [], _).
after([S|Ss], [D|Ds], E) :-
   E #>= S+D, after(Ss, Ds, E).

%% End of file
```

```
*/ TASK        DURATION  RESOURCE
   ====        ========  ========
   t1          16          2
   t2          6           9
   t3          13          3
   t4          7           7
   t5          5           10
   t6          18          1
   t7          4           11        /*
```

```
| ?- schedule(Ss,Rs,End).
Rs = [2,9,3,7,10,1,11],
Ss = [1,17,10,10,5,5,1],
End = 23 ?
yes
| ?-
```

Outline

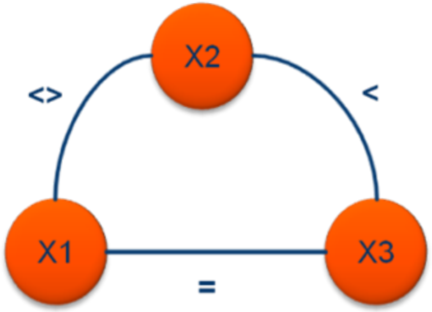
- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Commercial Application
- Pointers
- Summary

Optimisation

- Case A: If there are many solution to the problem
  - Use some criteria to select the best one
  - E.g a cost function

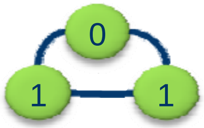
- Case B: If all constraints in a problem cannot be satisfied
  - Seek the “best” partial solution
  - E.g. use MAX-CSP or Constraint hierarchy

Optimization: Case A



Cost function

- The previous example simply found single solution



- A complete search discover two more solutions

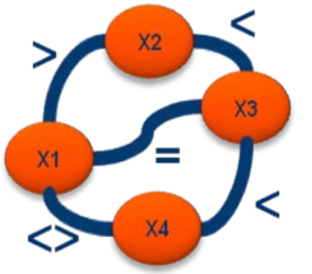
- We can use a simple cost function to find the optimal solution
  - As an example take
    - Cost function = 5X1 + 2X2 – 1X2

Optimization: Case B

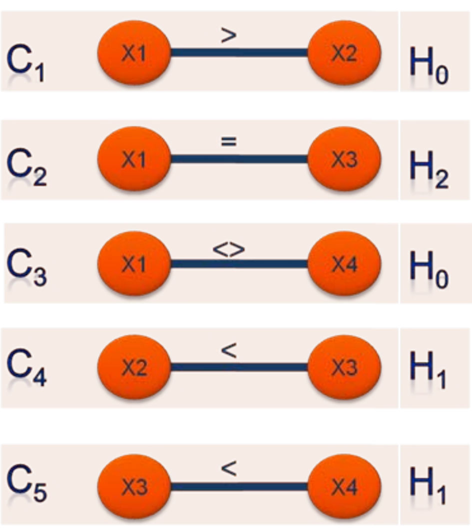
Over constrained problem:

- Not uncommon that real world problem is over-constrained
- Solution, relaxation by removing constraints.
  - Allow user to select constraint(s)
  - associate a cost with each constraint violation (known in mathematical programming as ‘Lagrangian relaxation’
  - Constraint hierarchies

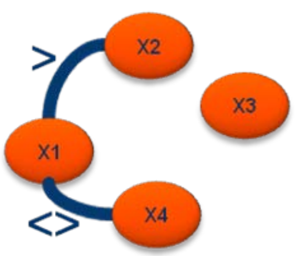
Constraint Hierarchies



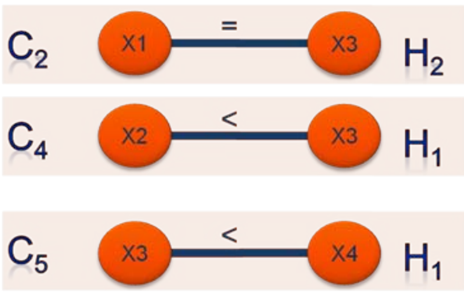
- constraints are partitioned into levels, {H0, H1, .., Hn}
  - H0 contain hard constraints that must be satisfied
  - H1 … Hn hold soft constraints of decreasing priority or strength
- If the CSP_{H0} have a solution start adding constraints from the next hierarchy.



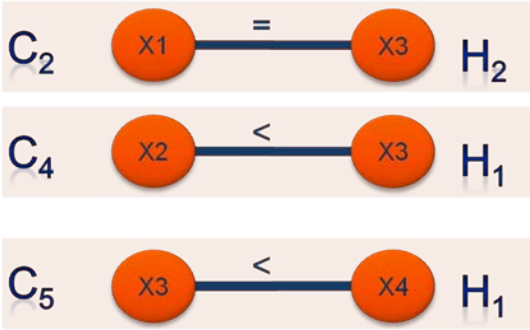
CSP_{H0}



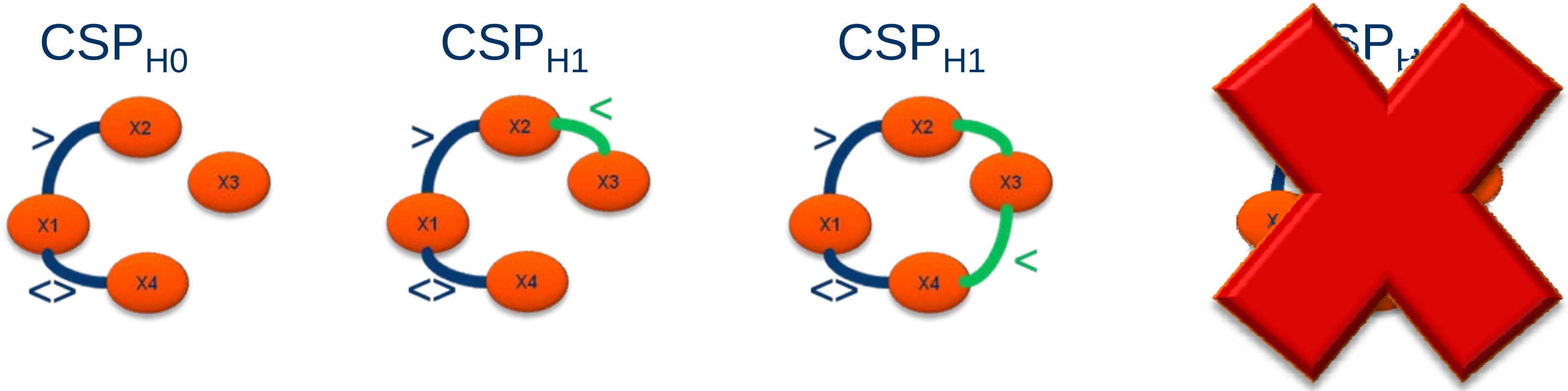
Soft Constraints



Constraint Hierarchies



- Solution Space Example



- A Solution?

Outline

- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Application areas
- Pointers
- Summary

Commercial Application: examples

- Airline crew scheduling - XLufthansa: (OR + CP)
- The counter allocation problem - Hong Kong International Airport.
- structure prediction of lattice proteins with extended alphabets
- Nurse rostering - GATsoft (Norweigen hospitals) (CP + LS)
- Planning and Scheduling space exploration - Component Library for NASA
- Staff planning – BanqueBuxelles Lambert
- Vihivle production optimization – Chrysler Corperation
- Planning medical appointments – FREMAP
- Task scheduling – Optichrome computer systems
- Resource allocation – SNCF
- From Push to Pull manufacturing – Whirlpool
- Utility service otimization – Long island lighting company
- Intelligent cabling of big building – France Telecom
- Financial DSS – Caisse des Depots
- Load Capacity constraint regulation – Eurocontrol
- Planning satellites mission – Alcatel Espace
- Optimization of configuration of telecom equipment – Alcatel CIT
- Production Scheduling of herbicides – Monsanto
- “Just in time” transport and logistic in food industri – Sun Valley
- Etc.

Pointers: Background reading

- http://www.cs.toronto.edu/~fbacchus/csc2512/biblio.html
- Nordlander, T.E., (2004) 'Constraint Relaxation Techniques & Knowledge Base Reuse', University of Aberdeen, PhD Thesis, pp. 246.

Pointers: Constraint Logic Languages

- **SCREAMER** (LISP, open software)
- B-Prolog (Prolog based, proprietary)
- CHIP V5 (Prolog based, also includes C++ and C libraries, proprietary)
- Ciao Prolog (Prolog based, Free software: GPL/LGPL)
- **ECLiPSe** (Prolog based, open source)
- **SICStus** (Prolog based, proprietary)
- GNU Prolog
- YAP Prolog
- SWI Prolog a free Prolog system containing several libraries for constraint solving
- Claire

Pointers: Libraries for CP

- **Choco** (Java library, free software: X11 style)
- **Comet** (C style language for constraint programming, free binaries available for non-commercial use)
- Disolver (C++ library, proprietary)
- Gecode (C++ library, free software: X11 style)
- Gecode/J (Java binding to Gecode, free software: X11 style)
- **ILOG CP** Optimizer (C++, Java, .NET libraries, proprietary)
- ILOG CP (C++ library, proprietary)
- **JaCoP** (Java library, open source)
- JOpt (Java library, free software)
- Koalog Constraint Solver (Java library, proprietary)
- **Minion** (C++ program, GPL)
- python-constraint (Python library, GPL)
- Cream (Java library, free software: LGPL)
- (Microsoft Solver Foundation (free up to a certain problem size))

Summary

- Constraint Programming
  - History
- Constraint Satisfaction
  - Constraint Satisfaction Problem (CSP)
  - Examples
- Optimisation
  - Many solution
  - Over constrained
- Application areas
- Pointers
- Summary

THE END

References:

(Bistarelli et al, 1997) S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. J. of the ACM, 44(2):201–236, 1997.

(Montanari, 1974) U. Montanari Networks of constraints: Fundamental properties and application to picture processing, Inf. Sci. 7, 95-132, 1974

(Sutherland , 1963) Sketchpad (aka Robot Draftsman) was a revolutionary computer new electronic editionPDF (3.90 MiB) was published in 2003

(Waltz, 1975) Waltz, D.L.: Understanding line drawings of scenes with shadows, in: Psychology of Computer Vision, McGraw- Hill, New York, 1975

(Burke & Kendal, 2005) Edmund Burke, Graham Kendall. Search methodologies: Introduction tutorials in optimization and decision support techniques

Tomas Eric Nordlander

tomas.nordlander@sintef.no