Methods of Artificial Intelligence

3. Advanced Constraint Satisfaction Problems

Nohayr Muhammad

Winter Term 2022/2023

November 18th, 2022

Today..

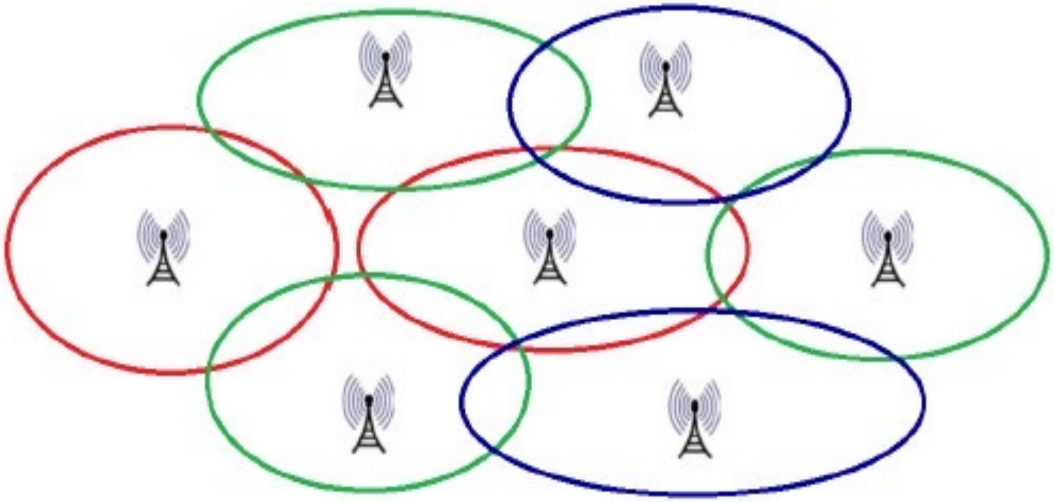
- Examples and Definitions
- Consistency Concepts for CSPs
- Local Search for CSPs

Constraint Satisfaction Problems

Examples and Definition

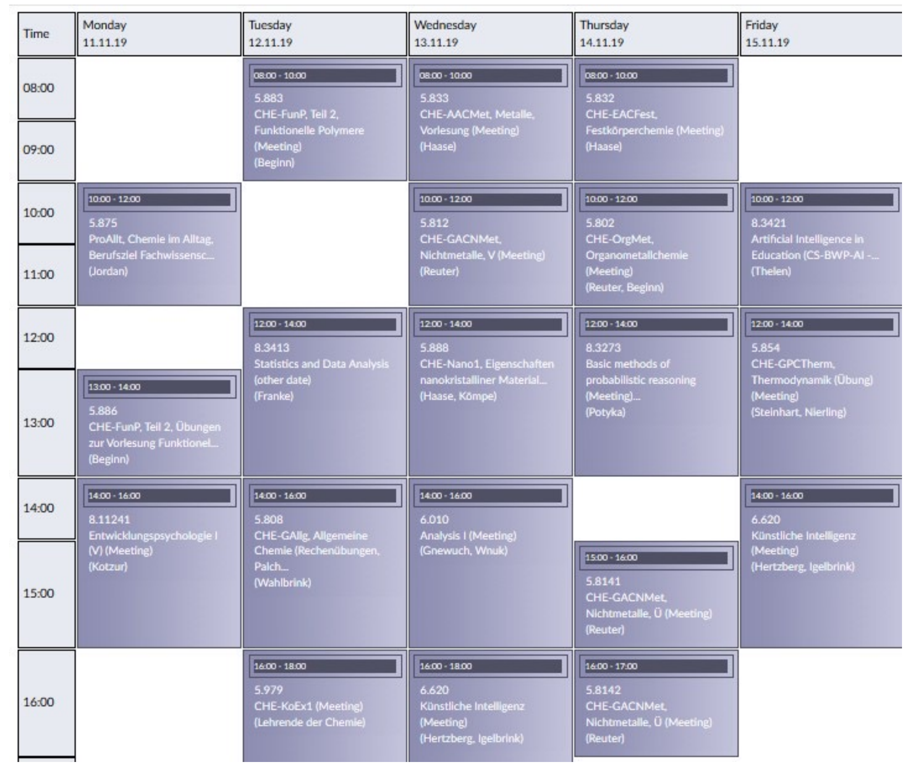
Frequency Assignment

- Assign frequency to every radio station (variable assignment)
- such that stations with overlapping regions use different frequencies (constraints)



Time Table Scheduling

- Assign (time slot, room) to every course *(variable assignment)*
- such that *(constraints)*
  - *No two courses take place at the same time slot in the same room*
  - *Room requirements for the course are met*
  - *Courses given by the same lecturer take place in different time slots*
  - *Courses offered for the same study program and semester take place in different time slots*



Constraint Satisfaction Problems (CSPs)

- A CSP is a tuple $(\{X_1, \ldots, X_n\}, \{D_1, \ldots, D_n\}, R)$ consisting of:
  - *variables* $X_1, \ldots, X_n$
  - Each variable X_i has an associated *domain* D_i
    - E.g. {true, false}, {red, blue, green}, [2,…,10], *N*, *Z*, *R*
  - Set of *constraints* R
    - Constraints are defined on variables and restrict the possible values that can be assigned to variables. For example,
    - $X_7 \in$ {red, blue, green} (unary constraint)
    - $X_1 \leq X_2$ (binary constraint)
    - $X_3 + X_4 \geq 4 * X_5 + 2 * X_6$ (4-ary constraint)

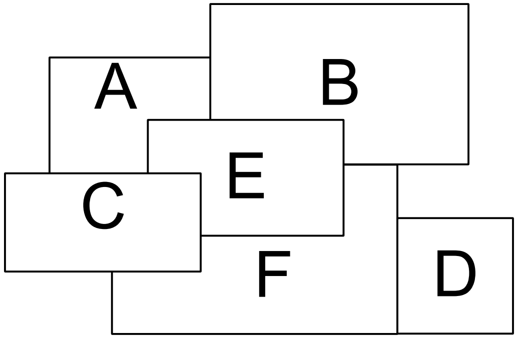
Solutions / Consistent Assignments

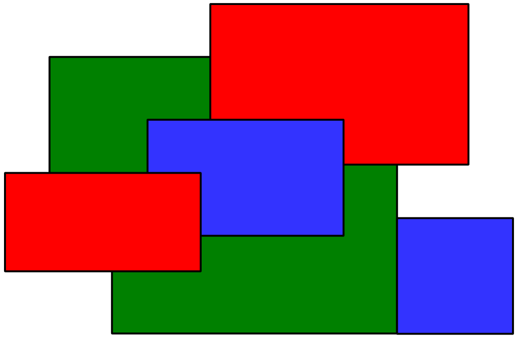
- (Complete) variable assignment:
  assigns to each variable a value from its domain
- Partial variable assignment:
  assigns values to a subset of all variables
- Consistent assignment: does not violate any constraints of CSP
- Solution: consistent complete assignment
- Consistent CSP: there exists a solution

- **Main problem:** Given CSP,
  - find a solution or
  - report that the CSP is inconsistent

Example: Map Coloring

- Variables: A, B, C, D, E, F
- Domains: {red, blue, green} for all variables
- Constraints
  - $A \neq B$
  - $A \neq C$
  - $A \neq E$
  - $B \neq E$
  - $B \neq F$
  - $C \neq E$
  - $C \neq F$
  - $D \neq F$
  - $E \neq F$





Representation of Constraints

- CSPs from a logical point of view:
  - Constraints correspond to first-order predicates $P(x_1, x_2,..., x_n)$
  - All constraints must evaluate to true for a solution
    - $X_1 \leq X_2$
      P contains all pairs (v_1, v_2) over domains of X_1 and X_2 such that $v_1 \leq v_2$
    - $X_3 + X_4 \geq 4 * X_5 + 2 * X_6$
      P contains all 4-tuples (v_3, v_4, v_5, v_6) over domains of variables such that $v_3 + v_4 \geq 4 * v_5 + 2 * v_6$
- Explicit representation: store all n-tuples in P
- Implicit representation: implement P as a function that takes n-tuples as arguments and returns true or false

Example

- Consider variables X_1, X_2 with domains {1,2,3}
- Consider constraint $X_1 < X_2$
- Logically, $X_1 < X_2$ is the binary predicate

  $P(X_1,X_2) = \{ (1,2), (1,3), (2,3)\}$

- { (1,2), (1,3), (2,3)} is the explicit representation of P.

  Assignment $(X_1=v_1, X_2=v_2)$ satisfies P iff $(v_1,v_2) \in P$

- the implicit representation of P takes a pair (v_1,v_2) as argument and returns true iff $v_1 < v_2$.

  Assignment $(X_1=v_1, X_2=v_2)$ satisfies P iff $P(v_1,v_2) = \text{true}$

Exercise

- Suppose X_1 and X_2 both have domain {1,2,..,20},
- $P(X_1, X_2)$ is the constraint $X_1 < X_2$,
- v is a variable assignment for X_1 and X_2.
- How many operations do we have to perform in the worst-case to test whether v satisfies P if
  - P is represented explicitly
  - P is represented implicitly
- Which representation is more efficient?

Solution

- Logically, v satisfies P iff $P(v(X_1),v(X_2))$ is true
- If P is represented explicitly, we may have to enumerate all tuples in P to test whether $(v(X_1),v(X_2))$ is in P
- In our example the explicit representation of P contains 19+18+…+1 = 190 tuples that we may have to enumerate
- If P is represented implicitly, we only have to perform a single comparison
- Test $v(X_1) < v(X_2)$
- So the implicit representation is much more efficient

- Remember: if P is defined by a criterion that can be computed easily, choose implicit representation

Solving CSPs

- Naive Approach: Enumerate all possible variable assignments
- Backtracking Search: Assign values variable by variable and do backtracking (change value of last variable assignment) on violations
- runtime can depend heavily on chosen selection functions, e.g. :
  - First-Fail Variable Selection Heuristic (Minimum Remaining Values): select variable with minimal domain size
  - Degree Variable Selection Heuristic: select variable that is involved in largest number of constraints for unassigned variables
  - Least Constraining Value Value Selection Heuristic: select value that rules out fewest choices for neighbors
- But approaches do not take implications of constraints into account, i.e. they check too many configurations that could be ruled out in advance

Normalized CSPs

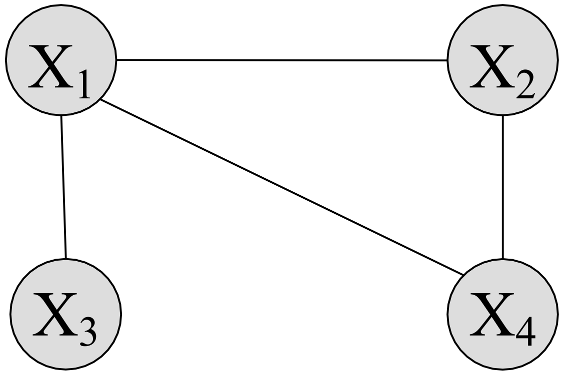
- We restrict our discussion to normalized CSPs that contain only unary and binary constraints

Domains:	$D_1, D_2, \ldots, D_n,$
Unary constraints:	$P_1\,(X_1), P_2\,(X_2), \ldots, P_n\,(X_n),$
Binary constraints:	$P_{1,2}(X_1, X_2), P_{1,3}(X_1, X_3), \ldots, P_{n\text{-}1,n}(X_{n\text{-}1}, X_n).$

- This does not mean any loss of generality (we can transform each CSP into an 'equivalent' normalized CSP in polynomial time)
(cf. e.g. http://ktiml.mff.cuni.cz/~bartak/constraints/binary.html)

Constraint Graph for Normalized CSPs

- Normalized CSPs can be easily represented in a constraint graph



- Node X_i can be identified with unary constraint P_i
- Edge between X_i and X_j can be identified with binary constraint P_{ij}

Consistency Concepts for CSPs

Node Consistency

Arc Consistency

Path Consistency

k-Consistency

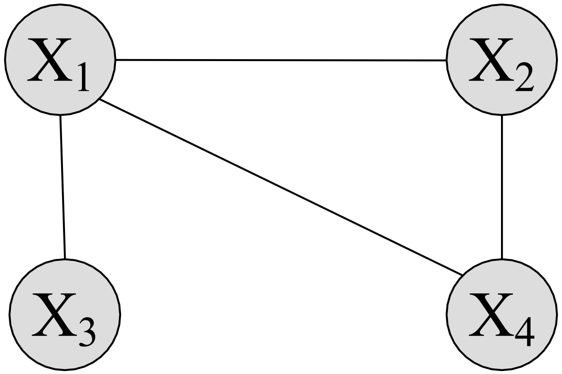
Consistency Concepts

- Basic idea: delete useless elements from domains of variables
- Can be used for pre-processing or in backtracking search after assigning a value to a variable

- The most important consistency concepts are
  - **Node Consistency**
  - **Arc Consistency**
- More involved concepts exist, but become harder to compute

- Tradeoff: If preprocessing time exceeds time for solving CSP directly, we do not gain anything

Node Consistency



- We call X_i node consistent iff all $x \in D_i$ are in P_i
- We call CSP node consistent iff all nodes are node consistent
- We can easily make CSP node consistent by letting $D_i = P_i$
- For example, consider frequency assignment problem: If P_i allows only particular frequencies for X_i, we just modify D_i accordingly

Exercise: Node Consistency

Consider frequency assignment problem with

- Variables: A, B, C, D, E, F
- Domains: $\{f_1, f_2, f_3\}$

- Suppose, we have a constraint $A \neq f_1$
  - Is problem node consistent?
  - How can you make CSP node consistent?

Solution: Node Consistency

Consider frequency assignment problem with

- Variables: A, B, C, D, E, F
- Domains: $\{f_1, f_2, f_3\}$

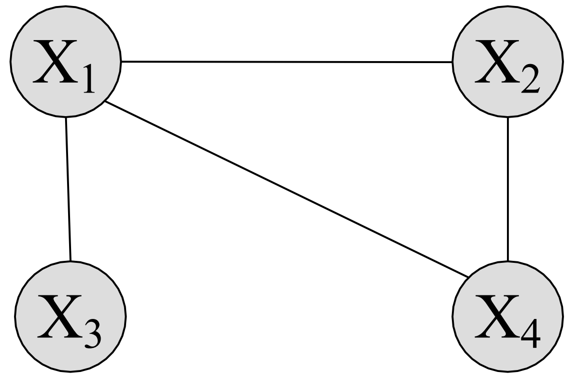
- Suppose, we have a constraint $A \neq f_1$
  - Is problem node consistent?

    Answer: No because the domain of A contains f_1.

  - How can you make CSP node consistent?

    Answer: We set the domain of A to $\{f_2, f_3\}$ (delete f_1)

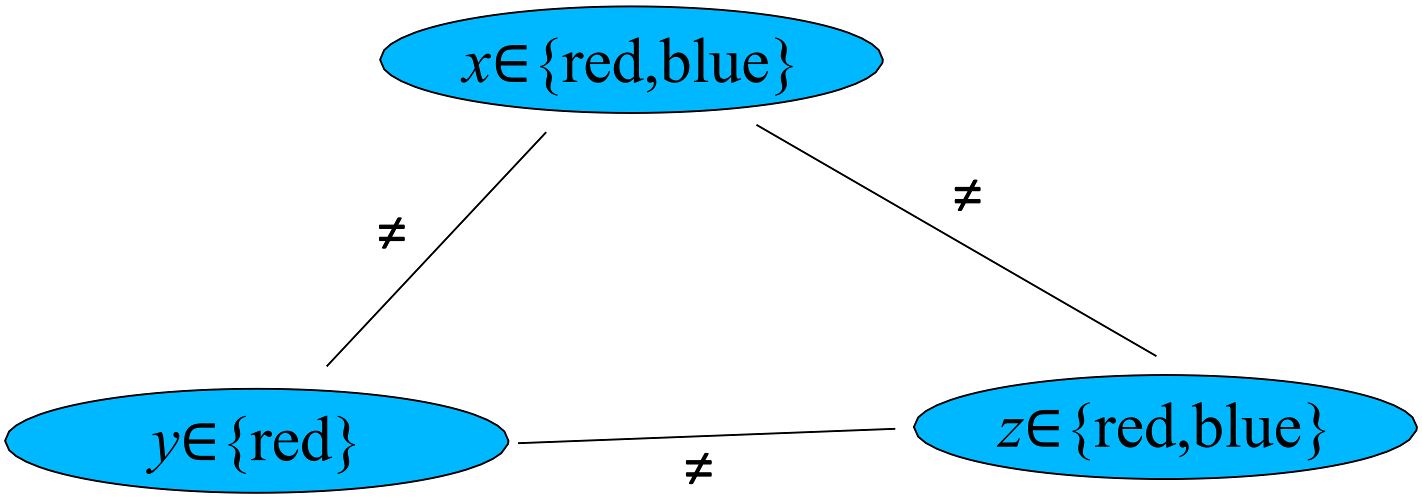
Arc Consistency



- We call X_i arc consistent with X_j iff for all $x \in D_i$ there is a $y \in D_j$ *such that* (x,y) is in P_{ij}
- Intuitively, for each value that X_i can take, X_j can take a value such that the binary constraint between X_i and X_j is satisfied
- We call CSP arc consistent iff all pairs of nodes are arc consistent
- Maintaining arc-consistency is more sophisticated

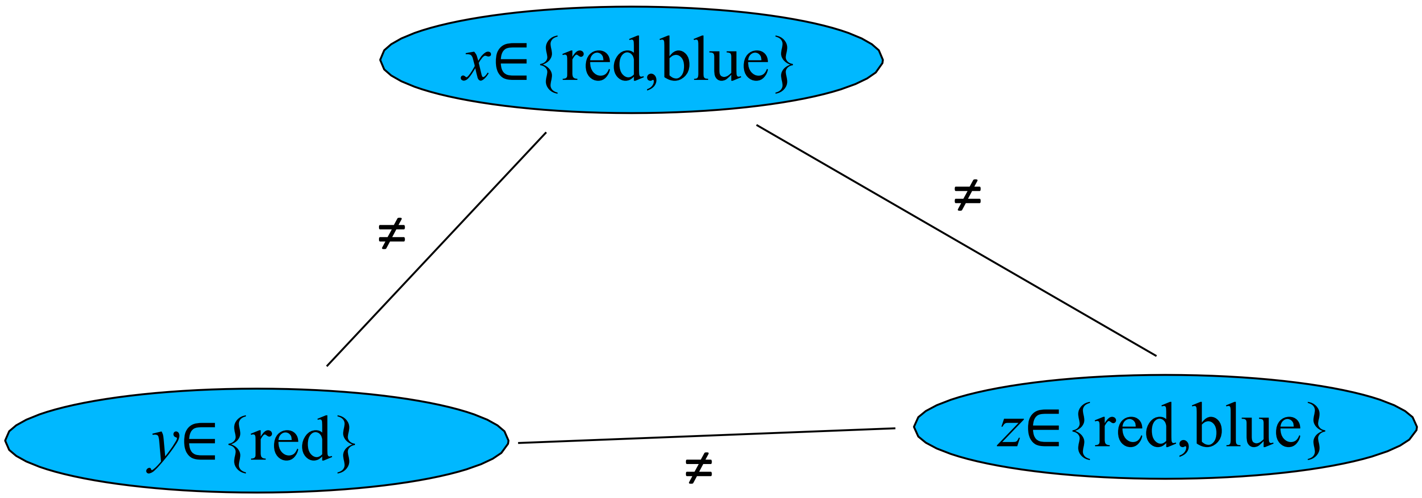
Exercise: Arc Consistency

- Consider the following Graph Coloring Problem



- What variables are arc consistent with each other?
- Is the CSP arc consistent?

Solution: Arc Consistency



- x is arc consistent with z
- y is arc consistent with x and z
- z is arc consistent with x
- x is not arc consistent with y
- z is not arc consistent with y
- Hence, the CSP is not arc consistent

AC-1 for establishing Arc Consistency

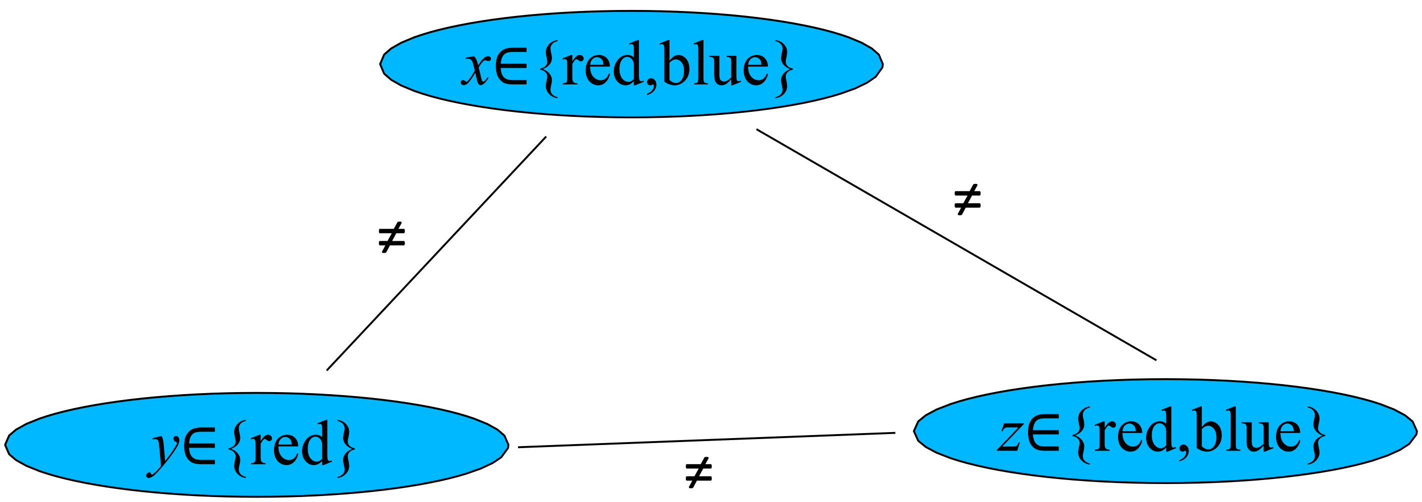
- AC-1 is the easiest (but not the most efficient) algorithm for making CSPs arc consistent

```
Revise(X_i, X_j)                    (make X_i arc-consistent with X_j)
  for each v_i in D_i
    if there is no v_j in D_j such that (v_i, v_j) in P_ij
      delete v_i from D_i
```

```
AC-1(X,D,R)                         (make all variables arc-consistent)
  do
    for each binary constraint P_ij
      Revise(X_i, X_j)
      Revise(X_j, X_i)
  until all domains remain unchanged or domain becomes empty
```

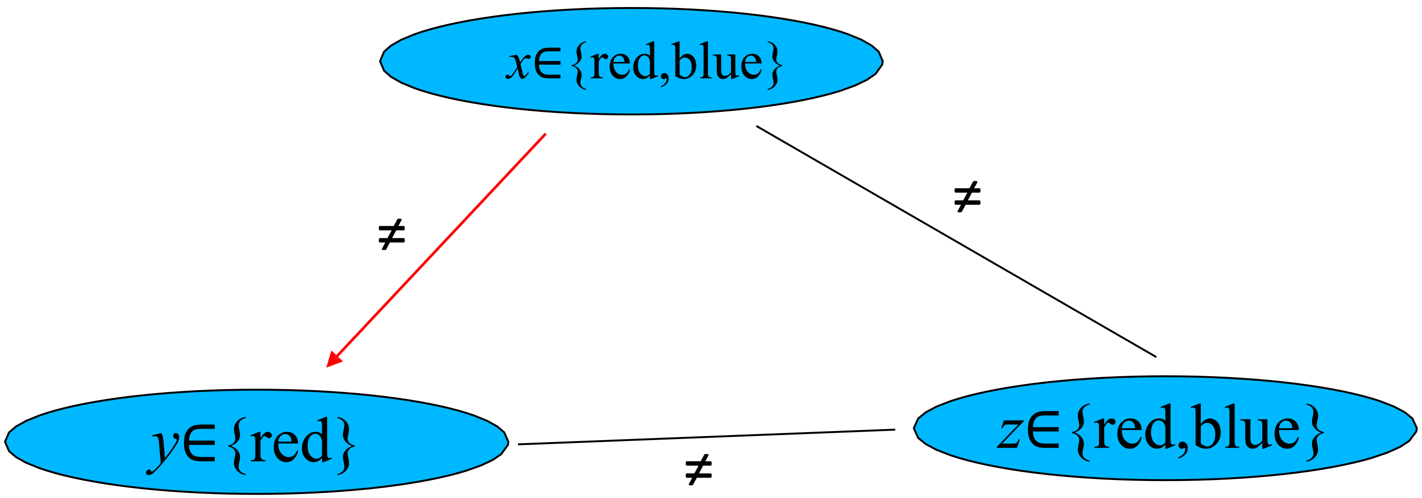
Example

- Consider again the following Graph Coloring Problem



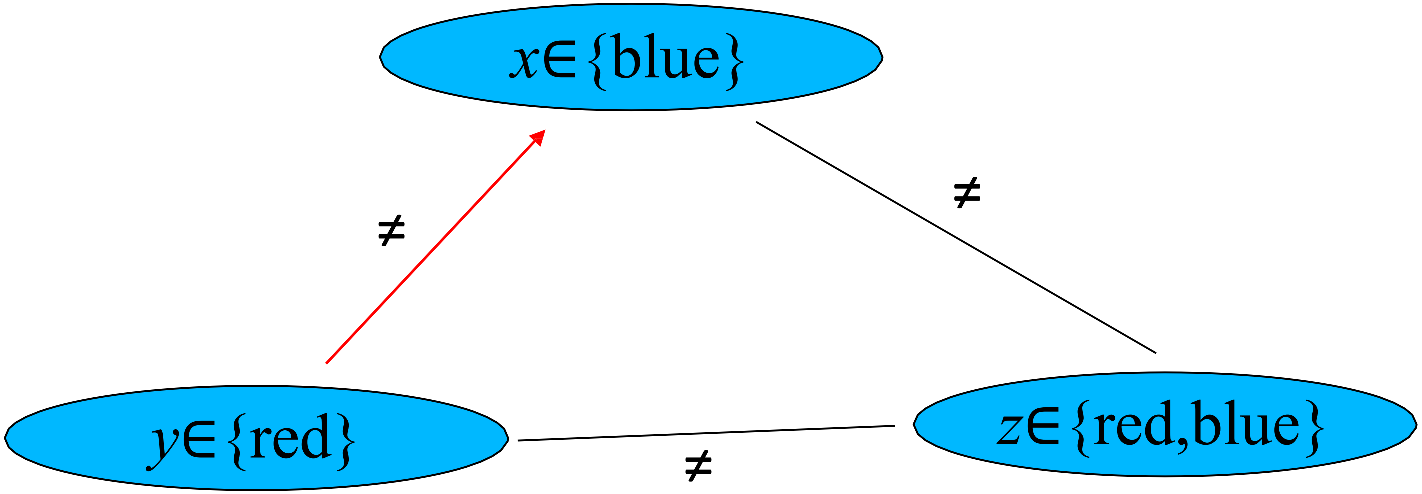
Example

- We start with P_{xy}
- We call Revise(x,y)
- For red in D_x there is no value v in D_y such that red $\neq$ v
- Hence, we delete red from D_x
- For blue in D_x, we can select red in D_y



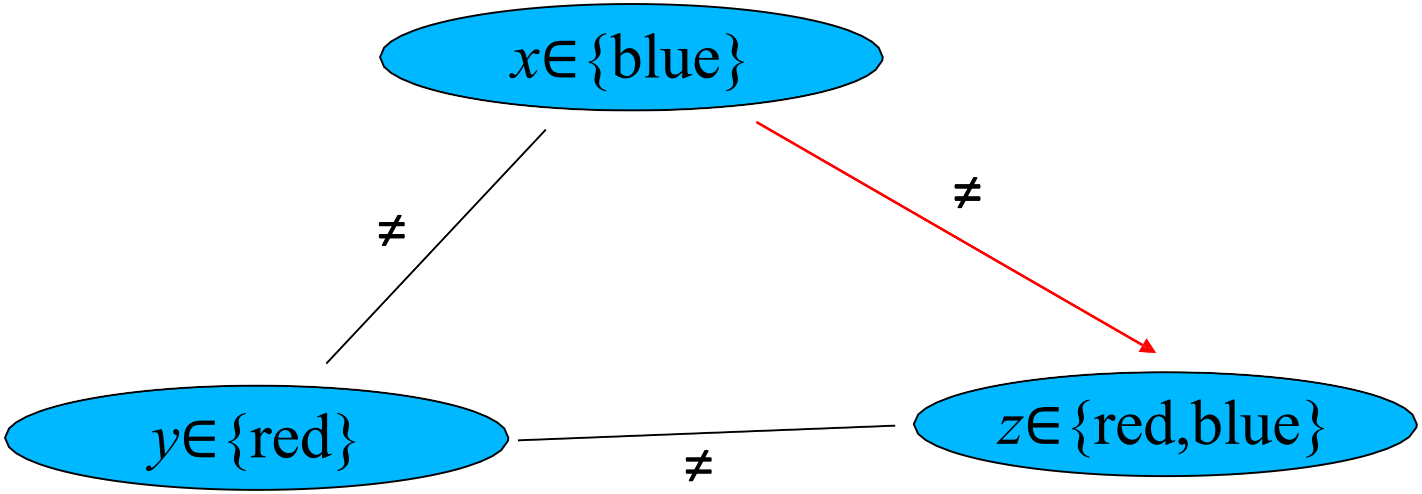
Example

- We now call Revise(y,x)
- For red in D_y, we can select blue in D_x



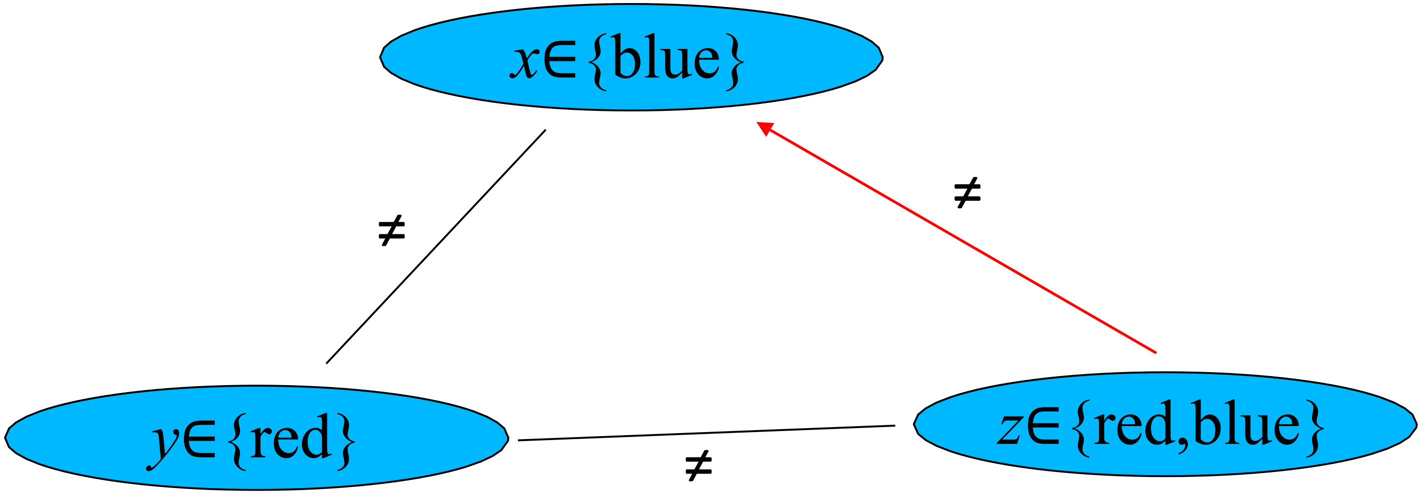
Example

- We next look at P_{xz}
- We call Revise(x,z)
- For blue in D_x, we can select red in D_z



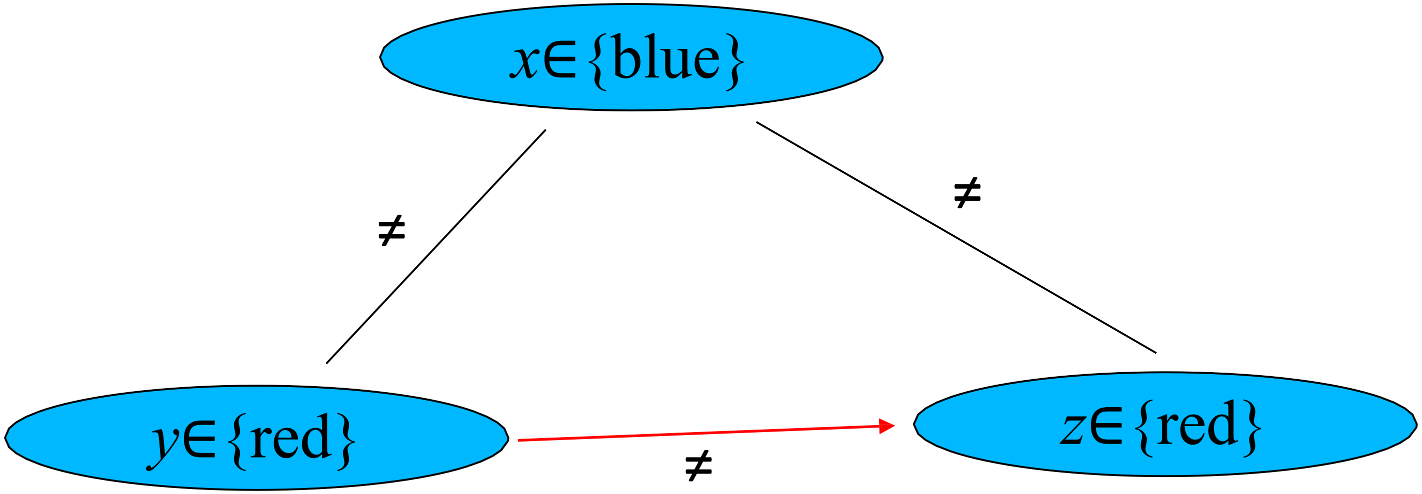
Example

- We now call Revise(z,x)
- For red in D_z, we can select blue in D_x
- For blue in D_z, there is no v in D_x such that blue $\neq$ v
- Hence, we delete blue from D_z



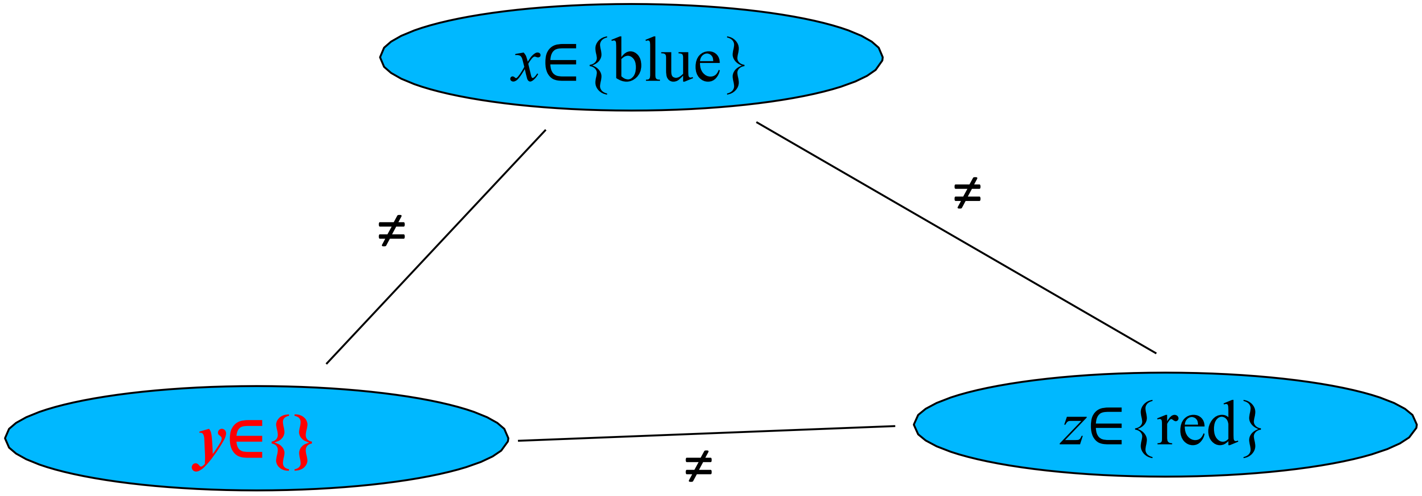
Example

- We next look at P_{yz}
- We call Revise(y,z)
- For red in D_y, there is no v in D_z such that red $\neq$ v
- Hence, we delete red from D_y



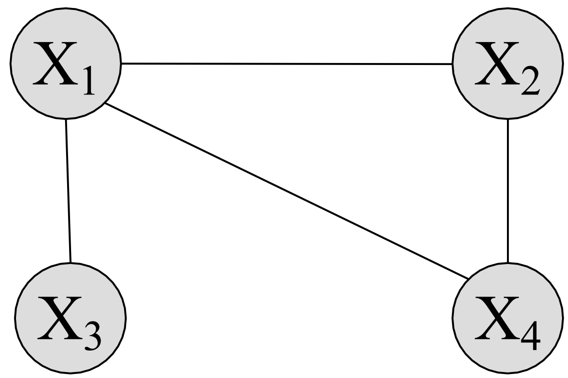
Example

- The domain of y is now empty
- Therefore, we know that the CSP is inconsistent and can stop



- Note: we found inconsistency without assigning any values

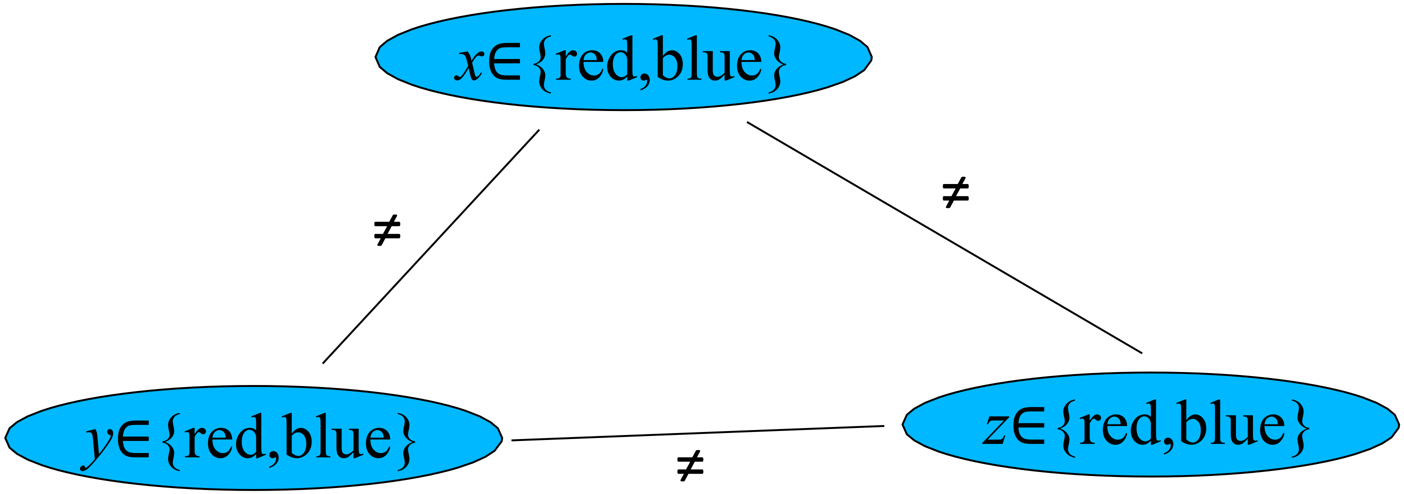
Consistency Concepts and Consistency



- If we get an empty domain, while establishing node consistency or arc consistency, our CSP must be inconsistent
- However, if a CSP is node consistent and arc consistent and has non-empty domains it is not necessarily (globally) consistent

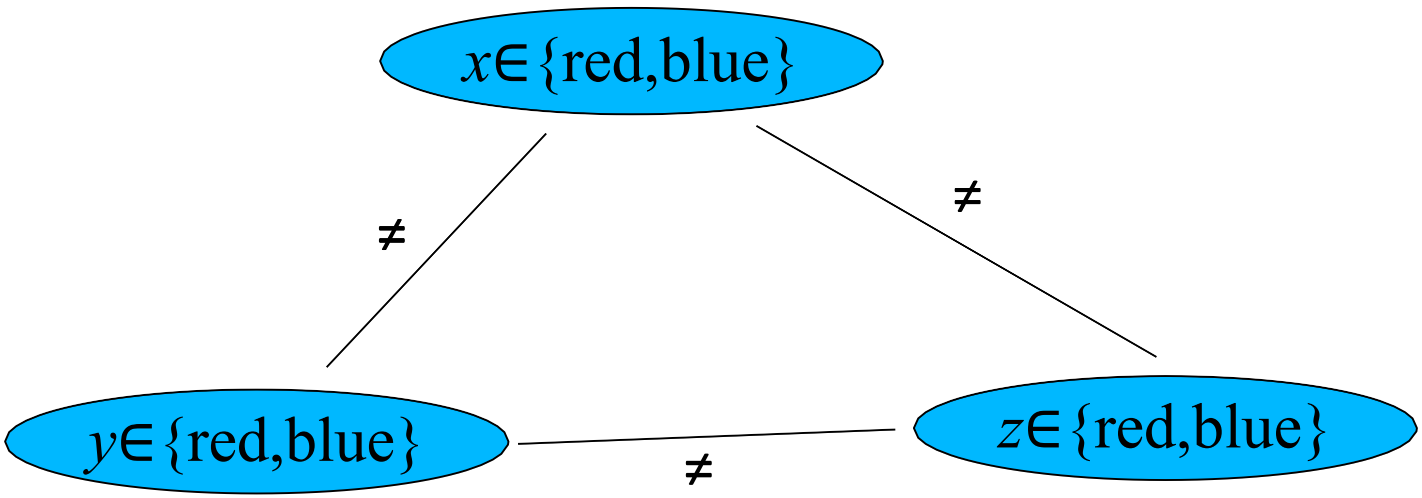
Example

- The following CSP is both node consistent and arc consistent



- However, since there are only two values and the variables must take pairwise distinct values, the CSP is inconsistent
- Hence, node consistency and arc consistency are not sufficient for testing consistency

Path Consistency

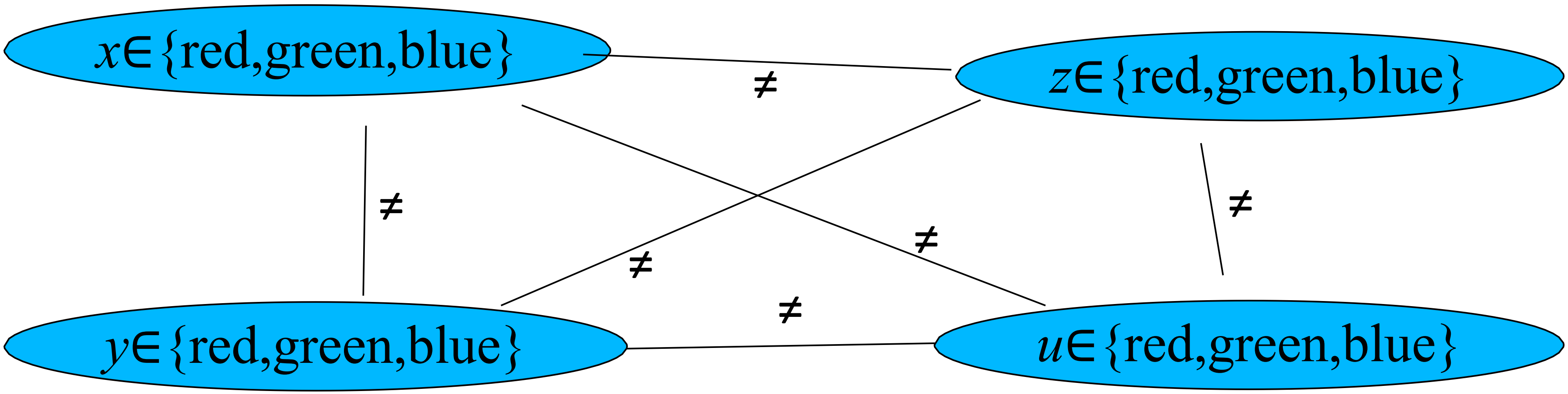


- Nodes i and j are path consistent with node m iff:

  For every arc consistent assignment of i and j, there is an assignment to m that is arc consistent with both i and j

- x and y are not path consistent with z
  - (x=red, y= blue) is arc-consistent assignment. However, z=red is not arc-consistent with x and z=blue is not arc-consistent with y

The following CSP is node, arc and path consistent



- However, since there are only three values and the variables must take pairwise distinct values, the CSP is inconsistent

k-Consistency and Strong k-Consistency

k-consistency

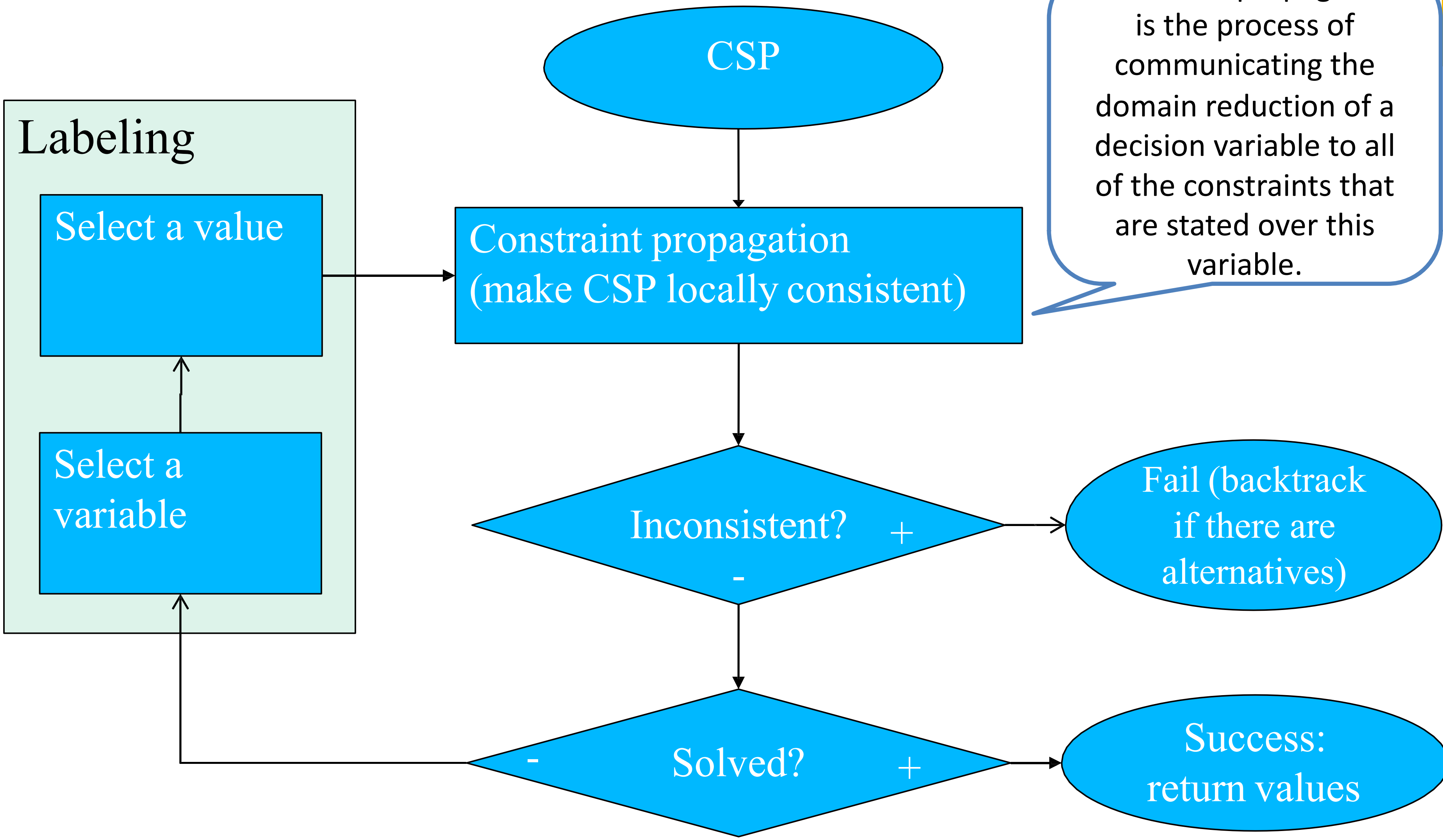
Every consistent (k-1)-assignment can be extended to a consistent k-assignment

Intuitively: (1-consistency = node consistency); (2-consistency = arc consistency) and (3-consistency = path consistency) provided node consistency is guaranteed

Strong *k-consistency*

The problem is *j-consistent* for all $j \leq k$

Solving CSPs with Backtracking Search



Constraint propagation is the process of communicating the domain reduction of a decision variable to all of the constraints that are stated over this variable.

Local Search for CSPs

Exercise: CSPs and Local Search

- Give a local search formulation for a general CSP
- Given variables with corresponding domains and constraints, define
  - State space
  - Neighborhood
  - Objective function

Solution: CSPs and Local Search

- State space: each complete variable assignment is a state
- Neighborhood: neighbors of a state (variable assignment) are obtained by changing the assignment of an arbitrary variable
- Hence, we have one neighbor for each
  - Variable and
  - each value in the domain of the variable (other than the current one)
- Objective function: value of state (variable assignment) is defined as
  - Number of constraints that are satisfied in state

Exercise: Genetic Algorithm for CSPs

- define a genetic algorithm for a general CSP
- Given variables with corresponding domains and constraints, define
  - Genes
  - Chromosomes
  - Fitness (value) of individuals
  - Mutation operation
- Illustrate reproduction by means of a small example

Solution: Genetic Algorithm for CSPs

- Genes: domain elements of all variables
- Chromosomes correspond to variable assignments
- Fitness: number of constraints that are satisfied
- Mutation operation: change a random gene
- Illustration: see Genetic Algorithms slides (8Queens, Traveling Salesman Problem)

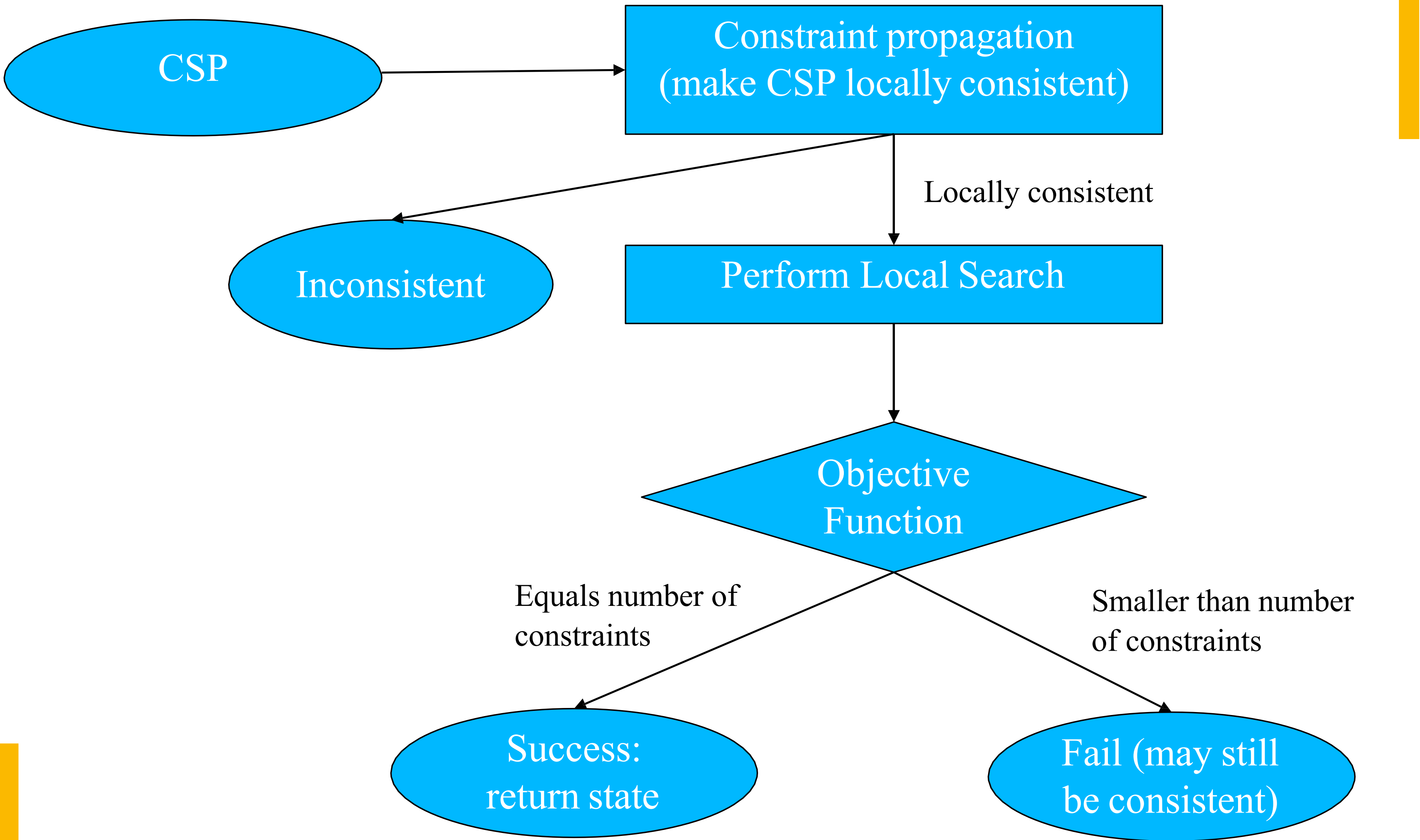
Exercise: Local Search Solution

- What can you conclude if local search algorithm finds
  - Globally optimal solution? (f(s) = number of constraints)
  - Locally optimal solution? (f(s) < number of constraints)

Solution: Local Search Solution

- Globally optimal solution
  - globally optimal solution satisfies all constraints by definition
  - Hence, we found a solution for the CSP (CSP is consistent)
- Locally optimal solution
  - Locally optimal solution violates some constraints
  - The CSP may be inconsistent, but we may also just got stuck in a local optimum
  - We cannot conclude anything in this case
  - However, solution may be sufficient (soft constraints)

Solving CSPs with Local Search



Summary

Summary: CSPs

- A constraint satisfaction problem (CSP) is given by
  - A set of *variables* $\{X_1,\ldots,X_n\}$
  - a *domain* D_i for each variable(the possible values), where the whole space $D = D_1 \times \ldots \times D_n$ is the assignment space
  - A set of *constraints*, i.e. relations $R_k \subseteq D_{k1} \times \ldots \times D_{km}$ for some domains $D_{k1}, \ldots, D_{km}$
- The goal is to find an assignment that satisfies all constraints
- If no such assignment exists, the CSP is called inconsistent (and consistent otherwise)

Summary: Algorithms

- Naïve Approach
  - Generate the whole search tree and test
  - usually not practical because of tree size
- Backtracking Search
  - start with empty assignment
  - Systematically extend assignment using heuristics
- Local Search
  - Move through space of complete variable assignment
  - Maximize number of satisfied constraints
- Consistency Concepts
  - can be applied to simplify problem initially (preprocessing)
  - can be applied to simplify problem during backtracking search

UNIVERSITÄT OSNABRÜCK

Summary: Consistency Concepts

- Node consistency: $\forall x : x \in D_i \rightarrow P_i(x)$
- Arc consistency: $\forall x : x \in D_i \rightarrow (\exists y \in D_j : P_{i,j}\ (x,y))$
- Path consistency:
  $\forall x \in D_i,\ z \in D_j : P_{i,j}(x,z) \rightarrow (\exists y \in D_m : P_{i,m}(x,y) \wedge P_{m,j}(y,z))$
- *k*-consistency
  - Any solution for *k*-1 variables can be extended to a solution for *k* variables
- Strong *k*-consistency
  - The problem is *j*-consistent for all $j \leq k$

Further Readings

The presented slides are manily Dr. Tobias Thelen slides

Most topics of this week can be found in:

Russell, S., Norvig, P. Artificial Intelligence - A modern approach. Pearson Education: 2010.

More details on representing and solving CSPs can be found in:

Dechter, R. Constraint processing. Morgan Kaufmann: 2003.

Rossi, F., Van Beek, P., & Walsh, T. Handbook of constraint programming. Elsevier: 2006.