

# Wissensrepräsentation

—

Christopher Habel, Hedda Schmidtké

Sommersemester 2004

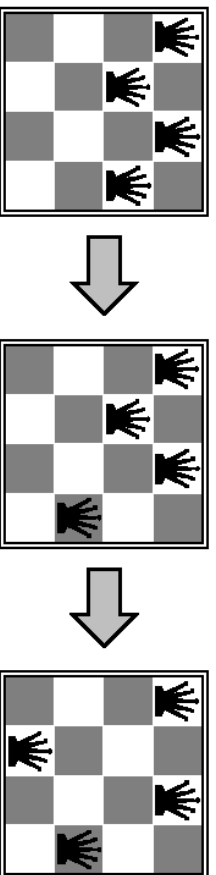
## Sitzung 10: Constraintverarbeitung

- Problemstellung

### Example: $n$ -queens

Put  $n$  queens on an  $n \times n$  board with no two queens on the same row, column, or diagonal

Move a queen to reduce number of conflicts



### Constraint satisfaction problems (CSPs)

Standard search problem:

state is a “black box”—any old data structure that supports goal test, eval, successor

CSP:

state is defined by *variables*  $X_i$  with *values* from *domain*  $D_i$

goal test is a set of *constraints* specifying allowable combinations of values for subsets of variables

Simple example of a *formal representation language*

Allows useful *general-purpose* algorithms with more power than standard search algorithms

Wissensrepräsentation, SoSe 2004

Ch. Habel / C. Eschenbach / H. Schmidtké

10 – 2

Russell & Norvig - ch.4

### Example: Map-Coloring



Variables  $WA, NT, Q, NSW, V, SA, T$

Domains  $D_i = \{red, green, blue\}$

Constraints: adjacent regions must have different colors

e.g.,  $WA \neq NT$  (if the language allows this), or

$(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), \dots\}$

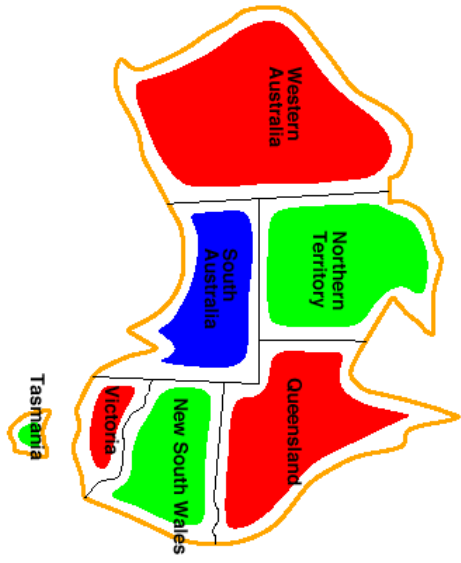
Wissensrepräsentation, SoSe 2004

Ch. Habel / C. Eschenbach / H. Schmidtké

10 – 4

Russell & Norvig - ch.4

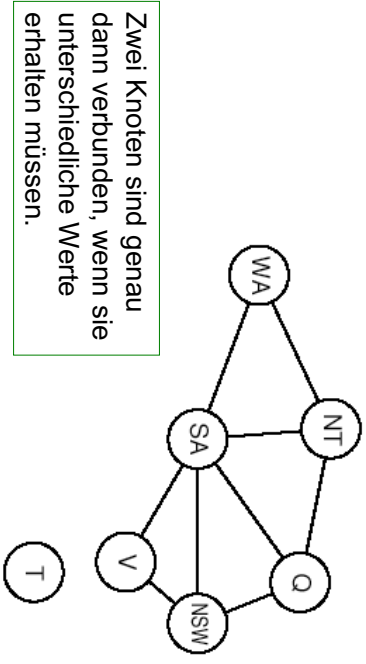
# Example: Map-Coloring contd.



Solutions are assignments satisfying all constraints, e.g.,  
 $\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$

# Constraint graph

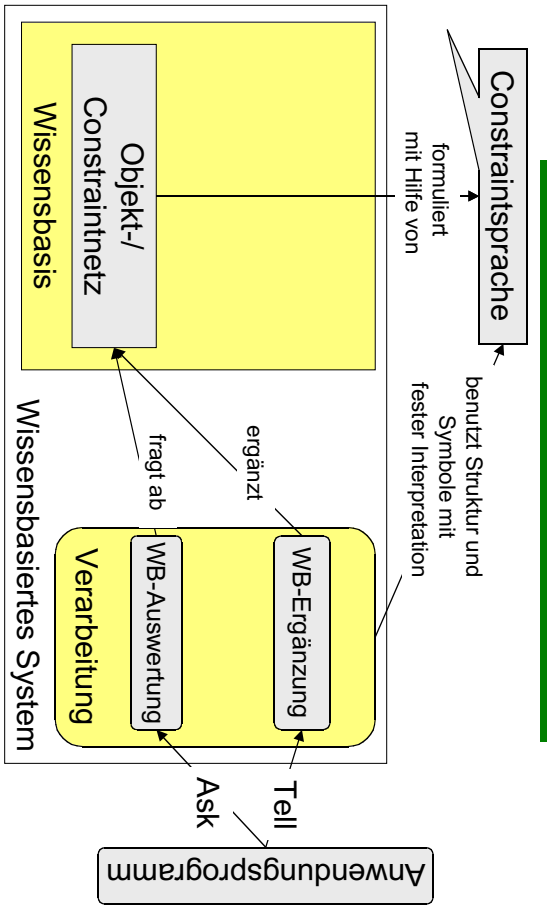
**Binary CSP:** each constraint relates at most two variables  
**Constraint graph:** nodes are variables, arcs show constraints



Zwei Knoten sind genau dann verbunden, wenn sie unterschiedliche Werte erhalten müssen.

General-purpose CSP algorithms use the graph structure to speed up search. E.g., Tasmania is an independent subproblem!

# Wissensbasiertes System mit Constraint

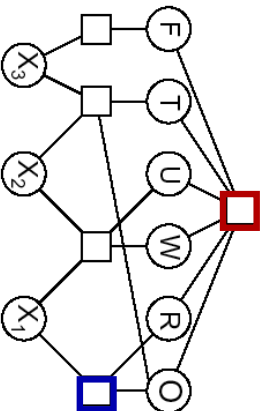


# Varieties of constraints

- Unary constraints** involve a single variable, e.g.,  $SA \neq green$  Wertebereichsangaben
- Binary constraints** involve pairs of variables, e.g.,  $SA \neq WA$  hard constraints
- Higher-order constraints** involve 3 or more variables, e.g., cryptarithmic column constraints
- Preferences** (soft constraints), e.g., *red* is better than *green* often representable by a cost for each variable assignment  
 → constrained optimization problems

## Example: Cryptarithmic

$$\begin{array}{r} T W O \\ + T W O \\ \hline F O U R \end{array}$$



Variables:  $F, T, U, W, R, O, X_1, X_2, X_3$

Domains:  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Constraints

$\text{allDiff}(F, T, U, W, R, O)$

$O + O = R + 10 \cdot X_1$  etc.

## Real-world CSPs

Assignment problems

e.g., who teaches what class

Timetabling problems

e.g., which class is offered when and where?

Hardware configuration

Spreadsheets

Transportation scheduling

Factory scheduling

Floorplanning

Notice that many real-world problems involve real-valued variables

## Varieties of CSPs

Discrete variables

finite domains: size  $d \Rightarrow O(d^n)$  complete assignments

◇ e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)

infinite domains (integers, strings, etc.)

◇ e.g., job scheduling, variables are start/end days for each job

◇ need a **constraint language**, e.g.,  $StartJob_1 + 5 \leq StartJob_2$

◇ **linear** constraints solvable, **nonlinear** undecidable

Continuous variables

◇ e.g., start/end times for Hubble Telescope observations

◇ **linear** constraints solvable in poly time by LP methods

## Standard search formulation (incremental)

Let's start with the straightforward, dumb approach, then fix it

States are defined by the values assigned so far

◇ **Initial state**: the empty assignment,  $\{\}$

◇ **Successor function**: assign a value to an unassigned variable that does not conflict with current assignment.  
 $\Rightarrow$  fail if no legal assignments (not fixable!)

◇ **Goal test**: the current assignment is complete

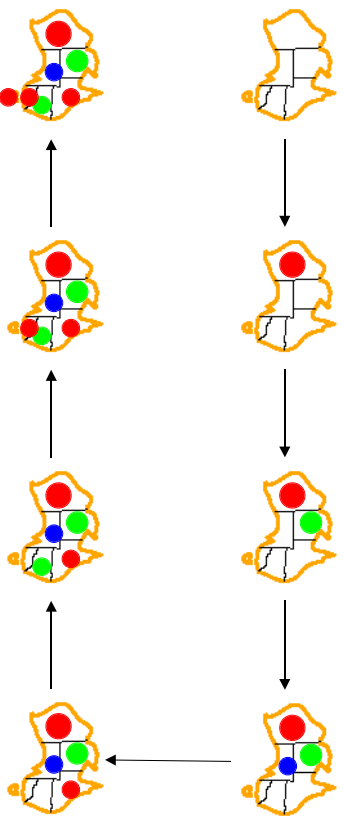
1) This is the same for all CSPs!

2) Every solution appears at depth  $n$  with  $n$  variables  
 $\Rightarrow$  use depth-first search

3) Path is irrelevant, so can also use complete-state formulation

4)  $b = (n - \ell)d$  at depth  $\ell$ , hence  $n!d^n$  leaves!!!!

## Map coloring: incremental search (1)



### Backtracking search

Variable assignments are commutative, i.e.,

$[WA = red \text{ then } NT = green]$  same as  $[NT = green \text{ then } WA = red]$

Only need to consider assignments to a single variable at each node

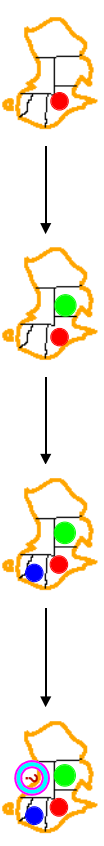
$\Rightarrow b = d$  and there are  $d^n$  leaves

Depth-first search for CSPs with single-variable assignments is called **backtracking** search

Backtracking search is the basic uninformed algorithm for CSPs

Can solve  $n$ -queens for  $n \approx 25$

## Map coloring: incremental search (2)



- Inkrementelle Suche erfordert
- **backtracking**
- und / oder
- Heuristiken / Strategien

### Backtracking search

```

function BACKTRACKING-SEARCH(csp) returns solution/failure
  return RECURSIVE-BACKTRACKING([], csp)

function RECURSIVE-BACKTRACKING(assigned, csp) returns solution/failure
  if assigned is complete then return assigned
  var ← SELECT-UNASSIGNED-VARIABLE(VARIALES[csp], assigned, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assigned, csp) do
    if value is consistent with assigned according to CONSTRAINTS[csp] then
      result ← RECURSIVE-BACKTRACKING(var = value | assigned[], csp)
      if result ≠ failure then return result
  end
  return failure
  
```

# Backtracking example

