Suche und Constraints

Eliminate all other factors,
and the one which remains,
must be the truth.

A. Conan Doyle

Constraints sind ein Wissensrepräsentationsformat

- für das es spezielle Suchverfahren gibt
- das zur „Vorbehandlung von Suchbereichen“ für allgemeine Suchverfahren verwendet werden kann

Definitionen zu Constraints

Ausgangspunkt: **Variablen** $\{X_1, \ldots, X_n\}$, je mit Werten aus **Wertebereichen** (*domains*) D_i $(i=1, \ldots, n)$

Ein k-stelliger **Constraint** C besteht aus einer Menge $\{X_1, \ldots, X_k\}$ von Variablen und einer k-stelligen Relation R_C auf den Variablen.

k Werte $\langle v_1, \ldots, v_k \rangle$ **erfüllen** C, wenn $v_i \in D_i$ und $\langle v_1, \ldots, v_k \rangle \in R_C$

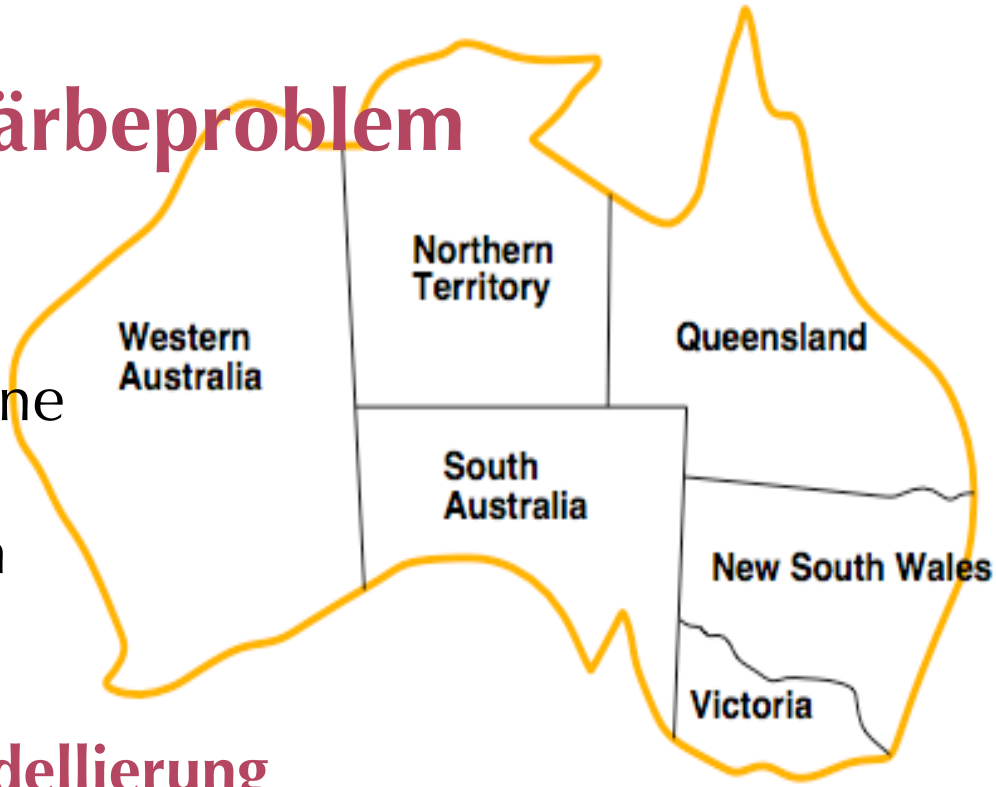
Ein **Constraintnetz** (*Constraint Satisfaction Problem*, CSP) besteht aus einer Menge $\{X_1, \ldots, X_n\}$ von Variablen und einer Menge $\{C_1, \ldots, C_m\}$ von Constraints auf Teilmengen dieser Variablen.

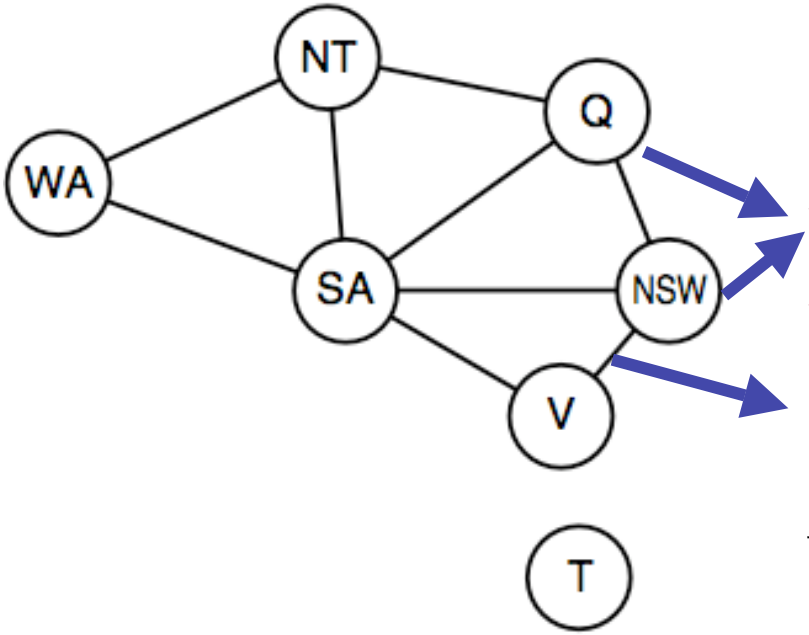
Eine **Lösung** eines CSP ist eine Belegung $\{X_1 = v_1, \ldots, X_n = v_n\}$, sodass $v_i \in D_i$ und alle Constraints aus $\{C_1, \ldots, C_m\}$ erfüllt sind.

Beispiel: Färbeproblem

Problem

Färbe die Regionen einer Karte mit drei Farben so ein, dass keine zwei aneinandergrenzende Regionen dieselbe Farbe haben





Modellierung

Variablen entspr. Regionen;

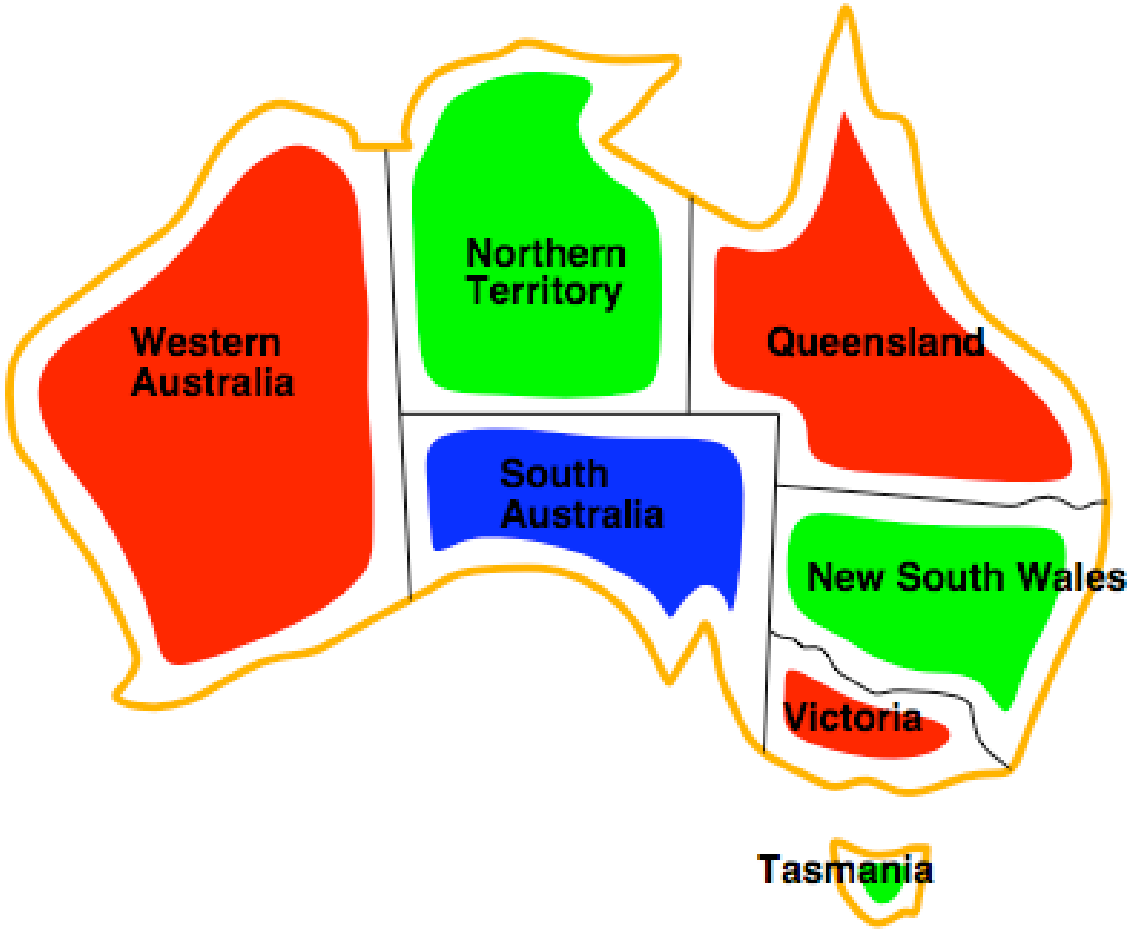
Wertebereiche jeweils {R, G, B};

Relation „ungleiche_Farbe“ zwischen je 2 angrenzenden Regionen

(Hier als Kante zwischen Variablen skizziert)

Eine Färbung Australiens

ungleiche_Farbe={⟨R,G⟩, ⟨R,B⟩, ⟨G,R⟩, ⟨G,B⟩, ⟨B,R⟩, ⟨B,G⟩}



Lösung

{WA=R,
NT=G,
Q=R,
NSW=G,
V=R,
SA=B,
T=G }

Klassen von Constraintproblemen

Diskrete vs. kontinuierliche Wertebereiche

- diskrete endliche Wertebereiche
  $\Rightarrow O(d^n)$ Belegungen bei max. Wertebereichgröße d
  - z.B. Färben, Boolesche Erfüllbarkeit (einschl. 3SAT)
- diskrete unendliche Wertebereiche
  - z.B. Scheduling-Probleme über Zeit*einheiten*
- kontinuierlich
  z.B. Scheduling-Probleme (Simple Temporal Networks, STNs)

Unäre vs. Binäre vs. k-äre CSPs

- binär z.B. Färbeprobleme; „universelle“ Darstellung!

Strikte vs. Präferenz-CSPs

- Relationen (z.B. Färbeproblem) oder Nutzen/Strafwerte

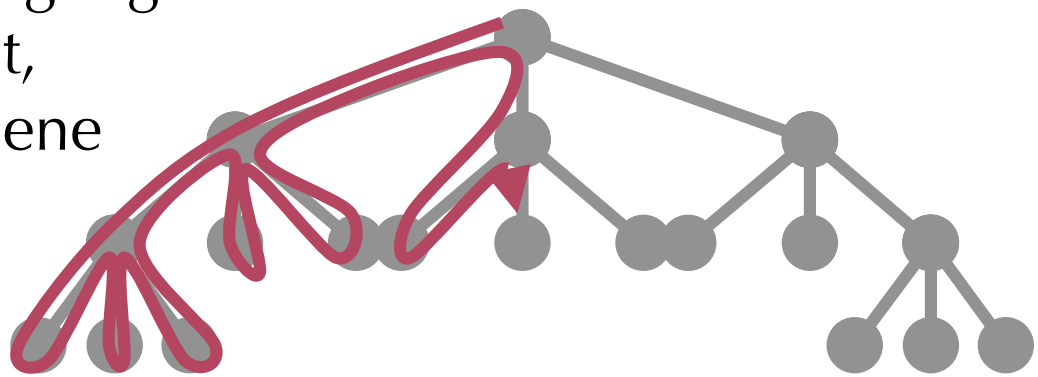
Backtracking zum Lösen von CSPs

Uninformierter Basisalgorithmus für diskrete endliche CSPs

Entsprechend Backtracking-Suche (Folie Nr.41):

Starte mit leerer Variablenbelegung und durchlaufe:

1. Wähle für einzelne(!) Variable nacheinander Zuweisungen, sodass deren Constraints erfüllt sind;
2. wenn alle Variablen belegt, Lösung gefunden!
3. wenn keine mögliche Belegung aktuellen Constraint erfüllt, nimm zuletzt vorgenommene Belegung zurück

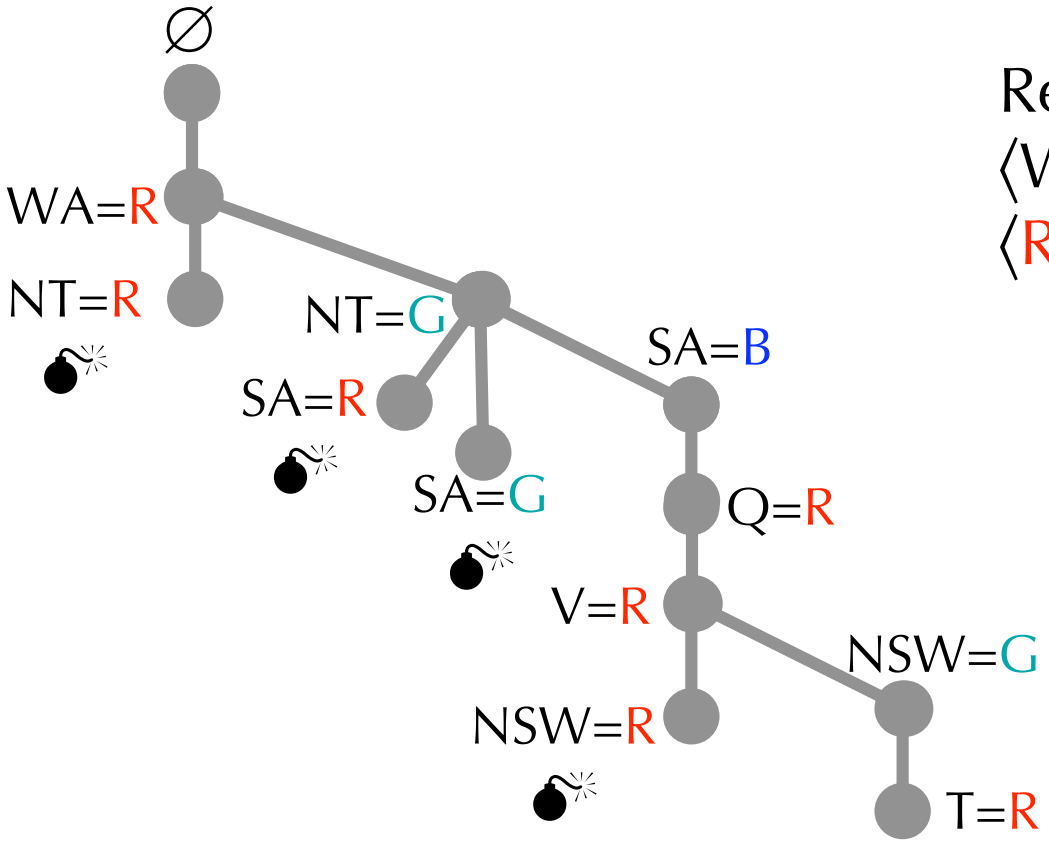


Ein Backtracking-CSP-Algorithmus

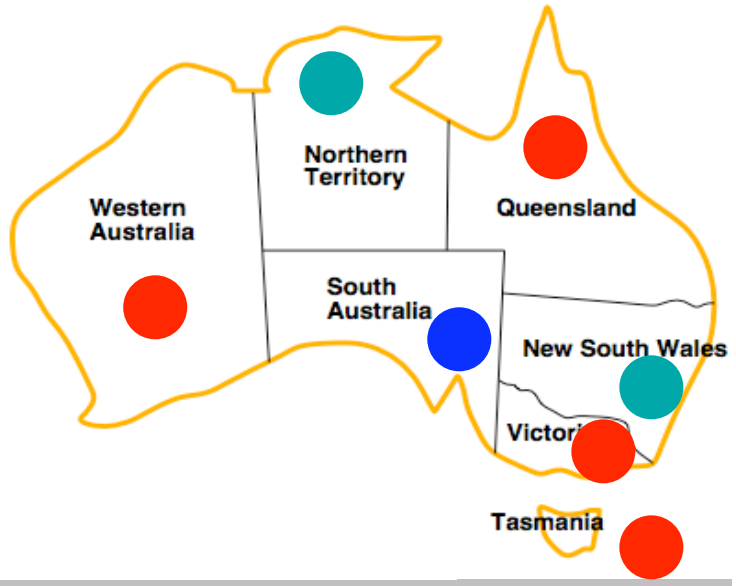
```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING([], csp)

function RECURSIVE-BACKTRACKING(assigned, csp) returns solution/failure
    if assigned is complete then return assigned
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[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
```

Beispiel: Australien per Backtracking

Wichtige Entscheidung: Reihenfolge, in der Variablen und ihre Werte versucht werden!



Reihenfolge hier:
⟨WA, NT, SA, Q, V, NSW, T⟩;
⟨R, G, B⟩ für alle Variablen



Eigenschaften des Backtracking-CSP-Lösers

☺ Vollständig für diskrete endliche Wertebereiche

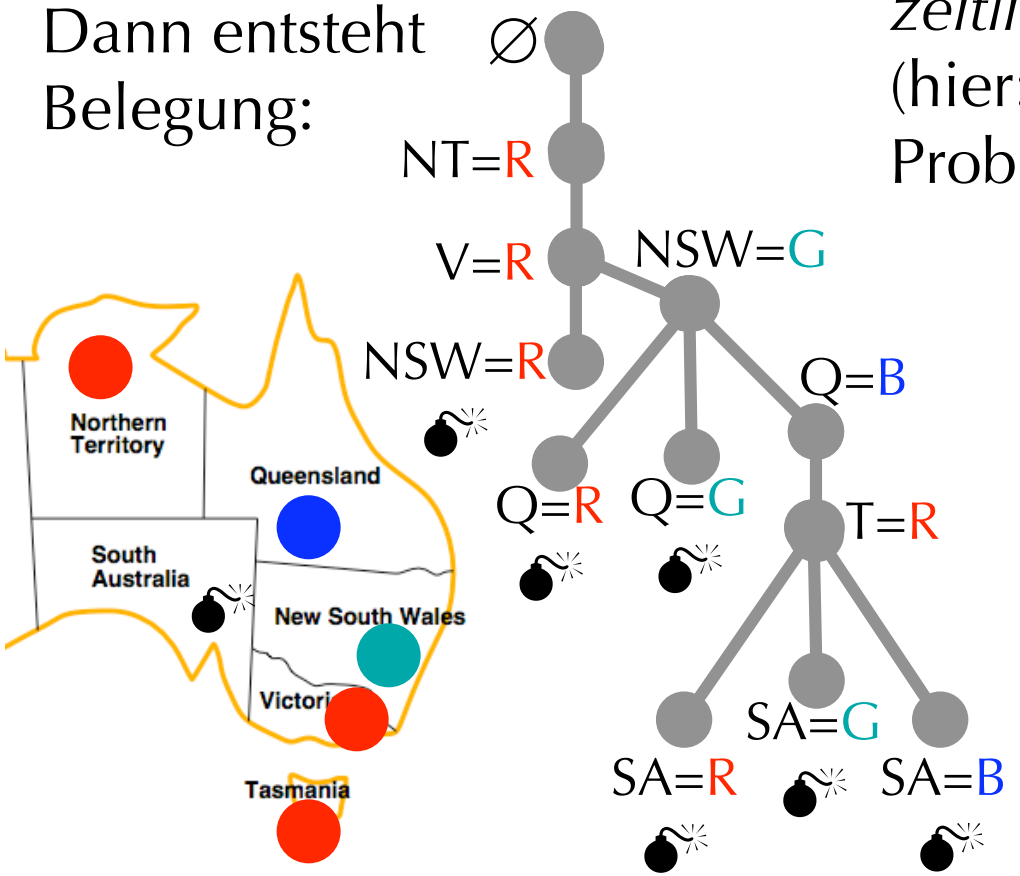
😐 Laufzeit $O(d^n)$ bei n Variablen und max. Wertebereichgröße d

☺ Speicher: Constraintnetz der Größe $O(n^2 d^2)$ (Binärconstraints!)

- für reale Probleme in der Regel zu ineffizient
- geeignet als Referenzverfahren

Rückspringen (“*backjumping*”)

Sei Variablenreihenfolge
⟨NT,V,NSW,Q,T,SA,WA⟩
Dann entsteht Belegung:



Chronologisches Rücksetzen

(BACKTRACKING-SEARCH) nimmt die *zeitlich* letzte Belegung zurück (hier: T≠R), was sofort zum selben Problem mit SA führt

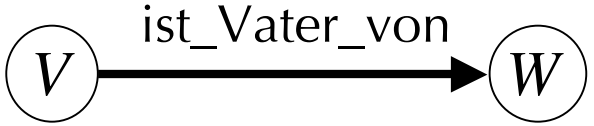
Rückspringen

nimmt die letzte Belegung einer Variablen aus der *Konfliktmenge* zurück: Menge der Variablen, die durch einen Constraint mit SA verbunden sind (hier: Q≠B ⇒ NSW≠G)

Variable Variablenreihenfolge

Beispiel: Finde einen Deutschen, der Vater eines Formel-1-Weltmeisters ist

Constraintnetz



D_V = {Holger Aalderink, … , Hella Zuther}: alle Deutschen, $|D_V| \approx 80$ Mio

D_W = {Giuseppe Farina, Juan Manuel Fangio, …, Jacques Villeneuve,
Mika Häkinnen, Michael Schumacher} $|D_W| \approx 20$

ist_Vater_von(X,Y) = {(Aaron Aanderud, Tomas Aanderud), …,
(Anton Abel, Susi Abel),…, (Johann S. Bach, Friedemann Bach), …
(Rolf Schumacher, Michael Schumacher),…
(Gilles Villeneuve, Jacques Villeneuve)…}

Heuristik der ***Minimum Remaining Values*** (MRV, *fail-first*)
in Funktion SELECT-UNASSIGNED-VARIABLE :

Wähle Variable mit kleinstem (aktuellem) Wertebereich!

Propagieren von Zuweisungen

Forward Checking (einfache Propagierung):

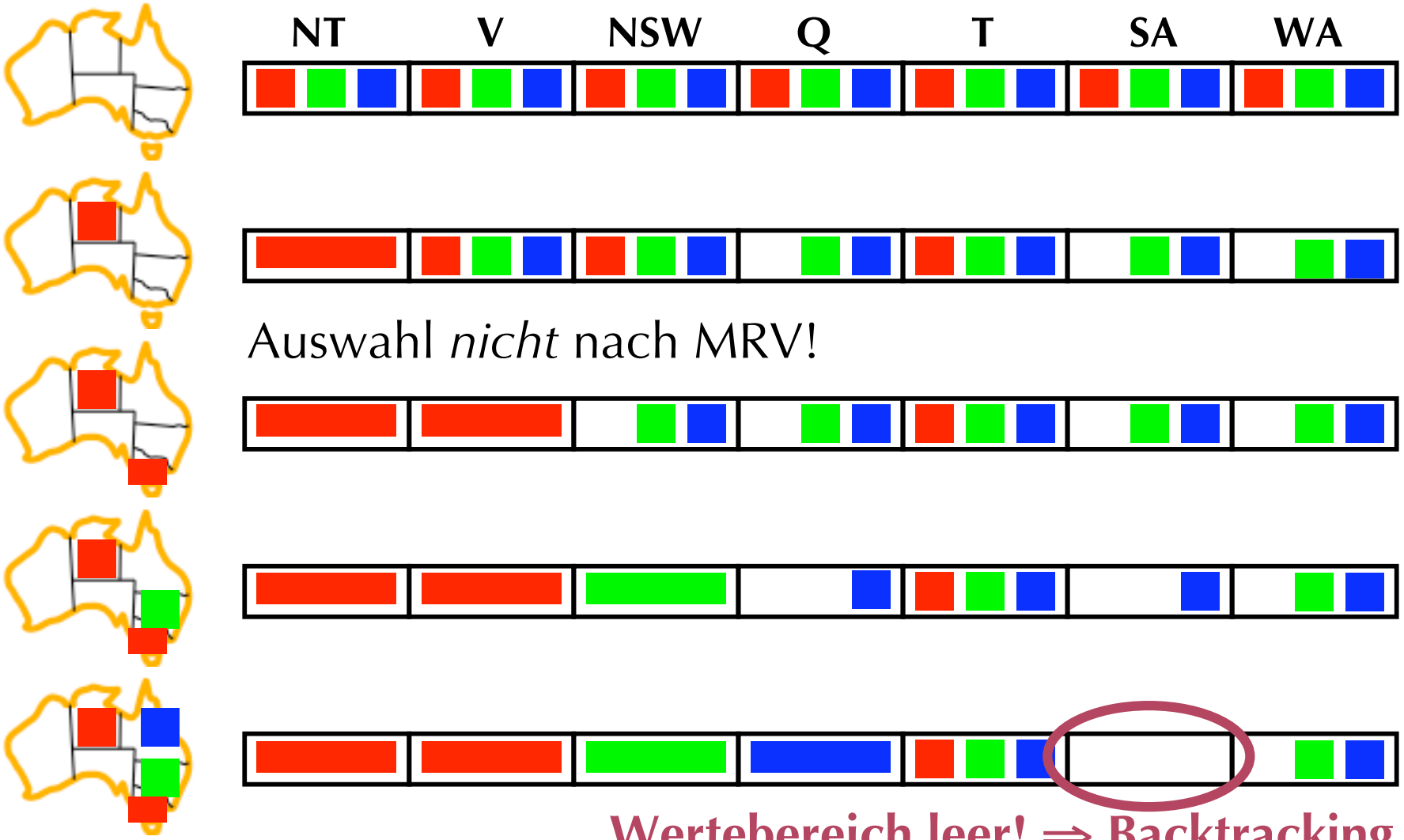
Wann immer ein Wert v an X zugewiesen wird, maskiere in allen Variablen, mit denen X durch einen Constraint C verbunden ist, alle mit v inkonsistenten Werte!

(Offensichtlich kombinierbar mit MRV;
„duales“ Verfahren zu Rückspringen)

Beispiel Formel-1-Weltmeister:

- Zuweisung „Jacques Villeneuve“ an W maskiert *alle* Werte in V
- Zuweisung „Michael Schumacher“ an W maskiert alle Werte in V außer „Rolf Schumacher“

Forward Checking in Australien



Auswahl *nicht* nach MRV!

Wertebereich leer! ⇒ Backtracking

Lokale Konsistenz

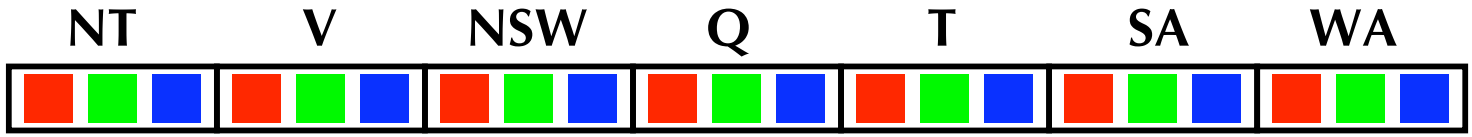
Seien X, Y Variablen, zwischen denen der k-stellige Constraint C gilt. Die **Kante** von X über C nach Y ist **konsistent**, gdw. es für jede mögliche Belegung x von X einen Wert y von Y und ein k-Tupel $T \in R_C$ gibt, sodass (x,y) die Projektion von T auf (X,Y) ist.

„Für jeden Wert x gibt es einen erlaubten Wert y“

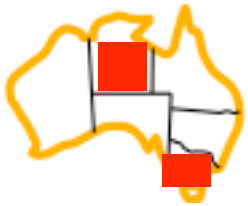
Ein Constraintnetz über den Variablen $\{X_1, \ldots, X_n\}$ ist **lokal konsistent** (**kantenkonsistent**), gdw. es bezüglich aller seiner Kanten konsistent ist und für alle X_i mit Wertebereich D_i gilt $D_i \neq \emptyset$.

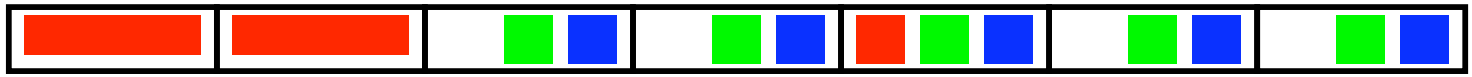
„Das Netz enthält lokal plausible Werte, und nur diese“

Beispiel im Färbeproblem

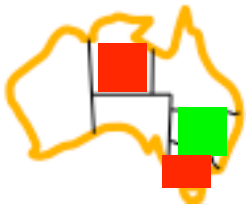


lokal konsistent!





lokal konsistent!



nicht lokal konsistent!

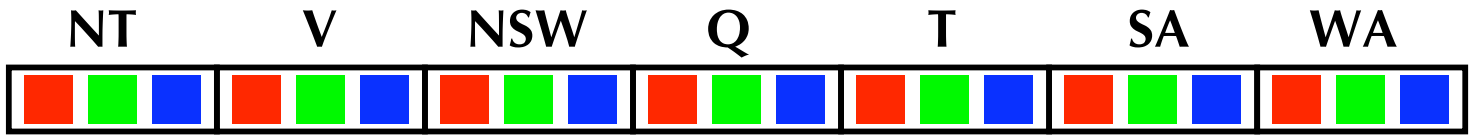
Q und SA können nicht beide blau sein
⇒ Kante zwischen Q und SA nicht konsistent

Der Algorithmus AC-3

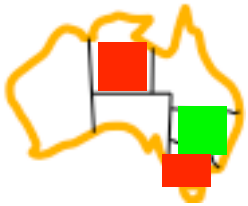
```
function AC3( csp) returns the CSP, possibly with reduced domains
   local variables: queue, a queue of arcs, initially all the arcs in csp

   loop while queue is not empty do
      (X_i, X_j) ← REMOVE-FRONT(queue)
      if REMOVE-INCONSISTENT(X_i, X_j) then
         for each X_k in NEIGHBORS[X_i] do
            add (X_k, X_i) to queue
```

```
function REMOVE-INCONSISTENT( X_i, X_j) returns true iff we remove a value
   removed ← false
   loop for each x in DOMAIN[X_i] do
      if no y in DOMAIN[X_j] allows (x,y) to fulfill the constraint between X_i, X_j
         then delete x from DOMAIN[X_i]; removed ← true
   return removed
```

Beispiel: AC-3 in Australien



… bleibt wie es ist!



… ergibt:

Eigenschaften des AC-3-Algorithmus

☺ Speicher: binäres Constraintnetz mit $O(n^2d^2)$
(n Variable, max. d Werte)

😐 Zeitbedarf für binäre Constraints: $O(n^2d^3)$
(für $O(n^2)$ Constraints packe max. d Mal d^2 Wertetupel an)

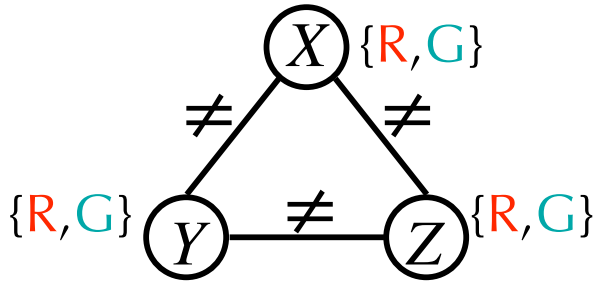
- Zeitbedarf verbesserbar auf $O(n^2d^2)$
(Buchführung, welche Löschungen Nachbarvariablenwerte beeinträchtigen)
- Reduktion der Komplexität bei Zyklenfreiheit (z.B. Baum) und unzusammenhängenden Teilnetzen
(im Australienbeispiel: Tasmanien unabhängig vom Rest färbbar)
- AC-3 terminiert unabhängig von der Sortierung in *queue* mit lokal konsistenter Belegung oder mit Ø für alle Variablen eines maximalen zusammenhängenden Teilnetzes

Lokale Konsistenz und Lösbarkeit

Polynomielle Laufzeit lässt vermuten, dass AC-3 im Allgemeinen Suchprobleme nicht *lösen* kann (geht nur exponenziell!):

Nicht alle Belegungen im lokal konsistenten Netz sind Lösungen!
Beispiel: Australien-Färbeproblem mit {R,G,B} für alle Variablen lok. konsistent

Es gibt lokal konsistente Netze, die keine Lösung haben!
Beispiel:



Schärfere Konsistenzbegriffe erweitern den „Horizont“ der lokalen Konsistenz: k-Konsistenz, strenge k-Konsistenz