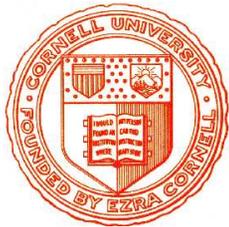


Automatisierte Logik und Programmierung

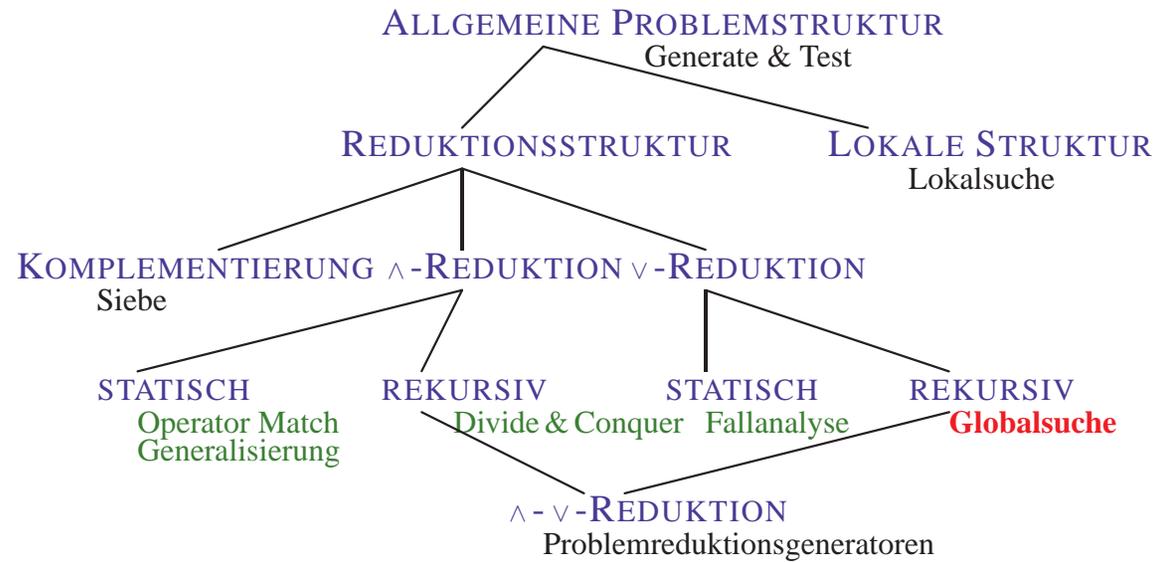
Einheit 19

Globalsuch-Algorithmen

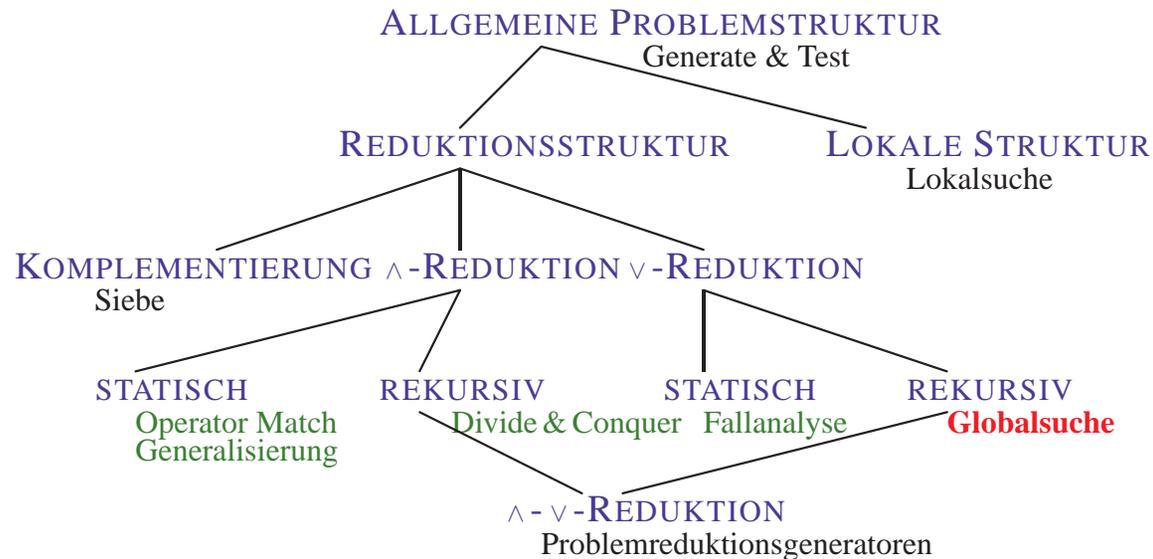


1. Algorithmenschema
2. Korrektheit
3. Wissensbasierte Unterstützung
4. Synthesestrategie

GLOBALSUCHALGORITHMEN



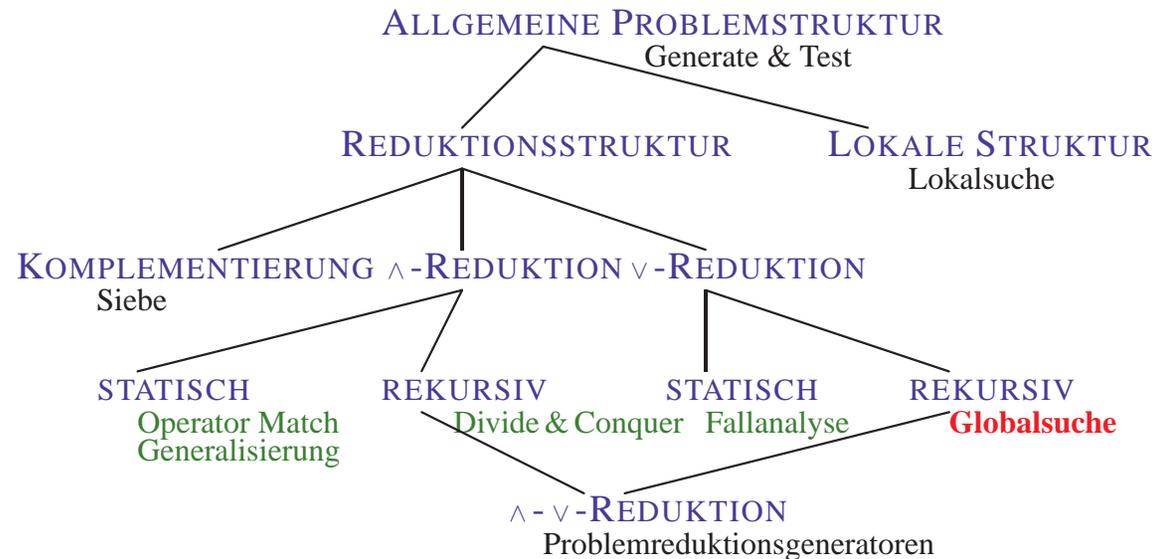
GLOBALSUCHALGORITHMEN



- **Bestimmung aller Lösungen eines Problems**

- Aufzählen von Kandidaten
- Eliminieren von Kandidaten, die keine Lösungen darstellen
- Verallgemeinert Backtracking, Binärsuche, Branch & Bound ...

GLOBALSUCHALGORITHMEN



- **Bestimmung aller Lösungen eines Problems**
 - Aufzählen von Kandidaten
 - Eliminieren von Kandidaten, die keine Lösungen darstellen
 - Verallgemeinert Backtracking, Binärsuche, Branch & Bound ...
- **Rekursive \vee -Reduktion des Problems**
 - Gesamtlösung ist Vereinigung unabhängiger Teillösungen
 - Gut geeignet für Parallelverarbeitung
 - Auch für sequentielle Suche nach erstmöglicher Lösung geeignet

EIN TYPISCHER GLOBALSUCHALGORITHMUS

- **Suche alle Indizes eines Wertes k in einer geordneten Liste L**

FUNCTION `osearch`($L, k : \text{Seq}(\mathbb{Z}) \times \mathbb{Z}$) WHERE $L \neq [] \wedge \text{ordered}(L)$
RETURNS $\{ i : \mathbb{N} \mid i \in \{1..|L|\} \wedge L_i = k \}$

EIN TYPISCHER GLOBALSUCHALGORITHMUS

- **Suche alle Indizes eines Wertes k in einer geordneten Liste L**

FUNCTION `osearch`($L, k : \text{Seq}(\mathbb{Z}) \times \mathbb{Z}$) WHERE $L \neq [] \wedge \text{ordered}(L)$

RETURNS $\{ i : \mathbb{N} \mid i \in \{1..|L|\} \wedge L_i = k \}$

- **Binäre Suche und Aufsammeln von Lösungen**

- Spalte Liste L in zwei Teile $[L_i \mid 1 \leq i \leq m]$ und $[L_i \mid m < i \leq |L|]$
- Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen

EIN TYPISCHER GLOBALSUCHALGORITHMUS

- **Suche alle Indizes eines Wertes k in einer geordneten Liste L**

FUNCTION `osearch`($L, k : \text{Seq}(\mathbb{Z}) \times \mathbb{Z}$) WHERE $L \neq [] \wedge \text{ordered}(L)$
RETURNS $\{ i : \mathbb{N} \mid i \in \{1..|L|\} \wedge L_i = k \}$

- **Binäre Suche und Aufsammeln von Lösungen**

- Spalte Liste L in zwei Teile $[L_i \mid 1 \leq i \leq m]$ und $[L_i \mid m < i \leq |L|]$
- Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen

- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**

- Signifikante Reduktion der zu verwaltenden Daten
- Grenzen l und r der Indexmengen sind **Repräsentanten des Suchraums**
- Repräsentanten sind nur dann **sinnvoll** wenn $1 \leq l \leq r \leq |L|$

EIN TYPISCHER GLOBALSUCHALGORITHMUS

- **Suche alle Indizes eines Wertes k in einer geordneten Liste L**

```
FUNCTION osearch( $L, k: \text{Seq}(\mathbb{Z}) \times \mathbb{Z}$ ) WHERE  $L \neq [] \wedge \text{ordered}(L)$   
  RETURNS  $\{ i: \mathbb{N} \mid i \in \{1..|L|\} \wedge L_i = k \}$ 
```

- **Binäre Suche und Aufsammeln von Lösungen**

- Spalte Liste L in zwei Teile $[L_i \mid 1 \leq i \leq m]$ und $[L_i \mid m < i \leq |L|]$
- Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen

- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**

- Signifikante Reduktion der zu verwaltenden Daten
- Grenzen l und r der Indexmengen sind Repräsentanten des Suchraums
- Repräsentanten sind nur dann sinnvoll wenn $1 \leq l \leq r \leq |L|$

- **Verwende Hilfsfunktion *aux* für rekursive Breitensuche**

```
FUNCTION aux( $L, k, l, r: \text{Seq}(\mathbb{Z}) \times \mathbb{Z} \times \mathbb{N} \times \mathbb{N}$ )  
  WHERE  $L \neq [] \wedge \text{ordered}(L) \wedge 1 \leq l \leq r \leq |L|$   
  RETURNS  $\{ i: \mathbb{N} \mid i \in \{1..r\} \wedge L_i = k \}$   
   $\equiv$  if  $l=r$  then if  $L_l=k$  then  $\{l\}$  else  $\emptyset$   
      else let  $m=(l+r)/2$  in aux( $L, k, l, m$ )  $\cup$  aux( $L, k, m+1, r$ )
```

EIN TYPISCHER GLOBALSUCHALGORITHMUS

- **Suche alle Indizes eines Wertes k in einer geordneten Liste L**

FUNCTION *osearch*($L, k: \text{Seq}(\mathbb{Z}) \times \mathbb{Z}$) WHERE $L \neq [] \wedge \text{ordered}(L)$
RETURNS $\{i: \mathbb{N} \mid i \in \{1..|L|\} \wedge L_i = k\}$

- **Binäre Suche und Aufsammeln von Lösungen**

- Spalte Liste L in zwei Teile $[L_i \mid 1 \leq i \leq m]$ und $[L_i \mid m < i \leq |L|]$
- Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen

- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**

- Signifikante Reduktion der zu verwaltenden Daten
- Grenzen l und r der Indexmengen sind Repräsentanten des Suchraums
- Repräsentanten sind nur dann sinnvoll wenn $1 \leq l \leq r \leq |L|$

- **Verwende Hilfsfunktion *aux* für rekursive Breitensuche**

FUNCTION *aux*($L, k, l, r: \text{Seq}(\mathbb{Z}) \times \mathbb{Z} \times \mathbb{N} \times \mathbb{N}$)
WHERE $L \neq [] \wedge \text{ordered}(L) \wedge 1 \leq l \leq r \leq |L|$
RETURNS $\{i: \mathbb{N} \mid i \in \{1..r\} \wedge L_i = k\}$
 \equiv if $l=r$ then if $L_l = k$ then $\{l\}$ else \emptyset
else let $m=(l+r)/2$ in *aux*(L, k, l, m) \cup *aux*($L, k, m+1, r$)

- **Initialisiere Hilfsfunktion mit gesamter Liste**

- *osearch*(L, k) \equiv *aux*($L, k, 1, |L|$)

OPTIMIERUNG DES GRUNDALGORITHMUS

- **Eliminiere Hilfsfunktionsaufrufe, wenn keine Lösung möglich**

- Suche berücksichtigt nicht, daß *Liste* geordnet ist (linearer Algorithmus)
- Suchraum $\{i \dots j\}$ ohne Lösung, falls $L_i > k$ oder $L_j < k$
- Ergänze Filter $L_i \leq k \leq L_j$ zu rekursivem Aufruf

```
FUNCTION aux(L,k,l,r:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$ )
  WHERE  $L \neq [] \wedge \text{ordered}(L) \wedge 1 \leq l \leq r \leq |L|$ 
  RETURNS  $\{i:\mathbb{N} \mid i \in \{1..r\} \wedge L_i = k\}$ 
   $\equiv$  if  $l=r$  then if  $L_l=k$  then  $\{l\}$  else  $\emptyset$ 
      else let  $m=(l+r)/2$ 
          in if  $L_m < k$  then aux(L,k,m+1,r)
              elseif  $L_{m+1} > k$  then aux(L,k,l,m)
                  else aux(L,k,l,m)  $\cup$  aux(L,k,m+1,r)
```

- Algorithmus hat nur noch logarithmische Laufzeit

OPTIMIERUNG DES GRUNDALGORITHMUS

- **Eliminiere Hilfsfunktionsaufrufe, wenn keine Lösung möglich**

- Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
- Suchraum $\{i \dots j\}$ ohne Lösung, falls $L_i > k$ oder $L_j < k$
- Ergänze Filter $L_i \leq k \leq L_j$ zu rekursivem Aufruf

```
FUNCTION aux(L,k,l,r:Seq(Z)×Z×N×N)
  WHERE L≠[] ∧ ordered(L) ∧ 1≤l≤r≤|L|
  RETURNS {i:N | i∈{1..r} ∧ Li=k}
≡ if l=r then if Ll=k then {l} else ∅
   else let m=(l+r)/2
        in if Lm<k then aux(L,k,m+1,r)
           elseif Lm+1>k then aux(L,k,l,m)
           else aux(L,k,l,m) ∪ aux(L,k,m+1,r)
```

- Algorithmus hat nur noch logarithmische Laufzeit

- **Ergänze Filter zur Initialisierung der Hilfsfunktion**

```
FUNCTION osearch(L,k:Seq(Z)×Z) WHERE L≠[] ∧ ordered(L)
  RETURNS {i:N | i∈{1..|L|} ∧ Li=k}
≡ if L1≤k≤L|L| then aux(L,k,1,|L|) else ∅
```

GLOBALSUCHE: GENERELLE IDEE

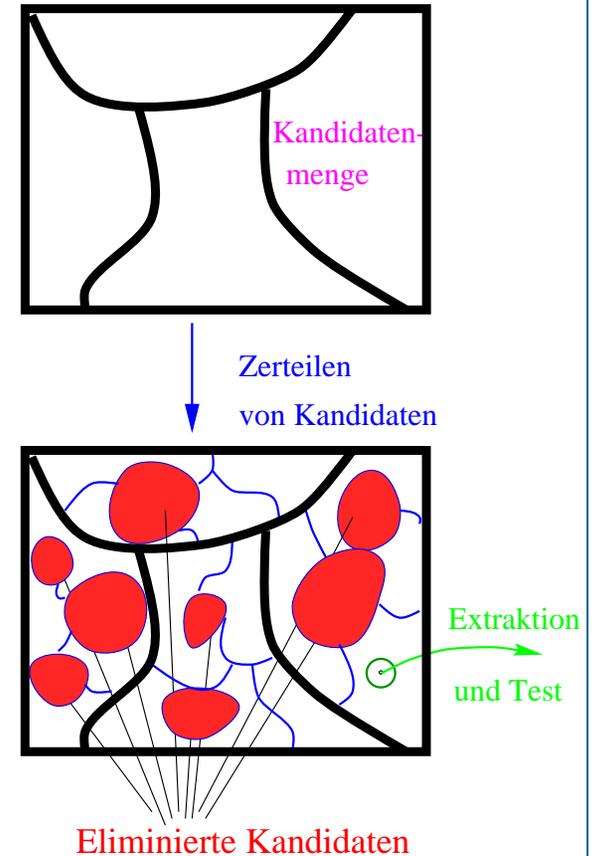
Durchsuchen eines gesamten Bildbereichs

GLOBALSUCHE: GENERELLE IDEE

Durchsuchen eines gesamten Bildbereichs

- **Suche von außen**

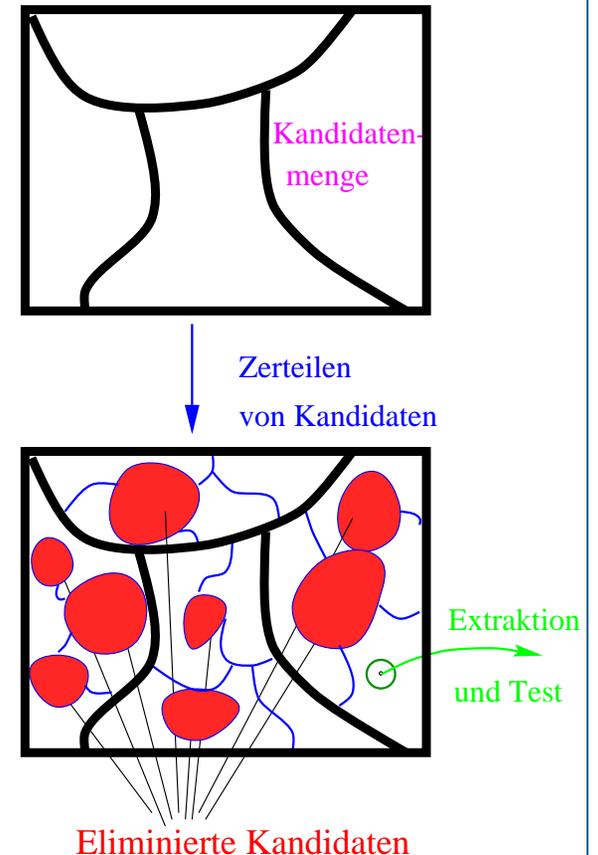
- **Global:** Untersuchung ganzer Mengen von Lösungskandidaten
- Wiederholtes **Aufteilen** von Kandidatenmengen
- **Extraktion** von tatsächlichen Lösungen
- **Elimination** von Kandidatenmengen ohne Lösung



GLOBALSUCHE: GENERELLE IDEE

Durchsuchen eines gesamten Bildbereichs

- **Suche von außen**
 - **Global:** Untersuchung ganzer Mengen von Lösungskandidaten
 - Wiederholtes **Aufteilen** von Kandidatenmengen
 - **Extraktion** von tatsächlichen Lösungen
 - **Elimination** von Kandidatenmengen ohne Lösung
- **Repräsentanten erforderlich**
 - Verarbeitung der Mengen selbst zu aufwendig
 - Codiere Kandidatenmengen durch **Deskriptoren**
 - Simuliere **Aufteilen** und **Filtern** auf Deskriptoren



GLOBALSUCHE: GENERELLE IDEE

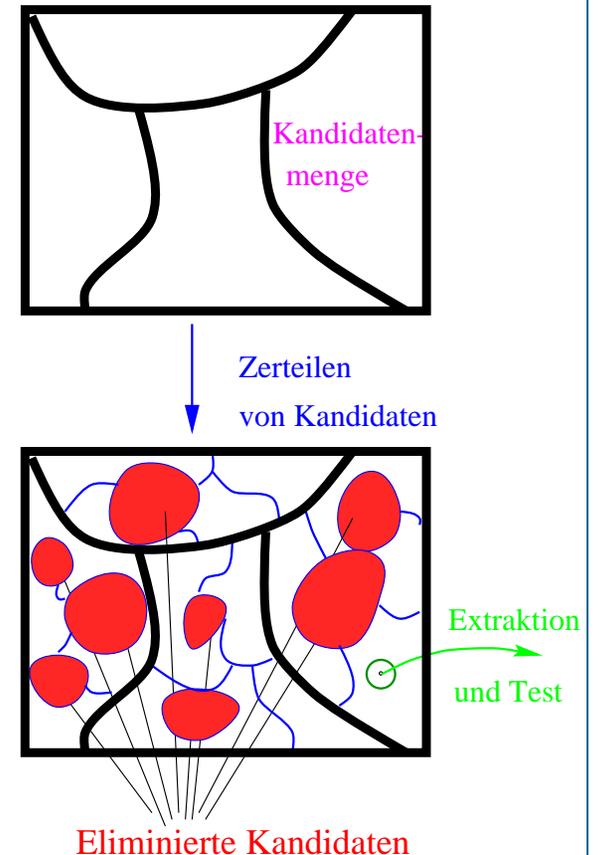
Durchsuchen eines gesamten Bildbereichs

- **Suche von außen**

- **Global:** Untersuchung ganzer Mengen von Lösungskandidaten
- Wiederholtes **Aufteilen** von Kandidatenmengen
- **Extraktion** von tatsächlichen Lösungen
- **Elimination** von Kandidatenmengen ohne Lösung

- **Repräsentanten erforderlich**

- Verarbeitung der Mengen selbst zu aufwendig
- Codiere Kandidatenmengen durch **Deskriptoren**
- Simuliere **Aufteilen** und **Filtern** auf Deskriptoren
- Notwendige Informationen bei der Spezifikation:
 - Wann ist ein Deskriptor eine **sinnvolle Beschreibung** einer Menge?
 - Wie beschreibt man **Zugehörigkeit zur Menge** mittels Deskriptoren?



EINHEITLICHE NOTATION FÜR GLOBALSUCHALGORITHMEN

```
FUNCTION aux(L,k,l,r:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$ ) ...  
≡ if l=r then if Ll=k then {l} else  $\emptyset$   
   else let m=(l+r)/2  
        in if Lm<k then aux(L,k,m+1,r)  
           elseif Lm+1>k then aux(L,k,l,m)  
           else aux(L,k,l,m)  $\cup$  aux(L,k,m+1,r)
```

EINHEITLICHE NOTATION FÜR GLOBALSUCHALGORITHMEN

```
FUNCTION aux(L,k,l,r:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$ ) ...  
≡ if l=r then if  $L_l=k$  then {l} else  $\emptyset$   
   else let m=(l+r)/2  
        in if  $L_m < k$  then aux(L,k,m+1,r)  
           elseif  $L_{m+1} > k$  then aux(L,k,l,m)  
           else aux(L,k,l,m)  $\cup$  aux(L,k,m+1,r)
```

- **Darstellung nur für binäre Aufspaltung des Suchraums geeignet**
 - Allgemeiner Fall ist Vereinigung vieler Lösungsmengen
 - Allgemeine Mengenschreibweise ist geeigneter als Darstellung
 - Indexgrenzen (i, j) werden aus **Aufspaltungsmenge** ausgewählt
 - Indexgrenzen werden ausgefiltert, wenn $L_i > k$ oder $L_j < k$
 - Direkte Lösung wird durch **Extraktion** aus $\{l..r\}$ erzeugt
 - Gesamtlösung ist **Vereinigung aller Einzellösungsmengen**

EINHEITLICHE NOTATION FÜR GLOBALSUCHALGORITHMEN

```
FUNCTION aux(L,k,l,r:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$ ) ...  
≡ if l=r then if  $L_l=k$  then {l} else  $\emptyset$   
   else let m=(l+r)/2  
       in if  $L_m < k$  then aux(L,k,m+1,r)  
          elseif  $L_{m+1} > k$  then aux(L,k,l,m)  
          else aux(L,k,l,m)  $\cup$  aux(L,k,m+1,r)
```

- **Darstellung nur für binäre Aufspaltung des Suchraums geeignet**
 - Allgemeiner Fall ist Vereinigung vieler Lösungsmengen
 - Allgemeine Mengenschreibweise ist geeigneter als Darstellung
 - Indexgrenzen (i, j) werden aus Aufspaltungsmenge ausgewählt
 - Indexgrenzen werden ausgefiltert, wenn $L_i > k$ oder $L_j < k$
 - Direkte Lösung wird durch Extraktion aus $\{l..r\}$ erzeugt
 - Gesamtlösung ist Vereinigung aller Einzellösungsmengen

- **Endform: wohlstrukturierter mathematischer Algorithmus**

```
FUNCTION aux(L,k,l,r:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$ ) ...  
≡ { i |  $i \in \{l\} \wedge i=r \wedge L_i=k$  }  
    $\cup \cup \{ \textit{aux}(L,k,n,m) \mid$   
          $(n,m) \in \{(l, (l+r)/2), ((l+r)/2+1, r) \mid l < r\} \wedge L_n \leq k \leq L_m \}$ 
```

ALLGEMEINES GLOBALSUCH-SCHEMA

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\}$
 $\cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ALLGEMEINES GLOBALSUCH-SCHEMA

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\}$
 $\quad \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

• 7 zentrale Komponenten der Algorithmentheorie

- $s:S$ Deskriptor für Kandidatenmengen
- $s_0: D \rightarrow S$ Initialdeskriptor
- $split:D \times S \rightarrow \text{Set}(S)$ Rekursive Aufteilung von Kandidatenmengen
- $\Phi:D \times S \rightarrow \mathbb{B}$ Filter zur Elimination unnötiger Deskriptoren
- $ext:S \rightarrow \text{Set}(R)$ Extraktion von Lösungskandidaten aus Deskriptoren
Selektion mit Ausgabebedingung $O[x,z]$
- $J:D \times S \rightarrow \mathbb{B}$ $J[x,s]$: Deskriptor s ist sinnvoll für Eingabewert x
- $sat:R \times S \rightarrow \mathbb{B}$ $sat[z,s]$: z gehört zu der durch s beschriebenen Menge

ALLGEMEINES GLOBALSUCH-SCHEMA

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\}$
 $\quad \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

• 7 zentrale Komponenten der Algorithmentheorie

- $s:S$ Deskriptor für Kandidatenmengen
- $s_0: D \rightarrow S$ Initialdeskriptor
- $split: D \times S \rightarrow \text{Set}(S)$ Rekursive Aufteilung von Kandidatenmengen
- $\Phi: D \times S \rightarrow \mathbb{B}$ Filter zur Elimination unnötiger Deskriptoren
- $ext: S \rightarrow \text{Set}(R)$ Extraktion von Lösungskandidaten aus Deskriptoren
Selektion mit Ausgabebedingung $O[x,z]$
- $J: D \times S \rightarrow \mathbb{B}$ $J[x,s]$: Deskriptor s ist sinnvoll für Eingabewert x
- $sat: R \times S \rightarrow \mathbb{B}$ $sat[z,s]$: z gehört zu der durch s beschriebenen Menge

Korrektheit folgt aus wenigen Voraussetzungen

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

3. Initialdeskriptor enthält alle Lösungen

$$I[x] \wedge O[x, z] \Rightarrow sat[z, s_0(x)]$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

3. Initialdeskriptor enthält alle Lösungen

$$I[x] \wedge O[x, z] \Rightarrow sat[z, s_0(x)]$$

4. Filter ist notwendig (keine Lösung wird eliminiert)

$$I[x] \wedge J[x, s] \Rightarrow (\Phi[x, s] \Leftarrow \exists z:R. sat[z, s] \wedge O[x, z])$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

3. Initialdeskriptor enthält alle Lösungen

$$I[x] \wedge O[x, z] \Rightarrow sat[z, s_0(x)]$$

4. Filter ist notwendig (keine Lösung wird eliminiert)

$$I[x] \wedge J[x, s] \Rightarrow (\Phi[x, s] \Leftarrow \exists z:R. sat[z, s] \wedge O[x, z])$$

5. Alle Lösungen in endlich vielen Schritten extrahierbar

$$I[x] \wedge O[x, z] \wedge J[x, s] \Rightarrow (sat[z, s] \Leftrightarrow \exists k:\mathbb{N}. \exists t \in split_{\Phi}^k[x, s]. z \in ext[t])$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

3. Initialdeskriptor enthält alle Lösungen

$$I[x] \wedge O[x, z] \Rightarrow sat[z, s_0(x)]$$

4. Filter ist notwendig (keine Lösung wird eliminiert)

$$I[x] \wedge J[x, s] \Rightarrow (\Phi[x, s] \Leftarrow \exists z:R. sat[z, s] \wedge O[x, z])$$

5. Alle Lösungen in endlich vielen Schritten extrahierbar

$$I[x] \wedge O[x, z] \wedge J[x, s] \Rightarrow (sat[z, s] \Leftrightarrow \exists k:\mathbb{N}. \exists t \in split_{\Phi}^k[x, s]. z \in ext[t])$$

6. Splitting (mit Filterung) ist wohlfundiert

$$I[x] \wedge J[x, s] \Rightarrow \exists k:\mathbb{N}. split_{\Phi}^k[x, s] = \emptyset$$

KORREKTHEIT DES GLOBALSUCH-SCHEMAS

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv let rec $aux(x,s) = \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$
in if $\Phi[x,s_0(x)]$ then $aux(x,s_0(x))$ else \emptyset

ist korrekt, wenn 6 Axiome erfüllt sind

$\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \Phi, ext)$ **wohlfundierte Globalsuchtheorie**

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$$I[x] \Rightarrow J[x, s_0(x)]$$

2. Splitting erhält sinnvoller Deskriptoren

$$I[x] \wedge J[x, s] \Rightarrow \forall t \in split[x, s]. J[x, t]$$

3. Initialdeskriptor enthält alle Lösungen

$$I[x] \wedge O[x, z] \Rightarrow sat[z, s_0(x)]$$

4. Filter ist notwendig (keine Lösung wird eliminiert)

$$I[x] \wedge J[x, s] \Rightarrow (\Phi[x, s] \Leftarrow \exists z:R. sat[z, s] \wedge O[x, z])$$

5. Alle Lösungen in endlich vielen Schritten extrahierbar

$$I[x] \wedge O[x, z] \wedge J[x, s] \Rightarrow (sat[z, s] \Leftrightarrow \exists k:\mathbb{N}. \exists t \in split_{\Phi}^k[x, s]. z \in ext[t])$$

6. Splitting (mit Filterung) ist wohlfundiert

$$I[x] \wedge J[x, s] \Rightarrow \exists k:\mathbb{N}. split_{\Phi}^k[x, s] = \emptyset$$

GLOBALSUCH-SCHEMA: KORREKTHEITSBEWIS

- **Abspalten und Spezifikation der Hilfsfunktion aux**

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

\equiv if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else \emptyset

FUNCTION $aux(x,s:D \times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

RETURNS $\{y:R \mid O[x,y] \wedge sat[y,s]\}$

$\equiv \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$

GLOBALSUCH-SCHEMA: KORREKTHEITSBEWEIF

- **Abspalten und Spezifikation der Hilfsfunktion *aux***

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

\equiv if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else \emptyset

FUNCTION $aux(x,s:D \times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

RETURNS $\{y:R \mid O[x,y] \wedge sat[y,s]\}$

$\equiv \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$

- **Korrektheit von *f* folgt aus der von *aux* mit Axiom 1, 3 & 4**

– Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)

– Aus $I[x]$ folgt $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y:R \mid O[x,y]\}$ (Axiom 3)

– Aus $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

GLOBALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- **Abspalten und Spezifikation der Hilfsfunktion *aux***

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

\equiv if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else \emptyset

FUNCTION $aux(x,s:D \times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

RETURNS $\{y:R \mid O[x,y] \wedge sat[y,s]\}$

$\equiv \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$

- **Korrektheit von *f* folgt aus der von *aux* mit Axiom 1, 3 & 4**

– Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)

– Aus $I[x]$ folgt $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y:R \mid O[x,y]\}$ (Axiom 3)

– Aus $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

- **Partielle Korrektheit von *aux* folgt aus Axiom 5 & 2**

Satz: Hält $aux[x,s]$ nach i Schritten an ($split_{\Phi}^i[x,s] = \emptyset$), so ist das Resultat

$$\bigcup \{ \{z \mid z \in ext[t] \wedge O[x,z]\} \mid t \in \bigcup \{split_{\Phi}^j[x,s] \mid 0 \leq j < i\} \}$$

(Menge der aus Deskriptoren extrahierbaren Lösungen, die zu einem $split_{\Phi}^j[x,s]$ gehören)

$$split_{\Phi}^k[x,s] \equiv \text{if } k=0 \text{ then } \{s\} \text{ else } \bigcup \{split_{\Phi}^{k-1}[x,t] \mid t \in split[x,s] \wedge \Phi[x,t]\}$$

Beweis: Induktion über i , Auffalten der Rekursion, Standardlemmata

GLOBALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- **Abspalten und Spezifikation der Hilfsfunktion *aux***

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

\equiv if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else \emptyset

FUNCTION $aux(x,s:D \times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

RETURNS $\{y:R \mid O[x,y] \wedge sat[y,s]\}$

$\equiv \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\}$

- **Korrektheit von f folgt aus der von aux mit Axiom 1, 3 & 4**

– Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)

– Aus $I[x]$ folgt $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y:R \mid O[x,y]\}$ (Axiom 3)

– Aus $\{y:R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

- **Partielle Korrektheit von aux folgt aus Axiom 5 & 2**

Satz: Hält $aux[x, s]$ nach i Schritten an ($split_{\Phi}^i[x, s] = \emptyset$), so ist das Resultat

$$\bigcup \{ \{z \mid z \in ext[t] \wedge O[x,z]\} \mid t \in \bigcup \{split_{\Phi}^j[x, s] \mid 0 \leq j < i\} \}$$

(Menge der aus Deskriptoren extrahierbaren Lösungen, die zu einem $split_{\Phi}^j[x, s]$ gehören)

$$split_{\Phi}^k[x, s] \equiv \text{if } k=0 \text{ then } \{s\} \text{ else } \bigcup \{split_{\Phi}^{k-1}[x, t] \mid t \in split[x, s] \wedge \Phi[x, t]\}$$

Beweis: Induktion über i , Auffalten der Rekursion, Standardlemmata

- **Terminierung von aux folgt aus Axiom 6**

BEISPIEL EINER GLOBALSUCHTHEORIE

• Theorie `gs_osearch` für `osearch`

<i>D</i>	\mapsto	$\text{Seq}(\mathbb{N}) \times \mathbb{N}$
<i>R</i>	\mapsto	$\text{Set}(\mathbb{N})$
<i>I</i>	\mapsto	$\lambda L, k. L \neq [] \wedge \text{ordered}(L)$
<i>O</i>	\mapsto	$\lambda L, k, i. i \in \{1.. L \} \wedge L_i = k$
<i>S</i>	\mapsto	$\mathbb{N} \times \mathbb{N}$
<i>J</i>	\mapsto	$\lambda L, k, l, r. 1 \leq l \leq r \leq L $
<i>s₀</i>	\mapsto	$\lambda L, k. (1, L)$
<i>sat</i>	\mapsto	$\lambda i, l, r. i \in \{1..r\}$
<i>split</i>	\mapsto	$\lambda L, k, l, r. \text{if } l < r \text{ then } \{(l, (l+r)/2), ((l+r)/2+1, r)\} \text{ else } \emptyset$
Φ	\mapsto	$\lambda L, k, l, r. L_l \leq k \leq L_r$
<i>ext</i>	\mapsto	$\lambda l, r. \text{if } l=r \text{ then } \{l\} \text{ else } \emptyset$

BEISPIEL EINER GLOBALSUCHTHEORIE

• Theorie *gs_osearch* für *osearch*

<i>D</i>	\mapsto	$\text{Seq}(\mathbb{N}) \times \mathbb{N}$
<i>R</i>	\mapsto	$\text{Set}(\mathbb{N})$
<i>I</i>	\mapsto	$\lambda L, k. L \neq [] \wedge \text{ordered}(L)$
<i>O</i>	\mapsto	$\lambda L, k, i. i \in \{1.. L \} \wedge L_i = k$
<i>S</i>	\mapsto	$\mathbb{N} \times \mathbb{N}$
<i>J</i>	\mapsto	$\lambda L, k, l, r. 1 \leq l \leq r \leq L $
<i>s₀</i>	\mapsto	$\lambda L, k. (1, L)$
<i>sat</i>	\mapsto	$\lambda i, l, r. i \in \{1..r\}$
<i>split</i>	\mapsto	$\lambda L, k, l, r. \text{if } l < r \text{ then } \{(1, (l+r)/2), ((l+r)/2+1, r)\} \text{ else } \emptyset$
Φ	\mapsto	$\lambda L, k, l, r. L_l \leq k \leq L_r$
<i>ext</i>	\mapsto	$\lambda l, r. \text{if } l=r \text{ then } \{1\} \text{ else } \emptyset$

• Alle 6 Axiome sind erfüllt

- $L \neq [] \wedge \text{ordered}(L) \Rightarrow 1 \leq l \leq |L| \leq |L|$
- $\dots \wedge 1 \leq l \leq r \leq |L| \Rightarrow \forall (x, y) \in \text{split}[L, k, l, r]. 1 \leq x \leq y \leq |L|$
- $\dots \wedge i \in \{1..|L|\} \wedge L_i = k \Rightarrow i \in \{1..|L|\}$
- $\dots \wedge 1 \leq l \leq r \leq |L| \Rightarrow L_l \leq k \leq L_r \Leftrightarrow \exists z: \mathbb{N}. z \in \{1..r\} \wedge z \in \{1..|L|\} \wedge L_z = k$
- $\dots \wedge 1 \leq l \leq r \leq |L| \wedge i \in \{1..|L|\} \wedge L_i = k \Rightarrow$
 $i \in \{1..r\} \Leftrightarrow \exists k: \mathbb{N}. \exists (x, y) \in \text{split}_{\Phi}^k[L, k, l, r]. i \in (\text{if } x=y \text{ then } \{x\} \text{ else } \emptyset)$
- $\dots \wedge 1 \leq l \leq r \leq |L| \Rightarrow \exists k: \mathbb{N}. \text{split}_{\Phi}^k[L, k, l, r] = \emptyset$

SCHEMATISCHER GLOBALSUCHALGORITHMUS FÜR osearch

```
FUNCTION osearch(L,k:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}$ ) WHERE L $\neq$ [ ]  $\wedge$  ordered(L)
  RETURNS { i: $\mathbb{N}$  | i $\in$ {1..|L|}  $\wedge$  Li=k }
 $\equiv$  let rec aux(L,k,l,r)
  = { z | z $\in$ (if l=r then {1} else  $\emptyset$ )  $\wedge$  z $\in$ {1..|L|}  $\wedge$  Lz=k }
     $\cup\cup$ { aux(L,k,n,m) |
      (n,m) $\in$ (if l<r then {(1,(l+r)/2), ((l+r)/2+1,r)}
        else  $\emptyset$ )  $\wedge$  Ln $\leq$ k $\leq$ Lm }
  in if L1 $\leq$ k $\leq$ L|L| then aux(L,k,1,|L|) else  $\emptyset$ 
```

SCHEMATISCHER GLOBALSUCHALGORITHMUS FÜR osearch

```
FUNCTION osearch(L,k:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}$ ) WHERE L $\neq$ [ ]  $\wedge$  ordered(L)
  RETURNS { i: $\mathbb{N}$  | i $\in$ {1..|L|}  $\wedge$  Li=k }
 $\equiv$  let rec aux(L,k,l,r)
  = { z | z $\in$ (if l=r then {1} else  $\emptyset$ )  $\wedge$  z $\in$ {1..|L|}  $\wedge$  Lz=k }
     $\cup\cup$ { aux(L,k,n,m) |
      (n,m) $\in$ (if l<r then {(1,(l+r)/2), ((l+r)/2+1,r)}
        else  $\emptyset$ )  $\wedge$  Ln $\leq$ k $\leq$ Lm }
  in if L1 $\leq$ k $\leq$ L|L| then aux(L,k,1,|L|) else  $\emptyset$ 
```

Nach (kontextabhängigen) Simplifikationen

```
FUNCTION osearch(L,k:Seq( $\mathbb{Z}$ ) $\times\mathbb{Z}$ ) WHERE L $\neq$ [ ]  $\wedge$  ordered(L)
  RETURNS { i: $\mathbb{N}$  | i $\in$ {1..|L|}  $\wedge$  Li=k }
 $\equiv$  let rec aux(L,k,l,r)
  = if l=r then if Ll=k then {1} else  $\emptyset$ 
    else let m = (l+r)/2 in
      (if k $\leq$ Lm then aux(L,k,l,m) else  $\emptyset$ )
       $\cup$  (if Lm+1 $\leq$ k then aux(L,k,m+1,r) else  $\emptyset$ )
  in if L1 $\leq$ k $\leq$ L|L| then aux(L,k,1,|L|) else  $\emptyset$ 
```

WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

Spezialisiere vorformuliertes Programmierwissen

WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie:** allgemeine Suchstruktur für R
 - Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt
 - Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, True, ext)$
 - Wissensbank speichert Globalsuchtheorien für Grunddatentypen

WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie: allgemeine Suchstruktur für R**
 - Vorgefertigte Zerlegungsstruktur, die **Axiome 1–5** erfüllt
 - Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, True, ext)$
 - Wissensbank speichert Globalsuchtheorien für Grunddatentypen
- **Filter Φ zur Verfeinerung der $split$ -Operation**
 - **Wohlfundiertheit**: Filter garantiert Terminierung von $split_\Phi$
Wissensbank speichert Wohlfundiertheitsfilter zu GS-Theorien \mapsto Ax 6
 - **Notwendigkeit**: Filter eliminiert keine Lösungen
System prüft **Axiom 4** zur Laufzeit
 - **Effizienzsteigerung**: System verfeinert notwendige Filter heuristisch

WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie: allgemeine Suchstruktur für R**
 - Vorgefertigte Zerlegungsstruktur, die **Axiome 1–5** erfüllt
 - Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_{\emptyset}, sat, split, True, ext)$
 - Wissensbank speichert Globalsuchtheorien für Grunddatentypen
- **Filter Φ zur Verfeinerung der $split$ -Operation**
 - **Wohlfundiertheit**: Filter garantiert Terminierung von $split_{\Phi}$
Wissensbank speichert Wohlfundiertheitsfilter zu GS-Theorien \mapsto Ax 6
 - **Notwendigkeit**: Filter eliminiert keine Lösungen
System prüft **Axiom 4** zur Laufzeit
 - **Effizienzsteigerung**: System verfeinert notwendige Filter heuristisch
- **Spezialisierungsmechanismen für \mathcal{G} und Φ**
 - Wähle \mathcal{G} passend zum Bildbereich der Spezifikation $spec = (D, R, I, O)$
 - Beweise $spec \ll spec_{\mathcal{G}}$ und extrahiere Substitution $\vartheta: D \rightarrow D_{\mathcal{G}}$

$$R \subseteq R_{\mathcal{G}} \wedge \forall x: D. I[x] \Rightarrow \exists x': D_{\mathcal{G}}. (I_{\mathcal{G}}[x'] \wedge \forall y: R. O[x, y] \Rightarrow O_{\mathcal{G}}[x', y])$$

WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie: allgemeine Suchstruktur für R**
 - Vorgefertigte Zerlegungsstruktur, die **Axiome 1–5** erfüllt
 - Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_{\emptyset}, sat, split, True, ext)$
 - Wissensbank speichert Globalsuchtheorien für Grunddatentypen
- **Filter Φ zur Verfeinerung der $split$ -Operation**
 - **Wohlfundiertheit**: Filter garantiert Terminierung von $split_{\Phi}$
Wissensbank speichert Wohlfundiertheitsfilter zu GS-Theorien \mapsto Ax 6
 - **Notwendigkeit**: Filter eliminiert keine Lösungen
System prüft **Axiom 4** zur Laufzeit
 - **Effizienzsteigerung**: System verfeinert notwendige Filter heuristisch
- **Spezialisierungsmechanismen für \mathcal{G} und Φ**
 - Wähle \mathcal{G} passend zum Bildbereich der Spezifikation $spec = (D, R, I, O)$
 - Beweise $spec \ll spec_{\mathcal{G}}$ und extrahiere Substitution $\vartheta: D \rightarrow D_{\mathcal{G}}$
 - **Modifiziere \mathcal{G} und Φ mit ϑ zu wohlfundierter Globalsuchtheorie für $spec$**

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L

- **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L

[]

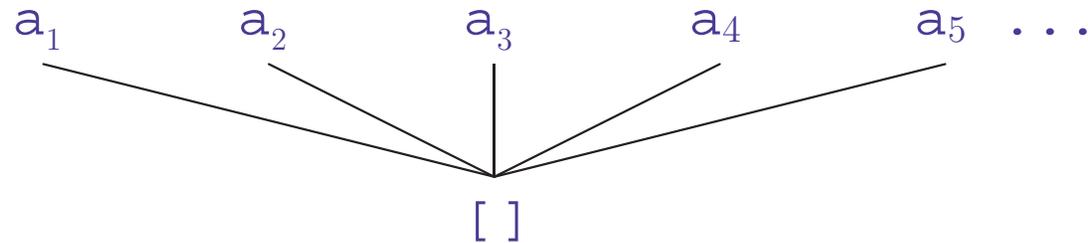
- **Deskriptoren: gemeinsamer Präfix s**
- **Initialdeskriptor: leerer Präfix**

$$\text{sat}[L, s] \equiv s \sqsubseteq L$$

$$s_0(M) \equiv []$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

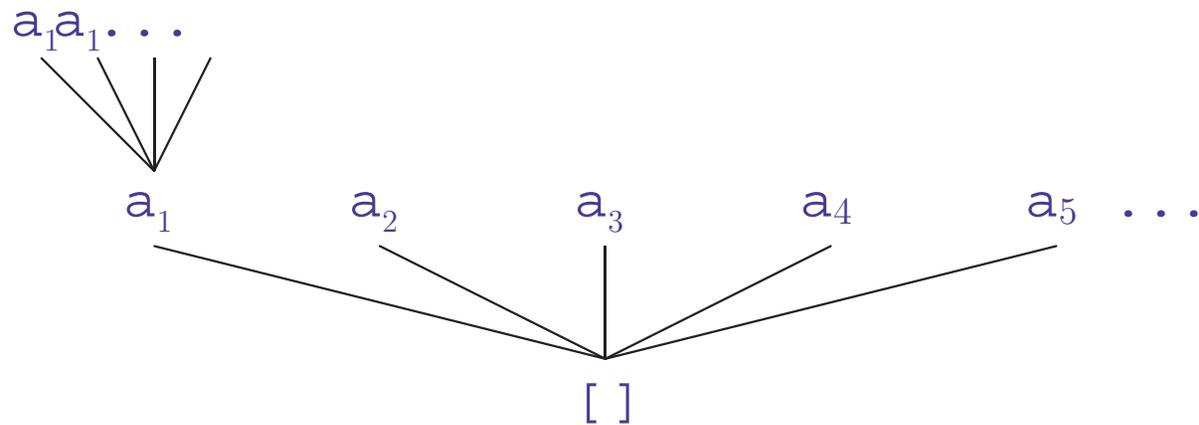
• **Initialdeskriptor: leerer Präfix**

$$s_0(M) \equiv []$$

• **Splitting: Verlängern des Präfix**

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

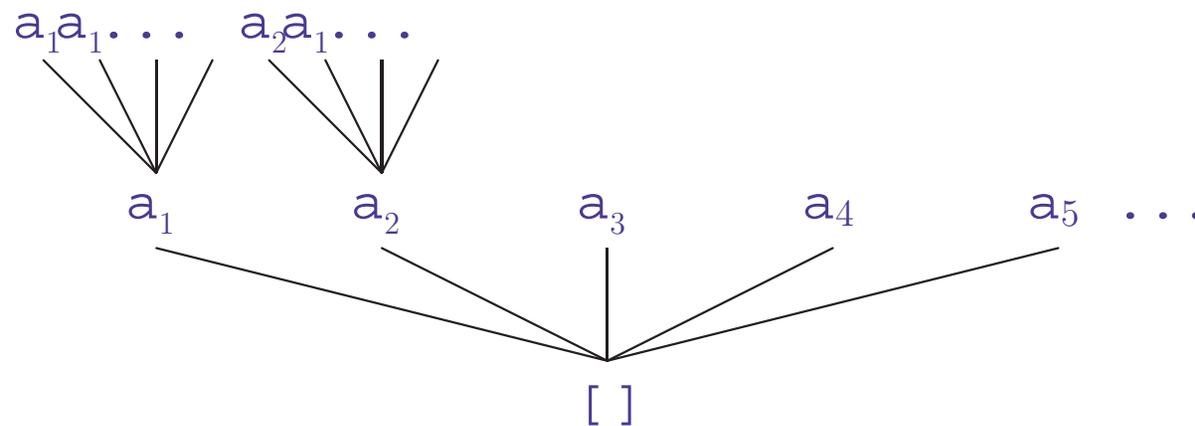
• **Initialdeskriptor: leerer Präfix**

$$s_0(M) \equiv []$$

• **Splitting: Verlängern des Präfix**

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

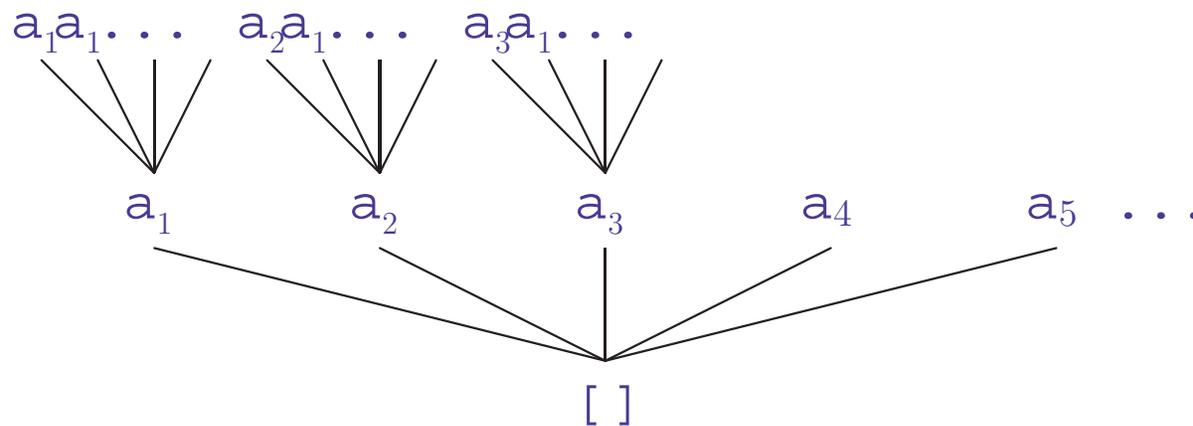
$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

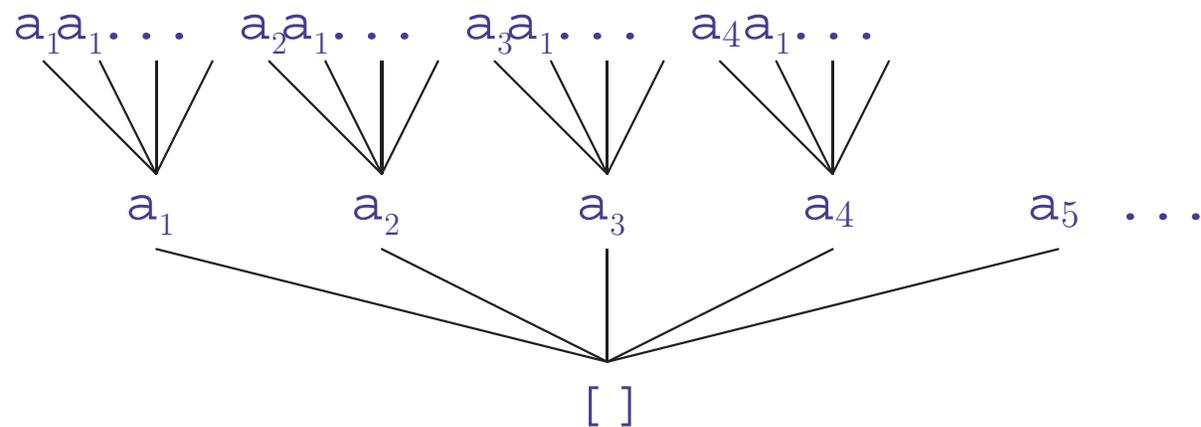
$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$\text{sat}[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

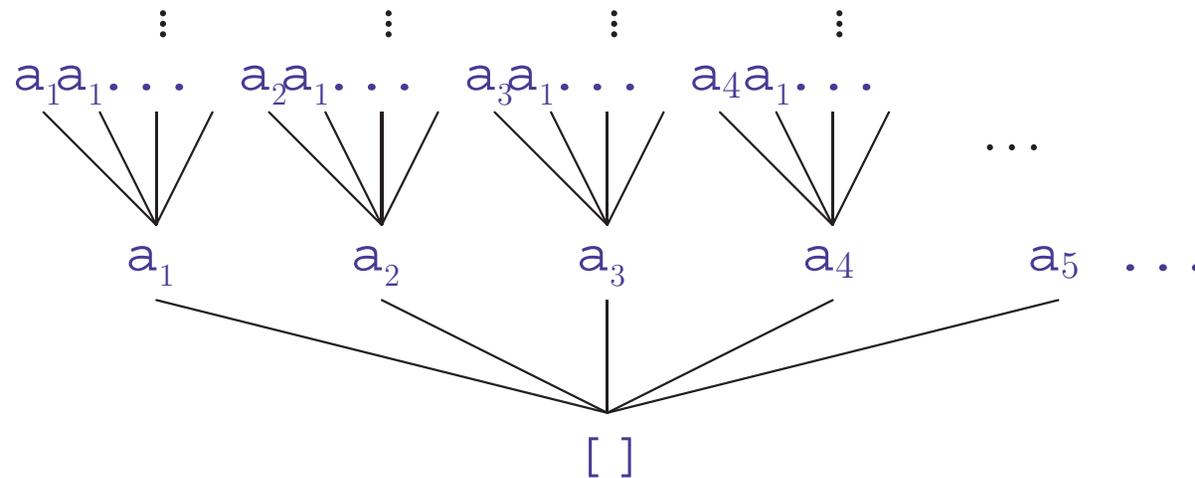
$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

$$\text{split}[M, s] \equiv \{s \cdot a \mid a \in M\}$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$sat[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

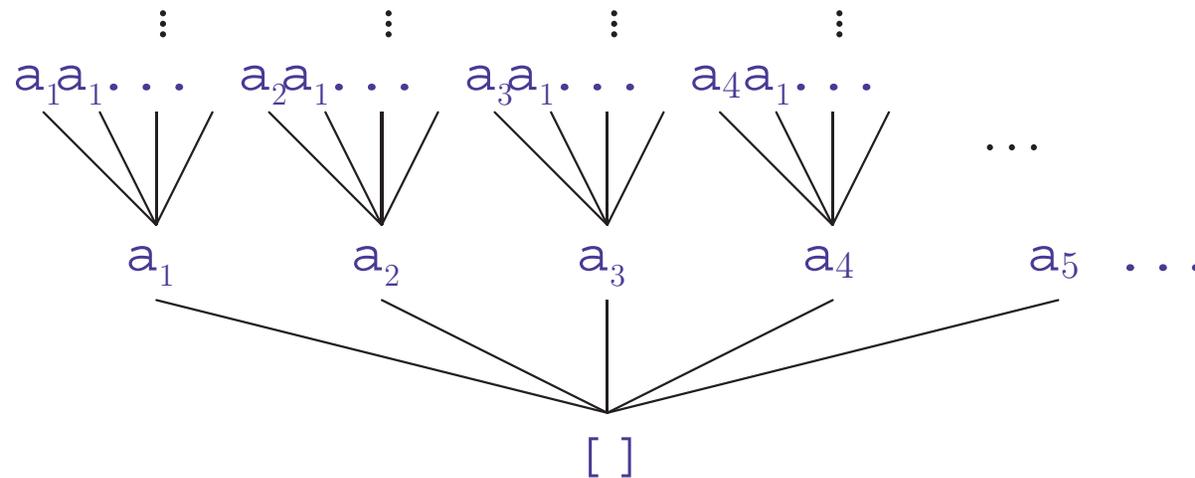
$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$\text{sat}[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

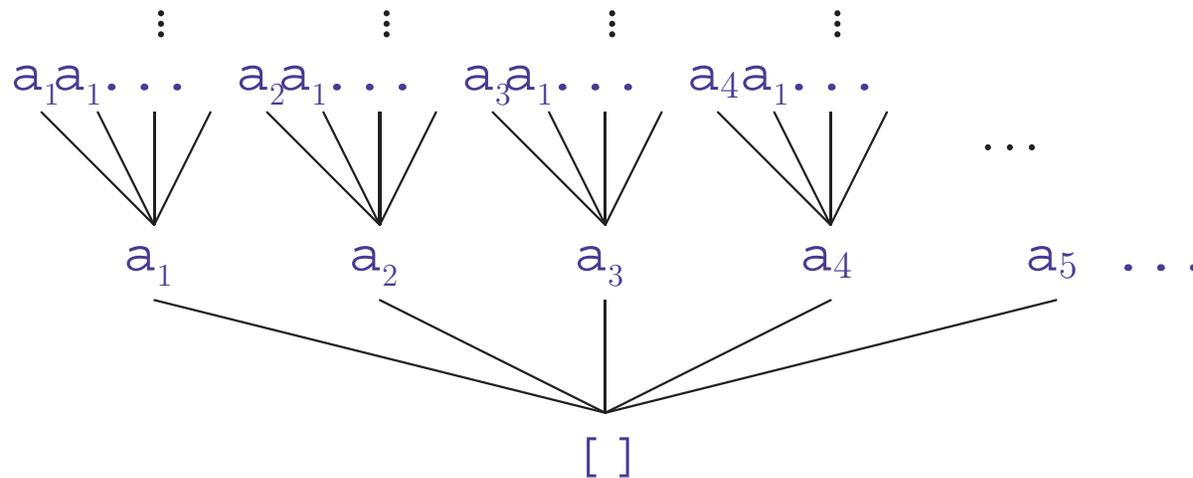
$$\text{split}[M, s] \equiv \{s \cdot a \mid a \in M\}$$

• **Extraktion: Gesamter Präfix**

$$\text{ext}[s] \equiv \{s\}$$

GS-THEORIE FÜR LISTEN ÜBER ENDLICHER MENGE $M \subseteq \alpha$

Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



• **Deskriptoren: gemeinsamer Präfix s**

$$\text{sat}[L, s] \equiv s \sqsubseteq L$$

• **Initialdeskriptor: leerer Präfix**

$$s_0(M) \equiv [\]$$

• **Splitting: Verlängern des Präfix**

$$\text{split}[M, s] \equiv \{s \cdot a \mid a \in M\}$$

• **Extraktion: Gesamter Präfix**

$$\text{ext}[s] \equiv \{s\}$$

• **Sinnvoll: nur Elemente aus M**

$$J[M, s] \equiv s' \sqsubseteq' M$$

FORMALE GS-THEORIE FÜR LISTEN ÜBER $M \subseteq \alpha$

- Deskriptoren: gemeinsamer Präfix s
- Initialdeskriptor: leerer Präfix
- Splitting: Verlängern des Präfix
- Extraktion: Gesamter Präfix
- Sinnvoll: nur Elemente aus M

$$sat[L, s] \equiv s \sqsubseteq L$$

$$s_0(M) \equiv []$$

$$split[M, s] \equiv \{s \cdot a \mid a \in M\}$$

$$ext[s] \equiv \{s\}$$

$$J[M, s] \equiv s \text{ '}\subseteq\text{' } M$$

FORMALE GS-THEORIE FÜR LISTEN ÜBER $M \subseteq \alpha$

- Deskriptoren: gemeinsamer Präfix s $sat[L, s] \equiv s \sqsubseteq L$
- Initialdeskriptor: leerer Präfix $s_0(M) \equiv []$
- Splitting: Verlängern des Präfix $split[M, s] \equiv \{s \cdot a \mid a \in M\}$
- Extraktion: Gesamter Präfix $ext[s] \equiv \{s\}$
- Sinnvoll: nur Elemente aus M $J[M, s] \equiv s \text{ '}\subseteq\text{' } M$

Darstellung als formales Objekt der Wissensbank

$gs_seq_set(\alpha)$	\equiv	D	\mapsto	$Set(\alpha)$
		R	\mapsto	$Seq(\alpha)$
		I	\mapsto	$\lambda M. true$
		O	\mapsto	$\lambda M, L. range(L) \subseteq M$
		S	\mapsto	$Seq(\alpha)$
		J	\mapsto	$\lambda M, s. range(s) \subseteq M$
		s_0	\mapsto	$\lambda M. []$
		sat	\mapsto	$\lambda L, s. s \sqsubseteq L$
		$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
		ext	\mapsto	$\lambda s. \{s\}$

WOHLFUNDIERTHEITSFILTER FÜR $gs_seq_set(\alpha)$

- $\Phi_1[M, s] \equiv |s| \leq k$
 - Filter testet absolute Längenbegrenzung der Deskriptorfolge
 - Einfacher, schnell auszuführender Test
 - Filter garantiert Terminierung nach k Schritten, Baumgröße $|M|^k$

WOHLFUNDIERTHEITSFILTER FÜR $gs_seq_set(\alpha)$

- $\Phi_1[M, s] \equiv |s| \leq k$
 - Filter testet absolute Längenbegrenzung der Deskriptorfolge
 - Einfacher, schnell auszuführender Test
 - Filter garantiert Terminierung nach k Schritten, Baumgröße $|M|^k$
- $\Phi_2[M, s] \equiv |s| \leq k * |M|$
 - Filter testet Längenbegrenzung relativ zur Größe von M
 - Einfacher, schnell auszuführender Test
 - Terminierung nach $k * |M|$ Schritten, Baumgröße $|M|^{k * |M|}$

WOHLFUNDIERTHEITSFILTER FÜR $gs_seq_set(\alpha)$

- $\Phi_1[M, s] \equiv |s| \leq k$
 - Filter testet absolute Längenbegrenzung der Deskriptorfolge
 - Einfacher, schnell auszuführender Test
 - Filter garantiert Terminierung nach k Schritten, Baumgröße $|M|^k$
- $\Phi_2[M, s] \equiv |s| \leq k * |M|$
 - Filter testet Längenbegrenzung relativ zur Größe von M
 - Einfacher, schnell auszuführender Test
 - Terminierung nach $k * |M|$ Schritten, Baumgröße $|M|^{k * |M|}$
- $\Phi_3[M, s] \equiv \text{nodups}(s)$
 - Filter testet Deskriptorfolge auf Duplikate
 - Test ist aufwendiger und sollte optimiert werden
 - Terminierung nach $|M|$ Schritten, Baumgröße $|M|!$

WOHLFUNDIERTHEITSFILTER FÜR $gs_seq_set(\alpha)$

- $\Phi_1[M, s] \equiv |s| \leq k$
 - Filter testet absolute Längenbegrenzung der Deskriptorfolge
 - Einfacher, schnell auszuführender Test
 - Filter garantiert Terminierung nach k Schritten, Baumgröße $|M|^k$
- $\Phi_2[M, s] \equiv |s| \leq k * |M|$
 - Filter testet Längenbegrenzung relativ zur Größe von M
 - Einfacher, schnell auszuführender Test
 - Terminierung nach $k * |M|$ Schritten, Baumgröße $|M|^{k * |M|}$
- $\Phi_3[M, s] \equiv \text{nodups}(s)$
 - Filter testet Deskriptorfolge auf Duplikate
 - Test ist aufwendiger und sollte optimiert werden
 - Terminierung nach $|M|$ Schritten, Baumgröße $|M|!$

Jeder Filter macht $gs_seq_set(\alpha)$ wohlfundiert

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :
 - Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenzentafel

2	4	1	6	5	3

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :
 - Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenztabelle

2	4	1	6	5	3
-2	3	-5	1	2	

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :
 - Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenzentafel

2	4	1	6	5	3
-2	3	-5	1	2	
1	-2	-4	3		
-4	-1	-2			
-3	1				
-1					

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :
 - Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenztabelle

2	4	1	6	5	3
-2	3	-5	1	2	
1	-2	-4	3		
-4	-1	-2			
-3	1				
-1					

- **Ziel: Berechne alle Costas Arrays der Größe n**

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :

- Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenzentafel

2	4	1	6	5	3
-2	3	-5	1	2	
1	-2	-4	3		
-4	-1	-2			
-3	1				
-1					

- **Ziel: Berechne alle Costas Arrays der Größe n**

- **Formalisierung vorkommender Begriffe:**

$$\text{dtrow}(L, j) \equiv [L_i - L_{i+j} \mid i \in [1..|L|-j]]$$

$$\text{perm}(L, S) \equiv \text{nodups}(L) \wedge \text{range}(L) = S$$

ENTWICKLUNG EINES GLOBALSUCH-ALGORITHMUS FÜR DAS COSTAS-ARRAYS PROBLEM

- **Costas Array** der Größe n :

- Permutation von $\{1..n\}$ ohne Duplikate in Zeilen der Differenztabelle

2	4	1	6	5	3
-2	3	-5	1	2	
1	-2	-4	3		
-4	-1	-2			
-3	1				
-1					

- **Ziel: Berechne alle Costas Arrays der Größe n**

- **Formalisierung vorkommender Begriffe:**

$dtrow(L, j) \equiv [L_i - L_{i+j} \mid i \in [1..|L| - j]]$

$perm(L, S) \equiv nodups(L) \wedge range(L) = S$

- **Spezifikation des Problems**

FUNCTION Costas ($n:\mathbb{Z}$) WHERE $n \geq 1$

RETURNS $\{p:Seq(\mathbb{Z}) \mid perm(p, \{1..n\}) \wedge \forall j < |p|. nodups(dtrow(p, j))\}$

MATHEMATISCHE GESETZE FÜR COSTAS ARRAYS

• Gesetze von Differenzentafeln

$\forall L, L' : \text{Seq}(\mathbb{Z}) . \forall i : \mathbb{Z} . \forall j : \mathbb{N} .$

1. $\text{dtrow}([], j) = []$
 2. $j \leq |L| \Rightarrow \text{dtrow}(i.L, j) = (i - L[j]) . \text{dtrow}(L, j)$
 3. $j \neq 0 \Rightarrow \text{dtrow}([i], j) = []$
 4. $L \sqsubseteq L' \Rightarrow \text{dtrow}(L, j) \sqsubseteq \text{dtrow}(L', j)$
 5. $j \geq |L| \Rightarrow \text{dtrow}(L, j) = []$
 6. $j \leq |L| \Rightarrow \text{dtrow}(L.i, j) = \text{dtrow}(L, j) \cdot (L[|L| + 1 - j] - i)$
- ⋮

MATHEMATISCHE GESETZE FÜR COSTAS ARRAYS

• Gesetze von Differenzentafeln

$\forall L, L' : \text{Seq}(\mathbb{Z}) . \forall i : \mathbb{Z} . \forall j : \mathbb{N} .$

1. $\text{dtrow}([], j) = []$
 2. $j \leq |L| \Rightarrow \text{dtrow}(i.L, j) = (i - L[j]).\text{dtrow}(L, j)$
 3. $j \neq 0 \Rightarrow \text{dtrow}([i], j) = []$
 4. $L \sqsubseteq L' \Rightarrow \text{dtrow}(L, j) \sqsubseteq \text{dtrow}(L', j)$
 5. $j \geq |L| \Rightarrow \text{dtrow}(L, j) = []$
 6. $j \leq |L| \Rightarrow \text{dtrow}(L.i, j) = \text{dtrow}(L, j) \cdot (L[|L|+1-j]-i)$
- ⋮

• Gesetze duplikatenfreier Listen

$\forall L, L' : \text{Seq}(\mathbb{Z}) . \forall i : \mathbb{Z} .$

1. $\text{nodups}([])$
 2. $\text{nodups}(i.L) = \text{nodups}(L) \wedge i \notin L$
 3. $\text{nodups}(L.i) = \text{nodups}(L) \wedge i \notin L$
 4. $L \sqsubseteq L' \Rightarrow \text{nodups}(L') \Rightarrow \text{nodups}(L)$
- ⋮

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n: $\mathbb{Z}$ ) WHERE n $\geq$ 1  
RETURNS {p:Seq( $\mathbb{Z}$ ) | perm(p, {1..n})  
        ^  $\forall j < |p|.nodups(dtrow(p, j))$ }
```

<i>D</i>	\mapsto	Set(α)
<i>R</i>	\mapsto	Seq(α)
<i>I</i>	\mapsto	$\lambda M. true$
<i>O</i>	\mapsto	$\lambda M, L. range(L) \subseteq M$
<i>S</i>	\mapsto	Seq(α)
<i>J</i>	\mapsto	$\lambda M, s. range(s) \subseteq M$
<i>s₀</i>	\mapsto	$\lambda M. []$
<i>sat</i>	\mapsto	$\lambda L, s. s \subseteq L$
<i>split</i>	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
<i>ext</i>	\mapsto	$\lambda s. \{s\}$

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n ≥ 1
RETURNS {p:Seq(ℤ) | perm(p, {1..n})
        ∧ ∀j < |p|. nodups(dtrow(p, j))}
```

1. Bildbereich stimmt mit $R_G = \text{Seq}(\mathbb{Z})$ überein

D	\mapsto	$\text{Set}(\alpha)$
R	\mapsto	$\text{Seq}(\alpha)$
I	\mapsto	$\lambda M. \text{true}$
O	\mapsto	$\lambda M, L. \text{range}(L) \subseteq M$
S	\mapsto	$\text{Seq}(\alpha)$
J	\mapsto	$\lambda M, s. \text{range}(s) \subseteq M$
s_0	\mapsto	$\lambda M. []$
sat	\mapsto	$\lambda L, s. s \subseteq L$
$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
ext	\mapsto	$\lambda s. \{s\}$

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n≥1  
RETURNS {p:Seq(ℤ) | perm(p, {1..n})  
          ∧ ∀j<|p|.nodups(dtrow(p, j))}
```

1. Bildbereich stimmt mit $R_G = \text{Seq}(\mathbb{Z})$ überein

2. Eingabebereiche \mathbb{Z} , $D_G = \text{Set}(\mathbb{Z})$ sind anzupassen

D	\mapsto	$\text{Set}(\alpha)$
R	\mapsto	$\text{Seq}(\alpha)$
I	\mapsto	$\lambda M. \text{true}$
O	\mapsto	$\lambda M, L. \text{range}(L) \subseteq M$
S	\mapsto	$\text{Seq}(\alpha)$
J	\mapsto	$\lambda M, s. \text{range}(s) \subseteq M$
s_0	\mapsto	$\lambda M. []$
sat	\mapsto	$\lambda L, s. s \subseteq L$
$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
ext	\mapsto	$\lambda s. \{s\}$

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n: $\mathbb{Z}$ ) WHERE n $\geq$ 1  
RETURNS {p:Seq( $\mathbb{Z}$ ) | perm(p, {1..n})  
        ^  $\forall j < |p|.nodups(dtrow(p, j))$ }
```

1. **Bildbereich stimmt mit $R_G = \text{Seq}(\mathbb{Z})$ überein**
2. **Eingabebereiche \mathbb{Z} , $D_G = \text{Set}(\mathbb{Z})$ sind anzupassen**
3. **Keine Eingabebedingung zu prüfen: $I_G(M) = \text{true}$**

D	\mapsto	$\text{Set}(\alpha)$
R	\mapsto	$\text{Seq}(\alpha)$
I	\mapsto	$\lambda M. \text{true}$
O	\mapsto	$\lambda M, L. \text{range}(L) \subseteq M$
S	\mapsto	$\text{Seq}(\alpha)$
J	\mapsto	$\lambda M, s. \text{range}(s) \subseteq M$
s_0	\mapsto	$\lambda M. []$
sat	\mapsto	$\lambda L, s. s \subseteq L$
$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
ext	\mapsto	$\lambda s. \{s\}$

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n≥1
RETURNS {p:Seq(ℤ) | perm(p, {1..n})
          ∧ ∀j<|p|.nodups(dtrow(p, j))}
```

<i>D</i>	↦	Set(α)
<i>R</i>	↦	Seq(α)
<i>I</i>	↦	$\lambda M. \text{true}$
<i>O</i>	↦	$\lambda M, L. \text{range}(L) \subseteq M$
<i>S</i>	↦	Seq(α)
<i>J</i>	↦	$\lambda M, s. \text{range}(s) \subseteq M$
<i>s₀</i>	↦	$\lambda M. []$
<i>sat</i>	↦	$\lambda L, s. s \subseteq L$
<i>split</i>	↦	$\lambda M, s. \{s \cdot a \mid a \in M\}$
<i>ext</i>	↦	$\lambda s. \{s\}$

1. Bildbereich stimmt mit $R_G = \text{Seq}(\mathbb{Z})$ überein
2. Eingabebereiche \mathbb{Z} , $D_G = \text{Set}(\mathbb{Z})$ sind anzupassen
3. Keine Eingabebedingung zu prüfen: $I_G(M) = \text{true}$
4. Zu zeigen ist also:

$\forall n: \mathbb{Z}. n \geq 1 \Rightarrow \exists M: \text{Set}(\mathbb{Z}). \forall p: \text{Seq}(\mathbb{Z}). O(n, p) \Rightarrow \text{range}(p) \subseteq M$

– Heuristik: Suche Folgerungen von $O(n, p)$, in denen $\text{range}(p)$ vorkommt

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n≥1
RETURNS {p:Seq(ℤ) | perm(p, {1..n})
          ∧ ∀j<|p|.nodups(dtrow(p, j))}
```

D	\mapsto	$Set(\alpha)$
R	\mapsto	$Seq(\alpha)$
I	\mapsto	$\lambda M. true$
O	\mapsto	$\lambda M, L. range(L) \subseteq M$
S	\mapsto	$Seq(\alpha)$
J	\mapsto	$\lambda M, s. range(s) \subseteq M$
s_0	\mapsto	$\lambda M. []$
sat	\mapsto	$\lambda L, s. s \subseteq L$
$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
ext	\mapsto	$\lambda s. \{s\}$

1. **Bildbereich stimmt mit $R_G = Seq(\mathbb{Z})$ überein**
2. **Eingabebereiche \mathbb{Z} , $D_G = Set(\mathbb{Z})$ sind anzupassen**
3. **Keine Eingabebedingung zu prüfen: $I_G(M) = true$**
4. **Zu zeigen ist also:**

$\forall n: \mathbb{Z}. n \geq 1 \Rightarrow \exists M: Set(\mathbb{Z}). \forall p: Seq(\mathbb{Z}). O(n, p) \Rightarrow range(p) \subseteq M$

- Heuristik: Suche Folgerungen von $O(n, p)$, in denen $range(p)$ vorkommt
- Auffalten von $perm$ liefert: $perm(p, \{1..n\}) \Rightarrow range(p) \subseteq \{1..n\}$
- Wähle $M \equiv \{1..n\}$ und extrahiere $\vartheta = \lambda n. \{1..n\}$

SPEZIALISIERE $gs_seq_set(\mathbb{Z}) / \Phi_3$ AUF COSTAS-ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n ≥ 1
RETURNS {p:Seq(ℤ) | perm(p, {1..n})
          ∧ ∀j < |p|. nodups(dtrow(p, j))}
```

D	\mapsto	$Set(\alpha)$
R	\mapsto	$Seq(\alpha)$
I	\mapsto	$\lambda M. true$
O	\mapsto	$\lambda M, L. range(L) \subseteq M$
S	\mapsto	$Seq(\alpha)$
J	\mapsto	$\lambda M, s. range(s) \subseteq M$
s_0	\mapsto	$\lambda M. []$
sat	\mapsto	$\lambda L, s. s \subseteq L$
$split$	\mapsto	$\lambda M, s. \{s \cdot a \mid a \in M\}$
ext	\mapsto	$\lambda s. \{s\}$

1. Bildbereich stimmt mit $R_G = Seq(\mathbb{Z})$ überein
2. Eingabebereiche \mathbb{Z} , $D_G = Set(\mathbb{Z})$ sind anzupassen
3. Keine Eingabebedingung zu prüfen: $I_G(M) = true$
4. Zu zeigen ist also:

$\forall n: \mathbb{Z}. n \geq 1 \Rightarrow \exists M: Set(\mathbb{Z}). \forall p: Seq(\mathbb{Z}). O(n, p) \Rightarrow range(p) \subseteq M$

- Heuristik: Suche Folgerungen von $O(n, p)$, in denen $range(p)$ vorkommt
- Auffalten von $perm$ liefert: $perm(p, \{1..n\}) \Rightarrow range(p) \subseteq \{1..n\}$
- Wähle $M \equiv \{1..n\}$ und extrahiere $\vartheta = \lambda n. \{1..n\}$

5. Modifiziere $gs_seq_set(\mathbb{Z})$ und Φ_3 mit ϑ und der Spezifikation

$G_\vartheta = (\mathbb{Z}, Seq(\mathbb{Z}), \lambda n. n \geq 1, O, Seq(\mathbb{Z}), \lambda n, s. range(s) \subseteq \{1..n\},$
 $\lambda n. [], \lambda L, s. s \subseteq L, \lambda n, s. \{s \cdot a \mid a \in \{1..n\}\}, \lambda s. \{s\})$

$\Phi_{3,\vartheta}(n, s) = \Phi_3(\{1..n\}, s) = nodups(s)$

$\Phi_{3,\vartheta}$ ist notwendig für G_ϑ

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x : D)$ WHERE $I[x]$ RETURNS $\{y : R \mid O[x, y]\}$

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x : D)$ WHERE $I[x]$ RETURNS $\{y : R \mid O[x, y]\}$

1. Wähle Globalsuchtheorie \mathcal{G} mit Ausgabotyp R (Wissensbank)

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x : D)$ WHERE $I[x]$ RETURNS $\{y : R \mid O[x, y]\}$

1. Wähle Globalsuchtheorie \mathcal{G} mit Ausgabety R (Wissensbank)
2. Beweise $(D, R, I, O) \ll \mathcal{G}$ und extrahiere Substitution ϑ
Verfeinere \mathcal{G} zu Globalsuchtheorie \mathcal{G}_ϑ für (D, R, I, O)

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x : D)$ WHERE $I[x]$ RETURNS $\{y : R \mid O[x, y]\}$

1. Wähle Globalsuchtheorie \mathcal{G} mit Ausgabety R (Wissensbank)
2. Beweise $(D, R, I, O) \ll \mathcal{G}$ und extrahiere Substitution ϑ
Verfeinere \mathcal{G} zu Globalsuchtheorie \mathcal{G}_ϑ für (D, R, I, O)
3. Wähle Wohlfundiertheitsfilter Φ für \mathcal{G} (Wissensbank)
Beweise ‘ Φ_ϑ notwendig für \mathcal{G}_ϑ ’ (Axiom 4)

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

1. Wähle Globalsuchtheorie \mathcal{G} mit Ausgabety R (Wissensbank)
2. Beweise $(D, R, I, O) \ll \mathcal{G}$ und extrahiere Substitution ϑ
Verfeinere \mathcal{G} zu Globalsuchtheorie \mathcal{G}_ϑ für (D, R, I, O)
3. Wähle Wohlfundiertheitsfilter Φ für \mathcal{G} (Wissensbank)
Beweise ‘ Φ_ϑ notwendig für \mathcal{G}_ϑ ’ (Axiom 4)
4. Bestimme zusätzlichen notwendigen Filter Ψ für \mathcal{G}_ϑ
– Leite Eigenschaften von x und s aus $sat[z, s] \wedge O[x, z]$ ab (Vorwärtsinferenz)

SYNTHESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$

1. Wähle Globalsuchtheorie \mathcal{G} mit Ausgabety R (Wissensbank)

2. Beweise $(D, R, I, O) \ll \mathcal{G}$ und extrahiere Substitution ϑ
Verfeinere \mathcal{G} zu Globalsuchtheorie \mathcal{G}_ϑ für (D, R, I, O)

3. Wähle Wohlfundiertheitsfilter Φ für \mathcal{G} (Wissensbank)

Beweise ‘ Φ_ϑ notwendig für \mathcal{G}_ϑ ’ (Axiom 4)

4. Bestimme zusätzlichen notwendigen Filter Ψ für \mathcal{G}_ϑ

– Leite Eigenschaften von x und s aus $sat[z,s] \wedge O[x,z]$ ab (Vorwärtsinferenz)

5. Instantiiere Globalsuch-Schema mit $\mathcal{G}_\vartheta, \Phi_\vartheta, \Psi$

FUNCTION $f(x:D)$ WHERE $I[x]$ RETURNS $\{y:R \mid O[x,y]\}$
 \equiv if $\Phi[\vartheta(x), s_0(\vartheta(x))] \wedge \Psi[x, s_0(\vartheta(x))]$ then $aux(x, s_0(\vartheta(x)))$ else \emptyset

FUNCTION $aux(x, s:D \times S)$ WHERE $I[x] \wedge J[\vartheta(x), s] \wedge \Phi[\vartheta(x), s] \wedge \Psi[x, s]$

RETURNS $\{y:R \mid O[x,y] \wedge sat[y,s]\}$

$\equiv \{z \mid z \in ext[s] \wedge O[x,z]\} \cup \bigcup \{aux(x,t) \mid t \in split[\vartheta(x), s] \wedge \Phi[\vartheta(x), t] \wedge \Psi[x,t]\}$

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

```
FUNCTION Costas (n:ℤ) WHERE n ≥ 1  
RETURNS {p:Seq(ℤ) | perm(p, {1..n}) ∧ ∀j < |p|. nodups(dtrow(p, j))}
```

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

FUNCTION `Costas` ($n:\mathbb{Z}$) WHERE $n \geq 1$
RETURNS $\{p:\text{Seq}(\mathbb{Z}) \mid \text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j))\}$

1. Wähle Globalsuchtheorie $\mathcal{G} = \text{gs_seq_set}(\mathbb{Z})$

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

FUNCTION *Costas* ($n:\mathbb{Z}$) WHERE $n \geq 1$
RETURNS $\{p:\text{Seq}(\mathbb{Z}) \mid \text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j))\}$

1. Wähle Globalsuchtheorie $\mathcal{G} = \text{gs_seq_set}(\mathbb{Z})$

2. Beweis für $(D, R, I, O) \ll \text{gs_seq_set}(\mathbb{Z})$ liefert $\vartheta[n] = \{1..n\}$

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

FUNCTION `Costas` ($n:\mathbb{Z}$) WHERE $n \geq 1$
RETURNS $\{p:\text{Seq}(\mathbb{Z}) \mid \text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j))\}$

1. Wähle Globalsuchtheorie $\mathcal{G} = \text{gs_seq_set}(\mathbb{Z})$
2. Beweis für $(D, R, I, O) \ll \text{gs_seq_set}(\mathbb{Z})$ liefert $\vartheta[n] = \{1..n\}$
3. Wähle WF-Filter Φ so, daß Φ_ϑ notwendig für G_ϑ beweisbar
 - $\text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j)) \wedge s \sqsubseteq p \Rightarrow \Phi[\{1..n\}, s]$
 - Leicht beweisbar nur für $\Phi_3[M, s] = \text{nodups}(s)$

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

FUNCTION *Costas* ($n:\mathbb{Z}$) WHERE $n \geq 1$
RETURNS $\{p:\text{Seq}(\mathbb{Z}) \mid \text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j))\}$

1. Wähle Globalsuchtheorie $\mathcal{G} = \text{gs_seq_set}(\mathbb{Z})$

2. Beweis für $(D, R, I, O) \ll \text{gs_seq_set}(\mathbb{Z})$ liefert $\vartheta[n] = \{1..n\}$

3. Wähle WF-Filter Φ so, daß Φ_{ϑ} notwendig für G_{ϑ} beweisbar

- $\text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j)) \wedge s \sqsubseteq p \Rightarrow \Phi[\{1..n\}, s]$
- Leicht beweisbar nur für $\Phi_3[M, s] = \text{nodups}(s)$

4. Leite zusätzlichen notwendigen Filter Ψ ab

- Aus $\text{perm}(p, \{1..n\}) \wedge \forall j < |p| . \text{nodups}(\text{dtrow}(p, j)) \wedge s \sqsubseteq p$
leite ab $\Psi[n, s] = \forall i < |s| . \text{nodups}(\text{dtrow}(s, i))$

GLOBALSUCHALGORITHMUS FÜR COSTAS ARRAYS

FUNCTION `Costas` ($n:\mathbb{Z}$) WHERE $n \geq 1$
RETURNS $\{p:\text{Seq}(\mathbb{Z}) \mid \text{perm}(p, \{1..n\}) \wedge \forall j < |p|. \text{nodups}(\text{dtrow}(p, j))\}$

1. Wähle Globalsuchtheorie $\mathcal{G} = \text{gs_seq_set}(\mathbb{Z})$

2. Beweis für $(D, R, I, O) \ll \text{gs_seq_set}(\mathbb{Z})$ liefert $\vartheta[n] = \{1..n\}$

3. Wähle WF-Filter Φ so, daß Φ_ϑ notwendig für G_ϑ beweisbar

- $\text{perm}(p, \{1..n\}) \wedge \forall j < |p|. \text{nodups}(\text{dtrow}(p, j)) \wedge s \sqsubseteq p \Rightarrow \Phi[\{1..n\}, s]$
- Leicht beweisbar nur für $\Phi_3[M, s] = \text{nodups}(s)$

4. Leite zusätzlichen notwendigen Filter Ψ ab

- Aus $\text{perm}(p, \{1..n\}) \wedge \forall j < |p|. \text{nodups}(\text{dtrow}(p, j)) \wedge s \sqsubseteq p$
leite ab $\Psi[n, s] = \forall i < |s|. \text{nodups}(\text{dtrow}(s, i))$

5. Instantiiere den Standard-Globalsuchalgorithmus

```
FUNCTION Costas ( $n:\mathbb{Z}$ ) WHERE  $n \geq 1$  RETURNS  $\{p:\text{Seq}(\mathbb{Z}) \mid \dots\}$   
 $\equiv$  let rec Costasgs( $n, s$ )  
    =  $\{p \mid p \in \{s\} \wedge \text{perm}(p, \{1..n\}) \wedge \forall j < |p|. \text{nodups}(\text{dtrow}(p, j))\}$   
       $\cup \bigcup \{\text{Costas}_{gs}(n, t) \mid t \in \{s \cdot i \mid i \in \{1..n\}\}$   
               $\wedge \text{nodups}(t) \wedge \forall j < |t|. \text{nodups}(\text{dtrow}(t, j))\}$   
    in if  $\text{nodups}([]) \wedge \forall j < |[]|. \text{nodups}(\text{dtrow}([], j))$   
        then Costasgs( $n, []$ ) else  $\emptyset$ 
```