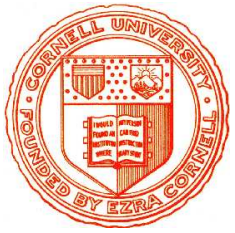
Automatisierte Logik und Programmierung

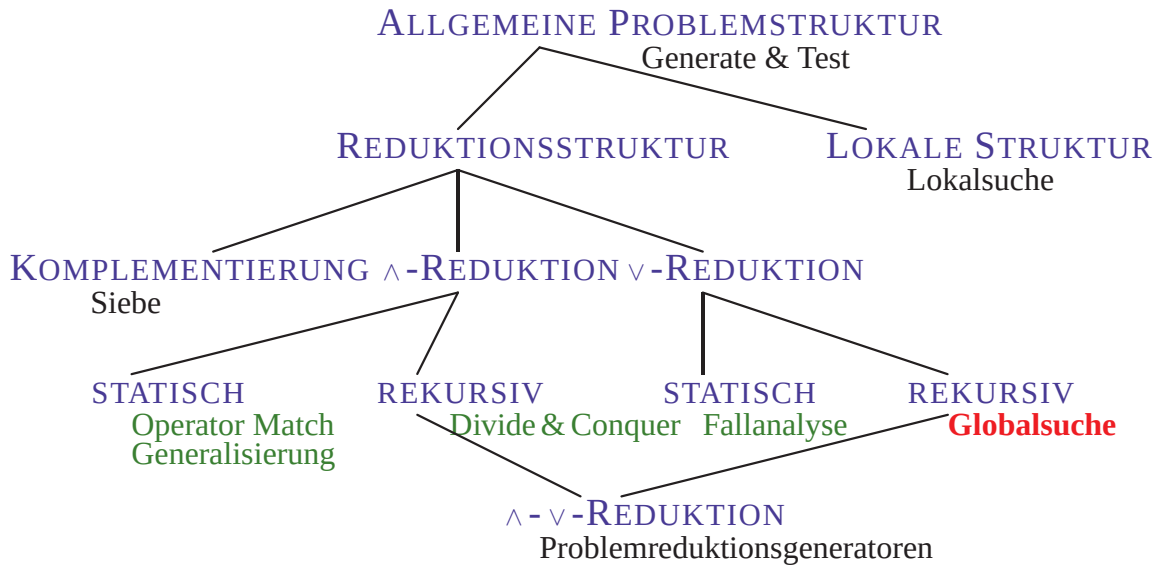
Einheit 19



Globalsuch-Algorithmen

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

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:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)`
  `  RETURNS {i:ℕ | i∈{1..|L|} ∧ L_i=k}`
- **Binäre Suche und Aufsammeln von Lösungen**
  - Spalte Liste `L` in zwei Teile `[L_i|1≤i≤m]` und `[L_i|m<i≤|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≤l≤r≤|L|`
- **Verwende Hilfsfunktion aux für rekursive Breitensuche**

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

- **Initialisiere Hilfsfunktion mit gesamter Liste**
  - `osearch(L,k) ≡ aux(L,k,1,|L|)`

Optimierung des Grundalgorithmus

- **Eliminiere Hilfsfunktionsaufrufe, wenn keine Lösung möglich**
  - Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
  - Suchraum $\{i..j\}$ ohne Lösung, falls $L_i>k$ oder $L_j<k$
  - Ergänze Filter $L_i \le k \le L_j$ 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∈{l..r} ∧ L_i=k}
≡ if l=r then if L_l=k then {l} else ∅
     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)∪aux(L,k,m+1,r)
```

  - Algorithmus hat nur noch logarithmische Laufzeit

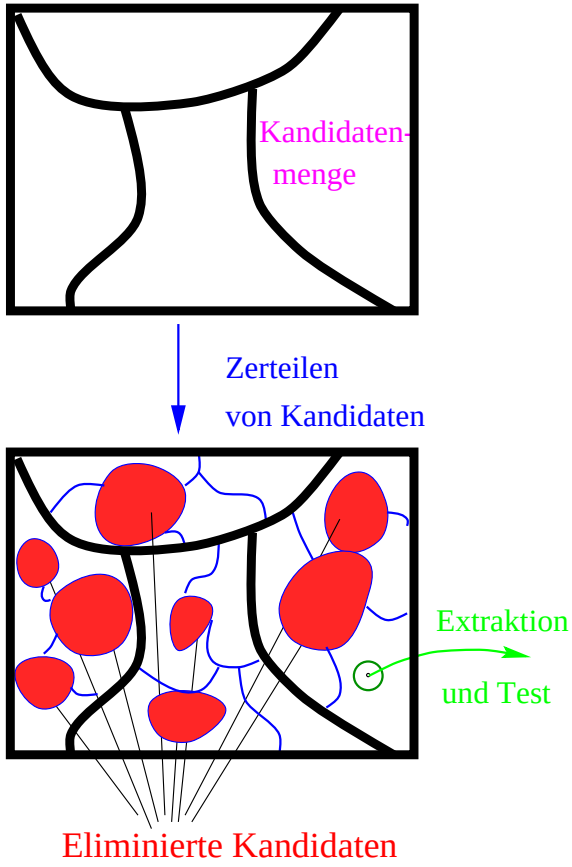
- **Ergänze Filter zur Initialisierung der Hilfsfunktion**

```
FUNCTION osearch(L,k:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)
  RETURNS {i:ℕ | i∈{1..|L|} ∧ L_i=k}
≡ if L_1≤k≤L_|L| then aux(L,k,1,|L|) else ∅
```

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(ℤ)×ℤ×ℕ×ℕ) ...
≡ if l=r then if L_l=k then {l} else ∅
      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)∪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 $\{l..l\}$ erzeugt
    - Gesamtlösung ist Vereinigung aller Einzellösungsmengen

- **Endform: wohlstrukturierter mathematischer Algorithmus**

```
FUNCTION aux(L,k,l,r:Seq(ℤ)×ℤ×ℕ×ℕ) ...
≡ {i| i∈{l} ∧ i=r ∧ L_i=k}
 ∪⋃{aux(L,k,n,m)|
    (n,m)∈{(l,(l+r)/2),((l+r)/2+1,r)|l<r} ∧ L_n≤k≤L_m}
```

Allgemeines Globalsuch-Schema

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

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

Korrektheit folgt aus wenigen Voraussetzungen

Korrektheit des Globalsuch-Schemas

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

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**
   $I[x] \wedge 0[x,z] \Rightarrow sat[z, s_0(x)]$
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**
   $I[x] \wedge 0[x,z] \wedge J[x,s] \Rightarrow (sat[z,s] \Leftrightarrow \exists k{:}\mathbb{N}.\, \exists t \in split^k_\Phi[x,s].\, z \in ext[t])$
6. **Splitting (mit Filterung) ist wohlfundiert**
   $I[x] \wedge J[x,s] \Rightarrow \exists k{:}\mathbb{N}.\; split^k_\Phi[x,s] = \emptyset$

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 ($split^i_\Phi[x,s]=\emptyset$), so ist das Resultat

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

  (Menge der aus Deskriptoren extrahierbaren Lösungen, die zu einem $split^j_\Phi[x,s]$ gehören)

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

  Beweis: Induktion über i, Auffalten der Rekursion, Standardlemmata

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

Beispiel einer Globalsuchtheorie

- **Theorie `gs_osearch` für `osearch`**

$$
\begin{array}{lll}
D & \mapsto & \mathtt{Seq}(\mathbb{N})\times\mathbb{N} \\
R & \mapsto & \mathtt{Set}(\mathbb{N}) \\
I & \mapsto & \lambda \mathtt{L,k}.\ \mathtt{L}\neq[]\wedge\mathtt{ordered(L)} \\
O & \mapsto & \lambda \mathtt{L,k,i}.\ \mathtt{i}\in\{1..|\mathtt{L}|\}\wedge \mathtt{L}_i=\mathtt{k} \\
S & \mapsto & \mathbb{N}\times\mathbb{N} \\
J & \mapsto & \lambda \mathtt{L,k,l,r}.\ 1\leq \mathtt{l}\leq \mathtt{r}\leq|\mathtt{L}| \\
s_0 & \mapsto & \lambda \mathtt{L,k}.\ (1,|\mathtt{L}|) \\
sat & \mapsto & \lambda \mathtt{i,l,r}.\ \mathtt{i}\in\{\mathtt{l..r}\} \\
split & \mapsto & \lambda \mathtt{L,k,l,r}.\ \mathtt{if\ l<r\ then}\ \{(\mathtt{l},(\mathtt{l}+\mathtt{r})/2),((\mathtt{l}+\mathtt{r})/2+1,\mathtt{r})\}\ \mathtt{else}\ \emptyset \\
\Phi & \mapsto & \lambda \mathtt{L,k,l,r}.\ \mathtt{L}_l\leq\mathtt{k}\leq\mathtt{L}_r \\
ext & \mapsto & \lambda \mathtt{l,r}.\ \mathtt{if\ l=r\ then}\ \{\mathtt{l}\}\ \mathtt{else}\ \emptyset
\end{array}
$$

- **Alle 6 Axiome sind erfüllt**

1. $\mathtt{L}\neq[]\wedge\mathtt{ordered(L)} \Rightarrow 1\leq 1\leq|\mathtt{L}|\leq|\mathtt{L}|$
2. $\ldots \wedge 1\leq\mathtt{l}\leq\mathtt{r}\leq|\mathtt{L}| \Rightarrow \forall(\mathtt{x},\mathtt{y})\in split[\mathtt{L,k,l,r}].1\leq\mathtt{x}\leq\mathtt{y}\leq|\mathtt{L}|$
3. $\ldots \wedge \mathtt{i}\in\{1..|\mathtt{L}|\}\wedge\mathtt{L}_i=\mathtt{k} \Rightarrow \mathtt{i}\in\{1..|\mathtt{L}|\}$
4. $\ldots \wedge 1\leq\mathtt{l}\leq\mathtt{r}\leq|\mathtt{L}| \Rightarrow \mathtt{L}_l\leq\mathtt{k}\leq\mathtt{L}_r \Leftarrow \exists\mathtt{z}{:}\mathbb{N}.\mathtt{z}\in\{\mathtt{l..r}\}\wedge\mathtt{z}\in\{1..|\mathtt{L}|\}\wedge\mathtt{L}_z=\mathtt{k}$
5. $\ldots \wedge 1\leq\mathtt{l}\leq\mathtt{r}\leq|\mathtt{L}|\wedge\mathtt{i}\in\{1..|\mathtt{L}|\}\wedge\mathtt{L}_i=\mathtt{k} \Rightarrow$
   $\mathtt{i}\in\{\mathtt{l..r}\} \Leftrightarrow \exists\mathtt{k}{:}\mathbb{N}.\exists(\mathtt{x},\mathtt{y})\in split^{\mathtt{k}}_{\Phi}[\mathtt{L,k,l,r}].\mathtt{i}\in(\mathtt{if\ x=y\ then}\{\mathtt{x}\}\ \mathtt{else}\ \emptyset)$
6. $\ldots \wedge 1\leq\mathtt{l}\leq\mathtt{r}\leq|\mathtt{L}| \Rightarrow \exists\mathtt{k}{:}\mathbb{N}.split^{\mathtt{k}}_{\Phi}[\mathtt{L,k,l,r}]=\emptyset$

Schematischer Globalsuchalgorithmus für osearch

```
FUNCTION osearch(L,k:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)
   RETURNS {i:ℕ | i∈{1..|L|}∧L_i=k}
≡ let rec aux(L,k,l,r)
   =     {z | z∈(if l=r then {l} else ∅) ∧ z∈{1..|L|} ∧ L_z=k}
      ∪⋃{ aux(L,k,n,m) |
           (n,m)∈(if l<r then{(l,(l+r)/2),((l+r)/2+1,r)}
                         else ∅)  ∧  L_n≤k≤L_m                }
   in  if L_1≤k≤L_|L| then aux(L,k,1,|L|) else ∅
```

Nach (kontextabhängigen) Simplifikationen

```
FUNCTION osearch(L,k:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)
   RETURNS {i:ℕ| i∈{1..|L|}∧L_i=k}
≡ let rec aux(L,k,l,r)
   = if l=r then if L_l=k then {l} else ∅
       else let m = (l+r)/2 in
              (if k≤L_m   then aux(L,k,l,m) else ∅)
            ∪ (if L_m+1≤k  then aux(L,k,m+1,r) else ∅)
   in  if L_1≤k≤L_|L| then aux(L,k,1,|L|) else ∅
```

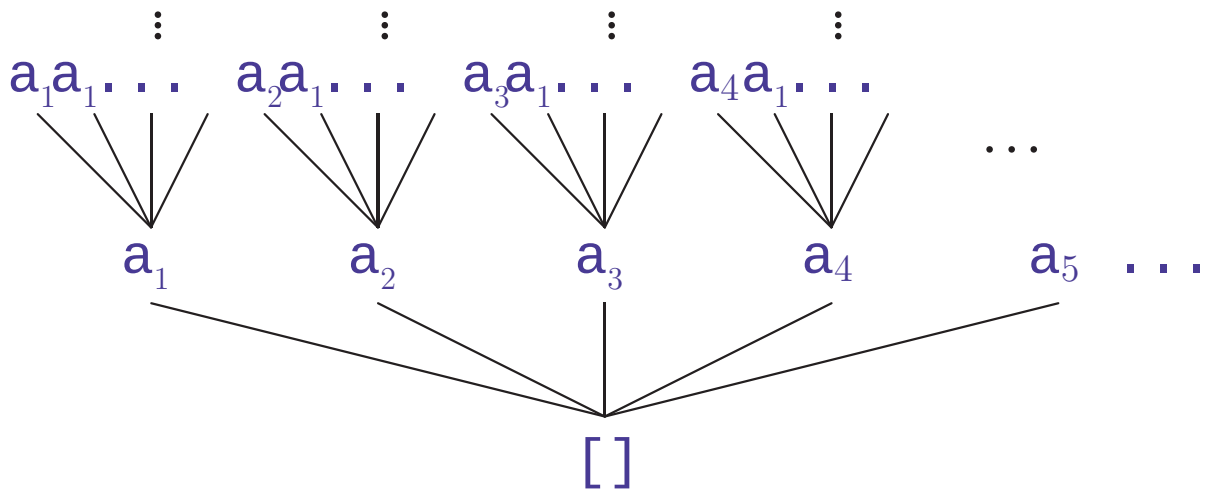
Wie erzeugt man Globalsuch-Algorithmen?

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie: allgemeine Suchstruktur für R**
  - Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt
  - Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \mathsf{True}, ext)$
  - 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 ↦ 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 $\vartheta: D \to D_{\mathcal{G}}$
  - 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** $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor: leerer Präfix** $\quad s_0(M) \equiv [\,]$
- **Splitting: Verlängern des Präfix** $\quad split[M,s] \equiv \{s{\cdot}a | a \in M\}$
- **Extraktion: Gesamter Präfix** $\quad ext[s] \equiv \{s\}$
- **Sinnvoll: nur Elemente aus M** $\quad J[M,s] \equiv s' \sqsubseteq' M$

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 — $split[M, s] \equiv \{s{\cdot}a | a \in M\}$
- Extraktion: Gesamter Präfix — $ext[s] \equiv \{s\}$
- Sinnvoll: nur Elemente aus M — $J[M, s] \equiv s \,`\subseteq' M$

Darstellung als formales Objekt der Wissensbank

`gs_seq_set(`α`)` $\equiv$

D	$\mapsto$	`Set(`α`)`
R	$\mapsto$	`Seq(`α`)`
I	$\mapsto$	λ`M. true`
O	$\mapsto$	λ`M,L. range(L)`$\subseteq$`M`
S	$\mapsto$	`Seq(`α`)`
J	$\mapsto$	λ`M,s. range(s)`$\subseteq$`M`
s_0	$\mapsto$	λ`M. []`
sat	$\mapsto$	λ`L,s. s`$\sqsubseteq$`L`
$split$	$\mapsto$	λ`M,s. {s·a|a`$\in$`M}`
ext	$\mapsto$	λ`s. {s}`

Wohlfundiertheitsfilter für `gs_seq_set(`α`)`

- $\Phi_1[M, s] \equiv |s| \leq k$
  - 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 $|M|^{k*|M|}$
- $\Phi_3[M, s] \equiv$ `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(`α`)` 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
-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} | i$\in$[1..|L|-j]]
  perm(L,S) $\equiv$ nodups(L) $\wedge$ range(L)=S

- **Spezifikation des Problems**

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

Mathematische Gesetze für Costas Arrays

- **Gesetze von Differenzentafeln**

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

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

$\vdots$

- **Gesetze duplikatenfreier Listen**

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

1. $nodups([])$
2. $nodups(i.L) = nodups(L) \wedge i \notin L$
3. $nodups(L \cdot i) = nodups(L) \wedge i \notin L$
4. $L \sqsubseteq L' \Rightarrow nodups(L') \Rightarrow nodups(L)$

$\vdots$

Spezialisiere gs_seq_set(ℤ) / Φ_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	↦	Set(α)
R	↦	Seq(α)
I	↦	λM. true
O	↦	λM,L. range(L)⊆M
S	↦	Seq(α)
J	↦	λM,s. range(s)⊆M
s_0	↦	λM. []
sat	↦	λL,s. s⊑L
$split$	↦	λM,s. {s·a|a∈M}
ext	↦	λs. {s}

1. **Bildbereich stimmt mit $R_\mathcal{G}$=Seq(ℤ) überein**
2. **Eingabebereiche ℤ, $D_\mathcal{G}$=Set(ℤ) sind anzupassen**
3. **Keine Eingabebedingung zu prüfen: $I_\mathcal{G}(M)$=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}) ⇒ range(p)⊆{1..n}

   – Wähle M ≡ {1..n} und extrahiere ϑ = λn.{1..n}

5. **Modifiziere gs_seq_set(ℤ) und Φ_3 mit ϑ und der Spezifikation**

   G_ϑ = (ℤ, Seq(ℤ), λn.n≥1, O, Seq(ℤ), λn,s.range(s)⊆{1..n},
   λn.[], λL,s. s⊑L, λn,s.{s·a|a∈{1..n}}, λs.{s})

   $\Phi_{3,\vartheta}$(n,s) = Φ_3({1..n},s) = nodups(s)

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

Synthesesestrategie 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 Ausgabetyp R** (Wissensbank)
2. **Beweise $(D,R,I,O) \ll \mathcal{G}$ und extrahiere Substitution ϑ**
   Verfeinere $\mathcal{G}$ zu Globalsuchtheorie $\mathcal{G}_\vartheta$ für (D,R,I,O)
3. **Wähle Wohlfundiertheitsfilter Φ für $\mathcal{G}$** (Wissensbank)
   Beweise 'Φ_ϑ notwendig für $\mathcal{G}_\vartheta$' (Axiom 4)
4. **Bestimme zusätzlichen notwendigen Filter Ψ für $\mathcal{G}_\vartheta$**
   – Leite Eigenschaften von x und s aus $sat[z,s] \wedge O[x,z]$ ab (Vorwärtsinferenz)
5. **Instantiiere Globalsuch-Schema mit $\mathcal{G}_\vartheta$, Φ_ϑ, Ψ**

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

1. **Wähle Globalsuchtheorie $\mathcal{G}$ = gs_seq_set($\mathbb{Z}$)**
2. **Beweis für $(D, R, I, O) \ll$ gs_seq_set($\mathbb{Z}$) liefert $\vartheta[n] = \{1..n\}$**
3. **Wähle WF-Filter Φ so, daß Φ_ϑ notwendig für G_ϑ beweisbar**
   – perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) $\wedge$ s$\sqsubseteq$p $\Rightarrow$ Φ[{1..n},s]
   – Leicht beweisbar nur für Φ_3[M,s] = nodups(s)
4. **Leite zusätzlichen notwendigen Filter Ψ ab**
   – Aus perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) $\wedge$ s$\sqsubseteq$p
     leite ab Ψ[n,s] = $\forall$i<|s|.nodups(dtrow(s,j))
5. **Instantiiere den Standard-Globalsuchalgorithmus**

```
FUNCTION Costas (n:ℤ) WHERE n≥1 RETURNS {p:Seq(ℤ)| ... }
≡ let rec Costas_gs(n,s)
    =      {p | p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
      ∪ ⋃{ Costas_gs(n,t) | t∈{s·i|i∈{1..n}}
                          ∧ nodups(t) ∧ ∀j<|t|.nodups(dtrow(t,j))}
    in if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
       then Costas_gs(n,[]) else ∅
```