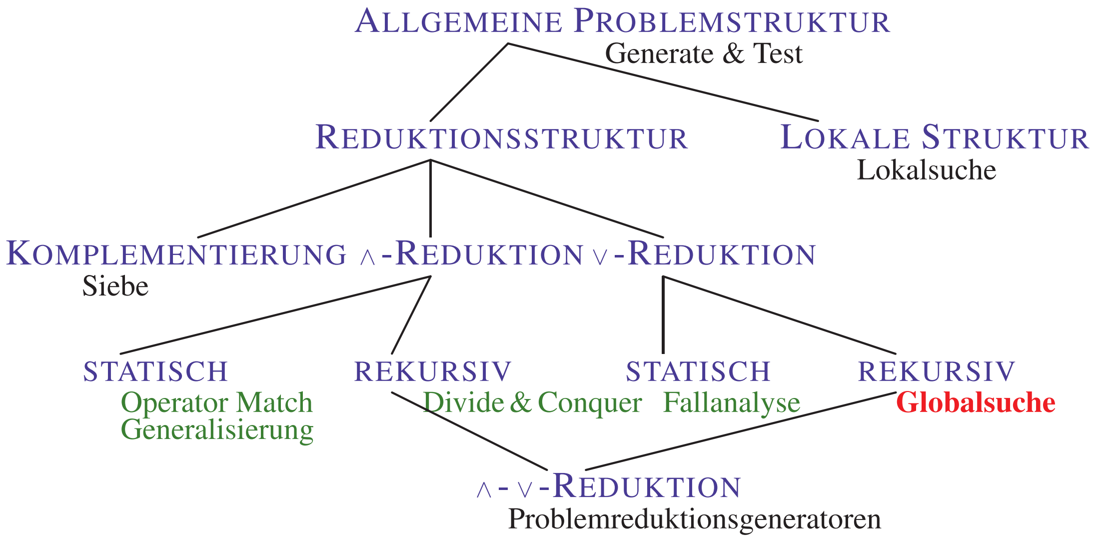
Automatisierte Logik und Programmierung

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 $[L_i | 1 \leq i \leq m]$ und $[L_i | m < i \leq |L|]$
  – 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: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 $L_i\leq k\leq L_j$ zu rekursivem Aufruf

$$
\begin{array}{l}
\texttt{FUNCTION } aux\texttt{(L,k,l,r:Seq(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}\texttt{)} \\
\quad \texttt{WHERE L}\neq\texttt{[]} \wedge \texttt{ordered(L)} \wedge 1\leq\texttt{l}\leq\texttt{r}\leq|\texttt{L}| \\
\quad \texttt{RETURNS } \{\texttt{i:}\mathbb{N} \mid \texttt{i}\in\{\texttt{l..r}\} \wedge \texttt{L}_i\texttt{=k}\} \\
\equiv \texttt{ if l=r then if } \texttt{L}_l\texttt{=k then } \{\texttt{l}\} \texttt{ else } \emptyset \\
\qquad \texttt{else let m=(l+r)/2} \\
\qquad\qquad \texttt{in if } \texttt{L}_m\texttt{<k then } aux\texttt{(L,k,m+1,r)} \\
\qquad\qquad \texttt{elseif } \texttt{L}_{m+1}\texttt{>k then } aux\texttt{(L,k,l,m)} \\
\qquad\qquad \texttt{else } aux\texttt{(L,k,l,m)} \cup aux\texttt{(L,k,m+1,r)}
\end{array}
$$

  - Algorithmus hat nur noch logarithmische Laufzeit

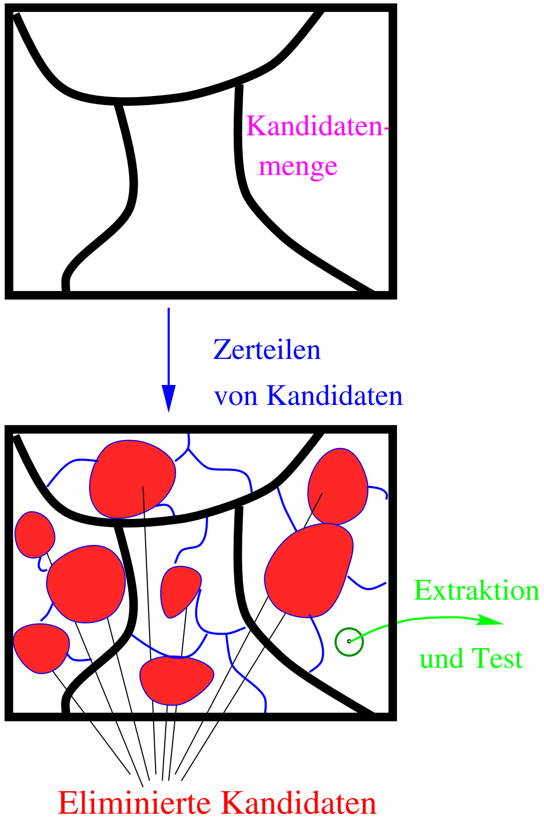
- **Ergänze Filter zur Initialisierung der Hilfsfunktion**

$$
\begin{array}{l}
\texttt{FUNCTION osearch(L,k:Seq(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{) WHERE L}\neq\texttt{[]} \wedge \texttt{ordered(L)} \\
\quad \texttt{RETURNS } \{\texttt{i:}\mathbb{N} \mid \texttt{i}\in\{\texttt{1..}|\texttt{L}|\} \wedge \texttt{L}_i\texttt{=k}\} \\
\equiv \texttt{ if } \texttt{L}_1\leq\texttt{k}\leq\texttt{L}_{|L|} \texttt{ then } aux\texttt{(L,k,1,}|\texttt{L}|\texttt{) else } \emptyset
\end{array}
$$

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

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

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

$$\texttt{in}\ \ \texttt{if}\ \Phi[x,s_0(x)]\ \texttt{then}\ aux(x,s_0(x))\ \texttt{else}\ \emptyset$$

- **7 zentrale Komponenten der Algorithmentheorie**
  - $s{:}S$ — Deskriptor für Kandidatenmengen
  - $s_0{:}\ D \to S$ — Initialdeskriptor
  - $split{:}D\times S \to \mathrm{Set}(S)$ — Rekursive Aufteilung von Kandidatenmengen
  - $\Phi{:}D\times S \to \mathbb{B}$ — Filter zur Elimination unnötiger Deskriptoren
  - $ext{:}S \to \mathrm{Set}(R)$ — 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

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

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

$$I[x] \wedge 0[x,z] \wedge J[x,s] \;\Rightarrow\; (sat[z,s] \;\Leftrightarrow\; \exists k{:}\mathbb{N}.\ \exists t \in split_\Phi^k[x,s].\ z \in ext[t])$$

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: 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`

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

- **Alle 6 Axiome sind erfüllt**

1. $L\neq[\,]\wedge \texttt{ordered}(L) \Rightarrow 1\leq 1\leq |L|\leq |L|$
2. $\ldots \wedge 1\leq l\leq r\leq |L| \Rightarrow \forall (x,y)\in split[L,k,l,r].\ 1\leq x\leq y\leq |L|$
3. $\ldots \wedge i\in\{1..|L|\}\wedge L_i=k \Rightarrow i\in\{1..|L|\}$
4. $\ldots \wedge 1\leq l\leq r\leq |L| \Rightarrow L_l\leq k\leq L_r \Leftarrow \exists z{:}\mathbb{N}.\ z\in\{l..r\}\wedge z\in\{1..|L|\}\wedge L_z=k$
5. $\ldots \wedge 1\leq l\leq r\leq |L| \wedge i\in\{1..|L|\}\wedge L_i=k \Rightarrow$
   $i\in\{l..r\} \Leftrightarrow \exists k{:}\mathbb{N}.\exists (x,y)\in split_\Phi^k[L,k,l,r].\ i\in(\texttt{if } x=y \texttt{ then } \{x\} \texttt{ else } \emptyset)$
6. $\ldots \wedge 1\leq l\leq r\leq |L| \Rightarrow \exists k{:}\mathbb{N}.\ split_\Phi^k[L,k,l,r]=\emptyset$

Schematischer Globalsuchalgorithmus für osearch

$$
\begin{array}{l}
\texttt{FUNCTION osearch(L,k:Seq}(\mathbb{Z})\times\mathbb{Z}\texttt{) WHERE L}\neq\texttt{[]} \wedge \texttt{ordered(L)}\\
\quad\texttt{RETURNS } \{\texttt{i}:\mathbb{N} \mid \texttt{i} \in \{1..|\texttt{L}|\} \wedge \texttt{L}_i\texttt{=k}\}\\
\equiv \texttt{ let rec } aux\texttt{(L,k,l,r)}\\
\quad = \quad \{\texttt{z} \mid \texttt{z} \in \texttt{(if l=r then } \{\texttt{l}\} \texttt{ else } \emptyset\texttt{)} \wedge \texttt{z} \in \{1..|\texttt{L}|\} \wedge \texttt{L}_z\texttt{=k}\}\\
\qquad \cup \bigcup\{ aux\texttt{(L,k,n,m)} \mid\\
\qquad\qquad \texttt{(n,m)} \in \texttt{(if l<r then} \{\texttt{(l,(l+r)/2), ((l+r)/2+1,r)}\}\\
\qquad\qquad\qquad\qquad \texttt{else } \emptyset\texttt{)} \quad \wedge \quad \texttt{L}_n \leq \texttt{k} \leq \texttt{L}_m \qquad \}\\
\quad \texttt{in if L}_1 \leq \texttt{k} \leq \texttt{L}_{|L|} \texttt{ then } aux\texttt{(L,k,1,}|\texttt{L}|\texttt{) else } \emptyset
\end{array}
$$

Nach (kontextabhängigen) Simplifikationen

$$
\begin{array}{l}
\texttt{FUNCTION osearch(L,k:Seq}(\mathbb{Z})\times\mathbb{Z}\texttt{) WHERE L}\neq\texttt{[]} \wedge \texttt{ordered(L)}\\
\quad\texttt{RETURNS } \{\texttt{i}:\mathbb{N} \mid \texttt{i} \in \{1..|\texttt{L}|\} \wedge \texttt{L}_i\texttt{=k}\}\\
\equiv \texttt{ let rec } aux\texttt{(L,k,l,r)}\\
\quad \texttt{= if l=r then if L}_l\texttt{=k then } \{\texttt{l}\} \texttt{ else } \emptyset\\
\qquad\quad \texttt{else let m = (l+r)/2 in}\\
\qquad\qquad\quad \texttt{(if k} \leq \texttt{L}_m \texttt{ then } aux\texttt{(L,k,l,m) else } \emptyset\texttt{)}\\
\qquad\qquad \cup \texttt{ (if L}_{m+1} \leq \texttt{k then } aux\texttt{(L,k,m+1,r) else } \emptyset\texttt{)}\\
\quad \texttt{in if L}_1 \leq \texttt{k} \leq \texttt{L}_{|L|} \texttt{ then } aux\texttt{(L,k,1,}|\texttt{L}|\texttt{) else } \emptyset
\end{array}
$$

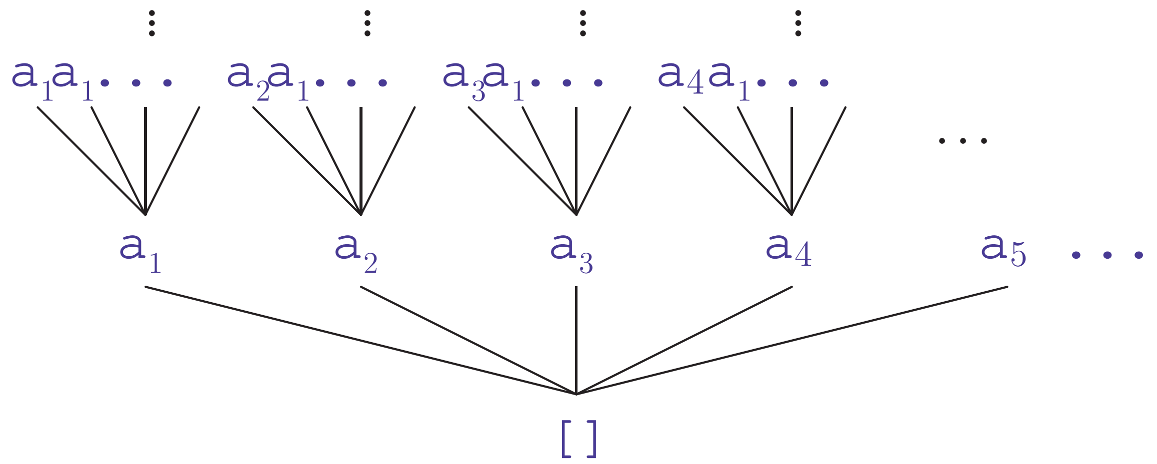
WIE ERZEUGT MAN GLOBALSUCH-ALGORITHMEN?

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 $\mapsto$ 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 — $sat[L,s] \equiv s \sqsubseteq L$
- **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 — $J[M,s] \equiv s\ '\!\subseteq'\ M$

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

$$
\texttt{gs\_seq\_set}(\alpha) \equiv
\begin{array}{lll}
D & \mapsto & \texttt{Set}(\alpha) \\
R & \mapsto & \texttt{Seq}(\alpha) \\
I & \mapsto & \lambda\texttt{M}.\ \texttt{true} \\
O & \mapsto & \lambda\texttt{M,L}.\ \texttt{range(L)} \subseteq \texttt{M} \\
S & \mapsto & \texttt{Seq}(\alpha) \\
J & \mapsto & \lambda\texttt{M,s}.\ \texttt{range(s)} \subseteq \texttt{M} \\
s_0 & \mapsto & \lambda\texttt{M}.\ \texttt{[]} \\
sat & \mapsto & \lambda\texttt{L,s}.\ \texttt{s} \sqsubseteq \texttt{L} \\
split & \mapsto & \lambda\texttt{M,s}.\ \{\texttt{s}{\cdot}\texttt{a} \mid \texttt{a} \in \texttt{M}\} \\
ext & \mapsto & \lambda\texttt{s}.\ \{\texttt{s}\}
\end{array}
$$

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

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

- **Spezifikation des Problems**

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

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

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≤|L| ⇒ dtrow(i.L,j) = (i−L[j]).dtrow(L,j)
3. j≠0 ⇒ dtrow([i],j) = []
4. L⊑L' ⇒ dtrow(L,j) ⊑ dtrow(L',j)
5. j≥|L| ⇒ dtrow(L,j) = []
6. j≤|L| ⇒ dtrow(L·i,j) = dtrow(L,j)·(L[|L|+1−j]−i)

⋮

• Gesetze duplikatenfreier Listen

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

1. nodups([])
2. nodups(i.L) = nodups(L) ∧ i∉L
3. nodups(L·i) = nodups(L) ∧ i∉L
4. L⊑L' ⇒ nodups(L') ⇒ nodups(L)

⋮

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	$\mapsto$	`Set(`α`)`
R	$\mapsto$	`Seq(`α`)`
I	$\mapsto$	λ`M. true`
O	$\mapsto$	λ`M,L. range(L)⊆M`
S	$\mapsto$	`Seq(`α`)`
J	$\mapsto$	λ`M,s. range(s)⊆M`
s_0	$\mapsto$	λ`M. []`
sat	$\mapsto$	λ`L,s. s⊑L`
split	$\mapsto$	λ`M,s. {s·a|a∈M}`
ext	$\mapsto$	λ`s. {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:ℤ. n≥1` $\Rightarrow$ $\exists$`M:Set(ℤ).`$\forall$`p:Seq(ℤ).` O`(n,p)` $\Rightarrow$ `range(p)⊆M`

– Heuristik: Suche Folgerungen von O`(n,p)`, in denen `range(p)` vorkommt

– Auffalten von `perm` liefert: `perm(p,{1..n})` $\Rightarrow$ `range(p)⊆{1..n}`

– Wähle `M` $\equiv$ `{1..n}` und extrahiere ϑ = λ`n.{1..n}`

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

G_ϑ = `(ℤ,Seq(ℤ),`λ`n.n≥1,`O`, Seq(ℤ),` λ`n,s.range(s)⊆{1..n},`
λ`n.[],` λ`L,s. s⊑L,` λ`n,s.{s·a|a∈{1..n}},` λ`s.{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:ℤ) WHERE n≥1
RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
```

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

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

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

– perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) ∧ s⊑p ⇒ 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}) ∧ ∀j<|p|.nodups(dtrow(p,j)) ∧ s⊑p
leite ab Ψ[n,s] = ∀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 ∅
```