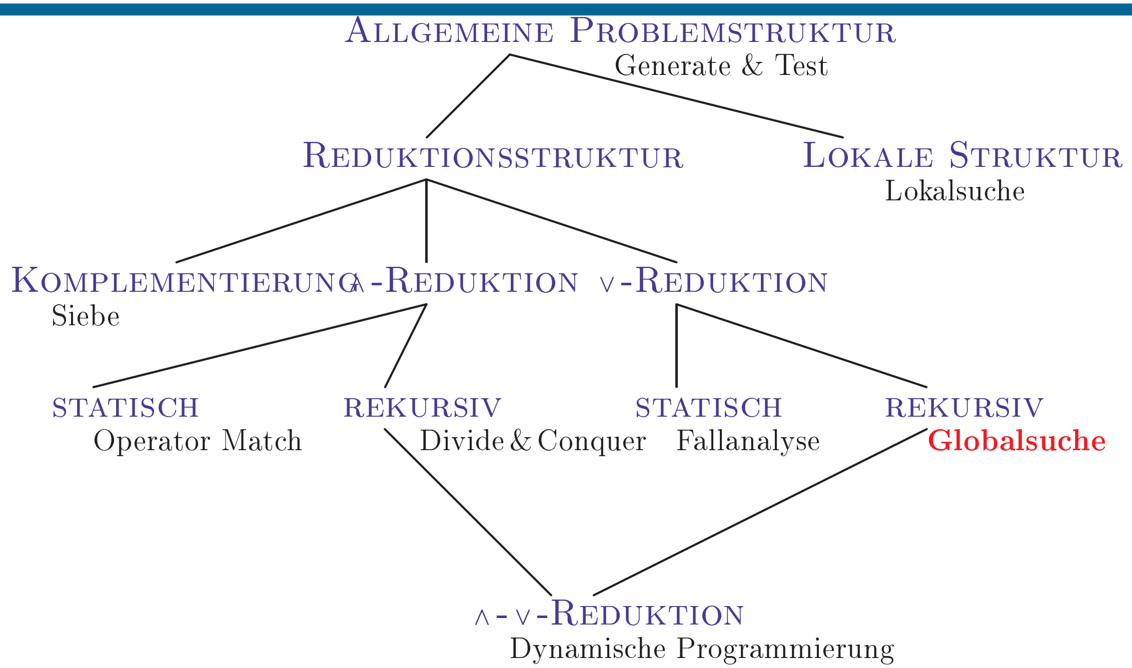
Globalsuchalgorithmen

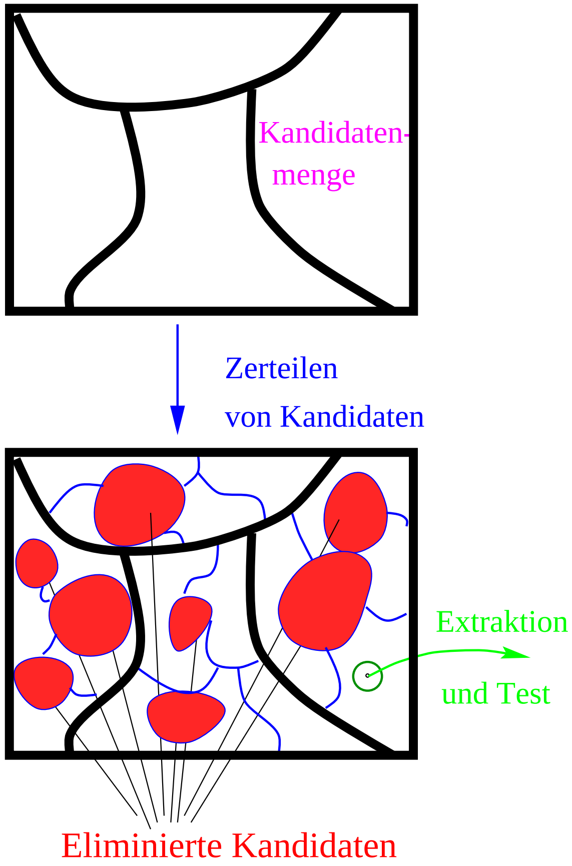


- **Bestimmung aller Lösungen eines Problems**
  - Aufzählen von Kandidaten
  - Eliminieren von Kandidaten, die keine Lösungen darstellen
  - Verallgemeinerung von Backtracking, Binärsuche, ...
- **∨-Reduktion des Problems**
  - Gesamtlösung ist Vereinigung unabhängiger Teillösungen
  - Gut geeignet für Parallelverarbeitung

Globalsuche: Generelle Idee

Durchsuchen des gesamten Bildbereichs

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



- **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?

Ein einfacher Globalsuchalgorithmus

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

  FUNCTION osearch(L,k:Seq($\mathbb{Z}$)×$\mathbb{Z}$) WHERE L≠[] ∧ ordered(L)
  RETURNS {i:$\mathbb{N}$| i∈{1..|L|}∧L_i=k}

- **Binäre Suche und Aufsammeln von Lösungen**
  - Spalte Indexmenge {1..|L|} in {1..m} und {m+1..|L|}
  - Durchsuche linke & rechte Hälfte und vereinige jeweilige Lösungsmengen

- **Vereinfache Verwaltungsaufwand mit Suchraumdeskriptoren**
  - Grenzen l und r der Indexmengen sind hinreichende Repräsentanten
  - Repräsentanten sind nur dann sinnvoll wenn 1≤l≤r≤|L|
  - Verwende Hilfsfunktion o_{aux}(L,k,l,r) mit Initialaufruf o_{aux}(L,k,1,|L|)

  FUNCTION o_{aux}(L,k,l,r:Seq($\mathbb{Z}$)×$\mathbb{Z}$×$\mathbb{N}$×$\mathbb{N}$)
  WHERE L≠[] ∧ ordered(L) ∧ 1≤l≤r≤|L|
  RETURNS {i:$\mathbb{N}$| i∈{l..r} ∧ L_i=k}

- **Hilfsfunktion durchläuft Suchraum rekursiv**

$$o_{aux}(L,k,l,r) = \begin{cases} \{l\} & \text{falls } l{=}r \;\wedge\; L_l{=}k \\ \emptyset & \text{falls } l{=}r \;\wedge\; L_l{\neq}k \\ o_{aux}(L,k,l,m) \cup o_{aux}(L,k,m{+}1,r) & \text{falls } l{<}r;\;\; m{=}(l{+}r)/2 \end{cases}$$

GLOBALSUCHALGORITHMEN: EINHEITLICHE DARSTELLUNG

- **Vereinheitlichung durch Mengendarstellung**

  $\mathtt{o}_{aux}\mathtt{(L,k,l,r)} = \{\mathtt{i} \mid \mathtt{i} \in \{\mathtt{l}\} \wedge \mathtt{i=r} \wedge \mathtt{L}_i\mathtt{=k}\}$
  $\qquad \cup \bigcup \{\mathtt{o}_{aux}\mathtt{(L,k,n,m)} \mid \mathtt{(n,m)} \in \{\mathtt{(l,(l+r)/2)}, \mathtt{((l+r)/2+1,r)} \mid \mathtt{l<r}\}\}$

  - Mengenschreibweise unabhängig von binärer Aufspaltung des Suchraums
  - (n,m) wird aus Aufspaltungsmenge ausgewählt
  - Lösungsmenge wird durch Vereinigung einer Lösungsfamilie gebildet
  - Direkte Lösung wird durch Extraktion aus $\{\mathtt{l}\} = \{\mathtt{l..l}\}$ erzeugt

- **Optimierung durch Einsatz von Filtern**
  - Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
  - Suchraum $\{\mathtt{n..m}\}$ ohne Lösung, falls $\mathtt{L}_n\mathtt{>k}$ oder $\mathtt{L}_m\mathtt{<k}$
  - Ergänze Filter $\mathtt{L}_n \leq \mathtt{k} \leq \mathtt{L}_m$ für Aufspaltungsmenge (logarithmischer Algorithmus)

- **Endform: effizienter, wohlstrukturierter Algorithmus**

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

Allgemeines Globalsuch-Schema

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

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

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

- **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
  - $ext{:}S \to R$ — 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

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

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

$\quad \mathtt{in}\ \ \mathtt{if}\ \Phi[x, s_0(x)]\ \ \mathtt{then}\ \ f_{gs}(x, s_0(x))\ \ \mathtt{else}\ \ \emptyset$

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 f_{gs}**

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

  $\texttt{FUNCTION}\ f_{gs}(x,s{:}D{\times}S)\ \texttt{WHERE}\ I[x]\wedge J[x,s]\wedge\Phi[x,s]\ \texttt{RETURNS}\ \{y{:}R|O[x,y]\wedge sat[y,s]\}$
  $\equiv\ \{\,z \mid z\in ext[s]\wedge O[x,z]\,\}\ \cup\ \bigcup\{\,f_{gs}(x,t) \mid t\in split[x,s]\wedge\ \Phi[x,t]\,\}$

- **Korrektheit von f folgt aus der von f_{gs} 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 f_{gs} folgt aus Axiom 5 & 2**

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

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

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

  (Lösungen, die aus Deskriptoren extrahierbar sind , die zu einem $split_\Phi^j[x,s]$ gehören)

  Beweis: Induktion über i, Auffalten der Rekursion, Standardlemmata

- **Terminierung von f_{gs} folgt aus Axiom 6**

Beispiel einer Globalsuchtheorie

- **Theorie gs_osearch für osearch**

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

- **Alle 6 Axiome sind erfüllt**

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

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 f_gs(L,k,l,r)
   =      {z | z∈(if l=r then {l} else ∅) ∧ z∈{1..|L|} ∧ L_z=k}
        ∪ ⋃{ o_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 o_aux(L,k, 1,|L|) else ∅
```

Nach Optimierung durch Simplifikationen

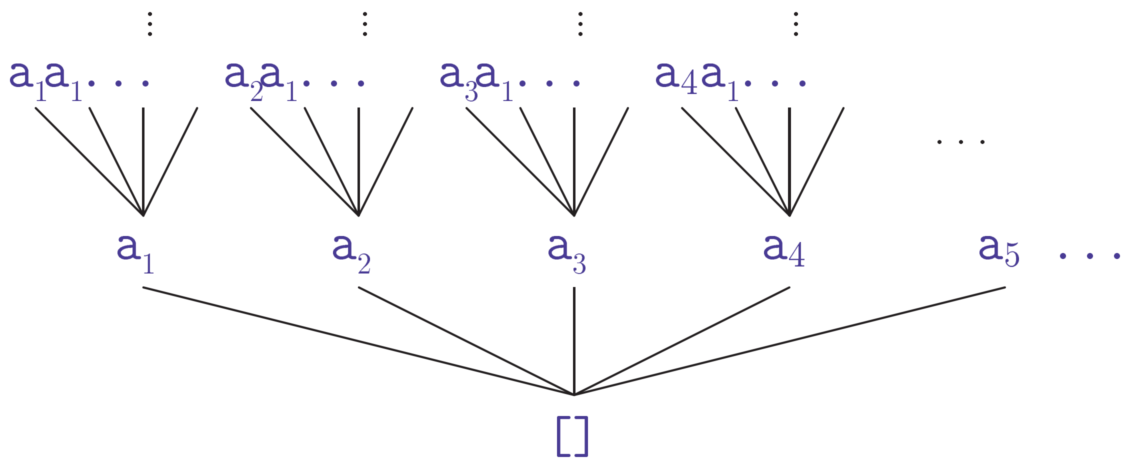
```
FUNCTION osearch(L,k:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)
   RETURNS {i:ℕ | i∈{1..|L|} ∧ L_i=k}
≡  let rec f_gs(L,k,l,r)
   = if l=r then if L_l=k then {l} else ∅
        else let m = (l+r)/2 in
               (if L_l≤k≤L_m then o_aux(L,k,l,m) else ∅)
             ∪ (if L[m+1]≤k≤L_r then o_aux(L,k,m+1,r)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

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 $\mapsto$ Axiom 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 $\theta{:}D \rightarrow D_{\mathcal{G}}$
  - 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 $\quad sat[L,s] \equiv s \sqsubseteq L$
- **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 $\quad J[M,s] \equiv s\,`\subseteq'\,M$

GS-Theorie für Listen über endlicher Menge $M\subseteq\alpha$

- 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 $\quad J[M,s] \equiv s\,`\!\subseteq'\,M$

Darstellung als formales Objekt der Wissensbank

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

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: Berechnung aller Costas Arrays der Größe n**

- **Formalisierung vorkommender Begriffe:**

  dtrow(L,j) $\equiv$ [L[i]-L[i+j] | i ∈ [1..|L|-j]]
  perm(L,S) $\equiv$ nodups(L) ∧ range(L)=S

- **Spezifikation des Problems**

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

SPEZIALISIERE gs_seq_set($\mathbb{Z}$) UND Φ_3 AUF COSTAS-ARRAYS

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

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

1. **Bildbereich stimmt mit $R_{\mathcal{G}}$=Seq($\mathbb{Z}$) überein**
2. **Eingabebereiche $\mathbb{Z}$, $D_{\mathcal{G}}$=Set($\mathbb{Z}$) 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}) $\Rightarrow$ range(p)$\subseteq${1..n}
   - Wähle M $\equiv$ {1..n} und extrahiere θ = λn.{1..n}
5. **Modifiziere gs_seq_set($\mathbb{Z}$) und Φ_3 mit θ und der Spezifikation**

   G_θ = ($\mathbb{Z}$,Seq($\mathbb{Z}$),λn.n$\geq$1,O, Seq($\mathbb{Z}$), λn,s.range(s)$\subseteq${1..n},
   λn.[], λL,s. s$\sqsubseteq$L, λn,s.{V·a|a$\in${1..n}}, λs.{s})

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

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

SYNTHESESESTRATEGIE FÜR GLOBALSUCHALGORITHMEN

Gegeben eine Problemspezifikation

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

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 $\mathcal{G}_\theta$ für (D,R,I,O)

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

Beweise 'Φ_θ notwendig für $\mathcal{G}_\theta$' (Axiom 4)

4. Bestimme zusätzlichen notwendigen Filter Ψ für $\mathcal{G}_\theta$

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

5. Instantiiere Globalsuch-Schema mit $\mathcal{G}_\theta$, Φ_θ, Ψ

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

$\equiv\ \texttt{if}\ \Phi[\theta(x),s_0(\theta(x))] \wedge \Psi[x,s_0(\theta(x))]\ \ \texttt{then}\ \ f_{gs}(x,s_0(\theta(x)))\ \ \texttt{else}\ \ \emptyset$

$\texttt{FUNCTION}\ f_{gs}(x,s{:}D{\times}S)\ \texttt{WHERE}\ I[x] \wedge J[\theta(x),s] \wedge \Phi[\theta(x),s] \wedge \Psi[x,s]$
$\texttt{RETURNS}\ \{y{:}R \mid O[x,y] \wedge sat[y,s]\}$

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

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$\in$domain(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 $\theta[n] = \{1..n\}$

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

– perm(p,{1..n}) $\wedge$ $\forall$j$\in$domain(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$\in$domain(p).nodups(dtrow(p,j)) $\wedge$ s$\sqsubseteq$p
leite ab Ψ[n,s] = $\forall$i$\in$domain(s).nodups(dtrow(s,j))

5. Instantiiere den Standard-Globalsuchalgorithmus

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

Rückblick: Synthese mit Algorithmenschemata

Wissensbasierte Techniken zur Softwareentwicklung

- **Erzeugte Algorithmen sind korrekt und effizient**
  - Formales theoretisches Fundament sichert Korrektheit
  - Gute algorithmische Struktur liefert Effizienz
  - Nachträgliche Optimierung des schematischen Algorithmus möglich/nötig
  - Mathematische Notation übersetzbar in Programmiersprachen
- **Synthesetechniken sind automatisierbar**
  - Jeder Schritt basiert auf logischer Inferenz
  - Wissen steuert alle Strategien des Algorithmenentwurfs
  - Ähnliche Techniken für Entwurf verschiedener Algorithmenstrukturen
- **Techniken sind praktisch erfolgreich**
  - KIDS erzeugt korrekte Scheduling Algorithmen in wenigen Stunden
  - Erzeugter Lisp Code 2000 mal schneller als existierende ADA Software