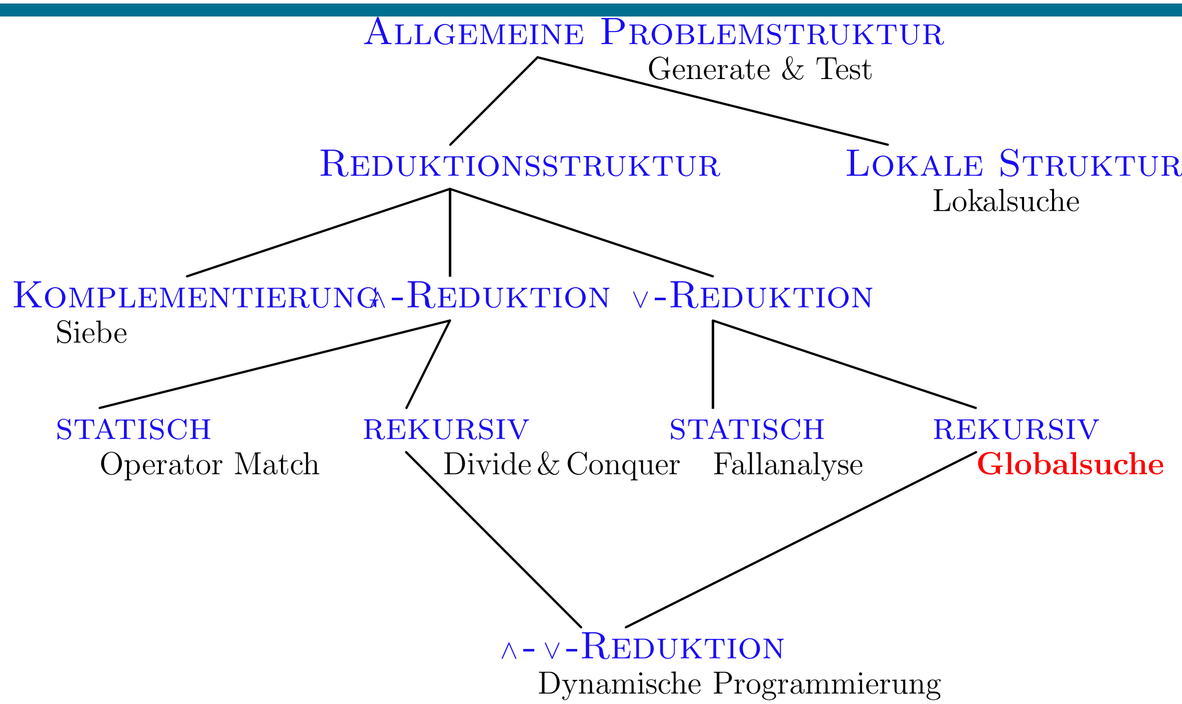
Globalsuchalgorithmen

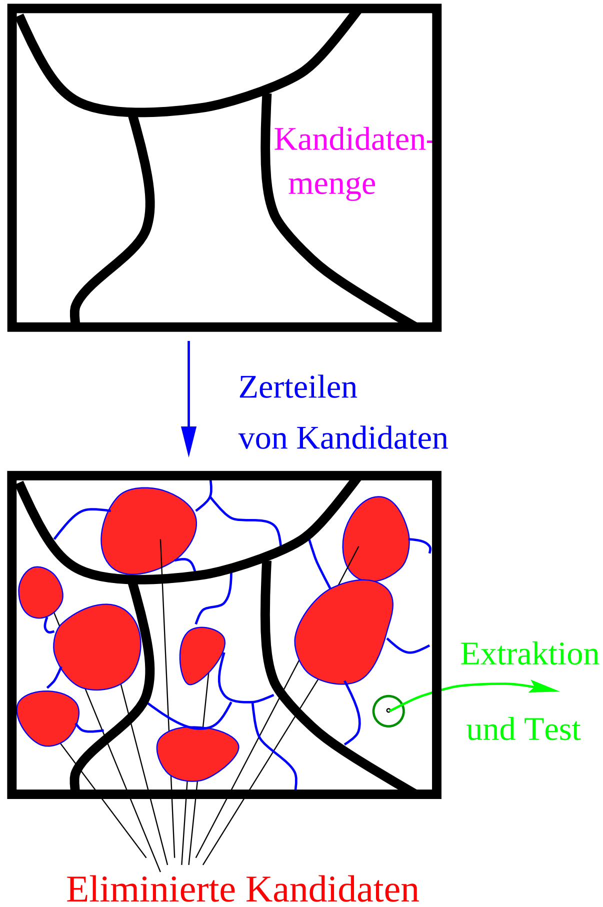


- **Bestimmung aller Lösungen eines Problems**
  - Aufzählen von Kandidaten
  - Eliminieren von Kandidaten, die keine Lösungen darstellen
  - Verallgemeinerung von Backtracking, Binärsuche, ...
- **∨-Reduktion des Problems**
  - Gesamtlösung ist Vereinigung unabhängiger Teillösungen
  - Gut geeignet 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($\mathbb{Z}$)$\times\mathbb{Z}$) WHERE L$\neq$[] $\wedge$ ordered(L)

  RETURNS $\{$ i:$\mathbb{N}$ | i$\in\{$1..|L|$\}\wedge$L$_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$\leq$l$\leq$r$\leq$|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($\mathbb{Z}$)$\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$)

    WHERE L$\neq$[] $\wedge$ ordered(L) $\wedge$ 1$\leq$l$\leq$r$\leq$|L|

    RETURNS $\{$ i:$\mathbb{N}$ | i$\in\{$l..r$\}$ $\wedge$ L$_i$=k $\}$

- **Hilfsfunktion durchläuft Suchraum rekursiv**

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

Globalsuchalgorithmen: einheitliche Darstellung

- **Vereinheitlichung durch Mengendarstellung**

  $\texttt{o}_{aux}\texttt{(L,k,l,r) = \{ i | i}\in\texttt{\{l\}}\wedge\texttt{i=r}\ \wedge\ \texttt{L}_i\texttt{=k \}}$

  $\quad\cup\bigcup\texttt{\{ o}_{aux}\texttt{(L,k,n,m) | (n,m)}\in\texttt{\{(l,(l+r)/2), ((l+r)/2+1,r) | l<r\}\}}$

  – 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 `{l} = {l..l}` erzeugt

- **Optimierung durch Einsatz von Filtern**

  – Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
  – Suchraum `{n..m}` ohne Lösung, falls $\texttt{L}_n\texttt{>k}$ oder $\texttt{L}_m\texttt{<k}$
  – Ergänze Filter $\texttt{L}_n\leq\texttt{k}\leq\texttt{L}_m$ für Aufspaltungsmenge (logarithmischer Algorithmus)

- **Endform: effizienter, wohlstrukturierter Algorithmus**

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

  $\quad\texttt{RETURNS \{ i:}\mathbb{N}\texttt{ | i}\in\texttt{\{1..|L|\}}\ \wedge\ \texttt{L}_i\texttt{=k \}}$

  $\equiv\ \texttt{let rec o}_{aux}\texttt{(L,k,l,r) = \{ i | i}\in\texttt{\{l\}}\ \wedge\ \texttt{i=r}\ \wedge\ \texttt{L}_i\texttt{=k\}}$

  $\quad\cup\bigcup\texttt{\{ o}_{aux}\texttt{(L,k,n,m) | (n,m)}\in\texttt{\{(l,(l+r)/2), ((l+r)/2+1,r) | l<r\}}$

  $\qquad\wedge\ \texttt{L}_n\leq\texttt{k}\leq\texttt{L}_m\texttt{ \}}$

  $\quad\texttt{in if L}_1\leq\texttt{k}\leq\texttt{L}_{|L|}\texttt{ then o}_{aux}\texttt{(L,k,1,|L|) else }\emptyset$

ALLGEMEINES GLOBALSUCH-SCHEMA

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

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

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

$\equiv \texttt{let rec}\ f_{gs}(x,s)\ \texttt{=}\ \{\, z \mid z \in ext[s] \wedge O[x,z]\,\} \cup \bigcup\{\, f_{gs}(x,t) \mid t \in split[x,s] \wedge \Phi[x,t]\,\}$
$\quad \texttt{in}\ \texttt{if}\ \Phi[x, s_0(x)]\ \texttt{then}\ f_{gs}(x, s_0(x))\ \texttt{else}\ \emptyset$

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

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

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: Korrektheitsbeweis

- **Abspalten und Spezifikation der Hilfsfunktion f_{gs}**

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

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

- **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**

  $split_\Phi^k[x,s] \equiv$ if k=0 then {s} 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`

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

- **Alle 6 Axiome sind erfüllt**

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

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 ∅
```

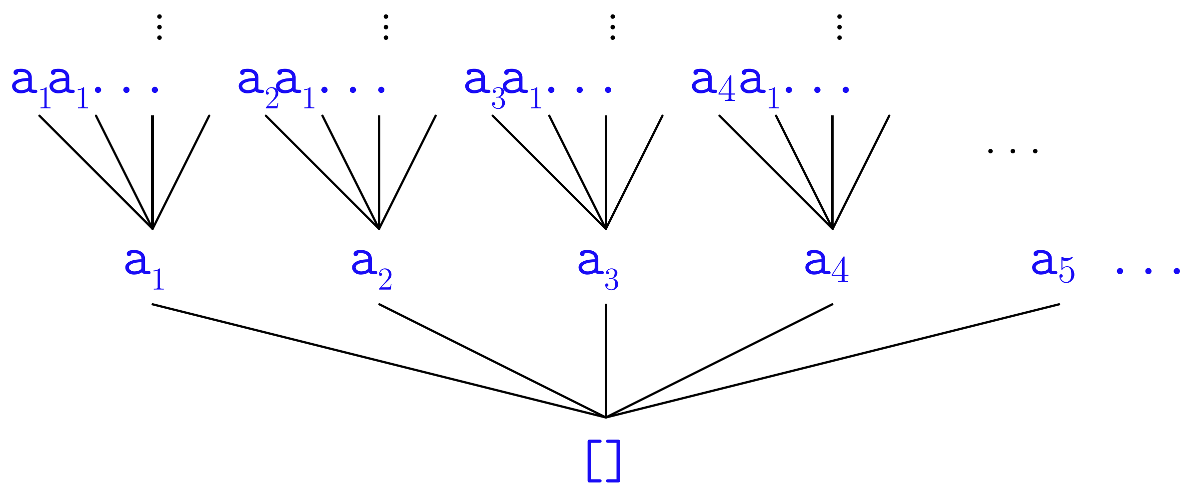
Synthese von Globalsuch-Algorithmen

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 $\mapsto$ 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 $\qquad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor:** **leerer Präfix** $\qquad s_0(M) \equiv$ `[]`
- **Splitting:** **Verlängern des Präfix** $\qquad split[M,s] \equiv \{s{\cdot}a | a \in M\}$
- **Extraktion:** **Gesamter Präfix** $\qquad ext[s] \equiv \{s\}$
- **Sinnvoll:** **nur Elemente aus** M $\qquad J[M,s] \equiv s \text{ ‘}\subseteq\text{’ } M$

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 $\quad s_0(M) \equiv \texttt{[]}$
- 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 $\quad J[M,s] \equiv s \text{ ‘}\subseteq\text{’ } M$

Darstellung als formales Objekt der Wissensbank

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

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 $|M|^{k*|M|}$
- $\Phi_3[M, s] \equiv \texttt{nodups}(s)$
  - 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)  ≡  [L[i]-L[i+j] | i ∈ [1..|L|-j]]
perm(L,S)   ≡  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

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 $\mathcal{G}_\theta$ 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 $sat[z,s] \land O[x,z]$ ab (Vorwärtsinferenz)

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

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

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

Globalsuchalgorithmus für Costas Arrays

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

1. **Wähle Globalsuchtheorie $\mathcal{G}$ = gs_seq_set(ℤ)**
2. **Beweis für $(D,R,I,O) \ll$ gs_seq_set(ℤ) liefert $\theta[n] = \{1..n\}$**
3. **Wähle WF-Filter Φ so, daß Φ_θ notwendig für G_θ beweisbar**
   – perm(p,{1..n}) ∧ ∀j ∈domain(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 ∈domain(p).nodups(dtrow(p,j)) ∧ s⊑p
   leite ab Ψ[n,s] = ∀i ∈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