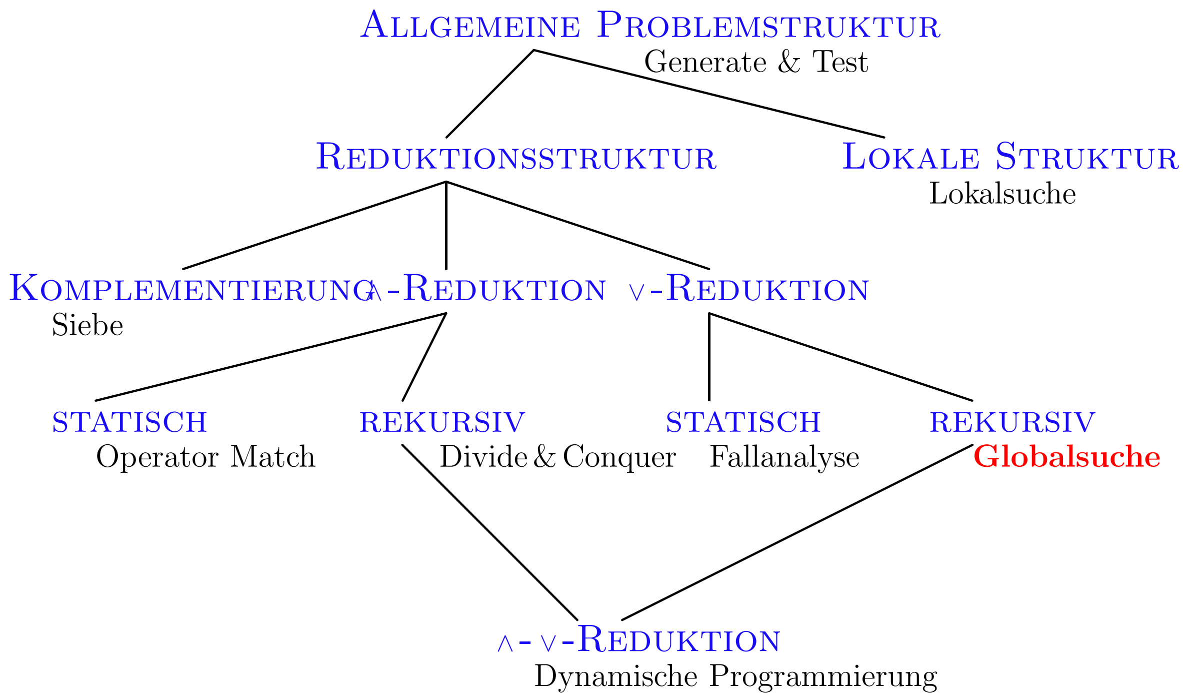
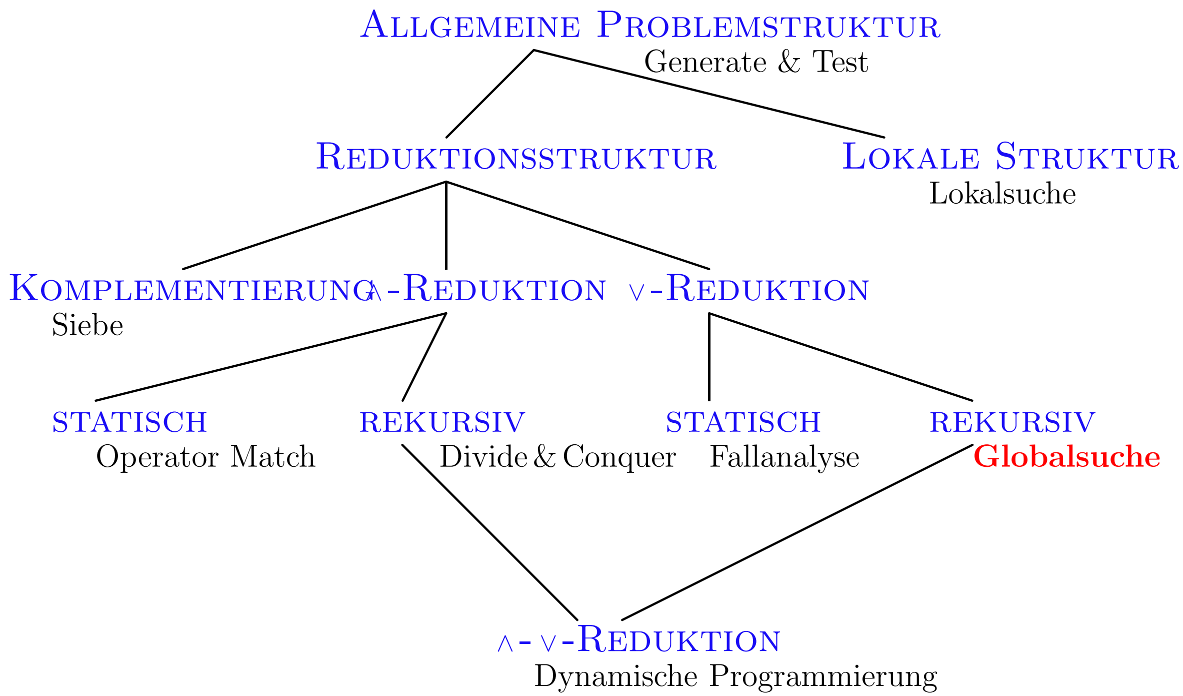
Globalsuchalgorithmen



- **Bestimmung aller Lösungen eines Problems**
  - Aufzählen von Kandidaten
  - Eliminieren von Kandidaten, die keine Lösungen darstellen
  - Verallgemeinerung von Backtracking, Binärsuche, …

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**

Globalsuche: Generelle Idee

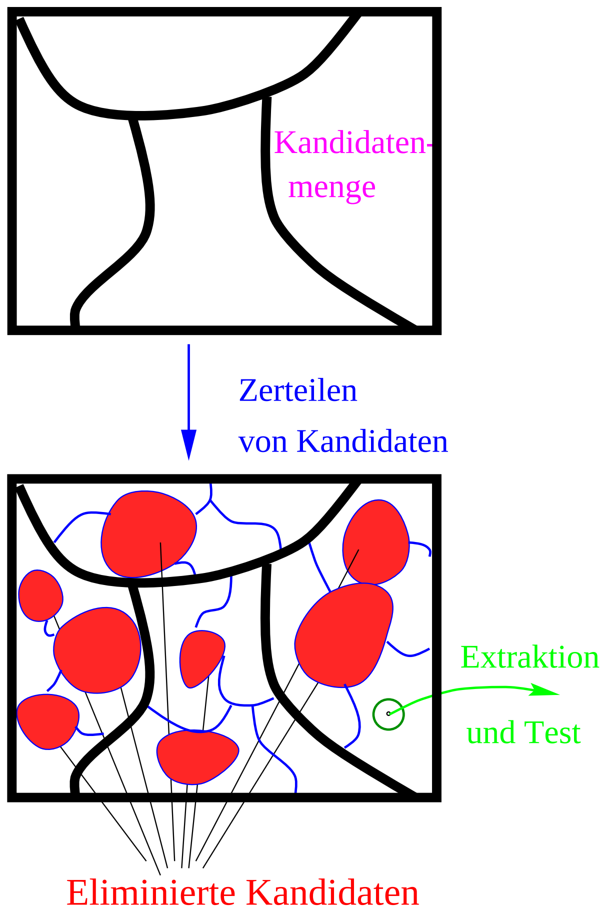
Durchsuchen des gesamten Bildbereichs

- **Suche von außen**
  - **Global:** Untersuchung von ganzen
    Mengen von Lösungskandidaten

Globalsuche: Generelle Idee

Durchsuchen des gesamten Bildbereichs

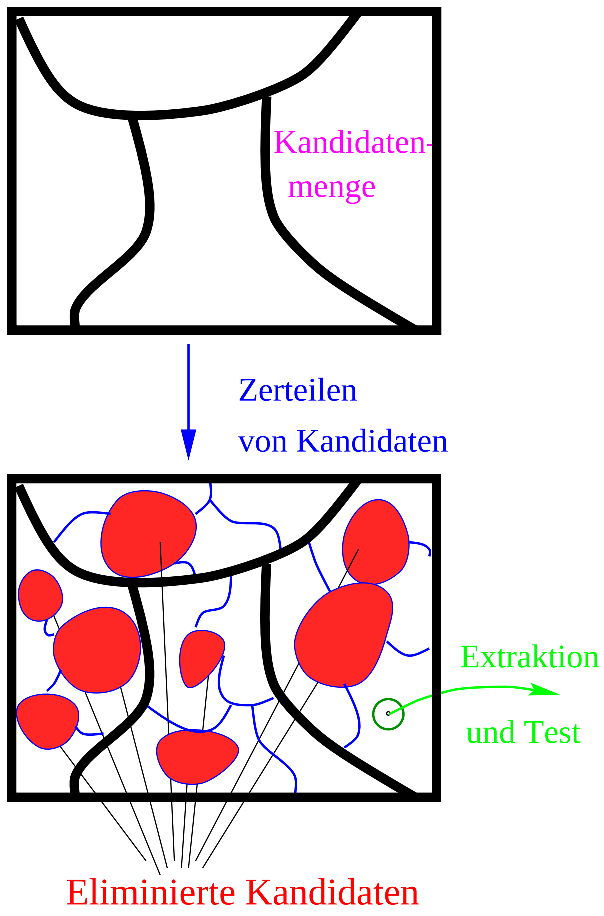
- **Suche von außen**
  - **Global:** Untersuchung von ganzen
    Mengen von Lösungskandidaten
  - Wiederholtes Aufteilen von Kandidatenmengen



Globalsuche: Generelle Idee

Durchsuchen des gesamten Bildbereichs

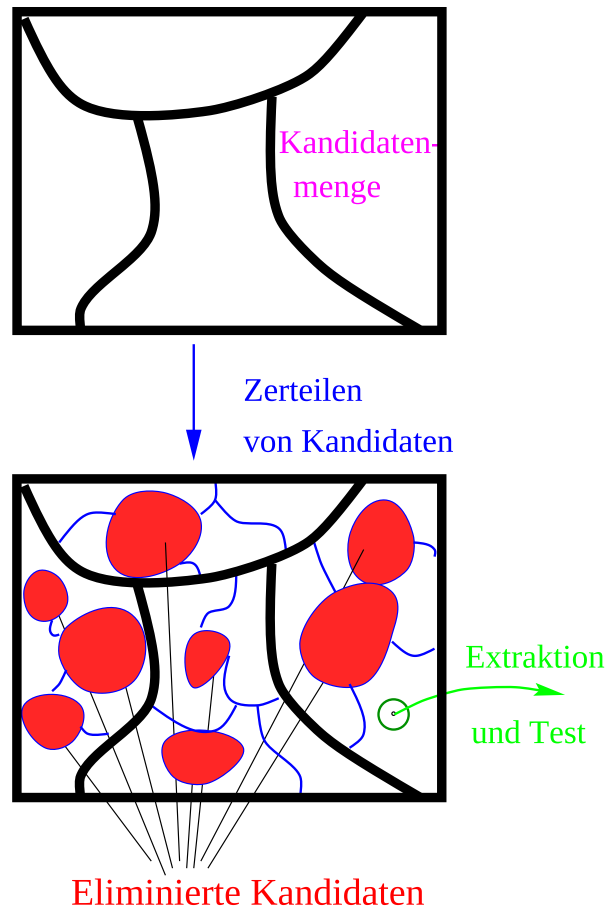
- **Suche von außen**
  - **Global:** Untersuchung von ganzen
    Mengen von Lösungskandidaten
  - Wiederholtes Aufteilen von Kandidatenmengen
  - Elimination von Kandidatenmengen ohne Lösung



Globalsuche: Generelle Idee

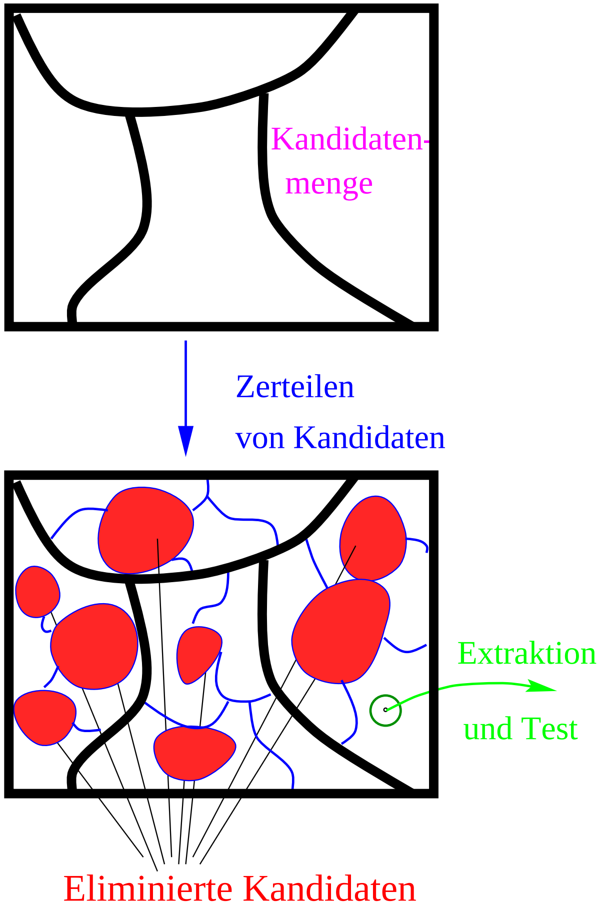
Durchsuchen des gesamten Bildbereichs

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



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

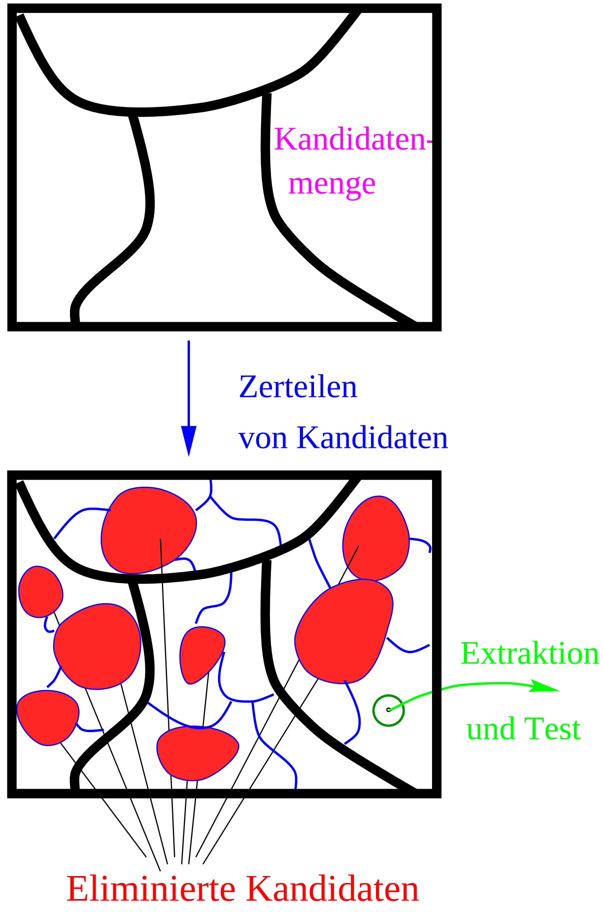


- **Repräsentanten erforderlich**
  - Verarbeitung der Mengen selbst zu aufwendig

GLOBALSUCHE: GENERELLE IDEE

Durchsuchen des gesamten Bildbereichs

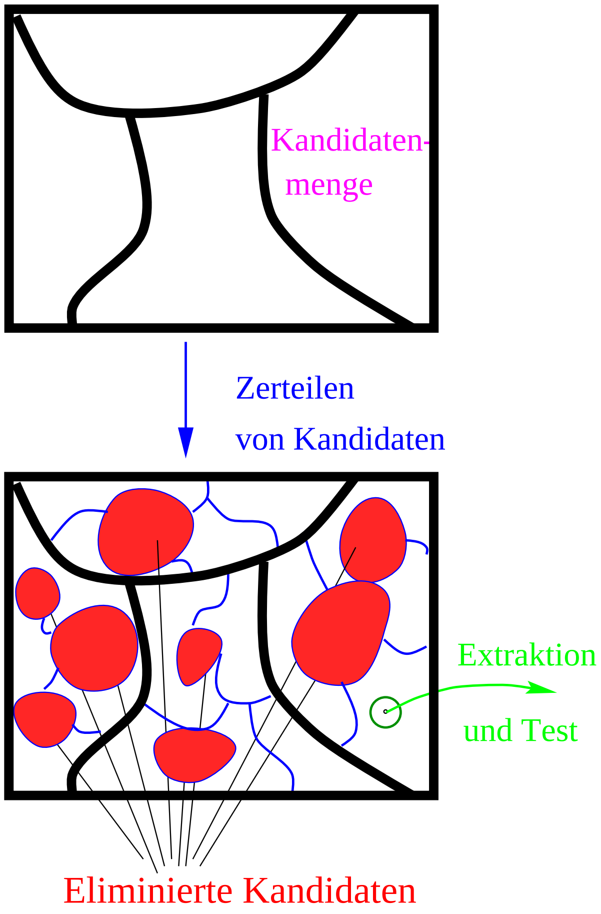
- **Suche von außen**
  - **Global:** Untersuchung von ganzen
    Mengen von Lösungskandidaten
  - Wiederholtes Aufteilen von Kandidatenmengen
  - Elimination von Kandidatenmengen ohne Lösung
  - Extraktion von tatsächlichen Lösungen



- **Repräsentanten erforderlich**
  - Verarbeitung der Mengen selbst zu aufwendig
  - Codiere Kandidatenmengen durch Deskriptoren

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

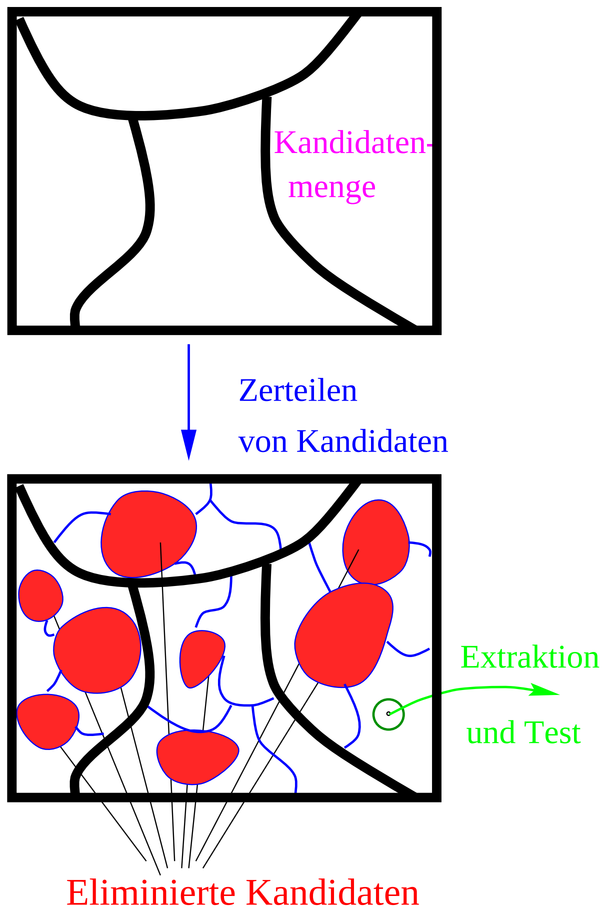


- **Repräsentanten erforderlich**
  - Verarbeitung der Mengen selbst zu aufwendig
  - Codiere Kandidatenmengen durch Deskriptoren
  - Simuliere Aufteilen und Filtern auf Deskriptoren

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 $\}$

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

Ein einfacher 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 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(ℤ)×ℤ×ℕ×ℕ)
   WHERE L≠[] ∧ ordered(L) ∧ 1≤l≤r≤|L|
   RETURNS {i:ℕ| i∈{l..r} ∧ L_i=k}
```

Ein einfacher 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 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(ℤ)×ℤ×ℕ×ℕ)`
  `WHERE L≠[] ∧ ordered(L) ∧ 1≤l≤r≤|L|`
  `RETURNS {i: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**

  $\mathtt{o}_{aux}\mathtt{(L,k,l,r)} = \{\, \mathtt{i} \mid \mathtt{i} \in \{\mathtt{l}\} \wedge \mathtt{i=r} \;\wedge\; \mathtt{L}_i\mathtt{=k} \,\}$

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

  – 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 `{l} = {l..l}` erzeugt

GLOBALSUCHALGORITHMEN: EINHEITLICHE DARSTELLUNG

- **Vereinheitlichung durch Mengendarstellung**

  o_{aux}(L,k,l,r) = { i | i ∈ {l} ∧ i=r ∧ L_i=k }
  ∪ ⋃{ o_{aux}(L,k,n,m) | (n,m) ∈ {(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 L_n>k oder L_m<k
  – Ergänze Filter $L_n \leq k \leq L_m$ für Aufspaltungsmenge (logarithmischer Algorithmus)

Globalsuchalgorithmen: einheitliche Darstellung

- **Vereinheitlichung durch Mengendarstellung**

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

  – 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**

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

Allgemeines Globalsuch-Schema

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

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

Allgemeines Globalsuch-Schema

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

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

$$\cup \bigcup \{\, f_{gs}(x,t) \mid t \in split[x,s] \;\wedge\; \Phi[x,t] \,\}$$

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

- **7 zentrale Komponenten der Algorithmentheorie**

Allgemeines Globalsuch-Schema

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

- **7 zentrale Komponenten der Algorithmentheorie**
  - $s{:}S$ — Deskriptor für Kandidatenmengen
  - s_0: $D \rightarrow S$ — Initialdeskriptor

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

Allgemeines Globalsuch-Schema

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

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

- **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

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

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: $D \rightarrow S$ — Initialdeskriptor
  - $split$:$D \times S \rightarrow \mathrm{Set}(S)$ — Rekursive Aufteilung von Kandidatenmengen
  - Φ:$D \times S \rightarrow \mathbb{B}$ — 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

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

Korrektheit des Globalsuch-Schemas

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

Korrektheit des Globalsuch-Schemas

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

---

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$\boldsymbol{I}[\boldsymbol{x}] \ \Rightarrow\ \boldsymbol{J}[\boldsymbol{x}, \boldsymbol{s}_0(\boldsymbol{x})]$

Korrektheit des Globalsuch-Schemas

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

---

1. Initialdeskriptor ist sinnvoll für zulässige Eingaben

$\boldsymbol{I}[\boldsymbol{x}] \ \Rightarrow\ \boldsymbol{J}[\boldsymbol{x}, \boldsymbol{s}_0(\boldsymbol{x})]$

2. Splitting erhält sinnvoller Deskriptoren

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

Korrektheit des Globalsuch-Schemas

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

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

$I[x] \wedge 0[x,z] \;\Rightarrow\; sat[z,s_0(x)]$

Korrektheit des Globalsuch-Schemas

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

---

1. **Initialdeskriptor ist sinnvoll für zulässige Eingaben**

   $\boldsymbol{I[x] \;\Rightarrow\; J[x,s_0(x)]}$

2. **Splitting erhält sinnvoller Deskriptoren**

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

3. **Initialdeskriptor enthält alle Lösungen**

   $\boldsymbol{I[x] \wedge 0[x,z] \;\Rightarrow\; sat[z,s_0(x)]}$

4. **Filter ist notwendig (keine Lösung wird eliminiert)**

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

Korrektheit des Globalsuch-Schemas

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

ist korrekt, wenn 6 Axiome erfüllt sind

---

1. **Initialdeskriptor ist sinnvoll für zulässige Eingaben**

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

2. **Splitting erhält sinnvoller Deskriptoren**

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

3. **Initialdeskriptor enthält alle Lösungen**

   $I[x] \land 0[x,z] \ \Rightarrow\ sat[z,s_0(x)]$

4. **Filter ist notwendig (keine Lösung wird eliminiert)**

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

5. **Alle Lösungen in endlich vielen Schritten extrahierbar**

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

Korrektheit des Globalsuch-Schemas

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

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

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

ist korrekt, wenn 6 Axiome erfüllt sind

---

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**

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

Korrektheit des Globalsuch-Schemas

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

Globalsuch-Schema: Korrektheitsbeweis

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

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

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

- **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)

GLOBALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- **Abspalten und Spezifikation der Hilfsfunktion $\boldsymbol{f_{gs}}$**

  `FUNCTION` $\boldsymbol{f}$`(`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` $\boldsymbol{f_{gs}}$`(`$x, s\!:\!D{\times}S$`)` `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 $\boldsymbol{f}$ folgt aus der von $\boldsymbol{f_{gs}}$ 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)

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)

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

- **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 $\{y{:}R \mid O[x,y] \land sat[y,s_0(x)]\} = \{y{:}R \mid O[x,y]\}$ (Axiom 3)
  - Aus $\{y{:}R \mid O[x,y] \land sat[y,s_0(x)]\} \neq \emptyset$ 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 $\bigcup\{split_\Phi^{k-1}[x,t] \mid t \in split[x,s] \land \Phi[x,t]\}$

Globalsuch-Schema: Korrektheitsbeweis

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

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

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

- **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**

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

  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)

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

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`

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

Beispiel einer Globalsuchtheorie

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

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

- **Alle 6 Axiome sind erfüllt**

Beispiel einer Globalsuchtheorie

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

D	$\mapsto$		$\mathtt{Seq}(\mathbb{N})\times\mathbb{N}$
R	$\mapsto$		$\mathbb{N}$
I	$\mapsto$	λL,k.	$\mathtt{L}\neq\mathtt{[]} \wedge \mathtt{ordered(L)}$
O	$\mapsto$	λL,k,i.	$\mathtt{i}\in\{\mathtt{1..|L|}\} \wedge \mathtt{L}_i\mathtt{=k}$
S	$\mapsto$		$\mathtt{Seq}(\mathbb{N})\times\mathtt{Seq}(\mathbb{N})$
J	$\mapsto$	λL,k,l,r.	$\mathtt{1}\leq\mathtt{l}\leq\mathtt{r}\leq\mathtt{|L|}$
s_0	$\mapsto$	λL,k.	$\mathtt{(1,|L|)}$
sat	$\mapsto$	λi,l,r.	$\mathtt{i}\in\{\mathtt{l..r}\}$
$split$	$\mapsto$	λL,k,l,r.	`if l<r then` $\{\mathtt{(l,(l+r)/2),((l+r)/2+1,r)}\}$ `else` $\emptyset$
Φ	$\mapsto$	λL,k,l,r.	$\mathtt{L}_l\leq\mathtt{k}\leq\mathtt{L}_r$
ext	$\mapsto$	λl,r.	`if l=r then` $\{\mathtt{l}\}$ `else` $\emptyset$

- **Alle 6 Axiome sind erfüllt**

1. $\mathtt{L}\neq\mathtt{[]} \wedge \mathtt{ordered(L)} \Rightarrow \mathtt{1}\leq\mathtt{1}\leq\mathtt{|L|}\leq\mathtt{|L|}$

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. $\texttt{L}\neq\texttt{[]} \wedge \texttt{ordered(L)} \wedge \texttt{1}\leq\texttt{l}\leq\texttt{r}\leq\texttt{|L|} \;\Rightarrow\; \forall\texttt{(x,y)}\in split\texttt{[L,k,l,r]}.\ \texttt{1}\leq\texttt{x}\leq\texttt{y}\leq\texttt{|L|}$

Beispiel einer Globalsuchtheorie

- **Theorie gs_osearch für osearch**

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

- **Alle 6 Axiome sind erfüllt**

1. $L\neq[\,] \wedge \mathtt{ordered}(L) \Rightarrow 1\leq 1\leq |L|\leq |L|$
2. $L\neq[\,] \wedge \mathtt{ordered}(L)\wedge 1\leq l\leq r\leq |L| \Rightarrow \forall (x,y)\in split[L,k,l,r].\ 1\leq x\leq y\leq |L|$
3. $L\neq[\,] \wedge \mathtt{ordered}(L)\wedge i\in\{1..|L|\}\wedge L_i=k \Rightarrow i\in\{1..|L|\}$

Beispiel einer Globalsuchtheorie

- **Theorie gs_osearch für osearch**

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

- **Alle 6 Axiome sind erfüllt**

1. $L\neq[] \wedge \mathtt{ordered}(L) \Rightarrow 1\leq 1\leq |L|\leq |L|$
2. $L\neq[] \wedge \mathtt{ordered}(L) \wedge 1\leq l\leq r\leq |L| \Rightarrow \forall (x,y)\in split[L,k,l,r].\ 1\leq x\leq y\leq |L|$
3. $L\neq[] \wedge \mathtt{ordered}(L) \wedge i\in\{1..|L|\} \wedge L_i=k \Rightarrow i\in\{1..|L|\}$
4. $L\neq[] \wedge \mathtt{ordered}(L) \wedge 1\leq l\leq r\leq |L| \Rightarrow$

   $L_l\leq k\leq L_r \Leftarrow \exists z{:}\mathbb{N}.\ z\in\{l..r\} \wedge z\in\{1..|L|\} \wedge L_z=k$

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. $\texttt{L}\neq\texttt{[]}\wedge\texttt{ordered(L)}\wedge\texttt{1}\leq\texttt{l}\leq\texttt{r}\leq\texttt{|L|}\wedge\texttt{i}\in\{\texttt{1..|L|}\}\wedge\texttt{L}_i\texttt{=k} \Rightarrow$
   $\texttt{i}\in\{\texttt{l..r}\} \Leftrightarrow \exists\texttt{k:}\mathbb{N}\texttt{.}\exists\texttt{(x,y)}\in split^k_\Phi\texttt{[L,k,l,r].}\ \texttt{i}\in\texttt{(if x=y then }\{\texttt{x}\}\texttt{ else }\emptyset\texttt{)}$

Beispiel einer Globalsuchtheorie

- **Theorie $\texttt{gs\_osearch}$ für $\texttt{osearch}$**

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

- **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

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

SCHEMATISCHER GLOBALSUCHALGORITHMUS FÜR osearch

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

Nach Optimierung durch Simplifikationen

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

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

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

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

Synthese von Globalsuch-Algorithmen

Spezialisiere vorformuliertes Programmierwissen

- **Globalsuchtheorie: allgemeine Suchstruktur für $\boldsymbol{R}$**
  - 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 $\boldsymbol{\Phi}$ 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

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 $spec = (D,R,I,O)$

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 $\theta{:}D \to D_\mathcal{G}$

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

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** $\quad sat[L,s] \;\equiv\; s \sqsubseteq L$

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

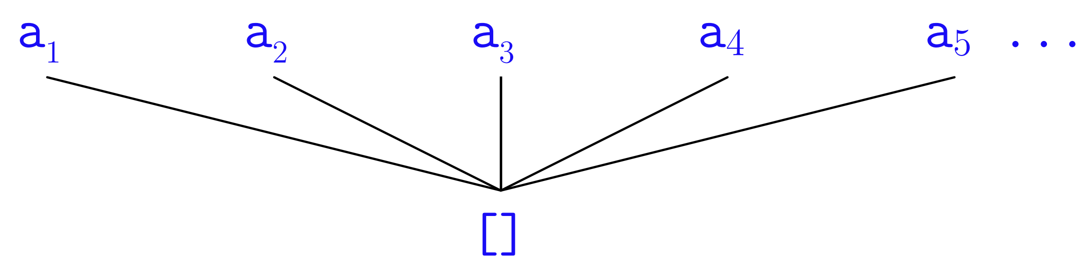
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L

`[]`

- **Deskriptoren:** **gemeinsamer Präfix s** $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor:** **leerer Präfix** $\quad s_0(M) \equiv$ `[]`

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

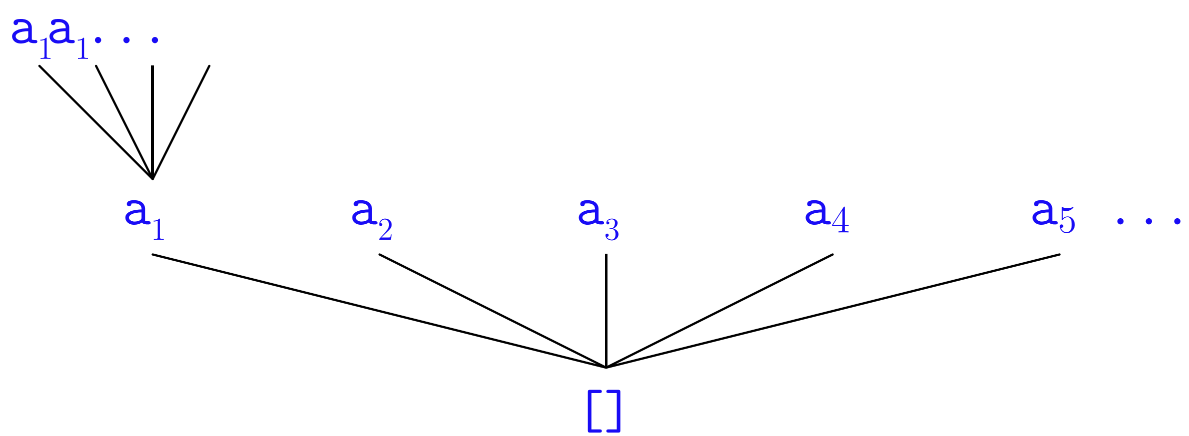
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren: gemeinsamer Präfix s** $\qquad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor: leerer Präfix** $\qquad s_0(M) \equiv$ `[]`
- **Splitting: Verlängern des Präfix** $\qquad split[M,s] \equiv \{s{\cdot}a \mid a \in M\}$

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

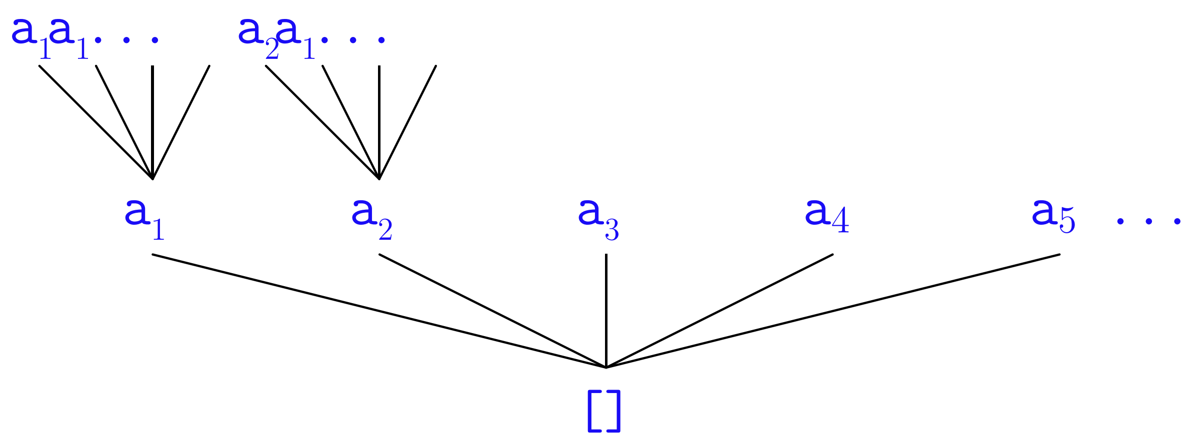
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren:** gemeinsamer Präfix s $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor:** leerer Präfix $\quad s_0(M) \equiv []$
- **Splitting:** Verlängern des Präfix $\quad split[M,s] \equiv \{s{\cdot}a | a \in M\}$

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

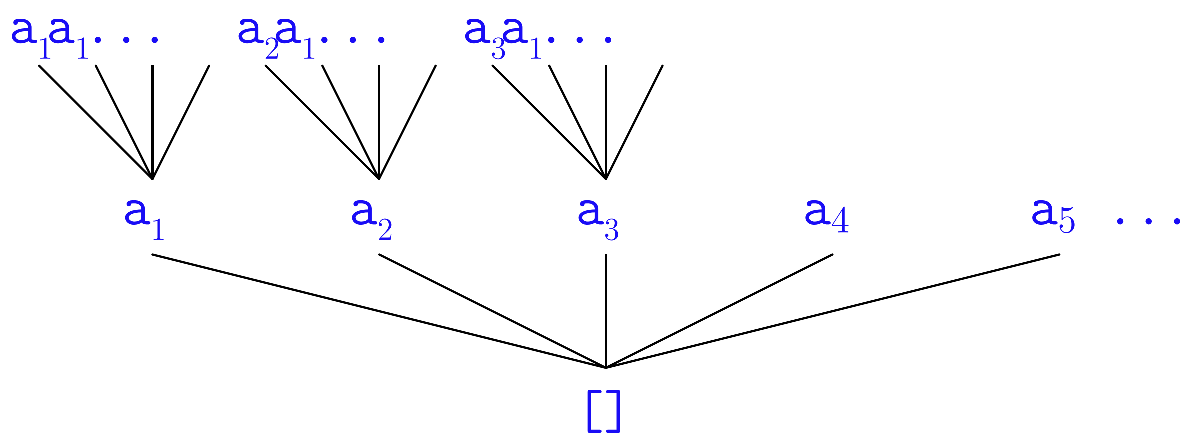
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren:** gemeinsamer Präfix s $\qquad sat[L,s] \;\equiv\; s \sqsubseteq L$
- **Initialdeskriptor:** leerer Präfix $\qquad s_0(M) \;\equiv\;$ `[]`
- **Splitting:** Verlängern des Präfix $\qquad split[M,s] \;\equiv\; \{s{\cdot}a | a \in M\}$

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

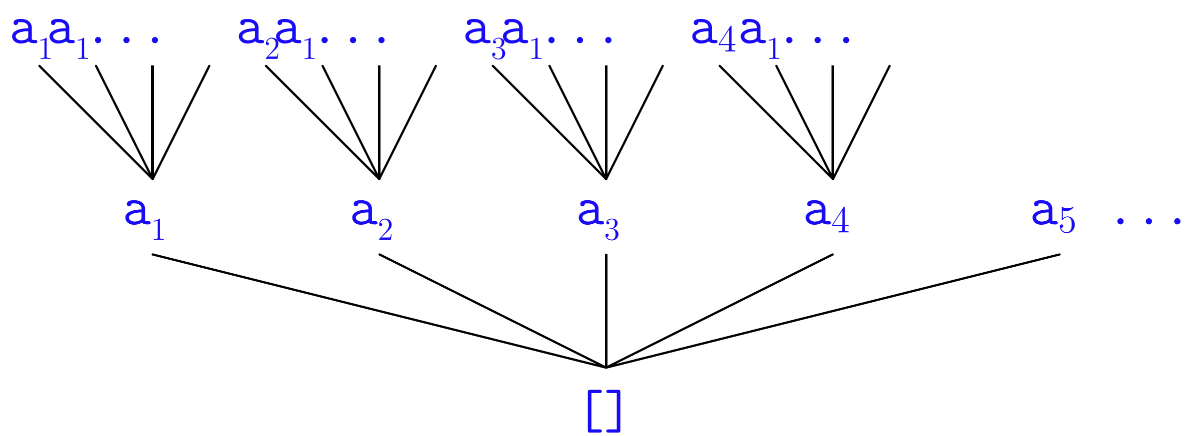
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren: gemeinsamer Präfix** s $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor: leerer Präfix** $\quad s_0(M) \equiv []$
- **Splitting: Verlängern des Präfix** $\quad split[M,s] \equiv \{s{\cdot}a \mid a \in M\}$

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

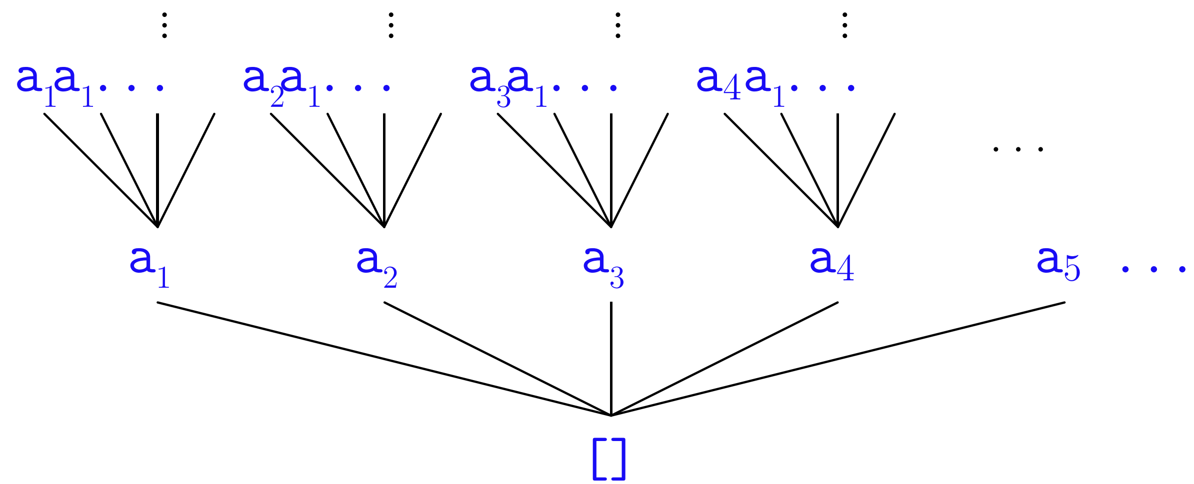
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren:** **gemeinsamer Präfix** s $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor:** **leerer Präfix** $\quad s_0(M) \equiv$ `[]`
- **Splitting:** **Verlängern des Präfix** $\quad split[M,s] \equiv \{s{\cdot}a \mid a \in M\}$

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

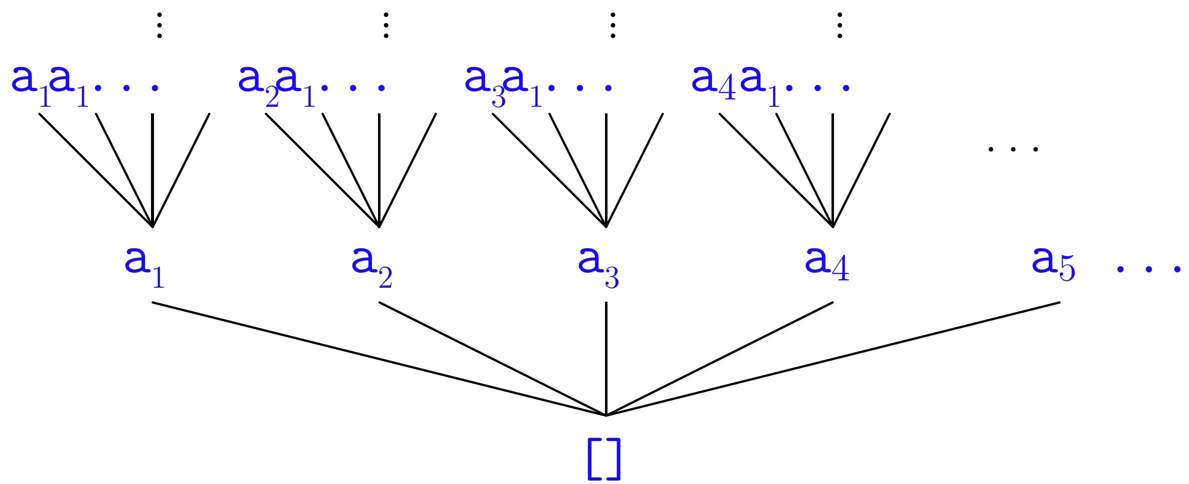
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren:** **gemeinsamer Präfix** s $\qquad sat[L,s] \;\equiv\; s \sqsubseteq L$
- **Initialdeskriptor:** **leerer Präfix** $\qquad s_0(M) \;\equiv\;$ `[]`
- **Splitting:** **Verlängern des Präfix** $\qquad split[M,s] \;\equiv\; \{s{\cdot}a \mid a \in M\}$

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

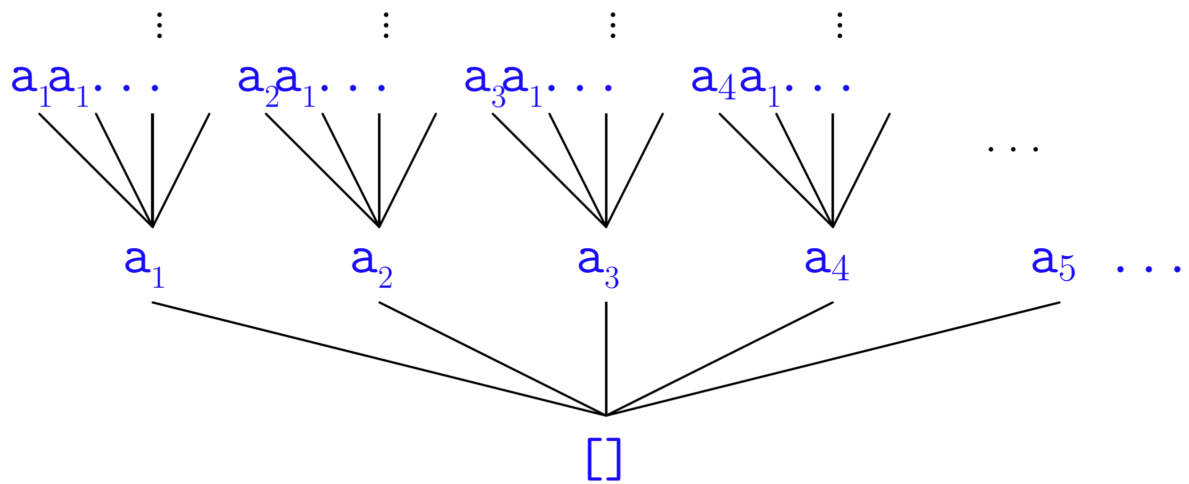
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren:** gemeinsamer Präfix s $\qquad sat[L,s] \;\equiv\; s \sqsubseteq L$
- **Initialdeskriptor:** leerer Präfix $\qquad s_0(M) \;\equiv\; []$
- **Splitting:** Verlängern des Präfix $\qquad split[M,s] \;\equiv\; \{s{\cdot}a \mid a \in M\}$
- **Extraktion:** Gesamter Präfix $\qquad ext[s] \;\equiv\; \{s\}$

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

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



- **Deskriptoren:** **gemeinsamer Präfix** s $\quad sat[L,s] \equiv s \sqsubseteq L$
- **Initialdeskriptor:** **leerer Präfix** $\quad s_0(M) \equiv$ []
- **Splitting:** **Verlängern des Präfix** $\quad split[M,s] \equiv \{s{\cdot}a \mid a \in M\}$
- **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 $\quad J[M,s] \equiv s \text{ `}{\subseteq}\text{' } M$

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

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

Wohlfundiertheitsfilter für `gs_seq_set`(α)

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

Wohlfundiertheitsfilter für gs_seq_set(α)

- $\mathbf{\Phi_1[M,s]} \;\equiv\; |s| \leq k$
  - Filter testet absolute Längenbegrenzung der Deskriptorfolge
  - Einfacher, schnell auszuführender Test
  - Filter garantiert Terminierung nach k Schritten, Baumgröße $|M|^k$
- $\mathbf{\Phi_2[M,s]} \;\equiv\; |s| \leq k * |M|$
  - 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`(α)

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

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**

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					

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				

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**	

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**		

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**			

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 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**

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 $\in$ [1..|L|-j]]

  perm(L,S) $\equiv$ nodups(L) $\wedge$ 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 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` $\in$ `[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∈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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM, s. range(s)⊆M
s_0	$\mapsto$	λM. []
sat	$\mapsto$	λL, s. s⊑L
$split$	$\mapsto$	λM, s. {s·a | a∈M}
ext	$\mapsto$	λs. {s}

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))}
```

1. Bildbereich stimmt mit $R_{\mathcal{G}}$=Seq($\mathbb{Z}$) überein

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

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM, s. range(s)⊆M
s_0	$\mapsto$	λM. []
sat	$\mapsto$	λL, s. s⊑L
split	$\mapsto$	λM, s. {s·a | a∈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

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))}
```

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**

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

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM, s. range(s)⊆M
s_0	$\mapsto$	λM. []
sat	$\mapsto$	λL, s. s⊑L
$split$	$\mapsto$	λM, s. {s·a | a∈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:**
   – ∀n:$\mathbb{Z}$. n≥1 ⇒ ∃M:Set($\mathbb{Z}$). ∀p:Seq($\mathbb{Z}$). O(n,p) ⇒ range(p)⊆M

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM, s. range(s)⊆M
s_0	$\mapsto$	λM. []
sat	$\mapsto$	λL, s. s⊑L
$split$	$\mapsto$	λM, s. {s·a | a∈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)⊆M
   - Heuristik: Suche Folgerungen von O(n,p), in denen range(p) vorkommt

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM, s. range(s)⊆M
s_0	$\mapsto$	λM. []
sat	$\mapsto$	λL, s. s⊑L
$split$	$\mapsto$	λM, s. {s·a | a∈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≥1 $\Rightarrow$ $\exists$M:Set($\mathbb{Z}$). $\forall$p:Seq($\mathbb{Z}$). 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}

SPEZIALISIERE $\texttt{gs\_seq\_set}(\mathbb{Z})$ 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	$\mapsto$	$\texttt{Set}(\alpha)$
R	$\mapsto$	$\texttt{Seq}(\alpha)$
I	$\mapsto$	$\lambda\texttt{M}.\,\texttt{true}$
O	$\mapsto$	$\lambda\texttt{M},\texttt{L}.\,\texttt{range(L)}\subseteq\texttt{M}$
S	$\mapsto$	$\texttt{Seq}(\alpha)$
J	$\mapsto$	$\lambda\texttt{M},\texttt{s}.\,\texttt{range(s)}\subseteq\texttt{M}$
s_0	$\mapsto$	$\lambda\texttt{M}.\,\texttt{[]}$
sat	$\mapsto$	$\lambda\texttt{L},\texttt{s}.\,\texttt{s}\sqsubseteq\texttt{L}$
$split$	$\mapsto$	$\lambda\texttt{M},\texttt{s}.\,\{\texttt{s}\cdot\texttt{a} \mid \texttt{a}\in\texttt{M}\}$
ext	$\mapsto$	$\lambda\texttt{s}.\,\{\texttt{s}\}$

1. Bildbereich stimmt mit $R_{\mathcal{G}}$=$\texttt{Seq}(\mathbb{Z})$ überein

2. Eingabebereiche $\mathbb{Z}$, $D_{\mathcal{G}}$=$\texttt{Set}(\mathbb{Z})$ sind anzupassen

3. Keine Eingabebedingung zu prüfen: $\texttt{I}_{\mathcal{G}}(M)$=$\texttt{true}$

4. Zu zeigen ist also:

- $\forall\texttt{n}:\mathbb{Z}.\,\texttt{n}\geq 1 \Rightarrow \exists\texttt{M}:\texttt{Set}(\mathbb{Z}).\,\forall\texttt{p}:\texttt{Seq}(\mathbb{Z}).\,O(\texttt{n,p}) \Rightarrow \texttt{range(p)}\subseteq\texttt{M}$
- Heuristik: Suche Folgerungen von $O(\texttt{n,p})$, in denen $\texttt{range(p)}$ vorkommt
- Auffalten von $\texttt{perm}$ liefert: $\texttt{perm(p,\{1..n\})} \Rightarrow \texttt{range(p)}\subseteq\{\texttt{1..n}\}$
- Wähle $\texttt{M} \equiv \{\texttt{1..n}\}$ und extrahiere $\theta = \lambda\texttt{n}.\{\texttt{1..n}\}$

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM,s.range(s)⊆M
s_0	$\mapsto$	λM.[]
sat	$\mapsto$	λL,s.s⊑L
$split$	$\mapsto$	λM,s.{s·a | a∈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≥1 $\Rightarrow$ $\exists$M:Set($\mathbb{Z}$). $\forall$p:Seq($\mathbb{Z}$). 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 = λn.{1..n}
5. **Modifiziere gs_seq_set($\mathbb{Z}$) und Φ_3 mit θ und der Spezifikation**

SPEZIALISIERE $\texttt{gs\_seq\_set}(\mathbb{Z})$ 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	$\mapsto$	$\texttt{Set}(\alpha)$
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	$\mapsto$	$\lambda\texttt{L,s.s}\sqsubseteq\texttt{L}$
$split$	$\mapsto$	$\lambda\texttt{M,s.}\{\texttt{s}\cdot\texttt{a}\,|\,\texttt{a}\in\texttt{M}\}$
ext	$\mapsto$	$\lambda\texttt{s.}\{\texttt{s}\}$

1. **Bildbereich stimmt mit $R_{\mathcal{G}}$=$\texttt{Seq}(\mathbb{Z})$ überein**
2. **Eingabebereiche $\mathbb{Z}$, $D_{\mathcal{G}}$=$\texttt{Set}(\mathbb{Z})$ sind anzupassen**
3. **Keine Eingabebedingung zu prüfen: $\texttt{I}_{\mathcal{G}}(M)$=$\texttt{true}$**
4. **Zu zeigen ist also:**
   - $\forall\texttt{n}:\mathbb{Z}.\,\texttt{n}\geq 1 \Rightarrow\ \exists\texttt{M}:\texttt{Set}(\mathbb{Z}).\,\forall\texttt{p}:\texttt{Seq}(\mathbb{Z}).\,O(\texttt{n,p}) \Rightarrow \texttt{range(p)}\subseteq\texttt{M}$
   - Heuristik: Suche Folgerungen von $O(\texttt{n,p})$, in denen $\texttt{range(p)}$ vorkommt
   - Auffalten von $\texttt{perm}$ liefert: $\texttt{perm}(\texttt{p},\{1..\texttt{n}\}) \Rightarrow \texttt{range(p)}\subseteq\{1..\texttt{n}\}$
   - Wähle $\texttt{M} \equiv \{1..\texttt{n}\}$ und extrahiere $\theta = \lambda\texttt{n}.\{1..\texttt{n}\}$
5. **Modifiziere $\texttt{gs\_seq\_set}(\mathbb{Z})$ und Φ_3 mit θ und der Spezifikation**

$$G_\theta = (\mathbb{Z},\texttt{Seq}(\mathbb{Z}),\lambda\texttt{n.n}\geq 1,O,\ \texttt{Seq}(\mathbb{Z}),\ \lambda\texttt{n,s.range(s)}\subseteq\{1..\texttt{n}\},\ \lambda\texttt{n.[]},\ \lambda\texttt{L,s.s}\sqsubseteq\texttt{L},\ \lambda\texttt{n,s.}\{\texttt{V}\cdot\texttt{a}\,|\,\texttt{a}\in\{1..\texttt{n}\}\},\ \lambda\texttt{s.}\{\texttt{s}\})$$

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)⊆M
S	$\mapsto$	Seq(α)
J	$\mapsto$	λM,s.range(s)⊆M
s_0	$\mapsto$	λM.[]
sat	$\mapsto$	λL,s.s⊑L
$split$	$\mapsto$	λM,s.{s·a|a∈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≥1 ⇒ $\exists$M:Set($\mathbb{Z}$). $\forall$p:Seq($\mathbb{Z}$). O(n,p) ⇒ range(p)⊆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($\mathbb{Z}$) und Φ_3 mit θ und der Spezifikation**

G_θ = ($\mathbb{Z}$,Seq($\mathbb{Z}$),λn.n≥1,O, Seq($\mathbb{Z}$), λn,s.range(s)⊆{1..n},
λn.[], λL,s. s⊑L, λn,s.{V·a|a∈{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 $\{y{:}R \mid O[x,y]\}$

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)

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)

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)

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 $sat[z,s] \wedge O[x,z]$ ab (Vorwärtsinferenz)

Synthesesestrategie für Globalsuchalgorithmen

Gegeben eine Problemspezifikation

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

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**

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

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

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))}
```

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)`

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 $\mathcal{G}$ = $\texttt{gs\_seq\_set}(\mathbb{Z})$

2. Beweis für $(D,R,I,O) \ll \texttt{gs\_seq\_set}(\mathbb{Z})$ liefert $\theta[n] = \{1..n\}$

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)

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))

Globalsuchalgorithmus für Costas Arrays

$\texttt{FUNCTION Costas (n:}\mathbb{Z}\texttt{) WHERE n}\geq\texttt{1}$
$\quad\texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \forall \texttt{j} \in \texttt{domain(p).nodups(dtrow(p,j))}\}$

1. Wähle Globalsuchtheorie $\mathcal{G}$ = $\texttt{gs\_seq\_set(}\mathbb{Z}\texttt{)}$

2. Beweis für $(D,R,I,O) \ll \texttt{gs\_seq\_set(}\mathbb{Z}\texttt{)}$ liefert $\theta[n] = \{1..n\}$

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

– $\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \forall \texttt{j} \in \texttt{domain(p).nodups(dtrow(p,j))} \wedge \texttt{s} \sqsubseteq \texttt{p} \Rightarrow \Phi[\{\texttt{1..n}\}\texttt{,s}]$

– Leicht beweisbar nur für $\Phi_3[\texttt{M,s}]$ = $\texttt{nodups(s)}$

4. Leite zusätzlichen notwendigen Filter Ψ ab

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

5. Instantiiere den Standard-Globalsuchalgorithmus

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

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

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

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