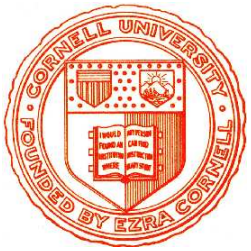
Automatisierte Logik und Programmierung

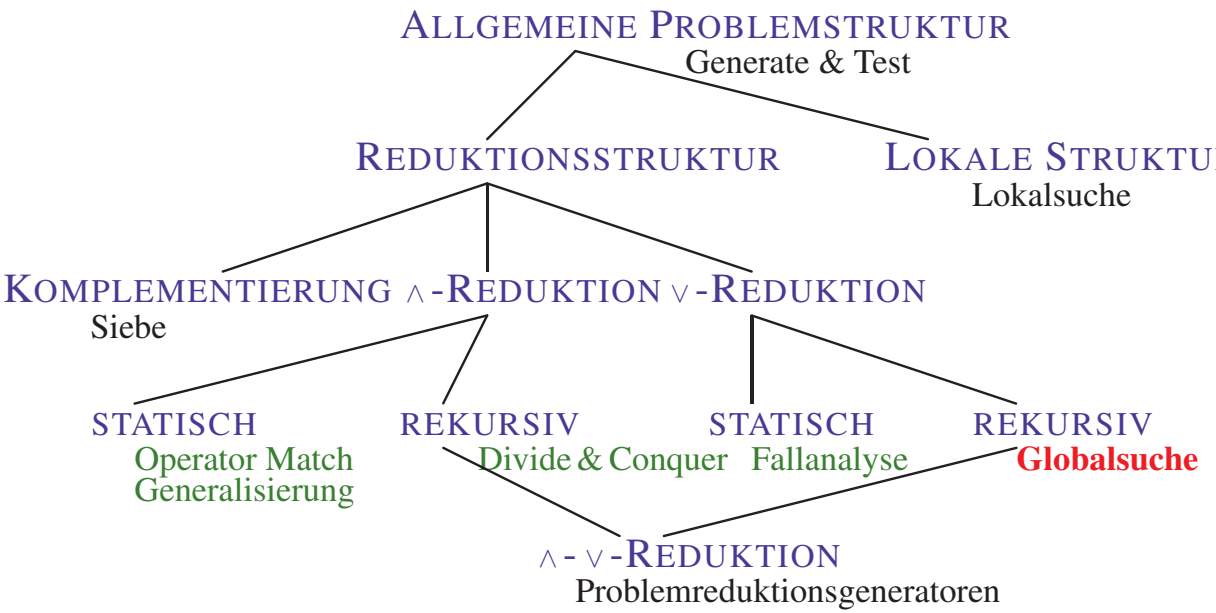
Einheit 20



Globalsuch-Algorithmen

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

GLOBALSUCHALGORITHMEN



- **Bestimmung aller Lösungen eines Problems**
  - Aufzählen von Kandidaten
  - Eliminieren von Kandidaten, die keine Lösungen darstellen
  - Verallgemeinert Backtracking, Binärsuche, Branch & Bound ...
- **Rekursive $\vee$-Reduktion des Problems**
  - Gesamtlösung ist Vereinigung unabhängiger Teillösungen
  - Gut geeignet für Parallelverarbeitung
  - Auch für sequentielle Suche nach erstmöglicher Lösung geeignet

Ein typischer Globalsuchalgorithmus

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

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

Ein typischer Globalsuchalgorithmus

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

  `FUNCTION osearch(L,k:Seq(ℤ)×ℤ) WHERE L≠[] ∧ ordered(L)`

  `    RETURNS {i:ℕ | i∈{1..|L|} ∧ L`$_i$`=k}`

- **Binäre Suche und Aufsammeln von Lösungen**
  - Spalte Liste `L` in zwei Teile `[L`$_i$`|1≤i≤m]` und `[L`$_i$`|m<i≤|L|]`
  - Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen

Ein typischer Globalsuchalgorithmus

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

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

- **Binäre Suche und Aufsammeln von Lösungen**
  - Spalte Liste `L` in zwei Teile `[L_i|1≤i≤m]` und `[L_i|m<i≤|L|]`
  - Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen
- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**
  - Signifikante Reduktion der zu verwaltenden Daten
  - Grenzen `l` und `r` der Indexmengen sind Repräsentanten des Suchraums
  - Repräsentanten sind nur dann sinnvoll wenn `1≤l≤r≤|L|`

Ein typischer 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 Liste L in zwei Teile [L_i|1$\leq$i$\leq$m] und [L_i|m<i$\leq$|L|]
  - Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen
- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**
  - Signifikante Reduktion der zu verwaltenden Daten
  - Grenzen l und r der Indexmengen sind Repräsentanten des Suchraums
  - Repräsentanten sind nur dann sinnvoll wenn 1$\leq$l$\leq$r$\leq$|L|
- **Verwende Hilfsfunktion aux für rekursive Breitensuche**

  FUNCTION aux(L,k,l,r:Seq($\mathbb{Z}$)$\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$)
  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}
  $\equiv$ if l=r then if L_l=k then {l} else $\emptyset$
  else let m=(l+r)/2 in aux(L,k,l,m) $\cup$ aux(L,k,m+1,r)

Ein typischer 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 Liste L in zwei Teile [L_i|1$\leq$i$\leq$m] und [L_i|m<i$\leq$|L|]
  - Durchsuche beide Hälften und vereinige jeweilige Lösungsmengen
- **Verwalte Indexgrenzen anstelle der tatsächlichen Teillisten**
  - Signifikante Reduktion der zu verwaltenden Daten
  - Grenzen l und r der Indexmengen sind Repräsentanten des Suchraums
  - Repräsentanten sind nur dann sinnvoll wenn 1$\leq$l$\leq$r$\leq$|L|
- **Verwende Hilfsfunktion aux für rekursive Breitensuche**

  FUNCTION aux(L,k,l,r:Seq($\mathbb{Z}$)$\times\mathbb{Z}\times\mathbb{N}\times\mathbb{N}$)
  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}
  $\equiv$ if l=r then if L_l=k then {l} else $\emptyset$
  else let m=(l+r)/2 in aux(L,k,l,m) $\cup$ aux(L,k,m+1,r)
- **Initialisiere Hilfsfunktion mit gesamter Liste**
  - osearch(L,k) $\equiv$ aux(L,k,1,|L|)

Optimierung des Grundalgorithmus

- **Eliminiere Hilfsfunktionsaufrufe, wenn keine Lösung möglich**
  - Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
  - Suchraum $\{i..j\}$ ohne Lösung, falls $L_i>k$ oder $L_j<k$
  - Ergänze Filter $L_i \le k \le L_j$ zum rekursivem Aufruf

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

  - Algorithmus hat nur noch logarithmische Laufzeit

Optimierung des Grundalgorithmus

- **Eliminiere Hilfsfunktionsaufrufe, wenn keine Lösung möglich**
  - Suche berücksichtigt nicht, daß Liste geordnet ist (linearer Algorithmus)
  - Suchraum $\{i..j\}$ ohne Lösung, falls $L_i>k$ oder $L_j<k$
  - Ergänze Filter $L_i \le k \le L_j$ zum rekursivem Aufruf

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

  - Algorithmus hat nur noch logarithmische Laufzeit

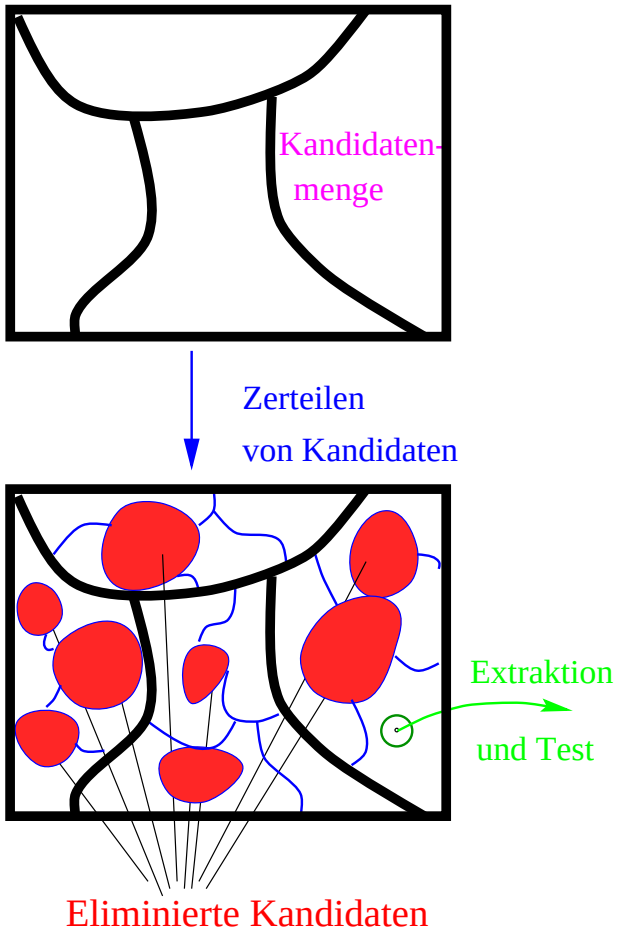
- **Integriere Filter in Initialaufruf der Hilfsfunktion**

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

Globalsuche: Generelle Idee

Durchsuchen eines gesamten Bildbereichs

- **Suche von außen**
  - **Global:** Untersuchung ganzer Mengen von Lösungskandidaten
  - Wiederholtes Aufteilen von Kandidatenmengen
  - Extraktion von tatsächlichen Lösungen
  - Elimination von Kandidatenmengen ohne Lösung
- **Repräsentanten erforderlich**
  - Verarbeitung der Mengen selbst zu aufwendig
  - Codiere Kandidatenmengen durch Deskriptoren
  - Simuliere Aufteilen und Filtern auf Deskriptoren
  - Notwendige Informationen bei der Spezifikation:
    - Wann ist ein Deskriptor eine sinnvolle Beschreibung einer Menge?
    - Wie beschreibt man Zugehörigkeit zur Menge mittels Deskriptoren?



Einheitliche Notation für Globalsuchalgorithmen

```
FUNCTION aux(L,k,l,r:Seq(ℤ)×ℤ×ℕ×ℕ) ...
≡ if l=r then if L_l=k then {l} else ∅
       else let m=(l+r)/2
            in if  L_m<k   then aux(L,k,m+1,r)
            elseif L_{m+1}>k then aux(L,k,l,m)
            else   aux(L,k,l,m) ∪ aux(L,k,m+1,r)
```

- **Darstellung nur für binäre Aufspaltung des Suchraums geeignet**
  - Allgemeiner Fall ist Vereinigung vieler Lösungsmengen
  - Allgemeine Mengenschreibweise ist geeigneter als Darstellung
    - Indexgrenzen `(i,j)` werden aus Aufspaltungsmenge ausgewählt
    - Indexgrenzen werden ausgefiltert, wenn $L_i > k$ oder $L_j < k$
    - Direkte Lösung wird durch Extraktion aus $\{l..l\}$ erzeugt
    - Gesamtlösung ist Vereinigung aller Einzellösungsmengen
- **Endform: wohlstrukturierter mathematischer Algorithmus**

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

Allgemeines Globalsuch-Schema

Suche durch Aufspalten, Filtern und Extrahieren

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

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

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

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

Korrektheit folgt aus wenigen Voraussetzungen

Korrektheit des Globalsuch-Schemas

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

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

in if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ 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**

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

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 explizite Spezifikation der Hilfsfunktion aux**

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

  $\equiv$ if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else $\emptyset$

  FUNCTION $aux(x,s{:}D\times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

  RETURNS $\{y{:}R \mid O[x,y] \wedge sat[y,s]\}$

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

Globalsuch-Schema: Korrektheitsbeweis

- **Abspalten und explizite Spezifikation der Hilfsfunktion aux**

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

  $\equiv$ if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else $\emptyset$

  FUNCTION $aux(x,s{:}D\times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

  RETURNS $\{y{:}R \mid O[x,y] \wedge sat[y,s]\}$

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

- **Korrektheit von f folgt aus der von aux mit Axiom 1, 3 & 4**

  – Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)

  $\boxed{\forall x{:}D.\ \ I[x] \ \Rightarrow\ J[x, s_0(x)]}$

  – Aus $I[x]$ folgt $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y{:}R \mid O[x,y]\}$ (Axiom 3)

  $\boxed{\forall x{:}D.\forall z{:}R.\ \ I[x] \wedge O[x,z] \ \Rightarrow\ sat[z, s_0(x)]}$

  – Aus $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

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

Globalsuch-Schema: Korrektheitsbeweis

- **Abspalten und explizite Spezifikation der Hilfsfunktion aux**

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

  $\equiv$ if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else $\emptyset$

  FUNCTION $aux(x,s{:}D\times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

  RETURNS $\{y{:}R \mid O[x,y] \wedge sat[y,s]\}$

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

- **Korrektheit von f folgt aus der von aux mit Axiom 1, 3 & 4**
  - Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)
  - Aus $I[x]$ folgt $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y{:}R \mid O[x,y]\}$ (Axiom 3)
  - Aus $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

- **Partielle Korrektheit von aux folgt aus Axiom 5 & 2** und folgendem

  Lemma: Hält $aux[x,s]$ nach i Schritten an ($split^i_\Phi[x,s]=\emptyset$), so ist das Resultat

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

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

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

  Beweis: Induktion über i, Auffalten der Rekursion, Standardlemmata

$$\forall x{:}D.\, \forall z{:}R.\, \forall s{:}S.\, I[x] \wedge O[x,z] \wedge J[x,s] \Rightarrow (sat[z,s] \Leftrightarrow \exists k{:}\mathbb{N}.\, \exists t \in split^k_\Phi[x,s].\, z \in ext[t])$$

Globalsuch-Schema: Korrektheitsbeweis

- **Abspalten und explizite Spezifikation der Hilfsfunktion aux**

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

  $\equiv$ if $\Phi[x, s_0(x)]$ then $aux(x, s_0(x))$ else $\emptyset$

  FUNCTION $aux(x,s{:}D\times S)$ WHERE $I[x] \wedge J[x,s] \wedge \Phi[x,s]$

  RETURNS $\{y{:}R \mid O[x,y] \wedge sat[y,s]\}$

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

- **Korrektheit von f folgt aus der von aux mit Axiom 1, 3 & 4**
  - Für den Startwert $s_0(x)$ gilt $J[x, s_0(x)]$ (Axiom 1)
  - Aus $I[x]$ folgt $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} = \{y{:}R \mid O[x,y]\}$ (Axiom 3)
  - Aus $\{y{:}R \mid O[x,y] \wedge sat[y, s_0(x)]\} \neq \emptyset$ folgt $\Phi[x, s_0(x)]$ (Axiom 4)

- **Partielle Korrektheit von aux folgt aus Axiom 5 & 2** und folgendem

  Lemma: Hält $aux[x,s]$ nach i Schritten an ($split^i_\Phi[x,s]=\emptyset$), so ist das Resultat

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

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

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

  Beweis: Induktion über i, Auffalten der Rekursion, Standardlemmata

- Terminierung von aux folgt aus Axiom 6

Globalsuche als Algorithmentheorie

Es sind 7 neue Komponenten zu bestimmen

- **Zusätzliche Datentypen**
  - Raum der Deskriptoren für den Suchraum `Set(`R`)` (S)
- **Zusätzliche Eingabebedingungen für Hilfsfunktion**
  - Charakterisierung sinnvoller Deskriptoren (J)
  - Zusammenhang zwischen Deskriptoren und Suchraum (sat)
- **Komponenten des Algorithmus**
  - s_0: $D \to S$ — Axiom 1,3
  - $split$:$D \times S \to$ Set(S) — Axiom 2,5,6
  - Φ:$D \times S \to \mathbb{B}$ — Axiom 4,5,6
  - ext:$S \to$ Set(R) — Axiom 5

Lösungen müssen Beweis der Axiome 1–6 liefern

Kompakte Bezeichnung: wohlfundierte Globalsuchtheorie

- Struktur "$\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \Phi, ext)$, die alle 6 Axiome erfüllt

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: 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 $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für osearch

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: 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 $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für osearch

1. Zahlenmengen sind Vereinigungen von Intervallen

– Intervalle $\{l..r\}$ sind durch Zahlenpaare (l,r) beschreibbar
Wähle $S \equiv \mathbb{N}\times\mathbb{N}$

– sat beschreibt Elemente von $\{$l..r$\}$, also $sat[i,(l,r)] \equiv i\in\{l..r\}$

– Sinnvolle Indexmengen sind nicht leer: $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq|L|$

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: 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 $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für osearch

1. Zahlenmengen sind Vereinigungen von Intervallen

– Intervalle $\{l..r\}$ sind durch Zahlenpaare (l,r) beschreibbar
Wähle $S \equiv \mathbb{N}\times\mathbb{N}$

– sat beschreibt Elemente von $\{$l..r$\}$, also $sat[i,(l,r)] \equiv i\in\{l..r\}$

– Sinnvolle Indexmengen sind nicht leer: $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq|L|$

– Intervalle müssen k enthalten: $\Phi(L,k),(l,r) \equiv L_l\leq k\leq L_r$ Axiom 4

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: `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` $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für `osearch`

1. Zahlenmengen sind Vereinigungen von Intervallen

– Intervalle $\{l..r\}$ sind durch Zahlenpaare (l,r) beschreibbar
Wähle $S \equiv \mathbb{N}\times\mathbb{N}$

– sat beschreibt Elemente von $\{$`1..r`$\}$, also $sat[i,(l,r)] \equiv i\in\{l..r\}$

– Sinnvolle Indexmengen sind nicht leer: $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq|L|$

– Intervalle müssen k enthalten: $\Phi(L,k),(l,r) \equiv L_l\leq k\leq L_r$ Axiom 4

2. Das Anfangsintervall muß alle möglichen Indizes umfassen

– $s_0[(L,k)] \equiv (1,|L|)$ liefert Axiom 1,3

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: 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 $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für osearch

1. Zahlenmengen sind Vereinigungen von Intervallen

– Intervalle $\{l..r\}$ sind durch Zahlenpaare (l,r) beschreibbar
Wähle $S \equiv \mathbb{N}\times\mathbb{N}$

– sat beschreibt Elemente von $\{$l..r$\}$, also $sat[i,(l,r)] \equiv i\in\{l..r\}$

– Sinnvolle Indexmengen sind nicht leer: $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq|L|$

– Intervalle müssen k enthalten: $\Phi(L,k),(l,r) \equiv L_l\leq k\leq L_r$ Axiom 4

2. Das Anfangsintervall muß alle möglichen Indizes umfassen

– $s_0[(L,k)] \equiv (1,|L|)$ liefert Axiom 1,3

3. Intervalle lassen sich binär aufspalten liefert Axiom 2,6

– $split[(L,k),(l,r)] \equiv$ if $l<r$ then $\{(l,(l+r)/2),((l+r)/2+1,r)\}$ else $\emptyset$

Synthese eines Globalsuch-Algorithmus
Indizes eines Wertes in einer geordneten Liste

Spezifikation: 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 $\}$

Ziel: Bestimme Komponenten des Globalsuchschemas für osearch

1. Zahlenmengen sind Vereinigungen von Intervallen

– Intervalle $\{l..r\}$ sind durch Zahlenpaare (l,r) beschreibbar
Wähle $S \equiv \mathbb{N}\times\mathbb{N}$

– sat beschreibt Elemente von $\{$l..r$\}$, also $sat[i,(l,r)] \equiv i\in\{l..r\}$

– Sinnvolle Indexmengen sind nicht leer: $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq|L|$

– Intervalle müssen k enthalten: $\Phi(L,k),(l,r) \equiv L_l\leq k\leq L_r$ Axiom 4

2. Das Anfangsintervall muß alle möglichen Indizes umfassen

– $s_0[(L,k)] \equiv (1,|L|)$ liefert Axiom 1,3

3. Intervalle lassen sich binär aufspalten liefert Axiom 2,6

– $split[(L,k),(l,r)] \equiv$ if $l<r$ then $\{(l,(l+r)/2),((l+r)/2+1,r)\}$ else $\emptyset$

4. Aus einelementigen Intervallen kann man extrahieren liefert Axiom 5

– $ext[(l,r)] \equiv$ if l=r then $\{l\}$ else $\emptyset$

Analyse liefert Globalsuchtheorie für osearch

- **Theorie gs_osearch faßt alle Komponenten zusammen**

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

- **Alle 6 Axiome sind erfüllt**

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

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 aux(L,k,l,r)
    =      {z | z∈(if l=r then {l} else ∅) ∧ z∈{1..|L|} ∧ L_z=k}
       ∪⋃{ aux(L,k,n,m) |
             (n,m)∈(if l<r then {(l,(l+r)/2),((l+r)/2+1,r)}
                           else ∅)   ∧   L_n≤k≤L_m                  }
    in  if L_1≤k≤L_|L| then aux(L,k,1,|L|) else ∅
```

Nach (kontextabhängigen) Simplifikationen

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

Wie erzeugt man Globalsuch-Theorien?

- **Viele Komponenten basieren auf Standard-Suchstrukturen**
  - Deskriptorenraum S, Aufspaltung $split$, Extraktion ext sowie die Prädikate J und sat hängen im wesentlichen nur vom Suchraum R ab
  - Synthese benötigt formales Wissen über solche Suchstrukturen
- **Filter und Initialdeskriptoren sind aus Axiomen herleitbar**
  - Aber auch hier geht Wissen über mögliche Initialsuchräume für s_0 ein, das auf die spezielle Situation angepaßt wird
- **Wohlfundiertheit ist nicht automatisch zu beweisen**
  - Beweis für Terminierung von $split_\Phi$ muß im Voraus geführt werden
  - Verlangt Wissen über "Mindestfilter", die eine Terminierung garantieren

Spezialisiere generische Globalsuch-Theorien auf Problem

Spezialisierung einer Suchstruktur für osearch

- **(Terminierende) Standard-Binärsuche auf Integer-Intervallen**
  - Durchsucht Elemente eines Intervalls $\{\mathtt{m}..\mathtt{n}\}$ durch binäres Aufspalten

$$
\begin{array}{lll}
D & \mapsto & \mathbb{Z}\times\mathbb{Z} \\
R & \mapsto & \mathbb{Z} \\
I & \mapsto & \lambda \mathtt{m},\mathtt{n}.\ \mathtt{m}\leq\mathtt{n} \\
O & \mapsto & \lambda \mathtt{m},\mathtt{n},\mathtt{k}.\ \mathtt{k}\in\{\mathtt{m}..\mathtt{n}\} \\
S & \mapsto & \mathbb{Z}\times\mathbb{Z} \\
J & \mapsto & \lambda \mathtt{m},\mathtt{n},\mathtt{i},\mathtt{j}.\ \{\mathtt{i}..\mathtt{j}\}\subseteq\{\mathtt{m}..\mathtt{n}\} \\
s_0 & \mapsto & \lambda \mathtt{m},\mathtt{n}.\ (\mathtt{m},\mathtt{n}) \\
sat & \mapsto & \lambda \mathtt{k},\mathtt{i},\mathtt{j}.\ \mathtt{k}\in\{\mathtt{i}..\mathtt{j}\} \\
split & \mapsto & \lambda \mathtt{m},\mathtt{n},\mathtt{i},\mathtt{j}.\ \mathtt{if\ i<j\ then}\ \{(\mathtt{i},(\mathtt{i}+\mathtt{j})/2),\ (1+(\mathtt{i}+\mathtt{j})/2,\mathtt{j})\}\ \mathtt{else}\ \emptyset \\
\Phi & \mapsto & \lambda \mathtt{m},\mathtt{n},\mathtt{i},\mathtt{j}.\ \{\mathtt{i}..\mathtt{j}\}\neq\emptyset \\
ext & \mapsto & \lambda \mathtt{i},\mathtt{j}.\ \mathtt{if\ i=j\ then}\ \{\mathtt{i}\}\ \mathtt{else}\ \emptyset
\end{array}
$$

- **GS-Theorie muß auf Spezifikation von osearch angepaßt werden**

$$
\begin{array}{lll}
D & \mapsto & \mathtt{Seq}(\mathbb{N})\times\mathbb{N} \\
R & \mapsto & \mathbb{N} \\
I & \mapsto & \lambda \mathtt{L},\mathtt{k}.\ \mathtt{L}\neq[]\wedge\mathtt{ordered}(\mathtt{L}) \\
O & \mapsto & \lambda \mathtt{L},\mathtt{k},\mathtt{i}.\ \mathtt{i}\in\{1..|\mathtt{L}|\}\wedge\mathtt{L}_i=\mathtt{k}
\end{array}
$$

  - Ausgabebedingung verwendet $\{1..|\mathtt{L}|\}$ statt $\{\mathtt{m}..\mathtt{n}\}$, Suchraum ist $\mathbb{N}$ anstelle von $\mathbb{Z}$
  - Anpassung liefert $S \equiv \mathbb{N}\times\mathbb{N}$, $J[(L,k),(l,r)] \equiv 1\leq l\leq r\leq |L|$, $s_0[(L,k)] \equiv (1,|L|)$

    sat, $split$, ext bleiben unverändert, $\Phi(L,k),(l,r) \equiv L_l\leq k\leq L_r$ folgt aus Axiom 4

Strategie: Spezialisiere Standard-Suchstrukturen

- **Wissensbank enthält relevantes Programmierwissen**

  Globalsuchtheorie: allgemeine Suchstruktur für $\boldsymbol{R}$

  – Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt

  – Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \mathsf{True}, ext)$

  – Speichere (mehrere) Globalsuchtheorien für alle Grunddatentypen

Strategie: Spezialisiere Standard-Suchstrukturen

- **Wissensbank enthält relevantes Programmierwissen**

  Globalsuchtheorie: allgemeine Suchstruktur für $\boldsymbol{R}$

  – Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt

  – Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \mathsf{True}, ext)$

  – Speichere (mehrere) Globalsuchtheorien für alle Grunddatentypen

  Wohlfundiertheitsfilter garantieren Terminierung von $split_\Phi$

  – Erforderlich, wenn GS-Theorie $\mathcal{G}$ Axiom 6 nicht erfüllt

    Oft gibt es dann mehrere mögliche wf-Filter für $\mathcal{G}$ ↦ Ax 6

Strategie: Spezialisiere Standard-Suchstrukturen

- **Wissensbank enthält relevantes Programmierwissen**

  Globalsuchtheorie: allgemeine Suchstruktur für $\boldsymbol{R}$

  – Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt

  – Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \mathsf{True}, ext)$

  – Speichere (mehrere) Globalsuchtheorien für alle Grunddatentypen

  Wohlfundiertheitsfilter garantieren Terminierung von $split_\Phi$

  – Erforderlich, wenn GS-Theorie $\mathcal{G}$ Axiom 6 nicht erfüllt

    Oft gibt es dann mehrere mögliche wf-Filter für $\mathcal{G}$ ↦ Ax 6

- **Unterstützende Techniken der Globalsuch-Strategie**

  Spezialisierungsmechanismus für GS-Theorien ↦ §18, Folie 20

  – Wähle $\mathcal{G}$ passend zum Bildbereich der Spezifikation $spec = (D,R,I,O)$

  – Beweise $spec \ll spec_{\mathcal{G}}$ und extrahiere Substitution $\vartheta: D \to D_{\mathcal{G}}$

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

Strategie: Spezialisiere Standard-Suchstrukturen

- **Wissensbank enthält relevantes Programmierwissen**

  Globalsuchtheorie: allgemeine Suchstruktur für $\boldsymbol{R}$
  – Vorgefertigte Zerlegungsstruktur, die Axiome 1–5 erfüllt
  – Formalisiert als Objekt $\mathcal{G} = (D, R, I, O, S, J, s_0, sat, split, \mathsf{True}, ext)$
  – Speichere (mehrere) Globalsuchtheorien für alle Grunddatentypen

  Wohlfundiertheitsfilter garantieren Terminierung von $split_\Phi$
  – Erforderlich, wenn GS-Theorie $\mathcal{G}$ Axiom 6 nicht erfüllt
    Oft gibt es dann mehrere mögliche wf-Filter für $\mathcal{G}$ ↦ Ax 6

- **Unterstützende Techniken der Globalsuch-Strategie**

  Spezialisierungsmechanismus für GS-Theorien ↦ §18, Folie 20
  – Wähle $\mathcal{G}$ passend zum Bildbereich der Spezifikation $spec = (D,R,I,O)$
  – Beweise $spec \ll spec_\mathcal{G}$ und extrahiere Substitution $\vartheta: D \to D_\mathcal{G}$
  – Modifiziere $\mathcal{G}$ mit ϑ zu wohlfundierter Globalsuchtheorie für $spec$

  Vorwärtsinferenz (heuristisch)
  – Wähle Φ für $\mathcal{G}$, so daß Axiom 4 für Φ_ϑ beweisbar ↦ Notwendigkeit
  – Verfeinere Φ_ϑ um weitere Folgerungen von $sat \wedge O$ ↦ Effizienzsteigerung

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

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

- **Deskriptoren: gemeinsamer Präfix s** $\qquad sat[L, s] \equiv s \sqsubseteq L$

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

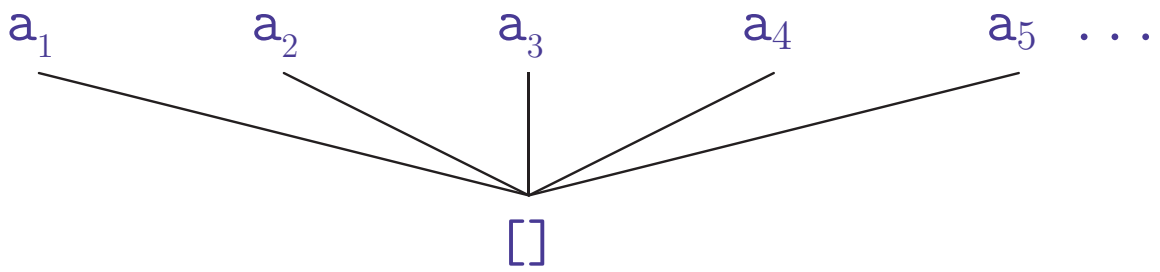
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L

$[]$

- **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 $M \subseteq \alpha$

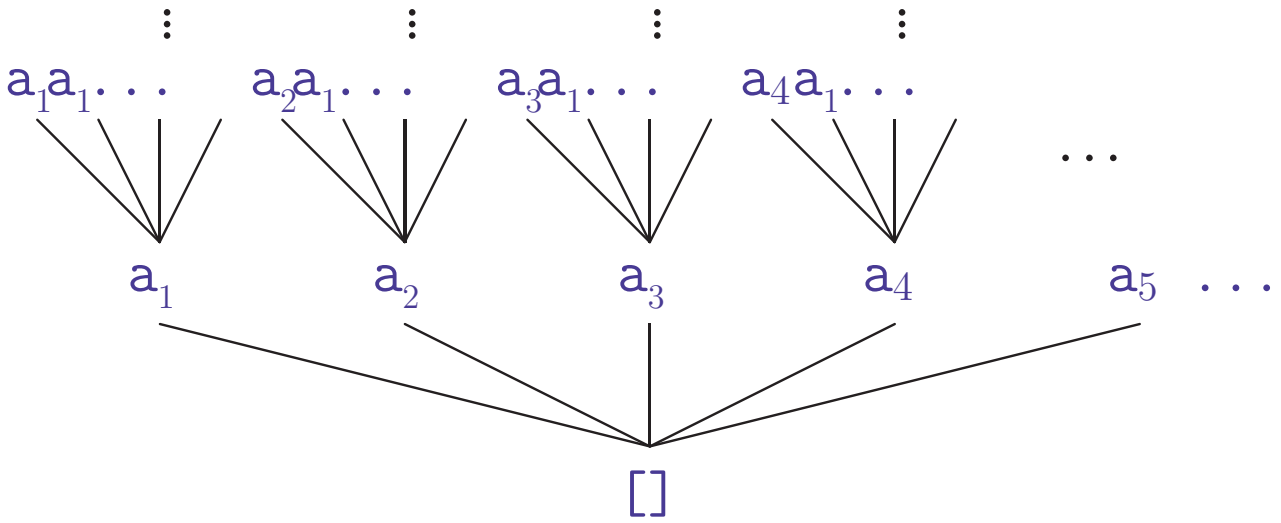
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren: gemeinsamer Präfix s** $\quad sat[L, s] \equiv s \sqsubseteq L$
- **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 $M \subseteq \alpha$

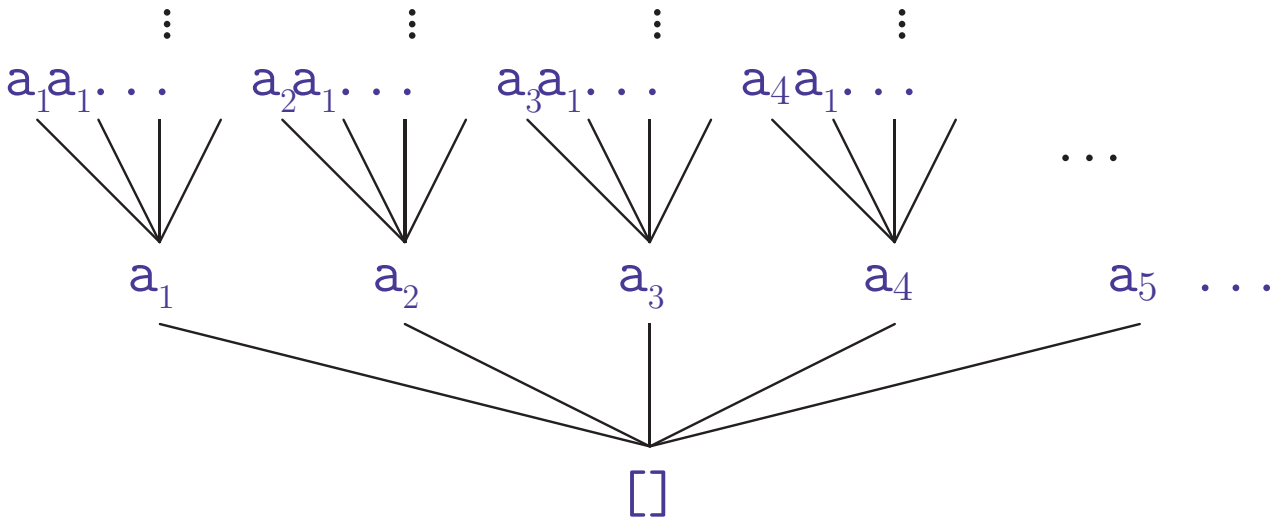
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



- **Deskriptoren: gemeinsamer Präfix s** $\quad sat[L, s] \equiv s \sqsubseteq L$
- **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 $M \subseteq \alpha$

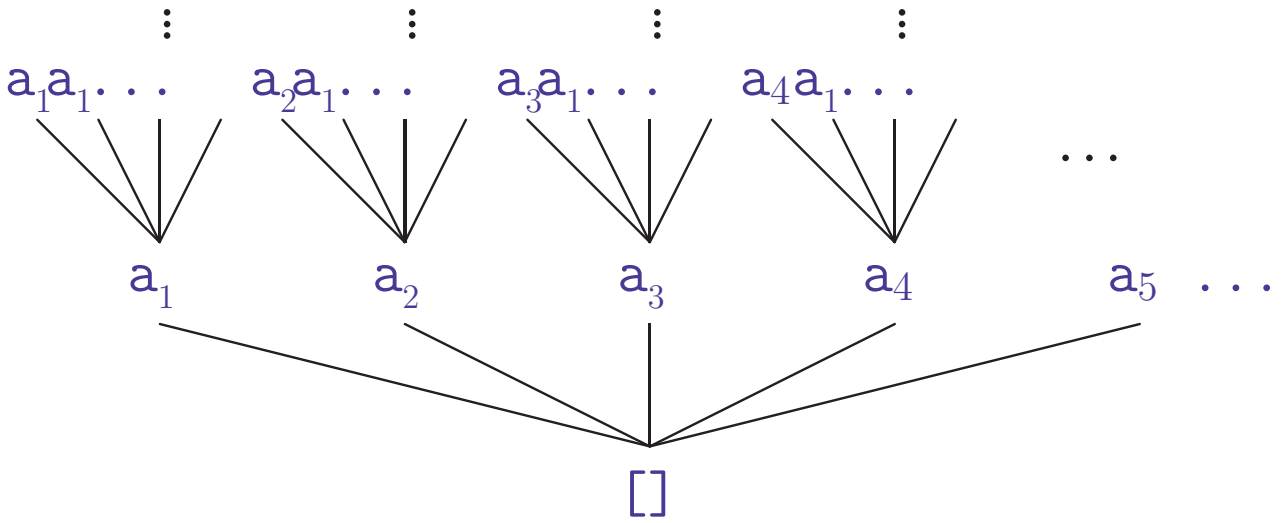
Suche $\hat{=}$ Aufzählung der Präfixe einer Liste L



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

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

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



- **Deskriptoren: gemeinsamer Präfix s** $\quad sat[L, s] \equiv s \sqsubseteq L$
- **Initialdeskriptor: leerer Präfix** $\quad s_0(M) \equiv []$
- **Splitting: Verlängern des Präfix** $\quad split[M, s] \equiv \{s \cdot a | a \in M\}$
- **Extraktion: Gesamter Präfix** $\quad ext[s] \equiv \{s\}$
- **Sinnvoll: nur Elemente aus M** $\quad J[M, s] \equiv s' \sqsubseteq' M$

Formale GS-Theorie für Listen über $M \subseteq \alpha$

- Deskriptoren: gemeinsamer Präfix s — $sat[L, s] \equiv s \sqsubseteq L$
- Initialdeskriptor: leerer Präfix — $s_0(M) \equiv []$
- Splitting: Verlängern des Präfix — $split[M, s] \equiv \{s \cdot a | a \in M\}$
- Extraktion: Gesamter Präfix — $ext[s] \equiv \{s\}$
- Sinnvoll: nur Elemente aus M — $J[M, s] \equiv s \,`\subseteq' M$

Darstellung als formales Objekt der Wissensbank

$\texttt{gs\_seq\_set}(\alpha) \equiv$

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	$\mapsto$	$\lambda \texttt{L,s}.\ \texttt{s} \sqsubseteq \texttt{L}$
$split$	$\mapsto$	$\lambda \texttt{M,s}.\ \{\texttt{s} \cdot \texttt{a} \mid \texttt{a} \in \texttt{M}\}$
Φ	$\mapsto$	$\lambda \texttt{M,s}.\ \texttt{true}$
ext	$\mapsto$	$\lambda \texttt{s}.\ \{\texttt{s}\}$

Axiome 1–5 sind formal beweisbar, Axiom 6 ist nicht erfüllt

Wohlfundiertheitsfilter für `gs_seq_set(`α`)`

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

Axiom 6 ist für jeden dieser Filter formal beweisbar

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	3	-5	1	2	
1	-2	-4	3		
-4	-1	-2			
-3	1				
-1					

- **Ziel: Berechne alle 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: Berechne alle Costas Arrays der Größe n**
- **Formalisierung vorkommender Begriffe:**

  $\texttt{dtrow}(L,j) \equiv [L_{\texttt{i}} - L_{\texttt{i}+j} \mid \texttt{i} \in [1..|L|-j]]$

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

- **Spezifikation des Problems**

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

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

Erforderliche zusätzliche mathematische Gesetze für Lösung des Costas Arrays Problems

- **Gesetze von Differenzentafeln**

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

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

$\vdots$

- **Gesetze duplikatenfreier Listen**

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

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

$\vdots$

Spezialisiere gs_seq_set($\mathbb{Z}$) / Φ_3 auf Costas-Arrays

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

D	$\mapsto$	Set(α)
R	$\mapsto$	Seq(α)
I	$\mapsto$	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$\cdot$a | a$\in$M}
ext	$\mapsto$	λs. {s}

Spezialisiere gs_seq_set($\mathbb{Z}$) / Φ_3 auf Costas-Arrays

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

D	$\mapsto$	Set(α)
R	$\mapsto$	Seq(α)
I	$\mapsto$	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$\cdot$a | a$\in$M}
ext	$\mapsto$	λs. {s}

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

Spezialisiere gs_seq_set($\mathbb{Z}$) / Φ_3 auf Costas-Arrays

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

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

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

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}

Spezialisiere gs_seq_set($\mathbb{Z}$) / Φ_3 auf Costas-Arrays

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

D	$\mapsto$	Set(α)
R	$\mapsto$	Seq(α)
I	$\mapsto$	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: $\mathtt{I}_{\mathcal{G}}(M)$=true**

Spezialisiere gs_seq_set($\mathbb{Z}$) / Φ_3 auf Costas-Arrays

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

D	$\mapsto$	Set(α)
R	$\mapsto$	Seq(α)
I	$\mapsto$	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}

↦ §18, Folie 28

Spezialisiere gs_seq_set(ℤ) / Φ_3 auf Costas-Arrays

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

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

1. Bildbereich stimmt mit R_G=Seq(ℤ) überein

2. Eingabebereiche ℤ, D_G=Set(ℤ) sind anzupassen

3. Keine Eingabebedingung zu prüfen: $I_G(M)$=true

4. Zu zeigen ist also:

$\forall n{:}\mathbb{Z}.\ n{\geq}1 \Rightarrow \exists M{:}\mathtt{Set}(\mathbb{Z}).\forall p{:}\mathtt{Seq}(\mathbb{Z}).\ O(n,p) \Rightarrow \mathtt{range}(p){\subseteq}M$

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

– Auffalten von perm liefert: perm(p,{1..n}) ⇒ range(p)⊆{1..n}

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

↦ §18, Folie 28

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

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

$\Phi_{3,\vartheta}(n,s) = \Phi_3(\{1..n\},s) = \mathtt{nodups}(s)$

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

Synthese einer Globalsuche für Costas Arrays

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

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

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

3. Wähle WF-Filter Φ so, daß Φ_ϑ notwendig für $\mathcal{G}_\vartheta$ beweisbar

– perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) $\wedge$ s$\sqsubseteq$p $\Rightarrow$ Φ[{1..n},s]

– Leicht beweisbar nur für Φ_3[M,s] = nodups(s)

4. Leite zusätzlichen notwendigen Filter Ψ ab

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

5. Instantiiere den Standard-Globalsuchalgorithmus

FUNCTION Costas (n:$\mathbb{Z}$) WHERE n$\geq$1 RETURNS {p:Seq($\mathbb{Z}$) | ... }
$\equiv$ let rec Costas_{gs}(n,s)
= {p | p$\in${s} $\wedge$ perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) }
$\cup \bigcup$ { Costas_{gs}(n,t) | t$\in${s·i|i$\in${1..n}}
$\wedge$ nodups(t) $\wedge$ $\forall$j<|t|.nodups(dtrow(t,j))}
in if nodups([]) $\wedge$ $\forall$j<|[]|.nodups(dtrow([],j))
then Costas_{gs}(n,[]) else $\emptyset$

Synthesesestrategie für Globalsuchalgorithmen

Gegeben eine Problemspezifikation

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

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

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

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

Passt Globalsuche zu Sortierverfahren?

Spezifikation muß mengenwertig formuliert werden

```
FUNCTION sort(L:Seq(ℤ)) WHERE true
RETURNS {S:Seq(ℤ) | rearranges(L,S) ∧ ordered(S)}
```

1. Wähle GS-theorie $\mathcal{G}$ = `gs_seq_set(ℤ)`, beweise $(D, R, I, O) \ll \mathcal{G}$

– Zeige: `∀L:Seq(ℤ). ∃M:Set(ℤ). ∀S:Seq(ℤ). SORT(L,S) ⇒ range(S)⊆M`

– Aus `rearranges(L,S)` folgt `range(S)=range(L)`, also ist ϑ`(L) = range(L)`

2. Wähle wf-Filter Φ so, daß Φ_ϑ notwendig für $\mathcal{G}_\vartheta$ beweisbar

– `rearranges(L,S) ∧ ordered(S) ∧ s⊑S ⇒ Φ[range(L),s]`

– Aus `rearranges(L,S)` folgt `|S|=|L|`, also `|s|≤|L|`

– Instanz von $\Phi_{1,\vartheta}$`[L,s] = |s|≤`k für k=`|L|`

3. Leite zusätzlichen notwendigen Filter Ψ ab

– Aus `ordered(S) ∧ s⊑S` folgt `ordered(s) =:` Ψ`[L,s]`

Weitere ableitbare Eigenschaften sind algorithmisch nicht verwertbar

4. Instantiiere und optimiere den Standard-Globalsuchalgorithmus

Sortieren als Globalsuchalgorithmus

- **Instantiiere Globalsuchschema mit $\mathcal{G}_\vartheta$, Φ_ϑ und Ψ**

```
FUNCTION sort(L:Seq(ℤ)) WHERE true
  RETURNS {S:Seq(ℤ) | rearranges(L,S) ∧ ordered(S)}
≡ if |[]|≤|L| ∧ ordered([]) then sort_gs(L,[]) else ∅
FUNCTION sort_gs(L:Seq(ℤ),s:Seq(ℤ)) WHERE |s|≤|L| ∧ ordered(s)
  RETURNS {S:Seq(ℤ) | rearranges(L,S) ∧ ordered(S)}
≡   {S | S∈{s} ∧ rearranges(L,S) ∧ ordered(S) }
  ∪ ⋃{ sort_gs(L,t) | t∈{ s·i|i∈range(L) } ∧ |t|≤|L| ∧ ordered(t)}
```

- **Nach (kontextabhängigen) Simplifikationen**

```
FUNCTION sort(L:Seq(ℤ)) WHERE true
  RETURNS {S:Seq(ℤ) | rearranges(L,S) ∧ ordered(S)}
≡ sort_gs(L,[])
FUNCTION sort_gs(L:Seq(ℤ),s:Seq(ℤ)) WHERE |s|≤|L| ∧ ordered(s)
  RETURNS {S:Seq(ℤ) | rearranges(L,S) ∧ ordered(S)}
≡ if rearranges(L,s) then {s}
      else ⋃{ sort_gs(L,s·i) | i∈L ∧ last(s)≤i}
```

- **Weitere Optimierungen sind möglich**
  - Dennoch ist Globalsuche nicht das beste Prinzip für Sortierverfahren

Integration der Methodik in eine Beweisumgebung

- **Formalisiere notwendige Begriffe**
  – GS-Theorie, Filter, Wohlfundiertheit, Notwendigkeit, Generalisierung,...
- **Formuliere und beweise Globalsuch-Synthesetheorem**

  $\forall$`spec=(`D`,`R`,I,O):SPEC.`

  `FUNCTION F(x:`D`):Set(`R`) WHERE I(x) RETURNS {z | O(x,z)}` ist erfüllbar

  $\Leftarrow$ $\exists$`G:GS. G is a GS-Theory`

  $\wedge$ $\exists\vartheta$`:`$D \not\to D_{\mathtt{G}}$`. G generalizes spec with` ϑ

  $\wedge$ $\exists\Phi$`:Filters(G).` Φ `wf-filter for G` $\wedge$ Φ_ϑ `necessary for` $\mathtt{G}_\vartheta$`(spec)`

  $\wedge$ $\exists\Psi$`:Filters(`$\mathtt{G}_\vartheta$`(spec)).` Ψ `necessary for` $\mathtt{G}_\vartheta$`(spec)`
- **Wende Synthesetheorem auf konkrete Probleme an**
  – Beweisaufgaben sind formal leichter zu lösen als bei D&C Synthesen
    · Wähle G (und Korrektheitsbeweis) aus Wissensbank
    · Bestimme ϑ durch Extraktion aus Spezialisierungsbeweis
    · Wähle Φ aus Wissensbank und beweise Notwendigkeit
    · Leite Folgerungen von $sat \wedge O$ ab und wähle Ψ entsprechend
  – Extrahiere Globalsuch-Algorithmus aus dem Beweis

GS-Theorien: Formalisierung von Struktur und Axiomen

$\mathtt{GS} \equiv \mathtt{spec{:}SPEC} \times \mathtt{S{:}TYPES} \times D{\times}S{\to}\mathbb{B} \times \ldots$

G is a GS-theory

$\equiv \mathtt{let}\ ((D,R,\mathtt{I},\mathtt{O}),\ \mathtt{S},\ \mathtt{J},\ \mathtt{s}_0,\ \mathtt{sat},\ \mathtt{split},\ \mathtt{ext}) = G\ \mathtt{in}$

$\forall \mathtt{x}{:}D.\ \mathtt{I(x)} \Rightarrow \mathtt{s}_0\mathtt{(x)}\ \text{hält} \wedge \mathtt{J(x,s}_0\mathtt{(x))}$

$\wedge\ \forall \mathtt{x}{:}D.\forall \mathtt{s}{:}\mathtt{S}.\ \mathtt{I(x)} \wedge \mathtt{J(x,s)} \Rightarrow \forall \mathtt{t} \in \mathtt{split(x,s)}.\mathtt{J(x,t)}$

$\wedge\ \forall \mathtt{x}{:}D.\forall \mathtt{z}{:}\mathtt{R}.\ \mathtt{I(x)} \wedge \mathtt{O(x,z)} \Rightarrow \mathtt{sat(z,s}_0\mathtt{(x))}$

$\wedge\ \forall \mathtt{x}{:}D.\forall \mathtt{s}{:}\mathtt{S}.\ \mathtt{I(x)} \wedge \mathtt{J(x,s)} \Rightarrow \forall \mathtt{z}{:}\mathtt{R}.\mathtt{O(x,z)} \Rightarrow$

$\mathtt{sat(z,s)} \Leftrightarrow \exists \mathtt{k}{:}\mathbb{N}.\exists \mathtt{t} \in \mathtt{split}^{\mathtt{k}}\mathtt{(x,s)}.\ \mathtt{z} \in \mathtt{ext(t)}$

Filters(G)

$\equiv \mathtt{let}\ \ldots{=}G\ \mathtt{in}\ D{\times}S{\to}\mathbb{B}$

Φ necessary for G

$\equiv \mathtt{let}\ \ldots{=}G\ \mathtt{in}\ \ \forall \mathtt{x}{:}D.\forall \mathtt{s}{:}\mathtt{S}.\ \exists \mathtt{z}{:}R.\,\mathtt{sat(z,s)} \wedge \mathtt{O(x,z)} \ \Rightarrow\ \Phi\mathtt{(x,s)}$

Φ wf-filter for G

$\equiv \mathtt{let}\ \ldots{=}G\ \mathtt{in}\ \ \forall \mathtt{x}{:}D.\forall \mathtt{s}{:}\mathtt{S}.\,\mathtt{I(x)} \wedge \mathtt{J(x,s)} \Rightarrow\ \exists \mathtt{k}{:}\mathbb{N}.\,\mathtt{split}_\Phi{}^{\mathtt{k}}\mathtt{(x,s)} = \emptyset$

G generalizes *spec* with ϑ

$\equiv \mathtt{let}\ \ldots{=}G.\ \mathtt{and}\ (D',R',\mathtt{I}',\mathtt{O}')=\mathit{spec}\ \mathtt{in}$

$R' \subset R \wedge \forall \mathtt{x}{:}D'.\ \mathtt{I}'\mathtt{(x)} \Rightarrow \mathtt{I}(\vartheta\mathtt{(x)}) \wedge \forall \mathtt{z}{:}R'.\,\mathtt{O}'\mathtt{(x,z)} \Rightarrow \mathtt{O}(\vartheta\mathtt{(x)},\mathtt{z})$

$G_\vartheta(\mathit{spec})$ ...

⋮

Synthese von Costas Arrays mit einem Beweissystem

```
# top
```

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

BY lemma 'GS-Synthese'

1. $\vdash$ $\exists$G. G is a GS-Theory
$\wedge$ $\exists\vartheta$. G generalizes spec with ϑ
$\wedge$ $\exists\Phi$. Φ wf-filter for G $\wedge$ Φ_ϑ necessary for G_ϑ(spec)
$\wedge$ $\exists\Psi$. Ψ necessary for G_ϑ(spec)
where spec = ($\mathbb{Z}$, Seq($\mathbb{Z}$), λn. n$\geq$1,
λn,p. perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)))

Typinformation zugunsten der Lesbarkeit unterdrückt (Display-Mechanismus)

Restliche Aufgabe: Bestimmung von G, ϑ, Φ und Ψ.

Extraktion des Algorithmus automatisch nach Lösung dieses Ziels

Costas Arrays Synthese (2): Wahl der GS-Theorie

```
# top  1
```

$\vdash \exists G.\ G$ is a GS-Theory
$\quad \wedge\ \exists \vartheta.\ G$ generalizes spec with ϑ
$\quad \wedge\ \exists \Phi.\ \Phi$ wf-filter for $G\ \wedge\ \Phi_\vartheta$ necessary for G_ϑ(spec)
$\quad \wedge\ \exists \Psi.\ \Psi$ necessary for G_ϑ(spec)
$\quad$ where spec $= (\mathbb{Z},\ \mathrm{Seq}(\mathbb{Z}),\ \lambda n.\ n \geq 1,$
$\qquad \lambda n,p.\ \mathrm{perm}(p, \{1..n\}) \wedge \forall j < |p|.\ \mathrm{nodups}(\mathrm{dtrow}(p,j)))$

BY $\exists$-intro gs_seq($\mathbb{Z}$) THEN unfold ‘perm‘

1. $\vdash$ gs_seq($\mathbb{Z}$) is a GS-Theory

2. $\vdash\ \exists \vartheta.$ gs_seq($\mathbb{Z}$) generalizes spec with ϑ
$\quad \wedge\ \exists \Phi.\ \Phi$ wf-filter for gs_seq($\mathbb{Z}$)
$\qquad \wedge\ \Phi_\vartheta$ necessary for gs_seq($\mathbb{Z}$)$_\vartheta$(spec)
$\quad \wedge\ \exists \Psi.\ \Psi$ necessary for gs_seq($\mathbb{Z}$)$_\vartheta$(spec)
$\quad$ where spec
$\quad = (\mathbb{Z},\ \mathrm{Seq}(\mathbb{Z}),\ \lambda n.\ n \geq 1,$
$\qquad \lambda n,p.\ \mathrm{nodups}(p) \wedge \mathrm{range}(p) = \{1..n\} \wedge \forall j < |p|.\ \mathrm{nodups}(\mathrm{dtrow}(p,j)))$

Teilziel 1 ist Lemma der Wissensbank

Costas Arrays Synthese (3): Bestimmung der Substitution ϑ

```
# top  1 2
```

$\vdash$ $\exists\vartheta$. gs_seq($\mathbb{Z}$) generalizes spec with ϑ
$\wedge$ $\exists\Phi$. Φ wf-filter for gs_seq($\mathbb{Z}$)
$\wedge$ Φ_ϑ necessary for gs_seq($\mathbb{Z}$)$_\vartheta$(spec)
$\wedge$ $\exists\Psi$. Ψ necessary for gs_seq($\mathbb{Z}$)$_\vartheta$(spec)
where spec
= ($\mathbb{Z}$, Seq($\mathbb{Z}$), λn. n$\geq$1,
λn,p. nodups(p) $\wedge$ range(p)={1..n} $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)))

BY andR THEN exR λn.{1..n}

1. $\vdash$ gs_seq($\mathbb{Z}$) generalizes spec with λn.{1..n}
where spec = ...

2. $\vdash$ $\exists\Phi$. Φ wf-filter for gs_seq($\mathbb{Z}$)
$\wedge$ Φ_ϑ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
$\wedge$ $\exists\Psi$. Ψ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
where spec = ...

Teilziel 1 lösbar durch einfachen Beweiser

In Teilziel 1 kann ϑ auch aus einem Beweis von gs_seq($\mathbb{Z}$) generalizes spec extrahiert werdend. Hierzu wird das konstruktive Auswahlaxiom (§9, Folie 39) angewandt und dann (nach Auffalten) folgendes Ziel bewiesen, aus dem sich dann ϑ = λn.{1..n} ergibt:

$\forall$n:$\mathbb{Z}$. n$\geq$1 $\Rightarrow$ $\exists$M:Set($\mathbb{Z}$).$\forall$p:Seq($\mathbb{Z}$). nodups(p)$\wedge$range(p)={1..n}$\wedge$$\forall$j<|p|.nodups(dtrow(p,j))
$\Rightarrow$ range(p)$\subseteq$M

Costas Arrays Synthese (4): Wahl des Filters Φ

```
# top  1 2 2
```

$\vdash$ $\exists\Phi.\ \Phi$ wf-filter for gs_seq($\mathbb{Z}$)
$\quad\wedge\ \Phi_\vartheta$ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
$\wedge\ \exists\Psi.\ \Psi$ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
where spec
= ($\mathbb{Z}$, Seq($\mathbb{Z}$), λn. n$\geq$1,
$\quad\lambda$n,p.nodups(p) $\wedge$ range(p)=$\{1..n\}$ $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)))

BY $\exists$-intro λS,V.nodups(V)

1. $\vdash$ λS,V.nodups(V) wf-filter for gs_seq($\mathbb{Z}$)

2. $\vdash$ λn,V.nodups(V) necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
   where spec = ...

3. $\vdash$ $\exists\Psi.\ \Psi$ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
   where spec = ...

Teilziel 1 ist Lemma der Wissensbank

COSTAS ARRAYS SYNTHESE (5): Φ_ϑ NOTWENDIG/ WAHL VON Ψ

```
# top  1 2 2 2
```

$\vdash$ λn,V.nodups(V) necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
  where spec
  = ($\mathbb{Z}$, Seq($\mathbb{Z}$), λn. n$\geq$1,
     λn,p. nodups(p) $\wedge$ range(p)={1..n} $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)))

BY undo_abstractions THEN unfold ‘specialize‘ THEN unfold ‘necessary‘

1. $\vdash$ $\forall$n.$\forall$V.
     ($\exists$p. nodups(p) $\wedge$ range(p)={1..n} $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) $\wedge$ V$\sqsubseteq$p)
       $\Rightarrow$ nodups(V)

     * BY Auto

Auto zerlegt logisch und konsultiert dabei Fakten der Wissensbank

```
# top  1 2 2 3
```

$\vdash$ $\exists\Psi$. Ψ necessary for gs_seq($\mathbb{Z}$)$_{\lambda n.\{1..n\}}$(spec)
  where spec = ...

BY undo_abstractions THEN unfold ‘specialize‘ THEN unfold ‘necessary‘

1. $\vdash$ $\exists\Psi$.$\forall$n.$\forall$V.
     ($\exists$p. nodups(p) $\wedge$ range(p)={1..n} $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) $\wedge$ V$\sqsubseteq$p)
       $\Rightarrow$ Ψ(n,V)

* BY $\exists$-intro λn,V. $\forall$j$\in$dom(V).nodups(dtrow(V,j) THEN Auto

Globalsuch-Synthese im Rückblick

- **Wissensbasierte Erzeugung eines Suchalgorithmus**
  - Sinnvoll, wenn Problem rekursiv zerlegbar in "additive" Teillösungen
  - Schematischer Algorithmus verwendet 4 Basiskomponenten
  - Komponenten lassen sich in einem Schritt aus generischen Suchstrukturen erzeugen und durch Filter verfeinern
  - Standard-Globalsuchtheorien und passende Wohlgeformtheitsfilter müssen in Wissensbank vorgespeichert sein
  - Techniken sind Spezialisierung und Vorwärtsinferenz
- **Erfolgreich in der Praxis**
  - Schematische Algorithmen sind schnell erzeugt und formal optimiert
  - Effiziente Lösung von Scheduling Problemen wurde 1993 demonstriert Entscheidend war Wahl der richtigen Algorithmenstruktur
  - Korrektheit der Algorithmen ist durch theoretische Vorarbeiten garantiert
  - Korrektheit der implementierten Methode kann durch Integration in Beweissysteme gesichert werden