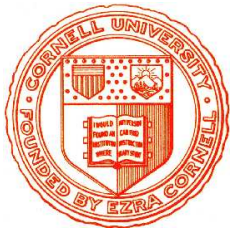
Automatisierte Logik und Programmierung

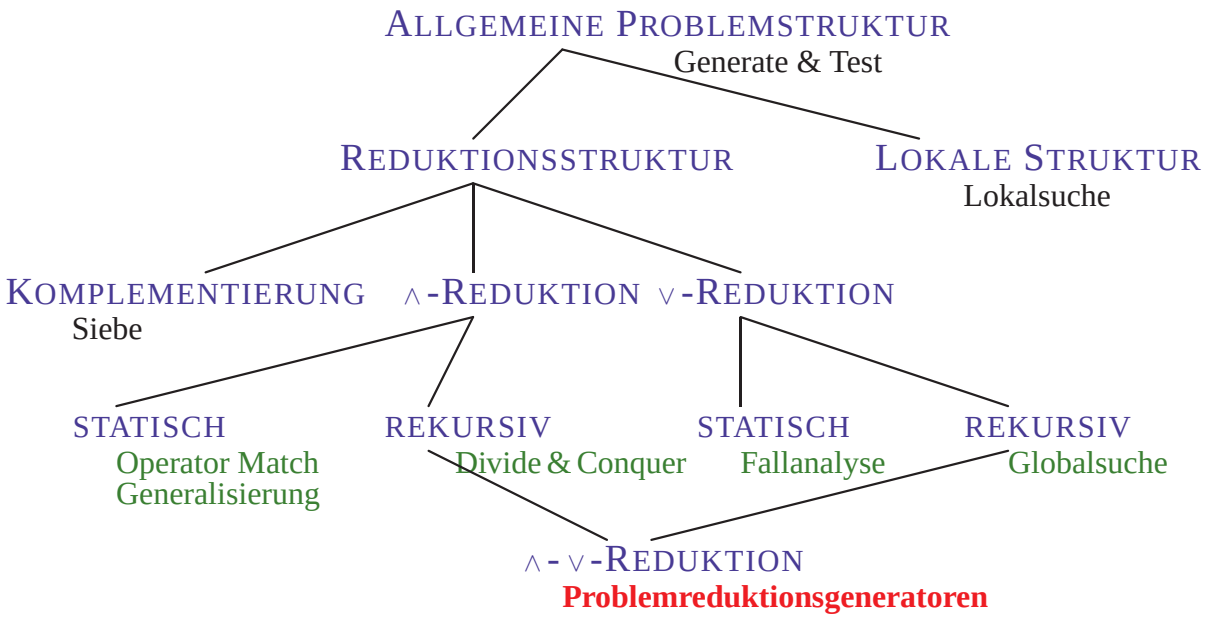
Einheit 20



Problemreduktionsgeneratoren

1. Algorithmenschema
2. Korrektheit
3. Synthesestrategie

Problemreduktionsgeneratoren



- **$\vee$-$\wedge$-Reduktion von Problemen**
  - Problem besitzt mehrere Lösungen
  - Gesamtlösung ist Summe unabhängiger Einzellösungen ($\vee$-Reduktion)
  - Einzellösungen aus Teillösungen zusammengesetzt ($\wedge$-Reduktion)
  - Verallgemeinert Dynamisches Programmieren, Spielbaumsuche, ...
- **Synthese ähnlich zu Divide & Conquer Techniken**

Ein typischer Problemreduktionsgenerator

Syntaxanalyse: bestimme alle Variablen einer Grammatik G in Chomsky-NF, aus denen ein Wort w ableitbar ist

- **Formale Problemspezifikation (abstrahiert)**

  `FUNCTION Parse(G=(V,T,P,S),w:Grammars×T*)`
  `  RETURNS {A:V | v` $\stackrel{*}{\longrightarrow}$ `w}`

- **Ansatz: CYK Table-Filling Algorithmus**

  Berechne Mengen $V_{i,j} = \{A \in V \mid A \stackrel{*}{\longrightarrow} w_i \ldots w_j\}$ für $i \leq j \leq |w|$ iterativ

  $V_{i,j} \equiv$ `if` $i{=}j$ `then` $\{A \in V \mid A \to w_i \in P\}$
  `else` $\{A \in V \mid \exists i \leq k < j.\ \exists A \to BC \in P.\ B \in V_{i,k} \wedge C \in V_{k+1,j}\}$

  Lösung ist die Menge $V_{1,|w|}$

- **Berechnung der $V_{i,j}$ ist $\vee$- $\wedge$-Reduktion**
  - $\wedge$-Reduktion für jedes k: Eintrag A benötigt $B \in V_{i,k}$ und $C \in V_{k+1,j}$
  - $\vee$-Reduktion: Vereinigung der Resultate aller k zwischen i und $j-1$

Vereinheitlichung der Notation

```
FUNCTION Parse(G=(V,T,P,S),w:Grammars×T*)
  RETURNS {A:V | v ⟶* w}
≡ aux(G,w,1,|w|)

FUNCTION aux(G,w,i,j:Grammars×T*×ℕ×ℕ) WHERE i,j∈{1..|w|}
  RETURNS {A:V | v ⟶* w_i..w_j}
≡ if i=j then {A | A⟶w_i∈P}
         else {A | ∃i≤k<j. ∃A⟶BC∈P.
                   B∈aux(G,w,i,k) ∧ C∈aux(G,w,k+1,j)}
```

- **Vereinheitlichung durch separierte Beschreibung**
  - – Dekomposition der Suche $i..j$ an Stelle k in $\{i..k,\ k{+}1..j\}$
  - – Rekursive Bestimmung der Lösung für $i..j$ wenn $i{<}j$

    Direkte Bestimmung der Lösung wenn $i{=}j$
  - – Komposition der Lösungen für k (Rückwärtsanwendung der Regeln)
  - – Gesamtresultat ist Vereinigung aller so entstandenen Lösungen

```
FUNCTION aux(G,w,i,j:Grammars×T*×ℕ×ℕ) ...
 ≡ ⋃_k( Compose_k ∘ (aux_k1 × aux_k2) ∘ Decompose_k )(i,j)
```

Problemreduktionsgeneratoren: Grundidee

- **Verallgemeinertes Divide & Conquer Schema**
  - Unabhängige Divide & Conquer Algorithmen für jede Einzellösung
    - Dekompositionen und Kompositionen verschieden
    - Teilprobleme können einander überlappen
    - Basisfall primitiver Eingaben wird einfaches Divide & Conquer
  - Lösung durch Vereinigung aller Einzellösungen berechnen
- **Allgemeines Algorithmenschema**

  $\texttt{FUNCTION } f(x\texttt{:}D)\texttt{ WHERE } I[x]\texttt{ RETURNS } \{y\texttt{:}R \mid O[x,y]\}$
  $\equiv \bigcup_i (\mathit{Compose}_i \circ (f_{i_1} \times .. \times f_{i_k}) \circ \mathit{Decompose}_i)(x)$
- **4 zentrale Komponenten der Algorithmentheorie**
  - $\mathit{Decompose}_i$: $D \to D_{i_1} \times .. \times D_{i_k}$ Aufspalten der Eingabe in Teilprobleme
  - Hilfsfunktionen f_{i_j}: $D_{i_j} \to R_{i_j}$ evtl. rekursiver Aufruf von f
  - $\mathit{Compose}_i$: $R_{i_1} \times .. \times R_{i_k} \to R$ Zusammensetzen der Teillösungen
  - Wohlfundierte Ordnung $\succ$ für Terminierungsgarantie
  - **Erweitertes Strong Problem Reduction Principle**

Syntaxanalyse als Problemreduktionsgenerator

- **Schematischer Algorithmus der Hilfsfunktion**

  FUNCTION $aux...$ $\equiv$ $\bigcup_k$ ($Compose_k$ $\circ$ ($aux_{k_1} \times aux_{k_2}$) $\circ$ $Decompose_k$) (i,j)

- **Formale Komponenten des Schemas**
  - Grammatik G und Wort w sind Konstante der Hilfsfunktion aux
  - $Decompose_k(i,j) \equiv ((i,k),\ (k{+}1,j))$
  - $aux_{k_1}(i,j) = aux_{k_2}(i,j) \equiv$ if $i{=}j$ then $\{A \in V \mid A \to w_i \in P\}$ else $aux(i,j)$
  - $Compose_k(V,V') \equiv \{A \in V \mid \exists A \to BC \in P.\ B \in V \wedge C \in V'\}$

- **Instantiierter Algorithmus (nach Simplifikationen)**

  FUNCTION Parse(G=(V,T,P,S),w:Grammars$\times$T*) RETURNS $\{$A:V$\mid$v $\xrightarrow{*}$ w$\}$

  $\equiv$ aux(G,w,1,|w|)

  FUNCTION aux(G,w,i,j:Grammars$\times$T$^*\times\mathbb{N}\times\mathbb{N}$) WHERE i,j$\in\{$1..|w|$\}$

  RETURNS $\{$A:V $\mid$ v $\xrightarrow{*}$ $w_i..w_j\}$

  $\equiv$ $\bigcup\{\{$A $\mid \exists$A$\to$BC$\in$P. (if i=k then B$\to w_i \in$P else B$\in aux$(G,w,i,k)) $\wedge$ (if k+1=j then C$\to w_j \in$P else C$\in aux$(G,w,k+1,j))$\}$ | k$\in\{$i..j-1$\}\}$

Korrektheit von Problemreduktionsgeneratoren

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

$\equiv \bigcup_i (Compose_i \circ (f_{i_1} \times .. \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

1. O **ist rekursiv zerlegbar in** O_{D_i}, $O_{i_1} \times .. \times O_{i_k}$ **und** O_{C_i} *(SPRP)*

$$O[x,z] \Leftrightarrow \exists i{:}\mathbb{N},\ \bar{y}_i{=}(y_{i_1},..,y_{i_k}){:}D_{i_1}\times..D_{i_k},\ \bar{w}_i{=}(w_{i_1},..,w_{i_k}){:}R_{i_1}\times..R_{i_k}.$$
$$O_{D_i}[x,\bar{y}_i] \wedge O_{i_1}[y_{i_1},w_{i_1}] \wedge .. \wedge O_{i_k}[y_{i_k},w_{i_k}] \wedge O_{C_i}[\bar{w}_i,z]$$

2. Dekompositionen erfüllen O_{D_i} **und 'verkleinern' Problem**

FUNCTION $f_{d_i}(x{:}D)$ WHERE $I[x]$ RETURNS $\{\bar{y}_i{:}D_{i_1}\times..D_{i_k} \mid O_{D_i}[x,\bar{y}_i] \wedge x \succ \bar{y}_i \wedge I_{i_1,..i_k}[\bar{y}_i]\}$

wobei $x \succ \bar{y}_i \equiv x \succ y_{i_j}$ für alle j mit $D_{i_j}{=}D$

3. Hilfsfunktionen f_{i_j} **erfüllen** O_{i_j}

FUNCTION $f_{i_j}(y_{i_j}{:}D_{i_j})$ WHERE $I_{i_j}[y_{i_j}]$ RETURNS $\{w_{i_j}{:}R_{i_j} \mid O_{i_j}[y_{i_j},w_{i_j}]\}$

4. Kompositionen erfüllen O_{C_i}

FUNCTION $f_{c_i}(\bar{w}_i{:}R_{i_1}\times..\times R_{i_k})$ WHERE true RETURNS $\{z_i{:}R \mid O_{C_i}[\bar{w}_i,z_i]\}$

5. Verkleinerungsrelation $\succ$ **ist wohlfundierte Ordnung auf** D

Korrektheitsbeweis analog zu Divide & Conquer

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

$\equiv \bigcup_i (\mathit{Compose}_i \circ (f_{i_1} \times .. \times f_{i_k}) \circ \mathit{Decompose}_i)(x)$

- **Partielle Korrektheit: strukturelle Induktion über $(D, \succ)$**
  - $z \in f(x) \Leftrightarrow \exists i.\ z \in (\mathit{Compose}_i \circ (f_{i_1} \times .. \times f_{i_k}) \circ \mathit{Decompose}_i)(x)$
  - $\mathit{Decompose}_i[x]$ liefert alle $\bar{y}_i = (y_{i_1}, .., y_{i_k})$ mit $O_{D_i}[x, \bar{y}_i]$ und $x \succ \bar{y}_i$ Axiom 2
  - Jedes $f_{i_j}(y_{i_j})$ liefert alle w_{i_j} mit $O_{i_j}[y_{i_j}, w_{i_j}]$ Axiom 3
  - $\mathit{Compose}_i[w_{i_1}, .., w_{i_k}]$ liefert alle z_i mit $O_{C_i}[w_{i_1}, .., w_{i_k}, z_i]$ Axiom 4
  - z ist eines dieser z_i und es gilt $O[x, z]$ Axiom 1

$$O[x,z] \Leftrightarrow \exists i{:}\mathbb{N},\ \bar{y}_i = (y_{i_1}, .., y_{i_k}){:}D_{i_1} \times .. D_{i_k},\ \bar{w}_i = (w_{i_1}, .., w_{i_k}){:}R_{i_1} \times .. R_{i_k}\ .\ O_{D_i}[x, \bar{y}_i] \wedge O_{i_1}[y_{i_1}, w_{i_1}] \wedge .. \wedge O_{i_k}[y_{i_k}, w_{i_k}] \wedge O_{C_i}[\bar{w}_i, z]$$

- **Terminierung: Wohlfundiertheit von $\succ$** Axiom 5

- **Rekursion und Basisfälle implizit in Hilfsfunktionen f_{i_j} enthalten**

Synthesestrategie ist analog zu Divide & Conquer

- **Grundstrategie**
  - Wähle $Decompose_i$ aus Wissensbank
  - Konstruiere Hilfsfunktionen f_{i_j} heuristisch und dann $Compose_i$
  - Wähle $\succ$ aus Wissensbank und verfiziere $Decompose_i$ und das SPRP
  - Instantiiere Algorithmenschema und simplifiziere Resultat
- **Umgekehrte Strategie**
  - Wähle $Compose_i$ aus Wissensbank, bestimme f_{i_j}, $Decompose_i$ und $\succ$
- **Aufspaltung des Reduktionsprinzips in 2 Axiome**
  - **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i{:}\mathbb{N}, x{:}D, \bar{y}_i{:}D_{i_1}\times..D_{i_k}, \bar{w}_i{:}R_{i_1}\times..R_{i_k}, z{:}R. \quad (1)$$
$$I[x]\wedge I_{i_1,..i_k}[\bar{y}_i]\wedge O_{i_1}[y_{i_1},w_{i_1}]\wedge..\wedge O_{i_k}[y_{i_k},w_{i_k}]\wedge O_{C_i}[\bar{w}_i,z] \;\Rightarrow\; (O_{D_i}[x,\bar{y}_i]\Leftrightarrow O[x,z])$$
$$\forall i{:}\mathbb{N}, x{:}D, \bar{y}_i{:}D_{i_1}\times..D_{i_k}, \bar{w}_i{:}R_{i_1}\times..R_{i_k}, z{:}R. \quad (2)$$
$$I[x]\wedge I_{i_1,..i_k}[\bar{y}_i]\wedge O_{D_i}[x,\bar{y}_i]\wedge O_{i_1}[y_{i_1},w_{i_1}]\wedge..\wedge O_{i_k}[y_{i_k},w_{i_k}] \;\Rightarrow\; (O_{C_i}[\bar{w}_i,z]\Leftrightarrow O[x,z])$$

  Hilfsmittel zur Konstruktion von $Decompose_i$ bzw. $Compose_i$

  - **Vollständigkeit** bzgl. Komposition

$$\forall i{:}\mathbb{N}, x{:}D, \bar{w}_i{:}R_{i_1}\times..R_{i_k}, z{:}R.$$
$$I[x]\wedge O[x,z]\wedge O_{C_i}[\bar{w}_i,z] \;\Rightarrow\; \exists \bar{y}_i{:}D_{i_1}\times..D_{i_k}.\,(I_{i_1,..i_k}[\bar{y}_i]\wedge O_{i_1}[y_{i_1},w_{i_1}]\wedge..\wedge O_{i_k}[y_{i_k},w_{i_k}])$$

Wichtige Hilfsmittel für die Synthesestrategie

- **Standard-Zerlegungen für Typen** $\mapsto$ *Decompose*$_i$
  - Endliche Listen, Intervalle: Spaltung in Teillisten/-intervalle an Stelle i
  - Endliche Mengen: Spaltung nach Größe der Elemente
  - Bäume: Spaltung in Wurzel und alle Teilbäume

  ⋮
- **Standard-Kompositionen für Typen** $\mapsto$ *Compose*
  - Endliche Folgen/Intervalle: Verkettung der Teile
  - Endliche Mengen: Vereinigung von Teilmengen(familien)
  - Bäume: Zusammensetzung von Teilbäumen mit neuer Wurzel

  ⋮
- **Standard-Wohlordnungen auf Datentypen** $\mapsto \succ$
  - Längen und Größenordnungen, Lexikographische Ordnung, etc
- **Heuristische Fixierung der Hilfsfunktionen** $\mapsto f_{i_j}$
  - Wähle $f_{i_j} := f$, wo möglich
  - Für minimale Elemente bzgl. $\succ$ berechne direkte Lösung

Synthesestrategie im Detail

- **Grundstrategie**
  1. Wähle Dekompositionsstruktur *Decompose*$_i$ aus Wissensbank
  2. Konstruiere Hilfsfunktionen f_{i_j} (Identität oder rekursiver Aufruf von f)
  3. Konstruiere Kompositionen *Compose*$_i$ mit Korrektheitsaxiom 2

     $I[x] \wedge I_{i_1,..i_k}[\bar{y}_i] \wedge O_{D_i}[x, \bar{y}_i] \wedge O_{i_1}[y_{i_1}, w_{i_1}] \wedge .. \wedge O_{i_k}[y_{i_k}, w_{i_k}] \Rightarrow (O_{C_i}[\bar{w}_i, z] \Leftrightarrow O[x, z])$

     Vereinfache $O[x, z]$ im Kontext zu einer Formel über $\bar{w}_i$ und z
  4. Wähle $\succ$ aus der Wissensbank und verifiziere *Decompose*$_i$
  5. Verifiziere Vollständigkeitsaxiom

     $I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i{:}D_{i_1} \times .. D_{i_k}. (I_{i_1,..i_k}[\bar{y}_i] \wedge O_{i_1}[y_{i_1}, w_{i_1}] \wedge .. \wedge O_{i_k}[y_{i_k}, w_{i_k}])$
  6. Instantiiere Algorithmenschema
- **Umgekehrte Strategie analog**
  – Wähle *Compose*$_i$ aus Wissensbank, bestimme f_{i_j}, *Decompose*$_i$ und $\succ$

Durchführung aufwendiger als bei Divide & Conquer

Anwendungen für Problemreduktionsgeneratoren

- **Suche alle kürzesten Verbindungen im Graphen G**
  - – GegebenVerbindungsgewichte $w(i,j)$ zwischen Knoten i und j bestimme Länge $d(i,j)$ aller kürzesten Pfade von i nach j
  - – Floyd Warshall Algorithm: Berechne $d(i,j) := d^{|V|}(i,j)$ mit $d^k(i,j)$ `= if k=0 then` $w(i,j)$ `else min{`$d^{k-1}(i,j)$`,` $d^{k-1}(i,k-1)$`+`$d^{k-1}(k-1,j)$`}`
- **Bestimme alle binären Suchbäume einer Menge S**
  - – Alle möglichen Arten, eine geordnete Menge zu durchlaufen
  - – Verwendung der umgekehrten Strategie mit Composition `mk_tree`
  - – `All-BST(`$\{s_i,..,s_j\}$`) = if i>j then` $\perp$ `else` $\bigcup_{k=i}^{j}$ `{let ltrees = All-BST(`$\{s_i,..,s_{k-1}\}$`) and rtrees = All-BST(`$\{s_{k+1},..,s_j\}$`) in mk_all_trees(ltrees,`s_k`,rtrees) }`
- **Beispiele ähnlicher Algorithmen**
  - – Bestimmung positiver Zyklen in gewichteten Graphen
  - – Maximale Segmentsumme, Längste aufsteigende Teilfolge, ...