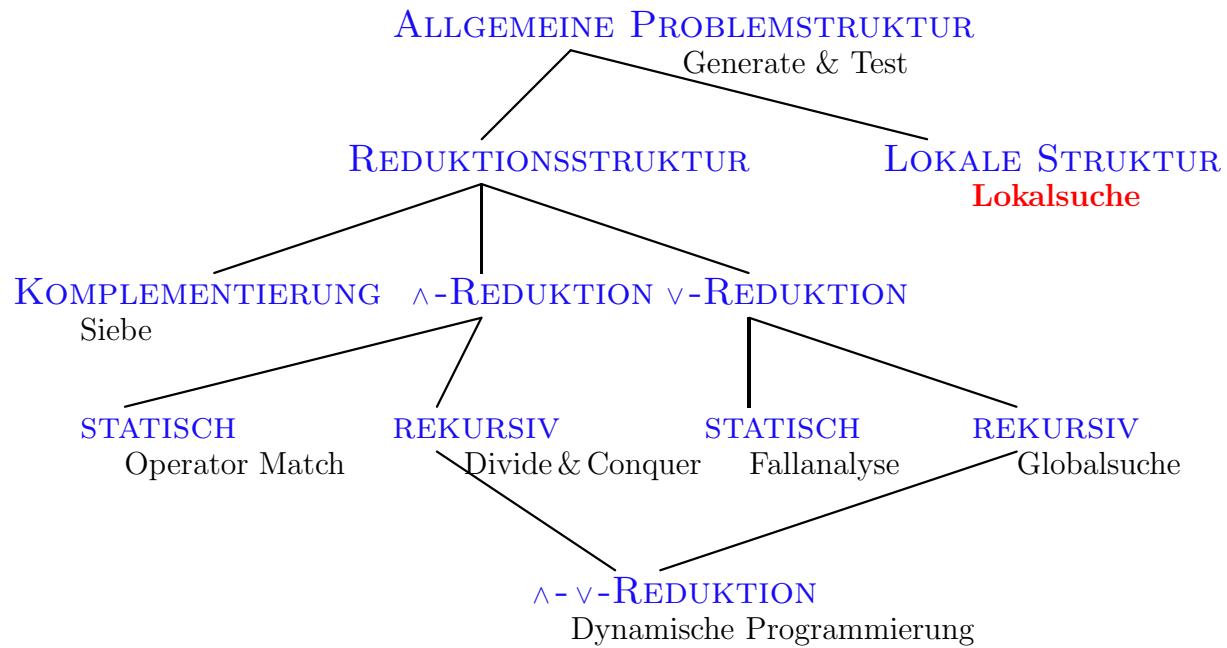
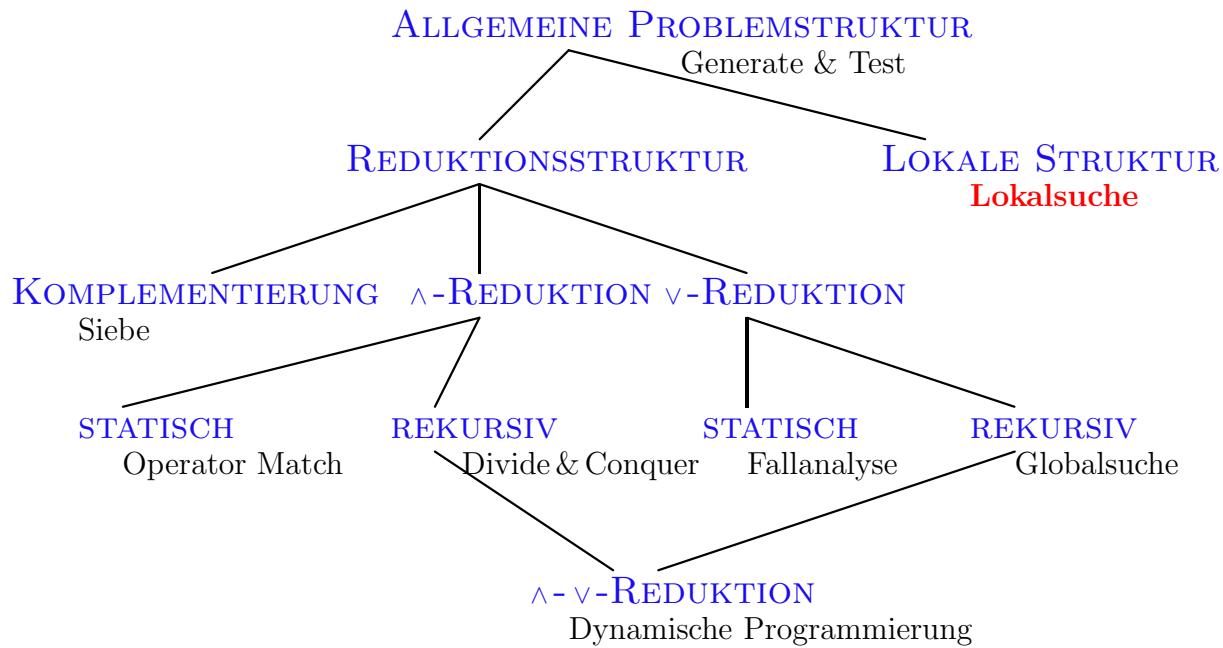


# LOKALSUCH-ALGORITHMEN



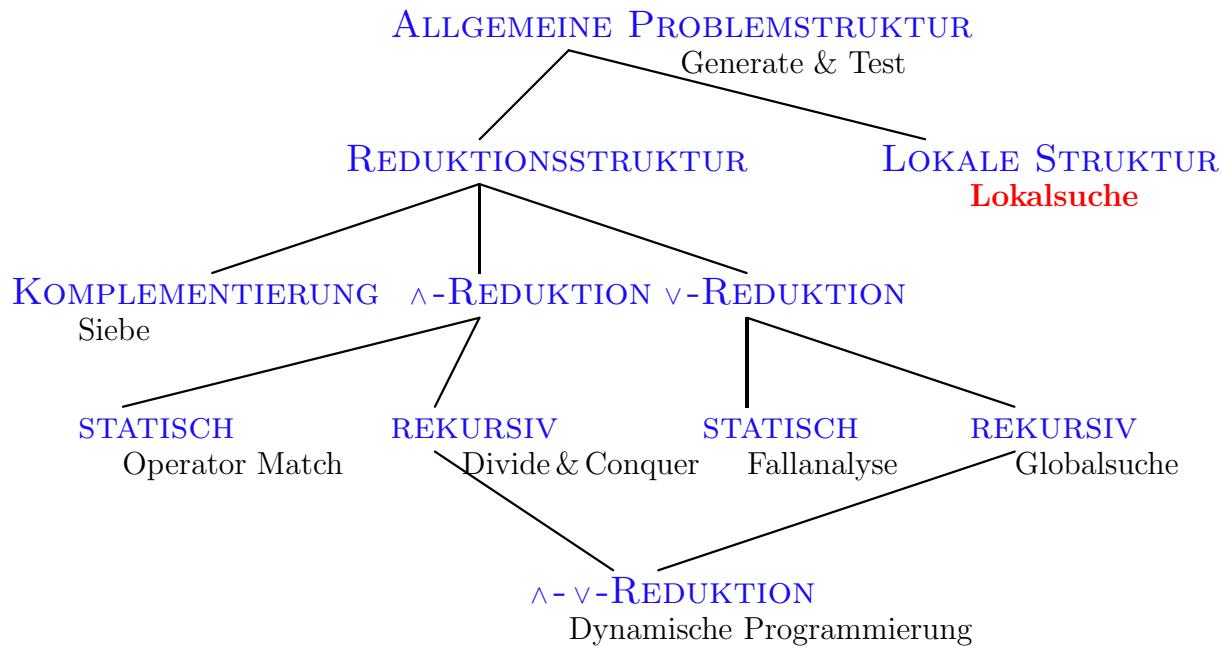
# LOKALSUCH-ALGORITHMEN



## ● Problemlösung durch kleine Veränderungen

- Bewertung der Qualität von Elementen des Bildbereichs
- Qualität einer (Teil-)Lösung wird schrittweise verbessert
- Gut für Optimierungsprobleme (Travelling Salesman, Scheduling, ...)

# LOKALSUCH-ALGORITHMEN



## ● Problemlösung durch kleine Veränderungen

- Bewertung der Qualität von Elementen des Bildbereichs
- Qualität einer (Teil-)Lösung wird schrittweise verbessert
- Gut für Optimierungsprobleme (Travelling Salesman, Scheduling, ...)

## ● Lösungsverfahren: Hillclimbing

- Beginne irgendwo im Lösungsraum
- Durchsuche lokale Nachbarschaft bis keine bessere Lösung zu finden

# LOKALSUCH-ALGORITHMEN: GENERELLE IDEE

- Optimierung als **Minimierung von Kosten**

- Optimierung als Minimierung von Kosten

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...

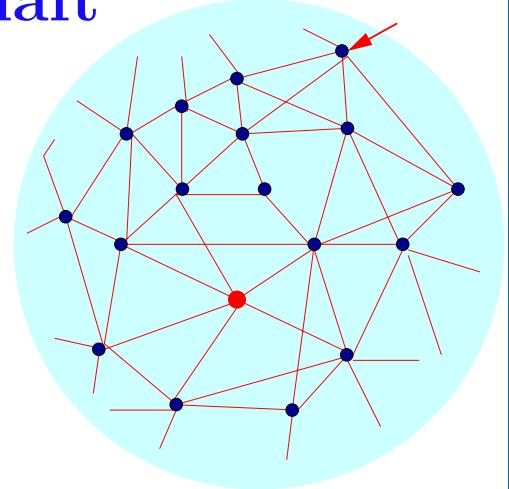
## ● Optimierung als Minimierung von Kosten

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...
- Gesucht ist Lösung mit optimaler (o.B.d.A. minimaler) Bewertung
- Exakte Optimierung oft  $\mathcal{NP}$ -vollständig

- **Optimierung als Minimierung von Kosten**

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...
- Gesucht ist Lösung mit optimaler (o.B.d.A. minimaler) Bewertung
- Exakte Optimierung oft  $\mathcal{NP}$ -vollständig

- **Lösungsverfahren durchsucht Nachbarschaft**

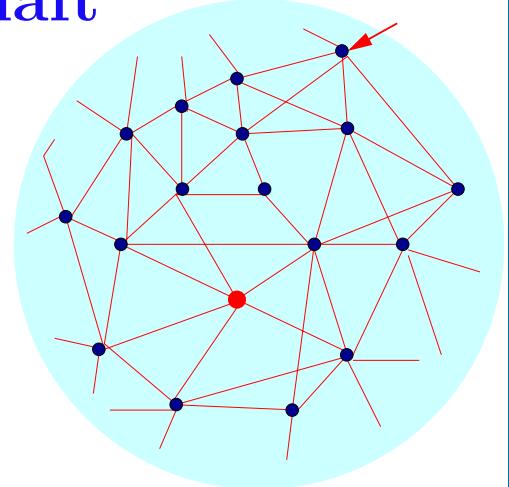


## ● Optimierung als Minimierung von Kosten

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...
- Gesucht ist Lösung mit optimaler (o.B.d.A. minimaler) Bewertung
- Exakte Optimierung oft  $\mathcal{NP}$ -vollständig

## ● Lösungsverfahren durchsucht Nachbarschaft

- Übergang auf Nachbarn, solange Verbesserungen möglich
- Verfahren endet in lokalen Optima

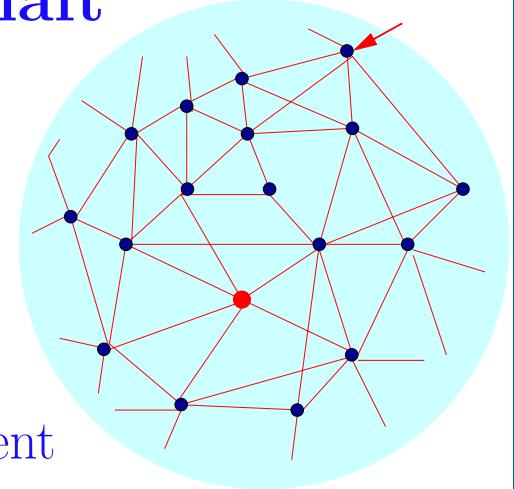


## ● Optimierung als Minimierung von Kosten

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...
- Gesucht ist Lösung mit optimaler (o.B.d.A. minimaler) Bewertung
- Exakte Optimierung oft  $\mathcal{NP}$ -vollständig

## ● Lösungsverfahren durchsucht Nachbarschaft

- Übergang auf Nachbarn, solange Verbesserungen möglich
- Verfahren endet in lokalen Optima
- Nachbarschaftstruktur entscheidet über Güte der Lösung
  - Zu fein  $\Rightarrow$  Verfahren führt nicht zu globalem Optimum
  - Zu grob  $\Rightarrow$  Suche + Test auf lokale Optimalität ineffizient

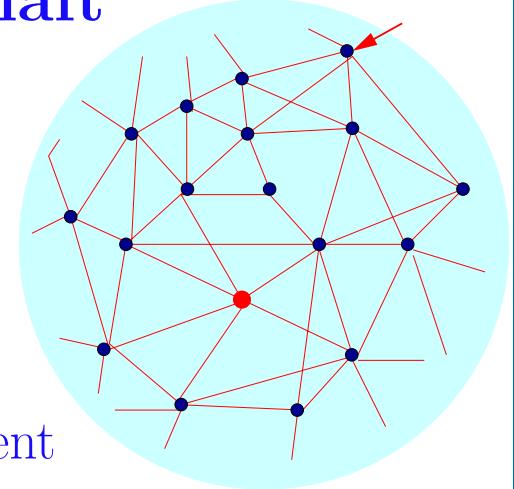


## ● Optimierung als Minimierung von Kosten

- Spezifikation des Problems besitzt viele mögliche “Lösungen”
- Lösungen werden bewertet nach Nutzen, Kosten, Korrektheitsgrad, ...
- Gesucht ist Lösung mit optimaler (o.B.d.A. minimaler) Bewertung
- Exakte Optimierung oft  $\mathcal{NP}$ -vollständig

## ● Lösungsverfahren durchsucht Nachbarschaft

- Übergang auf Nachbarn, solange Verbesserungen möglich
- Verfahren endet in lokalen Optima
- Nachbarschaftstruktur entscheidet über Güte der Lösung
  - Zu fein  $\Rightarrow$  Verfahren führt nicht zu globalem Optimum
  - Zu grob  $\Rightarrow$  Suche + Test auf lokale Optimalität ineffizient
- Bestimmung einer guten Nachbarschaftstruktur ist wichtig



# GRUNDSCHAEME VON LOKALSUCH-ALGORITHMEN

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  let  $f_{LS}(x, z)$   
= if  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$   
in  $f_{LS}(x, \text{Init}[x])$

# GRUNDSCHAEME VON LOKALSUCH-ALGORITHMEN

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  let  $f_{LS}(x, z)$   
= if  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$   
in  $f_{LS}(x, \text{Init}[x])$

- 3 zentrale Komponenten der Algorithmentheorie

# GRUNDSCHAFA VON LOKALSUCH-ALGORITHMEN

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  let  $f_{LS}(x, z)$   
= if  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$   
in  $f_{LS}(x, \text{Init}[x])$

- **3 zentrale Komponenten der Algorithmentheorie**
  - $\text{Init}: D \rightarrow R$  Initiallösung für Basisspezifikation  $(D, R, I, O)$

# GRUNDSCHAFA VON LOKALSUCH-ALGORITHMEN

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  let  $f_{LS}(x, z)$   
= if  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$   
in  $f_{LS}(x, \text{Init}[x])$

## • 3 zentrale Komponenten der Algorithmentheorie

- $\text{Init}: D \rightarrow R$  Initiallösung für Basisspezifikation ( $D, R, I, O$ )
- $c: R \rightarrow \mathcal{R}$  Kostenfunktion auf geordnetem Kostenraum ( $\mathcal{R}, \leq$ )  
Zusatzspezifikation des Optimierungsproblems

# GRUNDSCHAEME VON LOKALSUCH-ALGORITHMEN

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  let  $f_{LS}(x, z)$   
= if  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$   
in  $f_{LS}(x, \text{Init}[x])$

## • 3 zentrale Komponenten der Algorithmentheorie

- $\text{Init}: D \rightarrow R$  Initiallösung für Basisspezifikation ( $D, R, I, O$ )
- $c: R \rightarrow \mathcal{R}$  Kostenfunktion auf geordnetem Kostenraum ( $\mathcal{R}, \leq$ )  
Zusatzspezifikation des Optimierungsproblems
- $N: D \times R \rightarrow \text{Set}(R)$  Nachbarschaftsstruktur  
Suchraumbeschreibung für lokale Variationen

# KORREKTHEIT DES LOKALSUCH-SCHEMAS

```
FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$ 
  SUCH THAT  $O[x,y] \wedge \forall t:R. (O[x,t] \Rightarrow c[x,y] \leq c[x,t])$ 
 $\equiv$  let  $f_{LS}(x,z) = \text{if } \forall t \in N[x,z]. (O[x,t] \Rightarrow c[x,z] \leq c[x,t]) \text{ then } z$ 
       $\quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x,z] \wedge O[x,t] \wedge c[x,t] < c[x,z]\}))$ 
    in  $f_{LS}(x, \text{Init}[x])$ 
```

ist korrekt, wenn 4 Axiome erfüllt sind

---

# KORREKTHEIT DES LOKALSUCH-SCHEMAS

```
FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$ 
  SUCH THAT  $O[x,y] \wedge \forall t:R. (O[x,t] \Rightarrow c[x,y] \leq c[x,t])$ 
 $\equiv$  let  $f_{LS}(x,z) = \text{if } \forall t \in N[x,z]. (O[x,t] \Rightarrow c[x,z] \leq c[x,t]) \text{ then } z$ 
      else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x,z] \wedge O[x,t] \wedge c[x,t] < c[x,z]\}))$ 
  in  $f_{LS}(x, \text{Init}[x])$ 
```

ist korrekt, wenn 4 Axiome erfüllt sind

---

1.  $\text{Init}[x]$  berechnet gültige Initiallösung für  $O$

```
FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x,y]$ 
```

# KORREKTHEIT DES LOKALSUCH-SCHEMAS

```
FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$ 
  SUCH THAT  $O[x,y] \wedge \forall t:R. (O[x,t] \Rightarrow c[x,y] \leq c[x,t])$ 
 $\equiv$  let  $f_{LS}(x,z) = \text{if } \forall t \in N[x,z]. (O[x,t] \Rightarrow c[x,z] \leq c[x,t]) \text{ then } z$ 
      else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x,z] \wedge O[x,t] \wedge c[x,t] < c[x,z]\}))$ 
  in  $f_{LS}(x, \text{Init}[x])$ 
```

ist korrekt, wenn 4 Axiome erfüllt sind

---

1.  $\text{Init}[x]$  berechnet gültige Initiallösung für  $O$

```
FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x,y]$ 
```

2. Nachbarschaftsstruktur  $N$  ist reflexiv

$$\forall x:D. \forall y:R. I[x] \wedge O[x,y] \Rightarrow y \in N[x,y]$$

# KORREKTHEIT DES LOKALSUCH-SCHEMAS

FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x,y] \wedge \forall t:R. (O[x,t] \Rightarrow c[x,y] \leq c[x,t])$   
 $\equiv$  let  $f_{LS}(x,z) = \text{if } \forall t \in N[x,z]. (O[x,t] \Rightarrow c[x,z] \leq c[x,t]) \text{ then } z$   
else  $f_{LS}(x, \text{arb}(\{t \mid t \in N[x,z] \wedge O[x,t] \wedge c[x,t] < c[x,z]\}))$   
in  $f_{LS}(x, Init[x])$

ist korrekt, wenn 4 Axiome erfüllt sind

---

1.  $Init[x]$  berechnet gültige Initiallösung für  $O$

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x,y]$

2. Nachbarschaftsstruktur  $N$  ist reflexiv

$\forall x:D. \forall y:R. I[x] \wedge O[x,y] \Rightarrow y \in N[x,y]$

3. Lokale Optima sind exakt → Optimale Algorithmen

$\forall x:D. \forall y:R. I[x] \wedge O[x,y] \Rightarrow (\forall t \in N[x,y]. O[x,t] \Rightarrow c[x,y] \leq c[x,t])$   
 $\Rightarrow \forall z:R. (O[x,z] \Rightarrow c[x,y] \leq c[x,z])$

# KORREKTHEIT DES LOKALSUCH-SCHEMAS

FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$   
 SUCH THAT  $O[x,y] \wedge \forall t:R. (O[x,t] \Rightarrow c[x,y] \leq c[x,t])$   
 $\equiv$  let  $f_{LS}(x,z) = \text{if } \forall t \in N[x,z]. (O[x,t] \Rightarrow c[x,z] \leq c[x,t]) \text{ then } z$   
 $\quad \quad \quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x,z] \wedge O[x,t] \wedge c[x,t] < c[x,z]\}))$   
 in  $f_{LS}(x, Init[x])$

ist korrekt, wenn 4 Axiome erfüllt sind

---

1.  $Init[x]$  berechnet gültige Initiallösung für  $O$

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x,y]$

2. Nachbarschaftsstruktur  $N$  ist reflexiv

$\forall x:D. \forall y:R. I[x] \wedge O[x,y] \Rightarrow y \in N[x,y]$

3. Lokale Optima sind exakt → Optimale Algorithmen

$\forall x:D. \forall y:R. I[x] \wedge O[x,y] \Rightarrow (\forall t \in N[x,y]. O[x,t] \Rightarrow c[x,y] \leq c[x,t])$   
 $\Rightarrow \forall z:R. (O[x,z] \Rightarrow c[x,y] \leq c[x,z])$

4. Alle gültigen Lösungen sind endlich erreichbar

$\forall x:D. \forall y, z:R. I[x] \wedge O[x,y] \wedge O[x,z] \Rightarrow \exists k:\mathbb{N}. z \in N_O^k[x,y]$

---

$$N_O^0[x,y] = \{y\} \quad N_O^{k+1}[x,y] = \bigcup \{N^k[x,t] \mid t \in N[x,y] \wedge O(x,t)\}$$

## LOKALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- Abspalten und Spezifikation der Hilfsfunktion  $f_{ls}$

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv f_{LS}(x, Init[x])$

FUNCTION  $f_{ls}(x, z:D \times R) : R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv \text{if } \forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t]) \text{ then } z$   
 $\quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$

# LOKALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- Abspalten und Spezifikation der Hilfsfunktion  $f_{ls}$

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
 SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv f_{LS}(x, Init[x])$

FUNCTION  $f_{ls}(x, z:D \times R) : R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$   
 SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv \text{if } \forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t]) \text{ then } z$   
 $\quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$

- Korrektheit von  $f_{opt}$  folgt aus der von  $f_{ls}$  mit Axiomen 1 & 3
  - Für den Startwert  $z = Init[x]$  gilt  $O[x, z]$
  - Für  $y = f_{LS}(x, z)$  gilt  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$
  - Mit Axiom 3 folgt  $\forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

$$\begin{aligned} \forall x:D. I[x] &\Rightarrow O[x, Init[x]] \\ \forall x:D. \forall z:R. I[x] \wedge O[x, z] &\Rightarrow O[x, f_{ls}[x, z]] \\ &\quad \wedge \forall t \in N[x, f_{ls}[x, z]]. (O[x, t] \Rightarrow c[x, f_{ls}[x, z]] \leq c[x, t]) \\ \forall x:D. \forall y:R. I[x] \wedge O[x, y] &\Rightarrow (\forall t \in N[x, y]. O[x, t] \Rightarrow c[x, y] \leq c[x, t]) \\ &\Rightarrow \forall z:R. (O[x, z] \Rightarrow c[x, y] \leq c[x, z]) \end{aligned}$$

# LOKALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- Abspalten und Spezifikation der Hilfsfunktion  $f_{ls}$

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv f_{LS}(x, Init[x])$

FUNCTION  $f_{ls}(x, z:D \times R) : R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv \text{if } \forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t]) \text{ then } z$   
 $\quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$

- Korrektheit von  $f_{opt}$  folgt aus der von  $f_{ls}$  mit Axiomen 1 & 3

- Für den Startwert  $z = Init[x]$  gilt  $O[x, z]$
- Für  $y = f_{LS}(x, z)$  gilt  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$
- Mit Axiom 3 folgt  $\forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

- Partielle Korrektheit von  $f_{ls}$  folgt aus Programmkörper

- Hält  $f_{ls}$  mit Ausgabe  $z$ , so gilt  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$

# LOKALSUCH-SCHEMA: KORREKTHEITSBEWEIS

- Abspalten und Spezifikation der Hilfsfunktion  $f_{ls}$

FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
 SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv f_{LS}(x, Init[x])$

FUNCTION  $f_{ls}(x, z:D \times R) : R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$   
 SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv \text{if } \forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t]) \text{ then } z$   
 $\quad \text{else } f_{LS}(x, \text{arb}(\{t \mid t \in N[x, z] \wedge O[x, t] \wedge c[x, t] < c[x, z]\}))$

- Korrektheit von  $f_{opt}$  folgt aus der von  $f_{ls}$  mit Axiomen 1 & 3

- Für den Startwert  $z = Init[x]$  gilt  $O[x, z]$
- Für  $y = f_{LS}(x, z)$  gilt  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$
- Mit Axiom 3 folgt  $\forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

- Partielle Korrektheit von  $f_{ls}$  folgt aus Programmkörper

- Hält  $f_{ls}$  mit Ausgabe  $z$ , so gilt  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$

- Terminierung von  $f_{ls}$  folgt aus Ordnung  $(\mathcal{R}, \leq)$  und Axiom 4

$$\begin{aligned} \forall x:D. \exists z:R. O[x, z] \wedge c[x, z] &= \min\{c[x, t] \mid t \in R \wedge O[x, t]\} \\ \forall x:D. \forall y, z:R. I[x] \wedge O[x, y] \wedge O[x, z] &\Rightarrow \exists k:\mathbb{N}. z \in N_O^k[x, y] \end{aligned}$$

# SORTIEREN MIT LOKALSUCHALGORITHMEN

## ● Formuliere Sortierung als Ordnungsoptimierung

- Einfache Basisspezifikation  $O[L, S] = \text{rearranges}(L, S)$
- Kostenfunktion  $c[L, S] = \#_>(S)$ : Anzahl der Fehlstellungen  $S_i > S_{i+1}$
- Es gilt  $\#_>(S) \geq 0$  und  $\#_>(S) = 0 \Rightarrow \text{ordered}(S)$
- Spezifikation `FUNCTION sort(L:Seq(Z)):Seq(Z) WHERE true RETURNS S  
SUCH THAT rearranges(L,S)  $\wedge$   
 $\forall S':\text{Seq}(Z). \text{rearranges}(L,S') \Rightarrow \#_>(S) \leq \#_>(S')$`

## ● Formuliere Sortierung als Ordnungsoptimierung

- Einfache Basisspezifikation  $O[L, S] = \text{rearranges}(L, S)$
- Kostenfunktion  $c[L, S] = \#_>(S)$ : Anzahl der Fehlstellungen  $S_i > S_{i+1}$
- Es gilt  $\#_>(S) \geq 0$  und  $\#_>(S) = 0 \Rightarrow \text{ordered}(S)$
- Spezifikation `FUNCTION sort(L:Seq(Z)) : Seq(Z) WHERE true RETURNS S  
SUCH THAT rearranges(L, S) ∧  
∀S' : Seq(Z). rearranges(L, S') ⇒ #>(S) ≤ #>(S')`

## ● Lokalsuchalgorithmus

- Initiallösung  $Init[L] = L$  ↔ Axiom 1
- Nachbarschaft  $N[S, S'] = \text{permute}_{i,j}(S, S')$ : Vertauschen von  $S_i$  und  $S_j$ 
  - Vertauschen ist reflexiv, Lokale Minima sind exakt ↔ Axiom 2,3
  - Alle Umordnungen erreichbar durch iteratives Vertauschen ↔ Axiom 4
- Ergibt Sortieren durch beliebiges Austauschen von Elementen
  - Ineffizient, da zu viele Nachbarn zu prüfen (vermutlich  $\mathcal{O}(n^3)$ )

# SORTIEREN MIT LOKALSUCHALGORITHMEN

## ● Formuliere Sortierung als Ordnungsoptimierung

- Einfache Basisspezifikation  $O[L, S] = \text{rearranges}(L, S)$
- Kostenfunktion  $c[L, S] = \#_>(S)$ : Anzahl der Fehlstellungen  $S_i > S_{i+1}$
- Es gilt  $\#_>(S) \geq 0$  und  $\#_>(S) = 0 \Rightarrow \text{ordered}(S)$
- Spezifikation `FUNCTION sort(L:Seq(Z)) : Seq(Z) WHERE true RETURNS S SUCH THAT rearranges(L, S) ∧ ∀S' : Seq(Z). rearranges(L, S') ⇒ #>(S) ≤ #>(S')`

## ● Lokalsuchalgorithmus

- Initiallösung  $\text{Init}[L] = L$   $\mapsto$  Axiom 1
- Nachbarschaft  $N[S, S'] = \text{perm}_{i,j}(S, S')$ : Vertauschen von  $S_i$  und  $S_j$ 
  - Vertauschen ist reflexiv, Lokale Minima sind exakt  $\mapsto$  Axiom 2,3
  - Alle Umordnungen erreichbar durch iteratives Vertauschen  $\mapsto$  Axiom 4
- Ergibt Sortieren durch beliebiges Austauschen von Elementen
  - Ineffizient, da zu viele Nachbarn zu prüfen (vermutlich  $\mathcal{O}(n^3)$ )

## ● Lokalsuchalgorithmus mit kleinerer Nachbarschaft

- Restriktion auf benachbarte Komponenten
- Nachbarschaft  $N[S, S'] = \text{perm}_i(S, S') = \text{perm}_{i,i+1}(S, S')$
- Ergibt Bubblesort (nach algorithmischer Optimierung)

# SYNTHESE VON LOKALSUCH-ALGORITHMEN

Lokalsuche  $\hat{=}$  Nachbarschaft + Suchfilter

# SYNTHESE VON LOKALSUCH-ALGORITHMEN

**Lokalsuche  $\hat{=}$  Nachbarschaft + Suchfilter**

- Bestimme effektive Nachbarschaftsstruktur

- Beschreibe Nachbarschaft als Perturbation (Verwirbelung)
- Inkrementelle Veränderung von Werten aus  $R$ 
  - Numerik:  $\delta$ -Vektoren, Kombinatorik: Austausch von Komponenten

## Lokalsuche $\hat{=}$ Nachbarschaft + Suchfilter

- Bestimme effektive Nachbarschaftsstruktur

- Beschreibe Nachbarschaft als Perturbation (Verwirbelung)
- Inkrementelle Veränderung von Werten aus  $R$ 
  - Numerik:  $\delta$ -Vektoren, Kombinatorik: Austausch von Komponenten
- Formalisiert als  $N[x, y] \hat{=} \{ \text{Action}[i, j, x, y] \mid i, j \in \pi[x, y] \}$ 
  - Änderungsaktion  $\text{Action}[i, j, x, y]$  modifiziert Lösungspunkt  $(x, y) \in D \times R$
  - Parameter  $i, j \in \pi[x, y]$  sind minimale Bestandteile von  $(x, y)$

## Lokalsuche $\hat{=}$ Nachbarschaft + Suchfilter

### • Bestimme effektive Nachbarschaftsstruktur

- Beschreibe Nachbarschaft als Perturbation (Verwirbelung)
- Inkrementelle Veränderung von Werten aus  $R$ 
  - Numerik:  $\delta$ -Vektoren, Kombinatorik: Austausch von Komponenten
- Formalisiert als  $N[x, y] \hat{=} \{ \text{Action}[i, j, x, y] \mid i, j \in \pi[x, y] \}$ 
  - Änderungsaktion  $\text{Action}[i, j, x, y]$  modifiziert Lösungspunkt  $(x, y) \in D \times R$
  - Parameter  $i, j \in \pi[x, y]$  sind minimale Bestandteile von  $(x, y)$

### • Bestimme effiziente Suchfilter

- Optimiere Nachbarschaftsstruktur durch frühzeitiges Abschneiden
  - Feasibility Constraint für  $O[x, y]$
  - Optimality Constraint für  $\forall t \in N[x, z]. (O[x, t] \Rightarrow c[x, z] \leq c[x, t])$

## Spezialisierter vorformuliertes Programmierwissen

- **Lokalsuchtheorie:** allgemeine Suchstruktur für  $R$

- Vorgefertigte Nachbarschaftsstruktur, die Axiome 2–4 erfüllt
- Formalisiert als Objekt  $\mathcal{L} = (D, R, I, O, \pi, Action)$
- Wissensbank speichert Lokalsuchtheorien für Grunddatentypen

## Spezialisiere vorformuliertes Programmierwissen

- **Lokalsuchtheorie:** allgemeine Suchstruktur für  $R$

- Vorgefertigte Nachbarschaftsstruktur, die Axiome 2–4 erfüllt
- Formalisiert als Objekt  $\mathcal{L} = (D, R, I, O, \pi, Action)$
- Wissensbank speichert Lokalsuchtheorien für Grunddatentypen

- **Spezialisierungsmechanismen**

- Synthetisiere Initiallösung  $Init[x]$  für Spezifikation  $spec = (D, R, I, O)$
- Wähle  $\mathcal{L}$  für Bildbereich  $R$ , so daß  $spec \ll spec_{\mathcal{L}}$  beweisbar
- Extrahiere Substitution  $\theta: D \rightarrow D_{\mathcal{L}}$  und spezialisiere  $\mathcal{L}$  mit  $\theta$ 
  - Ergibt Nachbarschaftsstruktur  $N$  für Problemstellung
- Generiere Filter zur Beschränkung auf optimale Lösungen

## Spezialisiere vorformuliertes Programmierwissen

- **Lokalsuchtheorie:** allgemeine Suchstruktur für  $R$

- Vorgefertigte Nachbarschaftsstruktur, die Axiome 2–4 erfüllt
- Formalisiert als Objekt  $\mathcal{L} = (D, R, I, O, \pi, Action)$
- Wissensbank speichert Lokalsuchtheorien für Grunddatentypen

- **Spezialisierungsmechanismen**

- Synthetisiere Initiallösung  $Init[x]$  für Spezifikation  $spec = (D, R, I, O)$
- Wähle  $\mathcal{L}$  für Bildbereich  $R$ , so daß  $spec \ll spec_{\mathcal{L}}$  beweisbar
- Extrahiere Substitution  $\theta: D \rightarrow D_{\mathcal{L}}$  und spezialisiere  $\mathcal{L}$  mit  $\theta$ 
  - Ergibt Nachbarschaftsstruktur  $N$  für Problemstellung
- Generiere Filter zur Beschränkung auf optimale Lösungen

- **Eventuell Verzicht auf Exaktheit**

- Liefert effizienteren, aber suboptimalen Algorithmus

# STANDARD-LOKALSUCHTHEORIEN

## ● Umordnung von Folgen

- Suche  $\hat{=}$  Permutation einzelner Elemente einer Folge
- Änderungsparameter: Indizes der Eingabeliste  $L$
- Perturbation: Vertauschung zweier Elemente einer Ausgabeliste  $S$

# STANDARD-LOKALSUCHTHEORIEN

## ● Umordnung von Folgen

- Suche  $\hat{=}$  Permutation einzelner Elemente einer Folge
- Änderungsparameter: Indizes der Eingabeliste  $L$
- Perturbation: Vertauschung zweier Elemente einer Ausgabeliste  $S$

$$\begin{aligned} \text{LS\_seq\_re}(\alpha) \equiv & D \mapsto \text{Seq}(\alpha) \\ & R \mapsto \text{Seq}(\alpha) \\ & I \mapsto \lambda L. \text{true} \\ & O \mapsto \lambda L, S. \text{rearranges}(L, S) \\ & \pi \mapsto \lambda L, S. (\text{domain}(S), \text{domain}(S)) \\ & \text{Action} \mapsto \lambda i, j, L, S. [S_{(i \leftrightarrow j)(k)} \mid k \in \text{domain}(S)] \\ & (i \leftrightarrow j)(k) \hat{=} \text{if } k=i \text{ then } j \text{ else if } k=j \text{ then } i \text{ else } k \end{aligned}$$

# STANDARD-LOKALSUCHTHEORIEN

## ● Umordnung von Folgen

- Suche  $\hat{=}$  Permutation einzelner Elemente einer Folge
- Änderungsparameter: Indizes der Eingabeliste  $L$
- Perturbation: Vertauschung zweier Elemente einer Ausgabeliste  $S$

$$\begin{aligned} \text{LS\_seq\_re}(\alpha) &\equiv \begin{array}{lll} D & \mapsto & \text{Seq}(\alpha) \\ R & \mapsto & \text{Seq}(\alpha) \\ I & \mapsto & \lambda L. \text{true} \\ O & \mapsto & \lambda L, S. \text{rearranges}(L, S) \\ \pi & \mapsto & \lambda L, S. (\text{domain}(S), \text{domain}(S)) \\ \text{Action} & \mapsto & \lambda i, j, L, S. [S_{(i \leftrightarrow j)(k)} \mid k \in \text{domain}(S)] \end{array} \\ &\quad (i \leftrightarrow j)(k) \hat{=} \text{if } k=i \text{ then } j \text{ else if } k=j \text{ then } i \text{ else } k \end{aligned}$$

## ● Teilmengen fester Größe

- Suche  $\hat{=}$  Austausch einzelner Elemente einer Menge
- Änderungsparameter: Elemente der Ein- und Ausgabemenge
- Perturbation: Austausch zweier Elemente in Ausgabemenge

# STANDARD-LOKALSUCHTHEORIEN

## ● Umordnung von Folgen

- Suche  $\hat{=}$  Permutation einzelner Elemente einer Folge
- Änderungsparameter: Indizes der Eingabeliste  $L$
- Perturbation: Vertauschung zweier Elemente einer Ausgabeliste  $S$

$$\begin{aligned} \text{LS\_seq\_re}(\alpha) \equiv & D \mapsto \text{Seq}(\alpha) \\ & R \mapsto \text{Seq}(\alpha) \\ & I \mapsto \lambda L. \text{true} \\ & O \mapsto \lambda L, S. \text{rearranges}(L, S) \\ & \pi \mapsto \lambda L, S. (\text{domain}(S), \text{domain}(S)) \\ & \text{Action} \mapsto \lambda i, j, L, S. [S_{(i \leftrightarrow j)(k)} \mid k \in \text{domain}(S)] \\ & (i \leftrightarrow j)(k) \hat{=} \text{if } k=i \text{ then } j \text{ else if } k=j \text{ then } i \text{ else } k \end{aligned}$$

## ● Teilmengen fester Größe

- Suche  $\hat{=}$  Austausch einzelner Elemente einer Menge
- Änderungsparameter: Elemente der Ein- und Ausgabemenge
- Perturbation: Austausch zweier Elemente in Ausgabemenge

$$\begin{aligned} \text{LS\_subsets}(\alpha) \equiv & D \mapsto \text{Set}(\alpha) \times \mathbb{N} \\ & R \mapsto \text{Set}(\alpha) \\ & I \mapsto \lambda S, m. m \leq |S| \\ & O \mapsto \lambda S, m, S'. S' \subseteq S \wedge |S'| = m \\ & \pi \mapsto \lambda S, m, S'. (S \setminus S', S') \\ & \text{Action} \mapsto \lambda i, j, S, m, S'. S' = (S \cup \{i\}) - j \end{aligned}$$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

Start: FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

Start: FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

1. Wähle Lokalsuchtheorie  $\mathcal{L}$  mit Ausgabetyp  $R$  aus Wissensbank

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

**Start:** FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

1. Wähle Lokalsuchtheorie  $\mathcal{L}$  mit Ausgabetyp  $R$  aus Wissensbank
2. Beweise  $(D, R, I, O) \ll spec_{\mathcal{L}}$

– Extrahiere Substitution  $\theta$  und setze  $\mathcal{L}_{\theta} = (D, R, I, O, \pi_{\theta}, Action_{\theta})$

$$R \subseteq R' \wedge \forall x:D. I[x] \Rightarrow \exists x':D'. (I'[x'] \wedge \forall y:R. O[x, y] \Rightarrow O'[x', y'])$$

$$\pi_{\theta}[x, y] = \pi[\theta(x), y]$$

$$Action_{\theta}[i, j, x, y] = Action[i, j, \theta(x), y]$$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

**Start:** FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

1. Wähle Lokalsuchtheorie  $\mathcal{L}$  mit Ausgabetyp  $R$  aus Wissensbank
2. Beweise  $(D, R, I, O) \ll spec_{\mathcal{L}}$ 
  - Extrahiere Substitution  $\theta$  und setze  $\mathcal{L}_{\theta} = (D, R, I, O, \pi_{\theta}, Action_{\theta})$
3. Generiere Lösungs-Filter  $FC$  für  $\mathcal{L}_{\theta}$  (Feasibility Constraint)
  - Filter eliminiert Punkte, die keine gültigen Lösungen bzgl.  $O$  sind
    - $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \Rightarrow FC[i, j, x, y]$
  - Vorwärtsinferenz: Vereinfachung der linken Seite ergibt  $FC$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

**Start:** FUNCTION  $f_{opt}(x:D) : R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

1. Wähle Lokalsuchtheorie  $\mathcal{L}$  mit Ausgabetyp  $R$  aus Wissensbank
2. Beweise  $(D, R, I, O) \ll spec_{\mathcal{L}}$ 
  - Extrahiere Substitution  $\theta$  und setze  $\mathcal{L}_{\theta} = (D, R, I, O, \pi_{\theta}, Action_{\theta})$
3. Generiere Lösungs-Filter  $FC$  für  $\mathcal{L}_{\theta}$  *(Feasibility Constraint)*
  - Filter eliminiert Punkte, die keine gültigen Lösungen bzgl.  $O$  sind
    - $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \Rightarrow FC[i, j, x, y]$
  - Vorwärtsinferenz: Vereinfachung der linken Seite ergibt  $FC$
4. Generiere Optimalitäts-Filter  $OC$  für  $\mathcal{L}_{\theta}$  *(Optimality Constraint)*
  - Filter eliminiert kostenungünstigere Punkte
    - $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \wedge c[x, y] \leq c[x, Action[i, j, \theta(x), y]] \Rightarrow OC[i, j, x, y]$
  - Lokale Optima müssen Bedingung  $\forall i, j \in \pi(x, y). OC(i, j, x, y)$  erfüllen

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

**Start:** FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

1. Wähle Lokalsuchtheorie  $\mathcal{L}$  mit Ausgabetyp  $R$  aus Wissensbank
2. Beweise  $(D, R, I, O) \ll spec_{\mathcal{L}}$ 
  - Extrahiere Substitution  $\theta$  und setze  $\mathcal{L}_{\theta} = (D, R, I, O, \pi_{\theta}, Action_{\theta})$
3. Generiere Lösungs-Filter  $FC$  für  $\mathcal{L}_{\theta}$  (*Feasibility Constraint*)
  - Filter eliminiert Punkte, die keine gültigen Lösungen bzgl.  $O$  sind
    - $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \Rightarrow FC[i, j, x, y]$
  - Vorwärtsinferenz: Vereinfachung der linken Seite ergibt  $FC$
4. Generiere Optimalitäts-Filter  $OC$  für  $\mathcal{L}_{\theta}$  (*Optimality Constraint*)
  - Filter eliminiert kostenungünstigere Punkte
    - $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \wedge c[x, y] \leq c[x, Action[i, j, \theta(x), y]] \Rightarrow OC[i, j, x, y]$
  - Lokale Optima müssen Bedingung  $\forall i, j \in \pi(x, y). OC(i, j, x, y)$  erfüllen
5. Synthetisiere Initiallösung  $Init$  für Spezifikation

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x, y]$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

## 6. Instantiiere Schema für suboptimale Lokalsuch Algorithmen

FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv f_{LS}(x, Init[x])$

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x, y] \equiv Init[x]$

FUNCTION  $f_{ls}(x, z:D \times R):R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$   
SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$   
 $\equiv$  if  $\forall i, j \in \pi[\theta(x), z]. FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$   
 $\Rightarrow OC[i, j, x, z] \wedge c[x, z] \leq c[x, Action[i, j, \theta(x), y]]$  then  $z$   
else  $f_{LS}(x, \text{arb}(\{Action[i, j, \theta(x), y] \mid i, j \in \pi[\theta(x), z].$   
 $\wedge FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$   
 $\wedge \neg OC[i, j, x, z] \vee c[x, z] > c[x, Action[i, j, \theta(x), y]]\}))$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

## 6. Instantiiere Schema für suboptimale Lokalsuch Algorithmen

```
FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$ 
    SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$ 
 $\equiv f_{LS}(x, Init[x])$ 

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x, y] \equiv Init[x]$ 

FUNCTION  $f_{ls}(x, z:D \times R):R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$ 
    SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$ 
 $\equiv$  if  $\forall i, j \in \pi[\theta(x), z]. FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$ 
         $\Rightarrow OC[i, j, x, z] \wedge c[x, z] \leq c[x, Action[i, j, \theta(x), y]]$  then  $z$ 
    else  $f_{LS}(x, \text{arb}(\{Action[i, j, \theta(x), y] \mid i, j \in \pi[\theta(x), z].$ 
         $\wedge FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$ 
         $\wedge \neg OC[i, j, x, z] \vee c[x, z] > c[x, Action[i, j, \theta(x), y]]\}))$ 
```

Lösungs- und Optimalitätstests notwendig für Korrektheit

Algorithmus testet nur Parameter, welche die Filter  $FC$  und  $OC$  passieren

Effiziente Abarbeitung nutzt **andthen/orelse** Semantik von  $\wedge$  und  $\vee$

# SYNTHESESTRATEGIE FÜR LOKALSUCH-ALGORITHMEN

## 6. Instantiiere Schema für suboptimale Lokalsuch Algorithmen

FUNCTION  $f_{opt}(x:D):R$  WHERE  $I[x]$  RETURNS  $y$

SUCH THAT  $O[x, y] \wedge \forall t:R. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

$$\equiv f_{LS}(x, \textcolor{red}{Init}[x])$$

FUNCTION  $f(x:D):R$  WHERE  $I[x]$  RETURNS  $y$  SUCH THAT  $O[x,y] \equiv Init[x]$

FUNCTION  $f_{ls}(x, z : D \times R) : R$  WHERE  $I[x] \wedge O[x, z]$  RETURNS  $y$

SUCH THAT  $O[x, y] \wedge \forall t \in N[x, y]. (O[x, t] \Rightarrow c[x, y] \leq c[x, t])$

$\equiv$  if  $\forall i, j \in \pi[\theta(x), z] . \text{ } FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$

$\Rightarrow OC[i, j, x, z] \wedge c[x, z] \leq c[x, Action[i, j, \theta(x), y]]$  then  $z$

else  $f_{LS}(x, \text{ arb}(\{\text{Action}[i, j, \theta(x), y] \mid i, j \in \pi[\theta(x), z])$

$\wedge \text{ } FC[i, j, x, z] \wedge O[x, Action[i, j, \theta(x), y]]$

$\wedge \neg OC[i, j, x, z] \vee c[x, z] > c[x, Action[i, j, \theta(x), y]] \}$ )))

Lösungs- und Optimalitätstests notwendig für Korrektheit

Algorithmus testet nur Parameter, welche die Filter  $FC$  und  $OC$  passieren

Effiziente Abarbeitung nutzt **andthen**/**orelse** Semantik von  $\wedge$  und  $\vee$

## 7. Generiere Bedingungen für Exaktheit (*Global Optimality Constraint*)

- Zusätzlicher (optionaler) Filter eliminiert suboptimale Lösungen

- $I[x] \wedge O[x, y] \wedge O[x, Action[i, j, \theta(x), y]] \wedge c[x, y] \leq c[x, Action[i, j, \theta(x), y]]$   
 $\wedge \forall t:R. O[x, t] \Rightarrow c[x, y] \leq c[x, t] \Rightarrow GOC[x, y]$

- **Minimal Spanning Tree**

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar

### ● Minimal Spanning Tree

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar
- Initialwert: Erzeuge spannenden Baum
- Perturbation: Ergänze neue Kante, entferne eine andere

### ● Minimal Spanning Tree

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar
- Initialwert: Erzeuge spannenden Baum
- Perturbation: Ergänze neue Kante, entferne eine andere
- Feasibility Constraint: Entfernte Kante muß redundant sein
- Optimality Constraint: Hinzugefügte Kante ist teurer als bisheriger Weg

## ● Minimal Spanning Tree

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar
- Initialwert: Erzeuge spannenden Baum
- Perturbation: Ergänze neue Kante, entferne eine andere
- Feasibility Constraint: Entfernte Kante muß redundant sein
- Optimality Constraint: Hinzugefügte Kante ist teurer als bisheriger Weg

## ● Lineare Programmierung

- Minimiere lineare Funktion  $f(x_1, \dots, x_n) = \sum c_i x_i$  unter Restriktionen  $A_j[x_1, \dots, x_n]$   
Standard Darstellung: Minimiere  $c \star x$  unter  $A \star x = b \wedge \forall i \leq n. x_i \geq 0$

## ● Minimal Spanning Tree

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar
- Initialwert: Erzeuge spannenden Baum
- Perturbation: Ergänze neue Kante, entferne eine andere
- Feasibility Constraint: Entfernte Kante muß redundant sein
- Optimality Constraint: Hinzugefügte Kante ist teurer als bisheriger Weg

## ● Lineare Programmierung

- Minimiere lineare Funktion  $f(x_1, \dots, x_n) = \sum c_i x_i$  unter Restriktionen  $A_j[x_1, \dots, x_n]$   
Standard Darstellung: Minimiere  $c \star x$  unter  $A \star x = b \wedge \forall i \leq n. x_i \geq 0$
- Initiallösung: Setze  $x_{m+1}, \dots, x_n := 0$  und löse  $A_{1..m} \star x_{1..m}$  mit  $\forall i \leq m. x_i > 0$   
durch Gauß-Verfahren
- Perturbation: Setze ein  $x_i := 0$ , wähle neues  $x_j \neq 0$

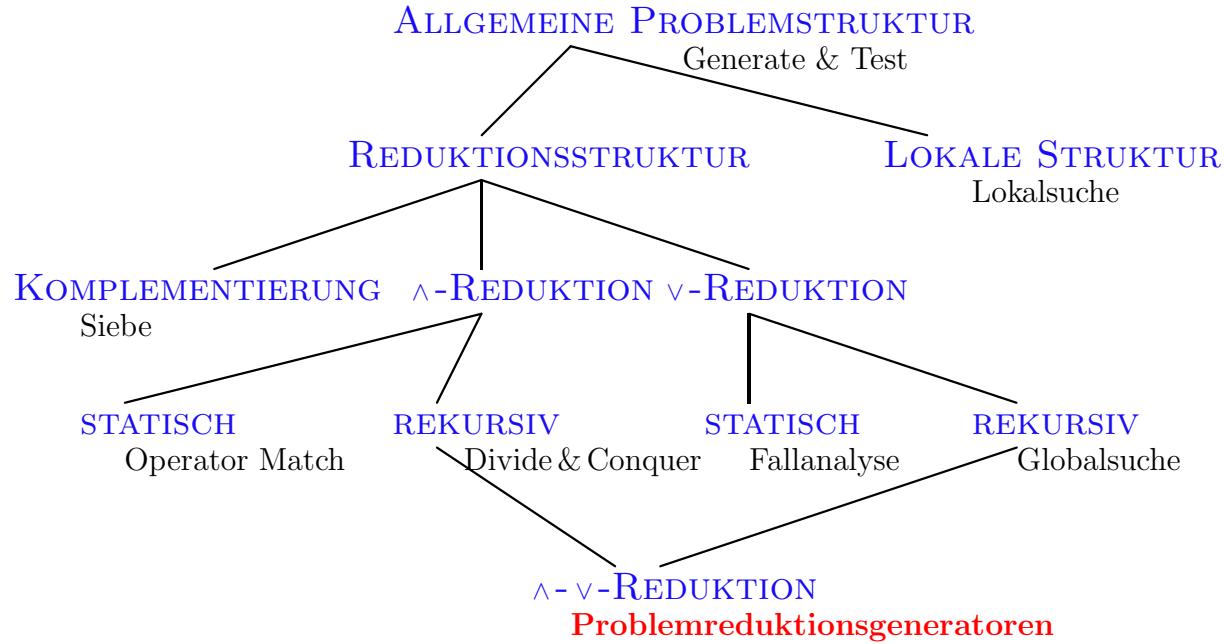
## ● Minimal Spanning Tree

- Gegeben: Graph mit gewichteten Kanten (Zugriffszeiten, Abstände,...)
- Gesucht: Baum, auf dem alle Knoten mit minimalen Kosten erreichbar
- Initialwert: Erzeuge spannenden Baum
- Perturbation: Ergänze neue Kante, entferne eine andere
- Feasibility Constraint: Entfernte Kante muß redundant sein
- Optimality Constraint: Hinzugefügte Kante ist teurer als bisheriger Weg

## ● Lineare Programmierung

- Minimiere lineare Funktion  $f(x_1, \dots, x_n) = \sum c_i x_i$  unter Restriktionen  $A_j[x_1, \dots, x_n]$   
Standard Darstellung: Minimiere  $c \star x$  unter  $A \star x = b \wedge \forall i \leq n. x_i \geq 0$
- Initiallösung: Setze  $x_{m+1}, \dots, x_n := 0$  und löse  $A_{1..m} \star x_{1..m}$  mit  $\forall i \leq m. x_i > 0$   
durch Gauß-Verfahren
- Perturbation: Setze ein  $x_i := 0$ , wähle neues  $x_j \neq 0$
- Feasibility Constraint: Alte + neue  $x$ -Komponenten nach Lösung positiv
- Optimality Constraint: Relative Kosten steigen durch Veränderung

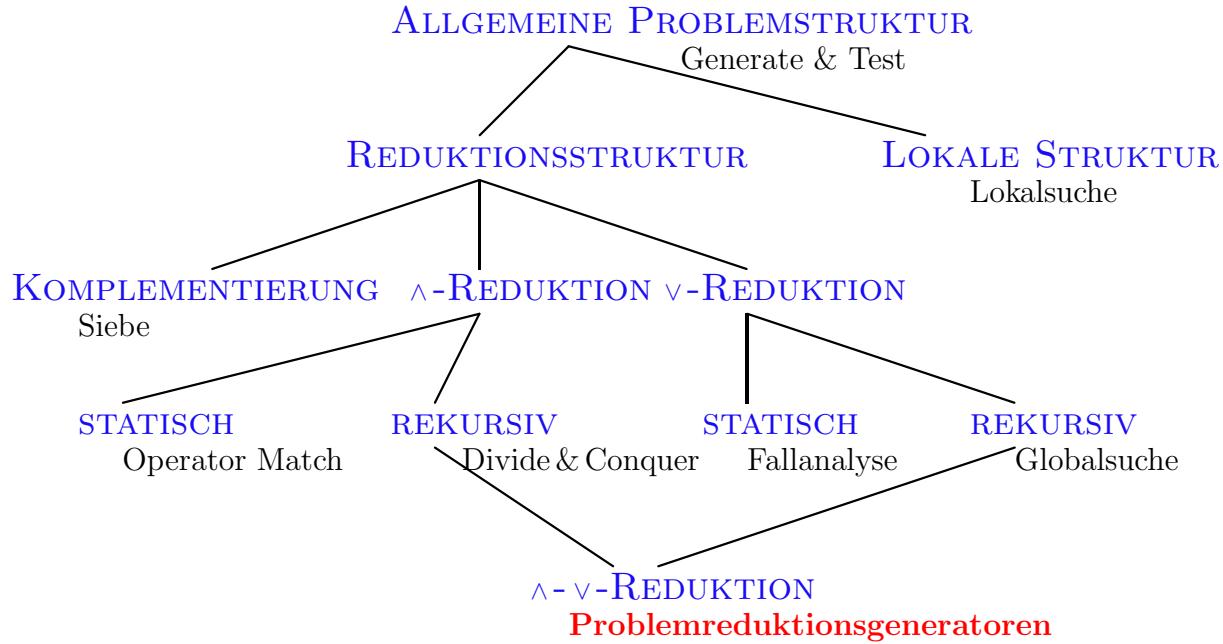
# PROBLEMREDUKTIONSGENERATOREN



## ● $\vee$ - $\wedge$ -Reduktion von Problemen

- Problem besitzt mehrere Lösungen
- Gesamtlösung ist Summe unabhängiger Einzellösungen (  $v$ -Reduktion)
- Einzellösungen aus Teillösungen zusammengesetzt (  $\wedge$ -Reduktion)
- Verallgemeinert Dynamisches Programmieren, Spielbaumsuche, ...

# 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

- **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

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

- **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**

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

- **4 zentrale Komponenten** der Algorithmentheorie

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

## ● 4 zentrale Komponenten der Algorithmentheorie

- $\text{Decompose}_i: D \rightarrow D_{i_1} \times \dots \times D_{i_k}$  Aufspalten der Eingabe in Teilprobleme

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

## ● 4 zentrale Komponenten der Algorithmentheorie

- $\text{Decompose}_i: D \rightarrow D_{i_1} \times \dots \times D_{i_k}$  Aufspalten der Eingabe in Teilprobleme
- Hilfsfunktionen  $f_{i,j}: D_{i_j} \rightarrow R_{i_j}$  evtl. rekursiver Aufruf von  $f$

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

## ● 4 zentrale Komponenten der Algorithmentheorie

- $\text{Decompose}_i: D \rightarrow D_{i_1} \times \dots \times D_{i_k}$  Aufspalten der Eingabe in Teilprobleme
- Hilfsfunktionen  $f_{i,j}: D_{i_j} \rightarrow R_{i_j}$  evtl. rekursiver Aufruf von  $f$
- $\text{Compose}_i: R_{i_1} \times \dots \times R_{i_k} \rightarrow R$  Zusammensetzen der Teillösungen

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

## ● 4 zentrale Komponenten der Algorithmentheorie

- $\text{Decompose}_i: D \rightarrow D_{i_1} \times \dots \times D_{i_k}$  Aufspalten der Eingabe in Teilprobleme
- Hilfsfunktionen  $f_{i,j}: D_{i_j} \rightarrow R_{i_j}$  evtl. rekursiver Aufruf von  $f$
- $\text{Compose}_i: R_{i_1} \times \dots \times R_{i_k} \rightarrow R$  Zusammensetzen der Teillösungen
- Wohlfundierte Ordnung  $\succ$  für Terminierungsgarantiee

## ● 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

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x,y]\}$   
 $\equiv \bigcup_{i,k} (\text{Compose}_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ \text{Decompose}_i)(x)$

## ● 4 zentrale Komponenten der Algorithmentheorie

- $\text{Decompose}_i: D \rightarrow D_{i_1} \times \dots \times D_{i_k}$  Aufspalten der Eingabe in Teilprobleme
- Hilfsfunktionen  $f_{i_j}: D_{i_j} \rightarrow R_{i_j}$  evtl. rekursiver Aufruf von  $f$
- $\text{Compose}_i: R_{i_1} \times \dots \times R_{i_k} \rightarrow R$  Zusammensetzen der Teillösungen
- Wohlfundierte Ordnung  $\succ$  für Terminierungsgarantie
- Erweitertes Strong Problem Reduction Principle

## PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

## PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

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

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

$$O_{i_1, \dots, i_k}[\bar{y}_{i_1}, \dots, \bar{y}_{i_k}, \bar{w}_{i_1}, \dots, \bar{w}_{i_k}] \equiv O_{i_1}(y_{i_1}, w_{i_1}) \wedge \dots \wedge O_{i_k}(y_{i_k}, w_{i_k})$$

# PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

1.  $O$  rekursiv zerlegbar in  $O_{D_i}$ ,  $O_{i_1} \times \dots \times O_{i_k}$  und  $O_{C_i}$  (SPRP)  
 $O[x, z] \Leftrightarrow \exists i:\mathbb{N}, \bar{y}_i:D_{i_1} \times \dots \times D_{i_k}, \bar{w}_i:R_{i_1} \times \dots \times R_{i_k}. O_{D_i}[x, \bar{y}_i] \wedge O_{i_1,..i_k}[\bar{y}_i, \bar{w}_i] \wedge O_{C_i}[\bar{w}_i, z]$   
 $O_{i_1,..i_k}[\bar{y}_{i_1}, \dots, \bar{y}_{i_k}, \bar{w}_{i_1}, \dots, \bar{w}_{i_k}] \equiv O_{i_1}(y_{i_1}, w_{i_1}) \wedge \dots \wedge O_{i_k}(y_{i_k}, w_{i_k})$

2. Dekompositionen erfüllen  $O_{D_i}$  und ‘verkleinern’ Problem

FUNCTION  $f_{d_i}(x:D)$  WHERE  $I[x]$  RETURNS  $\{ \bar{y}_i:D_{i_1} \times \dots \times D_{i_k} \mid O_{D_i}[x, \bar{y}_i] \wedge x \succ \bar{y}_i \wedge I_{i_1,..i_k}[\bar{y}_i] \}$   
 $x \succ \bar{y}_i \equiv x \succ y_{i_j}$  für alle  $j$  mit  $D_{i_j}=D$

# PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

1.  $O$  rekursiv zerlegbar in  $O_{D_i}$ ,  $O_{i_1} \times \dots \times O_{i_k}$  und  $O_{C_i}$  (SPRP)  
 $O[x, z] \Leftrightarrow \exists i:\mathbb{N}, \bar{y}_i:D_{i_1} \times \dots \times D_{i_k}, \bar{w}_i:R_{i_1} \times \dots \times R_{i_k}. O_{D_i}[x, \bar{y}_i] \wedge O_{i_1,..,i_k}[\bar{y}_i, \bar{w}_i] \wedge O_{C_i}[\bar{w}_i, z]$   
 $O_{i_1,..,i_k}[\bar{y}_{i_1}, \dots, \bar{y}_{i_k}, \bar{w}_{i_1}, \dots, \bar{w}_{i_k}] \equiv O_{i_1}(y_{i_1}, w_{i_1}) \wedge \dots \wedge O_{i_k}(y_{i_k}, w_{i_k})$

2. Dekompositionen erfüllen  $O_{D_i}$  und ‘verkleinern’ Problem

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

# PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

1.  $O$  rekursiv zerlegbar in  $O_{D_i}$ ,  $O_{i_1} \times \dots \times O_{i_k}$  und  $O_{C_i}$  (SPRP)  
 $O[x, z] \Leftrightarrow \exists i:\mathbb{N}, \bar{y}_i:D_{i_1} \times \dots \times D_{i_k}, \bar{w}_i:R_{i_1} \times \dots \times R_{i_k}. O_{D_i}[x, \bar{y}_i] \wedge O_{i_1,..,i_k}[\bar{y}_i, \bar{w}_i] \wedge O_{C_i}[\bar{w}_i, z]$   
 $O_{i_1,..,i_k}[\bar{y}_i, \dots, \bar{y}_i, \bar{w}_i, \dots, \bar{w}_i] \equiv O_{i_1}(y_{i_1}, w_{i_1}) \wedge \dots \wedge O_{i_k}(y_{i_k}, w_{i_k})$

2. Dekompositionen erfüllen  $O_{D_i}$  und ‘verkleinern’ Problem

FUNCTION  $f_{d_i}(x:D)$  WHERE  $I[x]$  RETURNS  $\{ \bar{y}_i:D_{i_1} \times \dots \times D_{i_k} \mid O_{D_i}[x, \bar{y}_i] \wedge x \succ \bar{y}_i \wedge I_{i_1,..,i_k}[\bar{y}_i] \}$   
 $x \succ \bar{y}_i \equiv x \succ y_{i_j}$  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  $f_{c_i}(\bar{w}_i:R_{i_1} \times \dots \times R_{i_k})$  WHERE true RETURNS  $\{z_i:R \mid O_{C_i}[\bar{w}_i, z_i]\}$

# PROBLEMREDUKTIONSGENERATOREN: KORREKTHEIT

FUNCTION  $f(x:D)$  WHERE  $I[x]$  RETURNS  $\{y:R \mid O[x, y]\}$   
 $\equiv \bigcup_{i,k} (Compose_i \circ (f_{i_1} \times \dots \times f_{i_k}) \circ Decompose_i)(x)$

ist korrekt, wenn 5 Axiome erfüllt sind

---

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

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

$$O_{i_1, \dots, i_k}[\bar{y}_i, \dots, \bar{y}_i, \bar{w}_i, \dots, \bar{w}_i] \equiv O_{i_1}(y_{i_1}, w_{i_1}) \wedge \dots \wedge O_{i_k}(y_{i_k}, w_{i_k})$$

2. Dekompositionen erfüllen  $O_{D_i}$  und ‘verkleinern’ Problem

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

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

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

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

4. Kompositionen erfüllen  $O_{C_i}$

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

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

# SYNTHESE VON PROBLEMREDUKTIONSGENERATOREN

- Aufspaltung des Reduktionsprinzips in 2 Axiome

# SYNTHESE VON PROBLEMREDUKTIONSGENERATOREN

- Aufspaltung des Reduktionsprinzips in 2 Axiome
  - Starke Korrektheit bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $Decompose_i$  aus Wissensbank

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $Decompose_i$  aus Wissensbank
2. Konstruiere Hilfsfunktionen  $f_{i_j}$  ( $id$  oder rekursiver Aufruf von  $f$ )

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $\text{Decompose}_i$  aus Wissensbank
2. Konstruiere Hilfsfunktionen  $f_{i,j}$  ( $id$  oder rekursiver Aufruf von  $f$ )
3. Konstruiere Dekompositionen  $\text{Compose}_i$  mit Korrektheitsaxiom 1

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $\text{Decompose}_i$  aus Wissensbank
2. Konstruiere Hilfsfunktionen  $f_{i_j}$  ( $id$  oder rekursiver Aufruf von  $f$ )
3. Konstruiere Dekompositionen  $\text{Compose}_i$  mit Korrektheitsaxiom 1
4. Wähle  $\succ$  aus der Wissensbank und verifiziere  $\text{Decompose}_i$

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $\text{Decompose}_i$  aus Wissensbank
2. Konstruiere Hilfsfunktionen  $f_{i_j}$  ( $id$  oder rekursiver Aufruf von  $f$ )
3. Konstruiere Dekompositionen  $\text{Compose}_i$  mit Korrektheitsaxiom 1
4. Wähle  $\succ$  aus der Wissensbank und verifizierte  $\text{Decompose}_i$
5. Verifizierte Vollständigkeitsaxiom

- **Aufspaltung des Reduktionsprinzips in 2 Axiome**

- **Starke Korrektheit** bzgl. Komposition und Dekomposition

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (1)$$

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

$$\forall i:\mathbb{N}, x:D, \bar{y}_i:D_{i_1} \times \dots D_{i_k}, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R. \quad (2)$$

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

- **Vollständigkeit** bzgl. Komposition

$$\forall i:\mathbb{N}, x:D, \bar{w}_i:R_{i_1} \times \dots R_{i_k}, z:R.$$

$$I[x] \wedge O[x, z] \wedge O_{C_i}[\bar{w}_i, z] \Rightarrow \exists \bar{y}_i:D_{i_1} \times \dots D_{i_k}. (I_{i_1 \dots i_k}[\bar{y}_i] \wedge O_{i_1 \dots i_k}[\bar{y}_i, \bar{w}_i])$$

- **Strategie analog zu Divide & Conquer Verfahren**

1. Wähle  $\text{Decompose}_i$  aus Wissensbank
2. Konstruiere Hilfsfunktionen  $f_{i_j}$  ( $id$  oder rekursiver Aufruf von  $f$ )
3. Konstruiere Dekompositionen  $\text{Compose}_i$  mit Korrektheitsaxiom 1
4. Wähle  $\succ$  aus der Wissensbank und verifizierte  $\text{Decompose}_i$
5. Verifizierte Vollständigkeitsaxiom
6. Instantiiere Algorithmenschema