# Automatisierte Logik und Programmierung

## Einheit 22

## Korrektheitserhaltende Optimierungen

1. Logische Vereinfachungen
2. Partielle Auswertung
3. Endliche Differenzierung
4. Fallanalyse
5. Datentyp-Verfeinerung

## Eliminiere überflüssige Berechnungen

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  - Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  – Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext
- **Partielle Auswertung**
  – Symbolische Auswertung von Ausdrücken mit konstanten Komponenten

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  – Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext
- **Partielle Auswertung**
  – Symbolische Auswertung von Ausdrücken mit konstanten Komponenten
- **Endliche Differenzierung**
  – Inkrementelle Berechnung von Teilausdrücken in Schleifen

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  – Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext
- **Partielle Auswertung**
  – Symbolische Auswertung von Ausdrücken mit konstanten Komponenten
- **Endliche Differenzierung**
  – Inkrementelle Berechnung von Teilausdrücken in Schleifen
- **Fallanalyse**
  – Analyse und Vereinfachung von Teilausdrücken

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  – Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext
- **Partielle Auswertung**
  – Symbolische Auswertung von Ausdrücken mit konstanten Komponenten
- **Endliche Differenzierung**
  – Inkrementelle Berechnung von Teilausdrücken in Schleifen
- **Fallanalyse**
  – Analyse und Vereinfachung von Teilausdrücken
- **Datentyp-Verfeinerung**
  – Bestimmung konkreter Implementierungen für abstrakte Datentypen

## Eliminiere überflüssige Berechnungen

- **Simplifikation: Logische Vereinfachung**
  – Äquivalenzumwandlung von Teilausdrücken, ggf. im Kontext
- **Partielle Auswertung**
  – Symbolische Auswertung von Ausdrücken mit konstanten Komponenten
- **Endliche Differenzierung**
  – Inkrementelle Berechnung von Teilausdrücken in Schleifen
- **Fallanalyse**
  – Analyse und Vereinfachung von Teilausdrücken
- **Datentyp-Verfeinerung**
  – Bestimmung konkreter Implementierungen für abstrakte Datentypen
- **Sprachabhängige Optimierung & Compilierung**
  – Ausnutzen der Besonderheiten einer konkreten Zielsprache

# Logische Vereinfachung von Teilausdrücken

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  – Term-Rewriting mit gerichteten Gleichungen
  – Gleichungen formuliert als Lemmata der Wissensbank
  – Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    `a` $\in$ `(x.[])`$\circ$`(b.L)`

$$(x.L_1)\circ L_2 = x.(L_1\circ L_2)$$ append / cons

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a} \in \texttt{x.([]} \circ \texttt{(b.L))}$

$(x.L_1) \circ L_2 = x.(L_1 \circ L_2)$ append / cons

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

$$\mathtt{a} \in \mathtt{(x.[])}\circ\mathtt{(b.L)} \mapsto \mathtt{a} \in \mathtt{x.([]}\circ\mathtt{(b.L))}$$

| | |
|---|---|
| $\mathtt{[]}\circ \boldsymbol{L} = \boldsymbol{L}$ | append / nil |

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

$$\texttt{a}\in\texttt{(x.[])}\circ\texttt{(b.L)} \mapsto \texttt{a}\in\texttt{x.([]}\circ\texttt{(b.L))} \mapsto \texttt{a}\in\texttt{x.(b.L)}$$

| | |
|---|---|
| $\texttt{[]}\circ L = L$ | append / nil |

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\mathtt{a} \in \mathtt{(x.[])} \circ \mathtt{(b.L)} \mapsto \mathtt{a} \in \mathtt{x.(b.L)}$

| | |
|---|---|
| $\boldsymbol{x} \in \boldsymbol{a.L} \;\Leftrightarrow\; \boldsymbol{x{=}a} \vee \boldsymbol{x} \in \boldsymbol{L}$ | member / cons |

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  – Term-Rewriting mit gerichteten Gleichungen
  – Gleichungen formuliert als Lemmata der Wissensbank
  – Identifikation geeigneter Lemmata über Operatoren im Ausdruck

$$\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a} \in \texttt{x.(b.L)} \mapsto \texttt{a=x} \vee \texttt{a} \in \texttt{b.L}$$

$\boldsymbol{x \in a.L \;\Leftrightarrow\; x{=}a \vee x \in L}$ member / cons

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a=x} \vee \texttt{a} \in \texttt{b.L}$

$\boldsymbol{x} \in \boldsymbol{a.L} \;\Leftrightarrow\; \boldsymbol{x{=}a} \vee \boldsymbol{x} \in \boldsymbol{L}$ member / cons

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a=x} \vee \texttt{a} \in \texttt{b.L} \mapsto \texttt{a=x} \vee \texttt{a=b} \vee \texttt{a} \in \texttt{L}$

$\boldsymbol{x} \in \boldsymbol{a.L} \;\Leftrightarrow\; \boldsymbol{x=a} \vee \boldsymbol{x} \in \boldsymbol{L}$ member / cons

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\mathtt{a} \in \mathtt{(x.[])} \circ \mathtt{(b.L)} \mapsto \mathtt{a=x} \lor \mathtt{a=b} \lor \mathtt{a} \in \mathtt{L}$

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    `a` $\in$ `(x.[])` $\circ$ `(b.L)` $\mapsto$ `a=x` $\vee$ `a=b` $\vee$ `a` $\in$ `L`
- **Kontextunabhängige CI-Simplifikation**
  - Anwendung einfacher Gleichungen ohne Rahmenbedingungen
  - Nur der aktuelle Teilausdruck muß betrachtet werden
  - Automatische Ausführung solange anwendbare Lemmata vorhanden

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a=x} \vee \texttt{a=b} \vee \texttt{a} \in \texttt{L}$
- **Kontextunabhängige CI-Simplifikation**
  - Anwendung einfacher Gleichungen ohne Rahmenbedingungen
  - Nur der aktuelle Teilausdruck muß betrachtet werden
  - Automatische Ausführung solange anwendbare Lemmata vorhanden
- **Kontextabhängige CD-Simplifikation**
  - Anwendung von Gleichungen mit Rahmenbedingungen
  - Rahmenbedingungen müssen durch Kontext erfüllt werden

## Logische Vereinfachung von Teilausdrücken

- **Transformation in äquivalente Ausdrücke**
  - Term-Rewriting mit gerichteten Gleichungen
  - Gleichungen formuliert als Lemmata der Wissensbank
  - Identifikation geeigneter Lemmata über Operatoren im Ausdruck

    $\texttt{a} \in \texttt{(x.[])} \circ \texttt{(b.L)} \mapsto \texttt{a=x} \vee \texttt{a=b} \vee \texttt{a} \in \texttt{L}$
- **Kontextunabhängige CI-Simplifikation**
  - Anwendung einfacher Gleichungen ohne Rahmenbedingungen
  - Nur der aktuelle Teilausdruck muß betrachtet werden
  - Automatische Ausführung solange anwendbare Lemmata vorhanden
- **Kontextabhängige CD-Simplifikation**
  - Anwendung von Gleichungen mit Rahmenbedingungen
  - Rahmenbedingungen müssen durch Kontext erfüllt werden
- **Benutzerinteraktion**
  - Auswahl von Teilausdruck und Art der Vereinfachung (CI/CD)
  - Optional bei CD: Begrenzung der Vor- und Rückwärtsinferenzen

# Optimierung des Costas-Arrays Algorithmus

## Ausgangspunkt: schematischer Globalsuchalgorithmus

# Optimierung des Costas-Arrays Algorithmus

## Ausgangspunkt: schematischer Globalsuchalgorithmus

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

# Optimierung des Costas-Arrays Algorithmus

## Ausgangspunkt: schematischer Globalsuchalgorithmus

$$\texttt{FUNCTION Costas (n:}\mathbb{Z}\texttt{) WHERE n}\geq\texttt{1}$$
$$\texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}$$
$$\equiv \texttt{ if nodups([])} \wedge \forall\texttt{j<|[]|.nodups(dtrow([],j))}$$
$$\texttt{then Costas}_{gs}\texttt{(n,[]) else } \emptyset$$

$$\texttt{FUNCTION Costas}_{aux}\texttt{ (n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}$$
$$\texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}$$
$$\texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}$$
$$\equiv \{\texttt{p}\mid\texttt{p}\in\{\texttt{s}\} \wedge \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))} \}$$
$$\cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,t)}\mid\texttt{t}\in\{\texttt{s}\cdot\texttt{i|i}\in\{\texttt{1..n}\}\} \wedge \texttt{nodups(t)}$$
$$\wedge \forall\texttt{j<|t|.nodups(dtrow(t,j))}\}$$

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

# CI-SIMPLIFIKATIONEN IM HAUPTALGORITHMUS COSTAS

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))$\}$
$\equiv$ if nodups([]) $\wedge$ $\forall$j<|[]|.nodups(dtrow([],j))
    then Costas$_{gs}$(n,[]) else $\emptyset$

---

**Domänenwissen:** nodups([]) $\equiv$ true

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([])∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** `nodups([])` ≡ `true`

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([])∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** |[]| ≡ 0

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([])∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true∧∀j<0.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** `|[]|` $\equiv$ `0`

**Domänenwissen:** $\forall x$`<0.`$P[x]$ $\equiv$ `true`

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true ∧ true
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** `|[]|` $\equiv$ `0`

**Domänenwissen:** $\forall x$`<0.`$P[x]$ $\equiv$ `true`

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true ∧ true
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** true $\wedge$ true $\equiv$ true

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** true ∧ true ≡ true

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([])∧∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if true
      then Costas_gs(n,[]) else ∅
```

---

**Domänenwissen:** `if true then` $a$ `else` $b$ $\equiv$ $a$

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

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

---

**Domänenwissen:** `if true then` $a$ `else` $b$ $\equiv$ $a$

# CI-Simplifikationen im Hauptalgorithmus Costas

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ if nodups([]) ∧ ∀j<|[]|.nodups(dtrow([],j))
      then Costas_gs(n,[]) else ∅
```

---

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

# CI-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡       {p|p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
   ∪ ⋃{Costas_gs(n,t)|t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                        ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

# CI-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     {p|p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
   ∪ ⋃{Costas_gs(n,t)|t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                          ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

---

**Domänenwissen:** $\{z \mid z \in \{x\} \wedge P[z]\} \equiv$ `if` $P[x]$ `then` $\{x\}$ `else` $\emptyset$

# CI-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      {p|p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
   ∪ ⋃{Costas_gs(n,t)|t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                         ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,t)|t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                         ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

---

**Domänenwissen:** $\{z \mid z \in \{x\} \wedge P[z]\} \;\equiv\;$ `if` $P[x]$ `then` $\{x\}$ `else` $\emptyset$

# CI-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

FUNCTION $\text{Costas}_{aux}$ (n,s:$\mathbb{Z}\times$Seq($\mathbb{Z}$))
  WHERE n$\geq$1 $\wedge$ range(s)$\subseteq${1..n} $\wedge$ nodups(s) $\wedge$ $\forall$j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq($\mathbb{Z}$)|perm(p,{1..n})$\wedge$ s$\sqsubseteq$p$\wedge$ $\forall$j<|p|.nodups(dtrow(p,j))}
$\equiv$ {p|p$\in${s} $\wedge$ perm(p,{1..n}) $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j)) }
  $\cup$ $\bigcup${$\text{Costas}_{gs}$(n,t)|t$\in${s·i|i$\in${1..n}} $\wedge$ nodups(t)
      $\wedge$ $\forall$j<|t|.nodups(dtrow(t,j))}

---

FUNCTION $\text{Costas}_{aux}$ (n,s:$\mathbb{Z}\times$Seq($\mathbb{Z}$))
  WHERE n$\geq$1 $\wedge$ range(s)$\subseteq${1..n} $\wedge$ nodups(s) $\wedge$ $\forall$j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq($\mathbb{Z}$)|perm(p,{1..n})$\wedge$ s$\sqsubseteq$p$\wedge$ $\forall$j<|p|.nodups(dtrow(p,j))}
$\equiv$ if perm(s,{1..n}) $\wedge$ $\forall$j<|s|.nodups(dtrow(s,j)) then {s} else $\emptyset$
  $\cup$ $\bigcup${$\text{Costas}_{gs}$(n,t)|t$\in${s·i|i$\in${1..n}} $\wedge$ nodups(t)
      $\wedge$ $\forall$j<|t|.nodups(dtrow(t,j))}

---

**Domänenwissen:** $\{\, f[x,t] \mid t \in \{g[x,y] \mid y \in S\,\} \wedge h[t]\}$

$$\equiv \quad \{\, f[x,\, g[x,y]\,] \mid y \in S \wedge h[\,g[x,y]\,]\}$$

# CI-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      {p | p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
   ∪ ⋃{Costasgs(n,t) | t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                         ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

---

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i) | i∈{1..n} ∧ nodups(s·i)
                                   ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** $\{\, f[x,t] \mid t \in \{g[x,y] \mid y \in S\,\} \wedge h[t]\}$

$$\equiv \quad \{\, f[x,\, g[x,y]\,] \mid y \in S \wedge h[\,g[x,y]\,]\}$$

# CI-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      {p|p∈{s} ∧ perm(p,{1..n}) ∧ ∀j<|p|.nodups(dtrow(p,j)) }
   ∪ ⋃{Costas_gs(n,t)|t∈{s·i|i∈{1..n}} ∧ nodups(t)
                                             ∧ ∀j<|t|.nodups(dtrow(t,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

# Vereinfachung unter Berücksichtigung von Kontext

# Kontextabhängige Simplifikation

## Vereinfachung unter Berücksichtigung von Kontext

- **Anwendung bedingter Gleichungen**
  - Gleichung $A \equiv B$ mit Vorbedingung $C$
  - Formuliert als Lemma der Form $C \Rightarrow A \equiv B$

# Kontextabhängige Simplifikation

## Vereinfachung unter Berücksichtigung von Kontext

- **Anwendung bedingter Gleichungen**
  - Gleichung $A \equiv B$ mit Vorbedingung $C$
  - Formuliert als Lemma der Form $C \;\Rightarrow\; A \equiv B$
- **Anwendbarbarkeit abhängig vom Kontext**
  - Ersetze Teilausdruck $A$ durch $B$, wenn $C$ aus dem Kontext folgt

# Kontextabhängige Simplifikation

## Vereinfachung unter Berücksichtigung von Kontext

- **Anwendung bedingter Gleichungen**
  - Gleichung $A \equiv B$ mit Vorbedingung $C$
  - Formuliert als Lemma der Form $C \Rightarrow A \equiv B$
- **Anwendbarbarkeit abhängig vom Kontext**
  - Ersetze Teilausdruck $A$ durch $B$, wenn $C$ aus dem Kontext folgt
  - Kontext ergibt sich aus Syntaxbaum des Gesamtausdrucks
    - · Bei Programmen: benachbarter Programmcode und Vorbedingung
  - Hauptanwendung: Elimination redundanter Teilausdrücke

# Kontextabhängige Simplifikation

## Vereinfachung unter Berücksichtigung von Kontext

- **Anwendung bedingter Gleichungen**
  - Gleichung $A \equiv B$ mit Vorbedingung $C$
  - Formuliert als Lemma der Form $C \;\Rightarrow\; A \equiv B$
- **Anwendbarbarkeit abhängig vom Kontext**
  - Ersetze Teilausdruck $A$ durch $B$, wenn $C$ aus dem Kontext folgt
  - Kontext ergibt sich aus Syntaxbaum des Gesamtausdrucks
    - Bei Programmen: benachbarter Programmcode und Vorbedingung
  - Hauptanwendung: Elimination redundanter Teilausdrücke
- **Komplizierter als CI-Simplifikation**
  - Vorbedingung kann auch Teil einer Gleichung (z.B. $C \wedge A \equiv B$) sein
  - Gleichung kann wird zuweilen auch rückwärts angewandt
  - Automatische Anwendung muß tiefenbeschränkt werden

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \qquad \texttt{if perm(s,}\{\texttt{1..n}\}\texttt{)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j)) then } \{\texttt{s}\} \texttt{ else } \emptyset \\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)} \mid \texttt{i}\in\{\texttt{1..n}\} \wedge \texttt{nodups(s}\cdot\texttt{i)} \\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<|s}\cdot\texttt{i|.nodups(dtrow(s}\cdot\texttt{i,j))}\}
\end{array}
$$

# CD-SIMPLIFIKATIONEN IN DER HILFSFUNKTION $\text{COSTAS}_{aux}$

FUNCTION $\texttt{Costas}_{aux}$ (n,s:$\mathbb{Z}\times$Seq($\mathbb{Z}$))
  WHERE n$\geq$1 $\wedge$ range(s)$\subseteq$$\{$1..n$\}$ $\wedge$ nodups(s) $\wedge$ $\forall$j<|s|.nodups(dtrow(s,j))
  RETURNS $\{$p:Seq($\mathbb{Z}$) | perm(p,$\{$1..n$\}$) $\wedge$ s$\sqsubseteq$p $\wedge$ $\forall$j<|p|.nodups(dtrow(p,j))$\}$

$\equiv$ if perm(s,$\{$1..n$\}$) $\wedge$ $\forall$j<|s|.nodups(dtrow(s,j)) then $\{$s$\}$ else $\emptyset$
  $\cup$ $\bigcup\{$ $\texttt{Costas}_{gs}$(n,s·i) | i$\in$$\{$1..n$\}$ $\wedge$ nodups(s·i)
    $\wedge$ $\forall$j<|s·i|.nodups(dtrow(s·i,j)) $\}$

---

**Domänenwissen:** $P \Rightarrow (Q \wedge P \equiv Q)$

**Kontext der Vorbedingung:** $\forall$j<|s|.nodups(dtrow(s,j))

# CD-SIMPLIFIKATIONEN IN DER HILFSFUNKTION COSTAS$_{aux}$

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                 ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                 ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** $P \Rightarrow (Q \wedge P \equiv Q)$

**Kontext der Vorbedingung:** ∀j<|s|.nodups(dtrow(s,j))

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                   ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                   ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** $\texttt{perm}(L,M) \equiv \texttt{range}(L)\subseteq M \wedge M\subseteq\texttt{range}(L) \wedge \texttt{nodups}(L)$

**Kontext der Vorbedingung:** range(s)⊆{1..n} ∧ nodups(s)

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}⊆range(s) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** perm$(L,M)$ ≡ range$(L)\subseteq M$ ∧ $M\subseteq$range$(L)$ ∧ nodups$(L)$

**Kontext der Vorbedingung:** range(s)⊆{1..n} ∧ nodups(s)

**Domänenwissen:** $M\subseteq M' \equiv M\setminus M'=\emptyset$

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∊{1..n} ∧ nodups(s·i)
                                ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∊{1..n} ∧ nodups(s·i)
                                ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** $\mathtt{perm}(L,M) \equiv \mathtt{range}(L)\subseteq M \wedge M\subseteq\mathtt{range}(L) \wedge \mathtt{nodups}(L)$

**Kontext der Vorbedingung:** $\mathtt{range(s)}\subseteq\{1..n\} \wedge \mathtt{nodups(s)}$

**Domänenwissen:** $M\subseteq M' \equiv M\setminus M'=\emptyset$

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** nodups($L{\cdot}x$) $\equiv$ nodups($L$) $\wedge$ $x \notin L$

**Kontext der Vorbedingung:** nodups(s)

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ i∉s
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** $\texttt{nodups}(L{\cdot}x) \equiv \texttt{nodups}(L) \wedge x \notin L$

**Kontext der Vorbedingung:** `nodups(s)`

**Domänenwissen:** $x \in M \wedge x \notin L \equiv x \in M\backslash\texttt{range}(L)$

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s)
                        ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** nodups($L{\cdot}x$) $\equiv$ nodups($L$) $\wedge$ $x \notin L$

**Kontext der Vorbedingung:** nodups(s)

**Domänenwissen:** $x \in M \;\wedge\; x \notin L \;\equiv\; x \in M\backslash$range$(L)$

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                    ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s)
                          ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

**Domänenwissen:** dtrow($L{\cdot}x$,$i$) $\equiv$ dtrow($L$,$i$)$\cdot$($L_{|L\cdot x|-i}$ - $x$)

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \qquad \texttt{if perm(s,}\{\texttt{1..n}\}\texttt{)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j)) then } \{\texttt{s}\} \texttt{ else } \emptyset \\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)} \mid \texttt{i}\in\{\texttt{1..n}\} \wedge \texttt{nodups(s}\cdot\texttt{i)} \\
\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<|s}\cdot\texttt{i|.nodups(dtrow(s}\cdot\texttt{i,j))}\}
\end{array}
$$

---

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \qquad \texttt{if } \{\texttt{1..n}\}\setminus\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset \\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)} \mid \texttt{i}\in\{\texttt{1..n}\}\setminus\texttt{range(s)} \\
\qquad\qquad \wedge \forall\texttt{j<|s}\cdot\texttt{i|.nodups(dtrow(s,j)}\cdot\texttt{(s}_{|s\cdot i|-j}\texttt{-i))}\}
\end{array}
$$

---

**Domänenwissen:** $\texttt{dtrow(}L\cdot x,i\texttt{)} \equiv \texttt{dtrow(}L,i\texttt{)}\cdot\texttt{(}L_{|L\cdot x|-i}\texttt{ - }x\texttt{)}$

$\texttt{nodups(}L\cdot x\texttt{)} \equiv \texttt{nodups(}L\texttt{)} \wedge x\notin L$

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                    ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i)|i∈{1..n}\range(s)
                    ∧ ∀j<|s·i|.nodups(dtrow(s,j)) ∧ s|s·i|-j -i ∉ dtrow(s,j)}
```

---

**Domänenwissen:** $\texttt{nodups}(L\cdot x) \equiv \texttt{nodups}(L) \wedge x \notin L$

$\forall i<|L\cdot x|.P[i] \equiv \forall i<|L|.P[i] \wedge P[|L|]$

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i) | i∈{1..n} ∧ nodups(s·i)
                                    ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costasaux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costasgs(n,s·i) | i∈{1..n}\range(s)
                        ∧ ∀j<|s|.nodups(dtrow(s,j)) ∧ s|s·i|-j -i ∉ dtrow(s,j)
                              ∧ nodups(dtrow(s,|s|)) ∧ s|s·i|-j -i ∉ dtrow(s,|s|)}
```

---

**Domänenwissen:** $\forall i{<}|L{\cdot}x|.P[i] \;\equiv\; \forall i{<}|L|.P[i] \wedge P[|L|]$

$\texttt{dtrow}(L,|L|) \;\equiv\; []$

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                    ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s)
                        ∧ ∀j<|s|.nodups(dtrow(s,j)) ∧ s_{|s·i|-j}-i ∉ dtrow(s,j)
                               ∧ nodups([]) ∧ s_{|s·i|-j}-i ∉ []}
```

---

**Domänenwissen:** dtrow($L$,|$L$|) ≡ []

nodups([]) ≡ true

$x$ ∉ [] ≡ true

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s)
                    ∧ ∀j<|s|.nodups(dtrow(s,j)) ∧ s_{|s·i|-j}-i∉dtrow(s,j)}
```

---

**Domänenwissen:** nodups([]) $\equiv$ true

$x \notin$ [] $\equiv$ true

$\forall x.\, P[x] \wedge Q[x] \quad \equiv \quad \forall x.\, P[x] \;\wedge\; \forall x.\, Q[x]$

## CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
                                           ∧ ∀j<|s|.s_{|s·i|-j}-i ∉ dtrow(s,j)}
```

---

**Domänenwissen:** $\forall x.\, P[x] \wedge Q[x] \quad \equiv \quad \forall x.\, P[x] \;\wedge\; \forall x.\, Q[x]$

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}\\
\quad\texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\quad\texttt{RETURNS }\{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)}\wedge \texttt{s}\sqsubseteq\texttt{p}\wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}\\
\equiv\qquad \texttt{if perm(s,}\{\texttt{1..n}\}\texttt{)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j)) then }\{\texttt{s}\}\texttt{ else }\emptyset\\
\quad\cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)}\mid\texttt{i}\in\{\texttt{1..n}\} \wedge \texttt{nodups(s}\cdot\texttt{i)}\\
\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<|s}\cdot\texttt{i|.nodups(dtrow(s}\cdot\texttt{i,j))}\}
\end{array}
$$

---

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}\\
\quad\texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\quad\texttt{RETURNS }\{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)}\wedge \texttt{s}\sqsubseteq\texttt{p}\wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}\\
\equiv\qquad \texttt{if }\{\texttt{1..n}\}\setminus\texttt{range(s)=}\emptyset\texttt{ then }\{\texttt{s}\}\texttt{ else }\emptyset\\
\quad\cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)}\mid\texttt{i}\in\{\texttt{1..n}\}\setminus\texttt{range(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<|s|.s}_{|s\cdot i|-j}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}
$$

---

**Kontext der Vorbedingung:** $\forall\texttt{j<|s|.nodups(dtrow(s,j))}$

# CD-Simplifikationen in der Hilfsfunktion $\text{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                   ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s) ∧ ∀j<|s|. s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

---

**Kontext der Vorbedingung:** `∀j<|s|.nodups(dtrow(s,j))`

# CD-Simplifikationen in der Hilfsfunktion Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if perm(s,{1..n}) ∧ ∀j<|s|.nodups(dtrow(s,j)) then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n} ∧ nodups(s·i)
                                  ∧ ∀j<|s·i|.nodups(dtrow(s·i,j))}
```

---

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s) ∧ ∀j<|s|. s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

## Auswertung von Ausdrücken mit Konstanten

# Partielle Auswertung

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren “Wert” bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden

# Partielle Auswertung

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren “Wert” bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation
    Je nach Inhalt der Wissensbank auch reine Simplifikation

# Partielle Auswertung

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

$|\texttt{[}x\texttt{;5]}\circ L|$ unfold append

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

$$
\begin{array}{ll}
\quad |\texttt{[}x\texttt{;5]}\circ L| & \text{unfold append} \\
\mapsto |x\texttt{.([5]}\circ L\texttt{)}| & \text{unfold append}
\end{array}
$$

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

| | | |
|---|---|---|
| | $\texttt{|[}x\texttt{;5]}\circ L\texttt{|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.([5]}\circ L\texttt{)|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.([]}\circ L\texttt{))|}$ | unfold append |

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

$$
\begin{array}{llr}
 & |[x;5] \circ L| & \text{unfold append} \\
\mapsto & |x.([5] \circ L)| & \text{unfold append} \\
\mapsto & |x.(5.([] \circ L))| & \text{unfold append} \\
\mapsto & |x.(5.L)| & \text{unfold length}
\end{array}
$$

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

| | | |
|---|---|---|
| | $\texttt{|[}x\texttt{;5]}\circ L\texttt{|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.([5]}\circ L\texttt{)|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.([]}\circ L\texttt{))|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.}L\texttt{)|}$ | unfold length |
| $\mapsto$ | $\texttt{1 + |5.}L\texttt{|}$ | unfold length |

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

| | | |
|---|---|---|
| | $\texttt{|[}x\texttt{;5]}\circ L\texttt{|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.([5]}\circ L\texttt{)|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.([]}\circ L\texttt{))|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.}L\texttt{)|}$ | unfold length |
| $\mapsto$ | $\texttt{1 + |5.}L\texttt{|}$ | unfold length |
| $\mapsto$ | $\texttt{1 + 1 + |}L\texttt{|}$ | simplify |

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

$$
\begin{array}{lll}
 & |[x;5] \circ L| & \text{unfold append} \\
\mapsto & |x.([5] \circ L)| & \text{unfold append} \\
\mapsto & |x.(5.([] \circ L))| & \text{unfold append} \\
\mapsto & |x.(5.L)| & \text{unfold length} \\
\mapsto & 1 + |5.L| & \text{unfold length} \\
\mapsto & 1 + 1 + |L| & \text{simplify} \\
\mapsto & 2 + |L| &
\end{array}
$$

## Auswertung von Ausdrücken mit Konstanten

- **Symbolische Auswertung zur Entwurfszeit**
  - Reduktion von Ausdrücken, deren "Wert" bestimmt werden kann
  - Anwendbar, solange ein konstanter Teilausdruck vorhanden
  - Formale Technik: Auffalten von Definitionen + Simplifikation

    Je nach Inhalt der Wissensbank auch reine Simplifikation

**Beispiel:**

| | | |
|---|---|---|
| | $\texttt{|[}x\texttt{;5]}\circ L\texttt{|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.([5]}\circ L\texttt{)|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.([]}\circ L\texttt{))|}$ | unfold append |
| $\mapsto$ | $\texttt{|}x\texttt{.(5.}L\texttt{)|}$ | unfold length |
| $\mapsto$ | $\texttt{1 + |5.}L\texttt{|}$ | unfold length |
| $\mapsto$ | $\texttt{1 + 1 + |}L\texttt{|}$ | simplify |
| $\mapsto$ | $\texttt{2 + |}L\texttt{|}$ | |

- **Benutzerinteraktion:**
  - Auswahl von auszuwertendem Ausdruck und Optimierungstechnik

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

$\texttt{FUNCTION Costas}_{aux}\texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}$ WHERE ... RETURNS ...

$\equiv\quad \texttt{if } \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset$

$\quad\cup\ \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)}\mid \texttt{i}\in\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \forall \texttt{j<|s|.s}_{|s\cdot i|-j}\texttt{-i}\notin\texttt{dtrow(s,j)}\}$

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

  FUNCTION $\texttt{Costas}_{aux}$(n,s:$\mathbb{Z}\times$Seq($\mathbb{Z}$)) WHERE ... RETURNS ...

  $\equiv$ if $\{$1..n$\}\backslash$range(s)=$\emptyset$ then $\{$s$\}$ else $\emptyset$

  $\cup \bigcup\{$ $\texttt{Costas}_{gs}$(n,s·i) | i $\in$ $\{$1..n$\}\backslash$range(s) $\wedge$ $\forall$j<|s|. $\texttt{s}_{|s\cdot i|-j}$ -i $\notin$ dtrow(s,j)$\}$

  ---

  - Benutzer identifiziert $\{$1..n$\}\backslash$range(s) als wiederkehrende Berechnung

# ENDLICHE DIFFERENZIERUNG

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))  WHERE ...  RETURNS ...}\\
\equiv \quad \texttt{if } \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset\\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)} \mid \texttt{i} \in \{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \forall\texttt{j<|s|.}\ \texttt{s}_{|s\cdot i|-j}\texttt{-i} \notin \texttt{dtrow(s,j)}\}
\end{array}
$$

---

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{))  WHERE ...  RETURNS ...}\\
\equiv \quad \texttt{if } \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset\\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,}\{\texttt{1..n}\}\backslash\texttt{range(s}\cdot\texttt{i))} \mid \texttt{i} \in \{\texttt{1..n}\}\backslash\texttt{range(s)}\\
\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<|s|.}\ \texttt{s}_{|s\cdot i|-j}\texttt{-i} \notin \texttt{dtrow(s,j)}\}
\end{array}
$$

---

  - Benutzer identifiziert $\{\texttt{1..n}\}\backslash\texttt{range(s)}$ als wiederkehrende Berechnung
  - System ändert $\texttt{Costas}_{aux}\texttt{(n,s)}$ in $\texttt{Costas}_{aux}\texttt{(n,s,pool)}$ und ändert Aufrufe von $\texttt{Costas}_{aux}$ entsprechend

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

```
FUNCTION Costas_aux(n,s:ℤ×Seq(ℤ))  WHERE ...   RETURNS ...
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i) | i∈{1..n}\range(s) ∧ ∀j<|s|. s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool ...)  WHERE ... pool = {1..n}\range(s)
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i,{1..n}\range(s·i)) | i∈{1..n}\range(s)
                                         ∧ ∀j<|s|. s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

---

  - Benutzer identifiziert `{1..n}\range(s)` als wiederkehrende Berechnung
  - System ändert `Costas`$_{aux}$`(n,s)` in `Costas`$_{aux}$`(n,s,pool)` und ändert Aufrufe von `Costas`$_{aux}$ entsprechend
  - System ergänzt `pool = {1..n}\range(s)` zur Eingabebedingung

# ENDLICHE DIFFERENZIERUNG

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

$$\texttt{FUNCTION Costas}_{aux}\texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}\quad\texttt{WHERE}\ \ldots\quad\texttt{RETURNS}\ \ldots$$
$$\equiv\quad \texttt{if}\ \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset\ \texttt{then}\ \{\texttt{s}\}\ \texttt{else}\ \emptyset$$
$$\cup\ \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)}\ |\ \texttt{i}\in\{\texttt{1..n}\}\backslash\texttt{range(s)}\wedge\forall\texttt{j<|s|.}\ \texttt{s}_{|s\cdot i|-j}\ \texttt{-i}\notin\texttt{dtrow(s,j)}\}$$

---

$$\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool}\ \ldots\texttt{)}\quad\texttt{WHERE}\ \ldots\ \texttt{pool} = \{\texttt{1..n}\}\backslash\texttt{range(s)}$$
$$\equiv\quad \texttt{if}\ \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset\ \texttt{then}\ \{\texttt{s}\}\ \texttt{else}\ \emptyset$$
$$\cup\ \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,}\{\texttt{1..n}\}\backslash\texttt{range(s}\cdot\texttt{i))}\ |\ \texttt{i}\in\{\texttt{1..n}\}\backslash\texttt{range(s)}$$
$$\wedge\forall\texttt{j<|s|.}\ \texttt{s}_{|s\cdot i|-j}\ \texttt{-i}\notin\texttt{dtrow(s,j)}\}$$

---

  - Benutzer identifiziert $\{\texttt{1..n}\}\backslash\texttt{range(s)}$ als wiederkehrende Berechnung
  - System ändert $\texttt{Costas}_{aux}\texttt{(n,s)}$ in $\texttt{Costas}_{aux}\texttt{(n,s,pool)}$ und ändert Aufrufe von $\texttt{Costas}_{aux}$ entsprechend
  - System ergänzt $\texttt{pool} = \{\texttt{1..n}\}\backslash\texttt{range(s)}$ zur Eingabebedingung
  - System vereinfacht Vorkommen von $\{\texttt{1..n}\}\backslash\texttt{range(s)}$ zu $\texttt{pool}$

# Endliche Differenzierung

- **Inkrementelle Berechnung statt ständiger Neuberechnung**
  - Ersetze Teilausdruck $E[x]$ in $f(x)$ durch neue Variable $c$
  - Initialisiere $c$ mit Ausdruck für Basiswerte
  - Bestimme differentielle Veränderungen bei rekursivem Aufruf
- **Endliche Differenzierung am Beispiel**

FUNCTION $\texttt{Costas}_{aux}$(n,s:$\mathbb{Z}\times$Seq($\mathbb{Z}$)) WHERE ... RETURNS ...
$\equiv$ if {1..n}\range(s)=$\emptyset$ then {s} else $\emptyset$
$\cup \bigcup${$\texttt{Costas}_{gs}$(n,s$\cdot$i) | i $\in$ {1..n}\range(s) $\wedge$ $\forall$j<|s|. $\texttt{s}_{|s\cdot i|-j}$ -i $\notin$ dtrow(s,j)}

---

FUNCTION $\texttt{Costas}_{aux}$(n,s,pool ...) WHERE ... pool = {1..n}\range(s)
$\equiv$ if pool=$\emptyset$ then {s} else $\emptyset$
$\cup \bigcup${$\texttt{Costas}_{gs}$(n,s$\cdot$i,pool\{i}) | i $\in$ pool
$\wedge$ $\forall$j<|s|. $\texttt{s}_{|s\cdot i|-j}$ -i $\notin$ dtrow(s,j)}

---

  - Benutzer identifiziert {1..n}\range(s) als wiederkehrende Berechnung
  - System ändert $\texttt{Costas}_{aux}$(n,s) in $\texttt{Costas}_{aux}$(n,s,pool) und ändert Aufrufe von $\texttt{Costas}_{aux}$ entsprechend
  - System ergänzt pool = {1..n}\range(s) zur Eingabebedingung
  - System vereinfacht Vorkommen von {1..n}\range(s) zu pool

# Endliche Differenzierung – Formalia

- **Abstraktion über Ausdruck $E[x]$ in Funktion $f(x)$**

  FUNCTION $main...$ WHERE ... RETURNS ... SUCH THAT... $\equiv$ $....f(x_0)....$

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

# Endliche Differenzierung – Formalia

- **Abstraktion über Ausdruck $E[x]$ in Funktion $f(x)$**

  `FUNCTION` $main...$ `WHERE ... RETURNS ... SUCH THAT...` $\equiv$ $....f(x_0)....$

  `FUNCTION` $f(x{:}D){:}R$ `WHERE` $I[x]$ `RETURNS` $y$ `SUCH THAT` $O[x,y]$
  $\equiv$ $....E[x]....f(\,t[x]\,)..$

  ---

  – Erweitere $f$ zu neuem $f'$, so daß $f(x) = f'(x, E[x])$

  – Ersetze alle Aufrufe der Art $f(\,t[x]\,)$ durch $f'(\,t[x],\ E[t[x]]\,)$

  – Ergänze Gleichung $c{=}E[x]$ zur Eingabebedingung von $f'$

# Endliche Differenzierung – Formalia

- **Abstraktion über Ausdruck $E[x]$ in Funktion $f(x)$**

  FUNCTION $main...$ WHERE ... RETURNS ... SUCH THAT... $\equiv$ $....f(x_0)....$

  FUNCTION $f(x{:}D){:}R$ WHERE $I[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ $....E[x]....f(\,t[x]\,)..$

  ---

  – Erweitere $f$ zu neuem $f'$, so daß $f(x) = f'(x, E[x])$

  – Ersetze alle Aufrufe der Art $f(\,t[x]\,)$ durch $f'(\,t[x],\ E[t[x]]\,)$

  – Ergänze Gleichung $c{=}E[x]$ zur Eingabebedingung von $f'$

  ---

  FUNCTION $main...$ WHERE ... RETURNS ... SUCH THAT... $\equiv$ $....f'(\,x_0,\ E[x_0]\,)....$

  FUNCTION $f(x,c{:}D{\times}D'){:}R$ WHERE $I[x]\wedge c{=}E[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ $....E[x]....f'(\,t[x],\ E[t[x]]\,)..$

# Endliche Differenzierung – Formalia

- **Abstraktion über Ausdruck $E[x]$ in Funktion $f(x)$**

  FUNCTION $main$... WHERE ... RETURNS ... SUCH THAT... $\equiv$ ....$f(x_0)$....
  FUNCTION $f$($x$:$D$):$R$ WHERE $I[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ ....$E[x]$....$f(\,t[x]\,)$..

  ---

  – Erweitere $f$ zu neuem $f'$, so daß $f(x) = f'(x, E[x])$

  – Ersetze alle Aufrufe der Art $f(\,t[x]\,)$ durch $f'(\,t[x],\ E[t[x]]\,)$

  – Ergänze Gleichung $c{=}E[x]$ zur Eingabebedingung von $f'$

  ---

  FUNCTION $main$... WHERE ... RETURNS ... SUCH THAT... $\equiv$ ....$f'(\,x_0,\ E[x_0]\,)$....
  FUNCTION $f$($x,c$:$D{\times}D'$):$R$ WHERE $I[x]\wedge c{=}E[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ ....$E[x]$....$f'(\,t[x],\ E[t[x]]\,)$..

- **Simplifikation mit Gleichung $c = E[x]$**

  – Transformiere Ausdrücke der Form $E[\,g[x]\,]$ in die Form $g'[\,E[x]\,]$

  – Ersetze alle Vorkommen von $E[x]$ durch $c$

# Endliche Differenzierung – Formalia

- **Abstraktion über Ausdruck $E[x]$ in Funktion $f(x)$**

  FUNCTION $main...$ WHERE ... RETURNS ... SUCH THAT... $\equiv$ $....f(x_0)....$
  FUNCTION $f(x\mathtt{:}D)\mathtt{:}R$ WHERE $I[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ $....E[x]....f(\,t[x]\,)..$

  ---

  – Erweitere $f$ zu neuem $f'$, so daß $f(x) = f'(x, E[x])$

  – Ersetze alle Aufrufe der Art $f(\,t[x]\,)$ durch $f'(\,t[x],\; E[t[x]]\,)$

  – Ergänze Gleichung $c{=}E[x]$ zur Eingabebedingung von $f'$

  ---

  FUNCTION $main...$ WHERE ... RETURNS ... SUCH THAT... $\equiv$ $....f'(\,x_0,\; E[x_0]\,)....$
  FUNCTION $f(x,c\mathtt{:}D{\times}D')\mathtt{:}R$ WHERE $I[x] \wedge c{=}E[x]$ RETURNS $y$ SUCH THAT $O[x,y]$
  $\equiv$ $....E[x]....f'(\,t[x],\; E[t[x]]\,)..$

- **Simplifikation mit Gleichung $c = E[x]$**

  – Transformiere Ausdrücke der Form $E[\,g[x]\,]$ in die Form $g'[\,E[x]\,]$

  – Ersetze alle Vorkommen von $E[x]$ durch $c$

- **Benutzerinteraktion:**

  – Auswahl des zu ersetzenden Teilausdrucks $E[x]$ in $f(x)$

# Endliche Differenzierung von Costas$_{aux}$

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

$\equiv$ if $\{$1..n$\}\setminus$range(s)=$\emptyset$ then $\{$s$\}$ else $\emptyset$
$\cup \bigcup\{$Costas$_{gs}$(n,s$\cdot$i) | i $\in\{$1..n$\}\setminus$range(s) $\wedge$ $\forall$j<|s|. s$_{|s\cdot i|-j}$ -i $\notin$ dtrow(s,j)$\}$

# Endliche Differenzierung von $\texttt{Costas}_{aux}$

$\texttt{FUNCTION Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))}$

$\quad\texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}$

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

$\equiv \quad \texttt{if } \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset$

$\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i) | i}\in\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \forall\texttt{j<|s|.s}_{|s\cdot i|-j}\texttt{-i}\notin\texttt{dtrow(s,j)}\}$

---

Benutzer identifiziert $\{\texttt{1..n}\}\backslash\texttt{range(s)}$ als wiederkehrende Berechnung

# Endliche Differenzierung von $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s) ∧ ∀j<|s|.s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool:ℤ×Seq(ℤ)×Set(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
             ∧ pool = {1..n}\range(s)
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i})|i∈pool
                                       ∧ ∀j<|s|.s_{|s·i|-j} -i ∉ dtrow(s,j)}
```

---

Benutzer identifiziert $\{1..n\}\backslash\texttt{range(s)}$ als wiederkehrende Berechnung

System ändert $\texttt{Costas}_{aux}\texttt{(n,s)}$ in $\texttt{Costas}_{aux}\texttt{(n,s,pool)}$ und ändert Aufrufe

System ergänzt $\texttt{pool} = \{1..n\}\backslash\texttt{range(s)}$ zur Eingabebedingung
und vereinfacht Vorkommen von $\{1..n\}\backslash\texttt{range(s)}$ zu pool

# ENDLICHE DIFFERENZIERUNG VON $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i)|i∈{1..n}\range(s) ∧ ∀j<|s|.s_{|s·i|−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool:ℤ×Seq(ℤ)×Set(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s)
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i})|i∈pool
                                         ∧ ∀j<|s|.s_{|s·i|−j} -i ∉ dtrow(s,j)}
```

---

Benutzer identifiziert `|s|` als wiederkehrende Berechnung

# ENDLICHE DIFFERENZIERUNG VON $\mathtt{Costas}_{aux}$

$$\begin{array}{l}
\texttt{FUNCTION } \mathtt{Costas}_{aux}\ \texttt{(n,s:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{))} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \qquad \texttt{if } \{\texttt{1..n}\}\backslash\texttt{range(s)=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset \\
\quad \cup \bigcup\{\mathtt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i)} \mid \texttt{i}\in\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \forall\texttt{j<|s|.}\,\texttt{s}_{|s\cdot i|-j}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}$$

---

$$\begin{array}{l}
\texttt{FUNCTION } \mathtt{Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\qquad\qquad\qquad \wedge \texttt{ pool = }\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \texttt{ssize=|s|} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \qquad \texttt{if pool=}\emptyset \texttt{ then } \{\texttt{s}\} \texttt{ else } \emptyset \\
\quad \cup \bigcup\{\mathtt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)} \mid \texttt{i}\in\texttt{pool} \\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge \forall\texttt{j<ssize.}\,\texttt{s}_{\texttt{ssize+1-j}}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}$$

---

Benutzer identifiziert `|s|` als wiederkehrende Berechnung

System ändert $\mathtt{Costas}_{aux}$`(n,s,pool)` in $\mathtt{Costas}_{aux}$`(n,s,pool,ssize)`

System ergänzt `ssize = |s|` zur Eingabebedingung

und vereinfacht Vorkommen von `|s|` zu `ssize`

# ENDLICHE DIFFERENZIERUNG VON Costas$_{aux}$

```
FUNCTION Costas_aux (n,s:ℤ×Seq(ℤ))
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
  RETURNS {p:Seq(ℤ) | perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if {1..n}\range(s)=∅ then {s} else ∅
   ∪ ⋃{Costas_gs(n,s·i) | i∈{1..n}\range(s) ∧ ∀j<|s|. s_{|s·i|−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
             ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ) | perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1) | i∈pool
                                              ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

### Modifizierter Aufruf aus Hauptfunktion

```
FUNCTION Costas (n:ℤ) WHERE n≥1
  RETURNS {p:Seq(ℤ) | perm(p,{1..n})∧ ∀j<|p|.nodups(dtrow(p,j))}
≡ Costas_gs(n,[],{1..n},0)
```

# Fallanalyse

- **Separate Analyse von Einzelfällen**
  – Auswertung von Tests aus anderen Programmteilen
  – Zweck: globale Vereinfachung durch lokale Einzelanalyse

# Fallanalyse

- **Separate Analyse von Einzelfällen**
  - Auswertung von Tests aus anderen Programmteilen
  - Zweck: globale Vereinfachung durch lokale Einzelanalyse
  - Formal: Erzeugung eines Kontexts + CD-Simplifikation

# Fallanalyse

- **Separate Analyse von Einzelfällen**
  - Auswertung von Tests aus anderen Programmteilen
  - Zweck: globale Vereinfachung durch lokale Einzelanalyse
  - Formal: Erzeugung eines Kontexts + CD-Simplifikation

- **Kontexterzeugung mit Prädikat $P$**
  - Ersetze Ausdruck $E[x]$ durch `if` $P[x]$ `then` $E[x]$ `else` $E[x]$
  - Vereinfache $E[x]$ in den entsprechenden Kontexten $P[x]$ und $\neg P[x]$
  - Distributiere `if` $P[x]$ `then`...`else`...
    über Ausdrücke außerhalb von $E[x]$

# Fallanalyse

- **Separate Analyse von Einzelfällen**
  - Auswertung von Tests aus anderen Programmteilen
  - Zweck: globale Vereinfachung durch lokale Einzelanalyse
  - Formal: Erzeugung eines Kontexts + CD-Simplifikation
- **Kontexterzeugung mit Prädikat $P$**
  - Ersetze Ausdruck $E[x]$ durch `if` $P[x]$ `then` $E[x]$ `else` $E[x]$
  - Vereinfache $E[x]$ in den entsprechenden Kontexten $P[x]$ und $\neg P[x]$
  - Distributiere `if` $P[x]$ `then...else...`
    über Ausdrücke außerhalb von $E[x]$
- **Benutzerinteraktion:**
  - Auswahl des zu ersetzenden Teilausdrucks $E[x]$
  - Auswahl des Prädikats $P[x]$
  - Auslösung der nachfolgenden Simplifikationen

# Fallanalyse in $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
             ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                 ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

# FALLANALYSE IN $\texttt{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad\texttt{WHERE n}\geq\texttt{1} \land \texttt{range(s)}\subseteq\{\texttt{1..n}\} \land \texttt{nodups(s)} \land \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\qquad\qquad\qquad \land \texttt{pool = }\{\texttt{1..n}\}\backslash\texttt{range(s)} \land \texttt{ssize=|s|}\\
\quad\texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)|perm(p,}\{\texttt{1..n}\}\texttt{)} \land \texttt{s}\sqsubseteq\texttt{p} \land \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}\\
\equiv \qquad \texttt{if pool=}\emptyset\texttt{ then }\{\texttt{s}\}\texttt{ else }\emptyset\\
\quad\cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)|i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad \land \forall\texttt{j<ssize. s}_{\texttt{ssize+1-j}}\texttt{ -i} \notin \texttt{dtrow(s,j)}\}
\end{array}
$$

---

Markierung von Ausdruck und Testprädikat

# Fallanalyse in $\mathtt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
               ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                       ∧ ∀j<ssize. s_(ssize+1−j) -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE ... RETURNS ...
≡    if pool=∅ then {s} else ∅
  ∪ if pool=∅ then ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                       ∧ ∀j<ssize. s_(ssize+1−j) -i ∉ dtrow(s,j)}
               else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                       ∧ ∀j<ssize. s_(ssize+1−j) -i ∉ dtrow(s,j)}
```

---

Aufspalten des Ausdrucks auf beide Fälle

# Fallanalyse in Costas$_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
             ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                      ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE ... RETURNS ...
≡   if pool=∅ then {s} else ∅
  ∪ if pool=∅ then ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                      ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
              else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                      ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

Domänenwissen: $\bigcup\{\, f(\mathtt{i}) \mid \mathtt{i} \in \emptyset\} = \emptyset$

Kontext der Fallanalyse: $\mathtt{pool}=\emptyset$

# Fallanalyse in Costas$_{aux}$

```
FUNCTION Costasaux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
             ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costasgs(n,s·i,pool\{i},ssize+1)|i∈pool
                                        ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costasaux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE ... RETURNS ...
≡   if pool=∅ then {s} else ∅
  ∪ if pool=∅ then ∅
              else ⋃{Costasgs(n,s·i,pool\{i},ssize+1)|i∈pool
                                        ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

Domänenwissen: $\bigcup\{\, f(\mathtt{i}) \mid \mathtt{i} \in \emptyset\} = \emptyset$

Kontext der Fallanalyse: $\mathtt{pool}=\emptyset$

# Fallanalyse in $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                      ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE ... RETURNS ...
≡   if pool=∅ then {s} else ∅
  ∪ if pool=∅ then ∅
              else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                      ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

Distribution von `if pool=∅ then ... else ..` über $\cup$

# Fallanalyse in $\texttt{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\qquad\qquad\qquad \wedge\ \texttt{pool = }\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \texttt{ssize=|s|}\\
\quad \texttt{RETURNS }\{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}\\
\equiv \qquad \texttt{if pool=}\emptyset\texttt{ then }\{\texttt{s}\}\texttt{ else }\emptyset\\
\quad \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)}\mid\texttt{i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \forall\texttt{j<ssize.}\ \texttt{s}_{\texttt{ssize+1-j}}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}
$$

---

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad \texttt{WHERE}\ \ldots\ \texttt{RETURNS}\ \ldots\\
\equiv \quad \texttt{if pool=}\emptyset\texttt{ then }\{\texttt{s}\}\cup\emptyset\\
\qquad\qquad\qquad \texttt{else }\emptyset \cup \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)}\mid\texttt{i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ \forall\texttt{j<ssize.}\ \texttt{s}_{\texttt{ssize+1-j}}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}
$$

---

Distribution von $\texttt{if pool=}\emptyset\texttt{ then ... else ..}$ über $\cup$

# Fallanalyse in $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡     if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                             ∧∀j<ssize. s_ssize+1−j -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE ... RETURNS ...
≡   if pool=∅ then {s}∪∅
              else ∅ ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                             ∧∀j<ssize. s_ssize+1−j -i ∉ dtrow(s,j)}
```

---

Domänenwissen: $S\cup\emptyset = \emptyset = \emptyset\cup S$

# Fallanalyse in $\texttt{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad\texttt{WHERE n}\geq\texttt{1} \land \texttt{range(s)}\subseteq\texttt{\{1..n\}} \land \texttt{nodups(s)} \land \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\qquad\qquad\qquad \land\ \texttt{pool = \{1..n\}}\backslash\texttt{range(s)} \land \texttt{ssize=|s|}\\
\quad\texttt{RETURNS \{p:Seq(}\mathbb{Z}\texttt{) | perm(p,\{1..n\})} \land \texttt{s}\sqsubseteq\texttt{p} \land \forall\texttt{j<|p|.nodups(dtrow(p,j))\}}\\
\equiv \qquad \texttt{if pool=}\emptyset\texttt{ then \{s\} else }\emptyset\\
\quad\cup \bigcup\texttt{\{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\texttt{\{i\},ssize+1) | i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land \forall\texttt{j<ssize. s}_{\texttt{ssize+1-j}}\texttt{ -i} \notin \texttt{dtrow(s,j)\}}
\end{array}
$$

---

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad\texttt{WHERE ... RETURNS ...}\\
\equiv \quad \texttt{if pool=}\emptyset\texttt{ then \{s\}}\\
\qquad\qquad\qquad \texttt{else }\bigcup\texttt{\{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\texttt{\{i\},ssize+1) | i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \land \forall\texttt{j<ssize. s}_{\texttt{ssize+1-j}}\texttt{ -i} \notin \texttt{dtrow(s,j)\}}
\end{array}
$$

---

Domänenwissen: $S\cup\emptyset = \emptyset = \emptyset\cup S$

# Fallanalyse in Costas$_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡      if pool=∅ then {s} else ∅
  ∪ ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                              ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡  if pool=∅ then {s}
              else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                              ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

# Datentypverfeinerung

**Ersetze abstrakte Datentypen**
**durch effiziente konkrete Implementierung**

**Ersetze abstrakte Datentypen durch effiziente konkrete Implementierung**

- **Endliche Mengen**
  - Listen: Standardimplementierung
  - Bitvektor: Mengen über endlichem Domain
  - Charakteristische Funktion: effiziente Elementrelation/Einfügen/Löschen

**Ersetze abstrakte Datentypen**
**durch effiziente konkrete Implementierung**

- **Endliche Mengen**
  - Listen: Standardimplementierung
  - Bitvektor: Mengen über endlichem Domain
  - Charakteristische Funktion: effiziente Elementrelation/Einfügen/Löschen
- **Folgen**
  - Verkettete Liste: Standardimplementierung
  - Umgekehrt verkettete Liste: gut für append-Operation

# Datentypverfeinerung

**Ersetze abstrakte Datentypen**
**durch effiziente konkrete Implementierung**

- **Endliche Mengen**
  - Listen: Standardimplementierung
  - Bitvektor: Mengen über endlichem Domain
  - Charakteristische Funktion: effiziente Elementrelation/Einfügen/Löschen
- **Folgen**
  - Verkettete Liste: Standardimplementierung
  - Umgekehrt verkettete Liste: gut für append-Operation
- **Benutzerinteraktion:**
  - System stellt Auswahl von Implementierungen bereit
  - Benutzer wählt Nichtstandard-Implementierung einzeln für jede Variable
  - System ersetzt abstrakte Notation durch konkrete Implementierung und fügt ggf. Konversionen ein

# DATENTYPVERFEINERUNG FÜR $\texttt{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)} \\
\quad \texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))} \\
\qquad\qquad \wedge\ \texttt{pool = }\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \texttt{ssize=|s|} \\
\quad \texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\} \\
\equiv \quad \texttt{if pool=}\emptyset\texttt{ then }\{\texttt{s}\} \\
\qquad\qquad \texttt{else } \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)} \mid \texttt{i}\in\texttt{pool} \\
\qquad\qquad\qquad\qquad\qquad \wedge\ \forall\texttt{j<ssize.s}_{\texttt{ssize+1}-\texttt{j}}\texttt{-i}\notin\texttt{dtrow(s,j)}\}
\end{array}
$$

---

# DATENTYPVERFEINERUNG FÜR $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
                ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡   if pool=∅ then {s}
               else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                        ∧∀j<ssize.s_(ssize+1−j) -i ∉ dtrow(s,j)}
```

---

n: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

# DATENTYPVERFEINERUNG FÜR $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ) | perm(p,{1..n}) ∧ s⊑p ∧ ∀j<|p|.nodups(dtrow(p,j))}
≡   if pool=∅ then {s}
              else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1) | i∈pool
                                        ∧ ∀j<ssize. s_{ssize+1−j} -i ∉ dtrow(s,j)}
```

---

$\texttt{n}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

$\texttt{s}$: $\texttt{Seq}(\mathbb{Z})$, Elemente werden hinten angehängt $\mapsto$ umgekehrt verkettete Liste

# DATENTYPVERFEINERUNG FÜR $\texttt{Costas}_{aux}$

$\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}$
$\quad\texttt{WHERE n}\geq\texttt{1} \land \texttt{range(s)}\subseteq\{\texttt{1..n}\} \land \texttt{nodups(s)} \land \forall\texttt{j<|s|.nodups(dtrow(s,j))}$
$\qquad\qquad \land \texttt{pool = }\{\texttt{1..n}\}\setminus\texttt{range(s)} \land \texttt{ssize=|s|}$
$\quad\texttt{RETURNS }\{\texttt{p:Seq(}\mathbb{Z}\texttt{)}\mid\texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \land \texttt{s}\sqsubseteq\texttt{p} \land \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}$
$\equiv\quad\texttt{if pool=}\emptyset\texttt{ then }\{\texttt{s}\}$
$\qquad\qquad\texttt{else }\bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\setminus\{\texttt{i}\}\texttt{,ssize+1)}\mid\texttt{i}\in\texttt{pool}$
$\qquad\qquad\qquad\qquad \land \forall\texttt{j<ssize.}\,\texttt{s}_{\texttt{ssize+1-j}}\texttt{-i}\notin\texttt{dtrow(s,j)}\}$

---

$\texttt{n}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

$\texttt{s}$: $\texttt{Seq(}\mathbb{Z}\texttt{)}$, Elemente werden hinten angehängt $\mapsto$ umgekehrt verkettete Liste

$\texttt{pool}$: $\texttt{Set(}\mathbb{Z}\texttt{)}$: Elemente werden aus fester Menge entnommen $\mapsto$ Bitvektor

# Datentypverfeinerung für $\texttt{Costas}_{aux}$

$$
\begin{array}{l}
\texttt{FUNCTION Costas}_{aux}\texttt{(n,s,pool,ssize:}\mathbb{Z}\times\texttt{Seq(}\mathbb{Z}\texttt{)}\times\texttt{Set(}\mathbb{Z}\texttt{)}\times\mathbb{Z}\texttt{)}\\
\quad\texttt{WHERE n}\geq\texttt{1} \wedge \texttt{range(s)}\subseteq\{\texttt{1..n}\} \wedge \texttt{nodups(s)} \wedge \forall\texttt{j<|s|.nodups(dtrow(s,j))}\\
\qquad\qquad\qquad \wedge \texttt{ pool = }\{\texttt{1..n}\}\backslash\texttt{range(s)} \wedge \texttt{ssize=|s|}\\
\quad\texttt{RETURNS } \{\texttt{p:Seq(}\mathbb{Z}\texttt{)} \mid \texttt{perm(p,}\{\texttt{1..n}\}\texttt{)} \wedge \texttt{s}\sqsubseteq\texttt{p} \wedge \forall\texttt{j<|p|.nodups(dtrow(p,j))}\}\\
\equiv\quad \texttt{if pool=}\emptyset\texttt{ then } \{\texttt{s}\}\\
\qquad\qquad\qquad \texttt{else } \bigcup\{\texttt{Costas}_{gs}\texttt{(n,s}\cdot\texttt{i,pool}\backslash\{\texttt{i}\}\texttt{,ssize+1)} \mid \texttt{i}\in\texttt{pool}\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\,\forall\texttt{j<ssize. s}_{\texttt{ssize+1-j}}\texttt{ -i} \notin \texttt{dtrow(s,j)}\}
\end{array}
$$

---

$\texttt{n}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

$\texttt{s}$: $\texttt{Seq(}\mathbb{Z}\texttt{)}$, Elemente werden hinten angehängt $\mapsto$ umgekehrt verkettete Liste

$\texttt{pool}$: $\texttt{Set(}\mathbb{Z}\texttt{)}$: Elemente werden aus fester Menge entnommen $\mapsto$ Bitvektor

$\texttt{ssize}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

# DATENTYPVERFEINERUNG FÜR $\texttt{Costas}_{aux}$

```
FUNCTION Costas_aux(n,s,pool,ssize:ℤ×Seq(ℤ)×Set(ℤ)×ℤ)
  WHERE n≥1 ∧ range(s)⊆{1..n} ∧ nodups(s) ∧ ∀j<|s|.nodups(dtrow(s,j))
              ∧ pool = {1..n}\range(s) ∧ ssize=|s|
  RETURNS {p:Seq(ℤ)|perm(p,{1..n})∧ s⊑p∧ ∀j<|p|.nodups(dtrow(p,j))}
≡   if pool=∅ then {s}
             else ⋃{Costas_gs(n,s·i,pool\{i},ssize+1)|i∈pool
                                       ∧∀j<ssize. s_(ssize+1−j) -i ∉ dtrow(s,j)}
```

---

$\texttt{n}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen

$\texttt{s}$: $\texttt{Seq}(\mathbb{Z})$, Elemente werden hinten angehängt $\mapsto$ umgekehrt verkettete Liste

$\texttt{pool}$: $\texttt{Set}(\mathbb{Z})$: Elemente werden aus fester Menge entnommen $\mapsto$ Bitvektor

$\texttt{ssize}$: $\mathbb{Z}$ $\mapsto$ Standardimplementierung positiver ganzen Zahlen