

Rekursionspräzise Intervallanalysen

Dirk Richter, Wolf Zimmermann

15. Kolloquium Programmiersprachen und Grundlagen der
Programmierung
13.10.2009

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III, Institut für Informatik,
Lehrstuhl Softwaretechnik und Programmiersprachen,
<http://swt.informatik.uni-halle.de/>

- Ziel: Programmverhalten präzise vorhersagen

z.B. Schätzung erwarteter Laufzeit

z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)

→ präzisere Analysen \Rightarrow weniger Fehlalarme

- hier: Variablenwerte statisch vorhersagen (bei Rekursion)

- Problem: unentscheidbar (Halteproblem¹)

→ Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

- Ziel: Programmverhalten präzise vorhersagen
- z.B. **Schätzung erwarteter Laufzeit**
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen ⇒ weniger Fehlalarme
- hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
 - Problem: unentscheidbar (Halteproblem¹)
- Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

- Ziel: Programmverhalten präzise vorhersagen
- z.B. Schätzung erwarteter Laufzeit
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen ⇒ weniger Fehlalarme
- hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
 - Problem: unentscheidbar (Halteproblem¹)
- Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

- Ziel: Programmverhalten präzise vorhersagen
- z.B. Schätzung erwarteter Laufzeit
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen ⇒ weniger Fehlalarme
 - hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
 - Problem: unentscheidbar (Halteproblem¹)
- Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

- Ziel: Programmverhalten präzise vorhersagen
- z.B. Schätzung erwarteter Laufzeit
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen \Rightarrow weniger Fehlalarme
- hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
- Problem: unentscheidbar (Halteproblem¹)
- Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

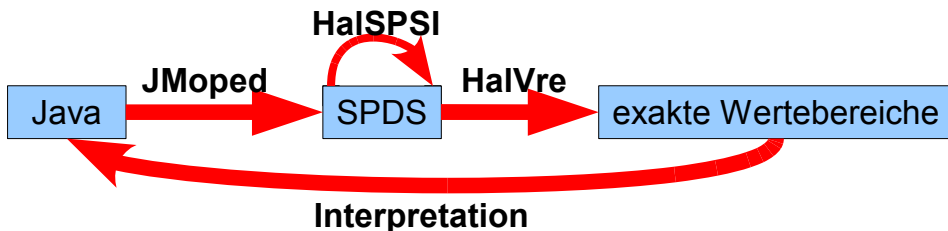
- Ziel: Programmverhalten präzise vorhersagen
- z.B. Schätzung erwarteter Laufzeit
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen ⇒ weniger Fehlalarme
- hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
 - Problem: **unentscheidbar** (Halteproblem¹)
- Abstraktion zu Rekursionsmodellen

¹formulierbar in vielen Hochsprachen

- Ziel: Programmverhalten präzise vorhersagen
- z.B. Schätzung erwarteter Laufzeit
- z.B. Fehler bei Arrayzugriff (ArrayOutOfBounds)
- präzisere Analysen \Rightarrow weniger Fehlalarme
- hier: Variablenwerte **statisch** vorhersagen (bei Rekursion)
 - Problem: unentscheidbar (Halteproblem¹)
- **Abstraktion zu Rekursionsmodellen**

¹formulierbar in vielen Hochsprachen

Idee: Rekursionsmodelle



Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

P ... Zustände

Γ ... Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*)$... Transitionen

- Kellerautomat ohne Eingabe
- (p, w) Konfiguration gdw. $p \in P, w \in \Gamma^*$
- (p, a) Kopf der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) Konfiguration gdw. $p \in P, w \in \Gamma^*$
- (p, a) Kopf der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kellularphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) Konfiguration gdw. $p \in P, w \in \Gamma^*$
- (p, a) Kopf der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kellularphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) Konfiguration gdw. $p \in P, w \in \Gamma^*$
- (p, a) Kopf der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- **Kellerautomat ohne Eingabe**
- (p, w) Konfiguration gdw. $p \in P, w \in \Gamma^*$
- (p, a) Kopf der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) **Konfiguration** gdw. $p \in P, w \in \Gamma^*$
- (p, a) **Kopf** der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) **Konfiguration** gdw. $p \in P, w \in \Gamma^*$
- (p, a) **Kopf** der Konfiguration $(p, aw), a \in \Gamma$
- symbolisch \rightarrow Rekursionsmodell

Rekursionsmodelle (SPDS)

Kellersystem (Pushdown System)

$\mathcal{P} = (P, \Gamma, \hookrightarrow)$

$P \dots$ Zustände

$\Gamma \dots$ Kelleralphabet

$\hookrightarrow \subseteq (P \times \Gamma) \times (P \times \Gamma^*) \dots$ Transitionen

- Kellerautomat ohne Eingabe
- (p, w) **Konfiguration** gdw. $p \in P, w \in \Gamma^*$
- (p, a) **Kopf** der Konfiguration $(p, aw), a \in \Gamma$
- **symbolisch** \rightarrow **Rekursionsmodell**

Neue Syntax: Remopla (Fragment)

```
void a(){
  int i=10;
  b(i);
}
```

```
module void a_a_V() {
  int v0(10);
  int s0(10);
  a_a_V:  s0=10;
  a_a_V2: v0=s0;
  a_a_V3: s0=v0;
  a_a_V4: a_b_I_V(s0);
  a_a_V7: return; }
```

```
void b(int i){
  i--;
  if (i>0)
    b(i);
}
```

```
module void a_b_I_V(int v0(10)) {
  int s0(10);
  a_b_I_V:  v0=v0-1;
  a_b_I_V3: s0=v0;
  a_b_I_V4:  if
    :: s0<=0 -> goto a_b_I_V11;
    :: else -> skip;
  fi;
  a_b_I_V7:  s0=v0;
  a_b_I_V8:  a_b_I_V(s0);
  a_b_I_V11: return;}
```

Neue Syntax: Remopla (Fragment)

```
void a(){
  int i=10;
  b(i);
}
```

```
module void a_a_V() {
  int v0(10);
  int s0(10);
  a_a_V: s0=10;
  a_a_V2: v0=s0;
  a_a_V3: s0=v0;
  a_a_V4: a_b_I_V(s0);
  a_a_V7: return; }
```

```
void b(int i){
  i--;
  if (i>0)
    b(i);
}
```

```
module void a_b_I_V(int v0(10)) {
  int s0(10);
  a_b_I_V: v0=v0-1;
  a_b_I_V3: s0=v0;
  a_b_I_V4: if
    :: s0<=0 -> goto a_b_I_V11;
    :: else -> skip;
  fi;
  a_b_I_V7: s0=v0;
  a_b_I_V8: a_b_I_V(s0);
  a_b_I_V11: return;}
```

Neue Syntax: Remopla (Fragment)

```
void a(){
  int i=10;
  b(i);
}
```

```
module void a_a_V() {
  int v0(10);
  int s0(10);
  a_a_V: s0=10;
  a_a_V2: v0=s0;
  a_a_V3: s0=v0;
  a_a_V4: a_b_I_V(s0);
  a_a_V7: return; }
```

```
void b(int i){
  i--;
  if (i>0)
    b(i);
}
```

```
module void a_b_I_V(int v0(10)) {
  int s0(10);
  a_b_I_V: v0=v0-1;
  a_b_I_V3: s0=v0;
  a_b_I_V4: if
    :: s0<=0 -> goto a_b_I_V11;
    :: else -> skip;
  fi;
  a_b_I_V7: s0=v0;
  a_b_I_V8: a_b_I_V(s0);
  a_b_I_V11: return;}
```

Neue Syntax: Remopla (Fragment)

```
void a(){
  int i=10;
  b(i);
}
```

```
void b(int i){
  i--;
  if (i>0)
    b(i);
}
```

```
module void a_a_V() {
  int v0(10);
  int s0(10);
  a_a_V:  s0=10;
  a_a_V2: v0=s0;
  a_a_V3: s0=v0;
  a_a_V4: a_b_I_V(s0);
  a_a_V7: return; }
```

```
module void a_b_I_V(int v0(10)) {
  int s0(10);
  a_b_I_V:  v0=v0-1;
  a_b_I_V3:  s0=v0;
  a_b_I_V4:  if
    :: s0<=0 -> goto a_b_I_V11;
    :: else -> skip;
  fi;
  a_b_I_V7:  s0=v0;
  a_b_I_V8:  a_b_I_V(s0);
  a_b_I_V11: return;}
```

Intervallanalyse

realisierbar

Variablenbelegung $\Phi^q : \text{Vars} \rightarrow \mathbb{Z}$ **realisierbar**

$\Leftrightarrow \exists$ Transitionenfolge von Startkonfiguration nach q (**Lauf**)

Intervallanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: $[a, b] \subset \mathbb{Z}$ mit $\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow a \leq \Phi^q(x) \leq b)$

- $[a] := [a, a]$

Intervallanalyse

realisierbar

Variablenbelegung $\Phi^q : \text{Vars} \rightarrow \mathbb{Z}$ **realisierbar**

$\Leftrightarrow \exists$ Transitionenfolge von Startkonfiguration nach q (**Lauf**)

Intervallanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: $[a, b] \subset \mathbb{Z}$ mit $\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow a \leq \Phi^q(x) \leq b)$

- $[a] := [a, a]$

Intervallanalyse

realisierbar

Variablenbelegung $\Phi^q : \text{Vars} \rightarrow \mathbb{Z}$ **realisierbar**

$\Leftrightarrow \exists$ Transitionenfolge von Startkonfiguration nach q (**Lauf**)

Intervallanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: $[a, b] \subset \mathbb{Z}$ mit $\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow a \leq \Phi^q(x) \leq b)$

- $[a] := [a, a]$

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;  
  a_a_V2: v0=s0;  
  a_a_V3: s0=v0;  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  

```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);  
  a_b_I_V:  v0=v0-1;  
  a_b_I_V3:  s0=v0;  
  a_b_I_V4:  if  
             :: s0<=0 -> goto a_b_I_V11;  
             :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return; }  

```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;      # s0:[10]  
  a_a_V2: v0=s0;  
  a_a_V3: s0=v0;  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);  
  a_b_I_V:  v0=v0-1;  
  a_b_I_V3:  s0=v0;  
  a_b_I_V4:  if  
             :: s0<=0 -> goto a_b_I_V11;  
             :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V: s0=10;      # s0:[10]  
  a_a_V2: v0=s0;    # v0:[10]  
  a_a_V3: s0=v0;  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);  
  a_b_I_V: v0=v0-1;  
  a_b_I_V3: s0=v0;  
  a_b_I_V4: if  
    :: s0<=0 -> goto a_b_I_V11;  
    :: else -> skip;  
  fi;  
  a_b_I_V7: s0=v0;  
  a_b_I_V8: a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;      # s0:[10]  
  a_a_V2: v0=s0;     # v0:[10]  
  a_a_V3: s0=v0;     # s0:[10]  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
# s0:[10]
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);  
  a_b_I_V:  v0=v0-1;  
  a_b_I_V3:  s0=v0;  
  a_b_I_V4:  if  
             :: s0<=0 -> goto a_b_I_V11;  
             :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return;}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;      # s0:[10]  
  a_a_V2: v0=s0;     # v0:[10]  
  a_a_V3: s0=v0;     # s0:[10]  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);          # v0:[1,10]  
  a_b_I_V:  v0=v0-1;  
  a_b_I_V3:  s0=v0;  
  a_b_I_V4:  if  
             :: s0<=0 -> goto a_b_I_V11;  
             :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;      # s0:[10]  
  a_a_V2: v0=s0;     # v0:[10]  
  a_a_V3: s0=v0;     # s0:[10]  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);          # v0:[1,10]  
  a_b_I_V:  v0=v0-1;   # v0:[0,9]  
  a_b_I_V3:  s0=v0;  
  a_b_I_V4:  if  
    :: s0<=0 -> goto a_b_I_V11;  
    :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V:  s0=10;      # s0:[10]  
  a_a_V2: v0=s0;      # v0:[10]  
  a_a_V3: s0=v0;      # s0:[10]  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);          # v0:[1,10]  
  a_b_I_V:  v0=v0-1;   # v0:[0,9]  
  a_b_I_V3:  s0=v0;    # s0:[0,9]  
  a_b_I_V4:  if  
    :: s0<=0 -> goto a_b_I_V11;  
    :: else -> skip;  
  fi;  
  a_b_I_V7:  s0=v0;  
  a_b_I_V8:  a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```

Intervallanalyse - Beispiel

```
module void a_a_V() {  
  int v0(10);  
  int s0(10);  
  a_a_V: s0=10;      # s0:[10]  
  a_a_V2: v0=s0;    # v0:[10]  
  a_a_V3: s0=v0;    # s0:[10]  
  a_a_V4: a_b_I_V(s0);  
  a_a_V7: return; }  
}
```

```
module void a_b_I_V(int v0(10)) {  
  int s0(10);          # v0:[1,10]  
  a_b_I_V: v0=v0-1;   # v0:[0,9]  
  a_b_I_V3: s0=v0;    # s0:[0,9]  
  a_b_I_V4: if  
    :: s0<=0 -> goto a_b_I_V11;  
    :: else -> skip;  
  fi;  
  a_b_I_V7: s0=v0;    # s0:[1,9]  
  a_b_I_V8: a_b_I_V(s0);  
  a_b_I_V11: return; }  
}
```


Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

\Rightarrow keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- entscheidbar @ Rekursionsmodelle

Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

\Rightarrow keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- entscheidbar @ Rekursionsmodelle

Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

\Rightarrow keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- entscheidbar @ Rekursionsmodelle

Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

\Rightarrow keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- entscheidbar @ Rekursionsmodelle

Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

⇒ keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- entscheidbar @ Rekursionsmodelle

Wertebereichsanalyse

Intervallmenge (disjunkt)

$\{[a_1, b_1], [a_2, b_2], \dots\}$ Intervallmenge, $a_i \leq b_i < a_{i+1}$

Wertebereichsanalyse (konservativ)

Geg: Variable x an Marke q .

Ges: Intervallmenge $\{[a_1, b_1], [a_2, b_2], \dots\}$ mit

$$\forall \Phi^q : (\Phi^q \text{ realisierbar} \Rightarrow \exists i : a_i \leq \Phi^q(x) \leq b_i)$$

exakte Wertebereichsanalyse

$\forall i : \forall v \in [a_i, b_i] : \exists \Phi^q : \Phi^q(x) = v \wedge \Phi^q \text{ realisierbar}$

⇒ keine Approximation (analog exaktes Intervall $[a_1, b_n]$)

- unentscheidbar @ viele Hochsprachen
- **entscheidbar** @ **Rekursionsmodelle**

exakte Wertbereichsanalyse für Rekursionsmodelle

Verfahren:

1 **Esparza: Rekursionsmodell \rightarrow Post*-Automat**

\rightarrow Beschreibt Menge aller erreichbaren Konfigurationen

! unendlich

2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q

! endlich

3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

exakte Wertebereichsanalyse für Rekursionsmodelle

Verfahren:

- 1 Esparza: Rekursionsmodell \rightarrow Post*-Automat
 \rightarrow Beschreibt Menge aller erreichbaren Konfigurationen
! unendlich
- 2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q
! endlich
- 3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

exakte Wertebereichsanalyse für Rekursionsmodelle

Verfahren:

- 1 Esparza: Rekursionsmodell \rightarrow Post*-Automat
 \rightarrow Beschreibt Menge aller erreichbaren Konfigurationen
! **unendlich**
- 2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q
! endlich
- 3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

exakte Wertebereichsanalyse für Rekursionsmodelle

Verfahren:

- 1 Esparza: Rekursionsmodell \rightarrow Post*-Automat
 \rightarrow Beschreibt Menge aller erreichbaren Konfigurationen
! unendlich
- 2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q
! endlich
- 3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

exakte Wertebereichsanalyse für Rekursionsmodelle

Verfahren:

- 1 Esparza: Rekursionsmodell \rightarrow Post*-Automat
 \rightarrow Beschreibt Menge aller erreichbaren Konfigurationen
! unendlich
- 2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q
! endlich
- 3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

exakte Wertebereichsanalyse für Rekursionsmodelle

Verfahren:

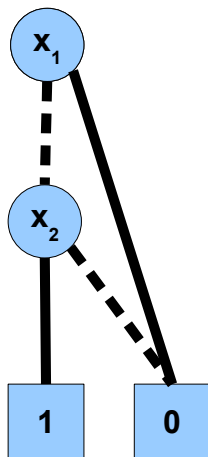
- 1 Esparza: Rekursionsmodell \rightarrow Post*-Automat
 \rightarrow Beschreibt Menge aller erreichbaren Konfigurationen
! unendlich
- 2 Extraktion char. Funktion Menge aller erreichbaren Köpfe f^q
! endlich
- 3 Bestimmung Intervallmengen direkt aus ROBDD² von f^q

²reduziert geordnetes Entscheidungsdiagramm

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

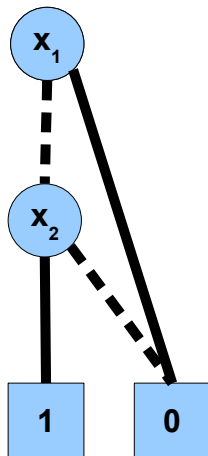
$$f^q(x) = \sum_{i=1}^4 x_i \cdot 2^{i-1} \Rightarrow f^q(x_1, x_2)$$

$$f^q(x_1, x_2) = \sum_{i=1}^4 x_i \cdot 2^{i-1} = x_1 + 2x_2$$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



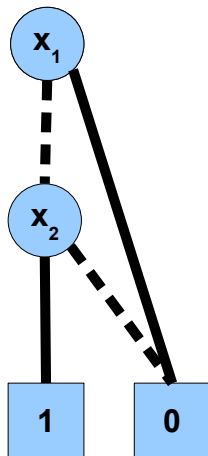
x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$
- ON-Menge (1-Pfade) $ON := \{01 **\}$
- exaktes Intervall $[\min(ON), \max(ON)]$
- min: * \mapsto 0 max: * \mapsto 1
- $[4, 7]$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



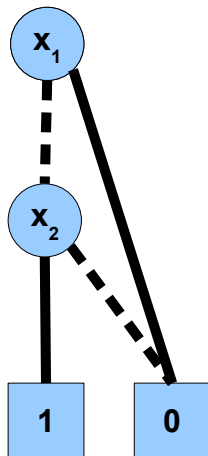
x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$
- ON-Menge (1-Pfade) $ON := \{01 **\}$
- exaktes Intervall $[\min(ON), \max(ON)]$
 $\rightarrow \min: * \mapsto 0$ $\max: * \mapsto 1$
 $\Rightarrow [4, 7]$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$

- ON-Menge (1-Pfade) $ON := \{01 **\}$

- exaktes Intervall $[\min(ON), \max(ON)]$

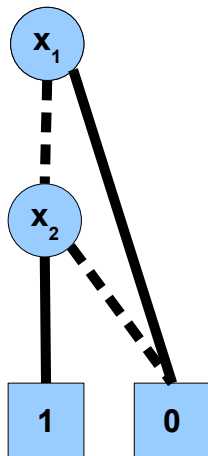
\rightarrow min: $* \mapsto 0$ max: $* \mapsto 1$

$\Rightarrow [4, 7]$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$

- **ON-Menge (1-Pfade) $ON := \{01**\}$**

- exaktes Intervall $[\min(ON), \max(ON)]$

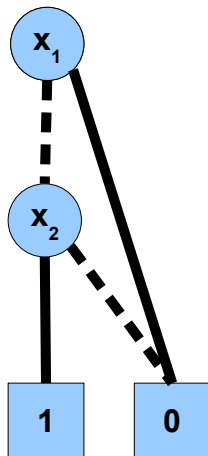
\rightarrow min: $* \mapsto 0$ max: $* \mapsto 1$

$\Rightarrow [4, 7]$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$
- ON-Menge (1-Pfade) $ON := \{01 **\}$
- **exaktes Intervall** $[\min(ON), \max(ON)]$

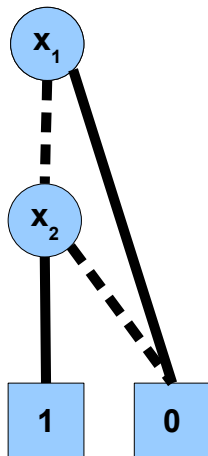
\rightarrow min: $* \mapsto 0$ max: $* \mapsto 1$

$\Rightarrow [4, 7]$

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$
- ON-Menge (1-Pfade) $ON := \{01 **\}$
- exaktes Intervall $[\min(ON), \max(ON)]$

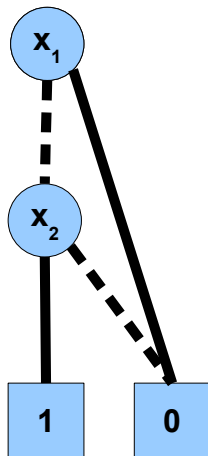
→ min: * \mapsto 0 max: * \mapsto 1

⇒ [4,7]

Beispiel zur Bestimmung Intervallmengen

Geg.: char. Fkt. f^q aller erreichbaren Köpfe an Marke q

$$f^q : \{0, 1\}^4 \rightarrow \{0, 1\}$$



x_1	x_2	x_3	x_4	f^q
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

- $\exists \Phi^q : \Phi^q(x) = \sum x_i \cdot 2^{4-i} \Leftrightarrow f^q(x_1..x_4)$
 - ON-Menge (1-Pfade) $ON := \{01 **\}$
 - exaktes Intervall $[\min(ON), \max(ON)]$
- min: * \mapsto 0 max: * \mapsto 1
- ⇒ **[4,7]**

Aufwand exakter Wertebereichsanalyse

Komplexität exakter Wertebereichsanalyse @ SPDS

- exakte Wertebereichsanalyse \supseteq Erreichbarkeit^a
- plausibel: löst Erreichbarkeit für jede Marke
- polynomial in Rekursionsmodellgröße

^aErreichbarkeitstest = einfache Form Modellprüfung

Aufwand exakter Wertebereichsanalyse

Komplexität exakter Wertebereichsanalyse @ SPDS

- exakte Wertebereichsanalyse \asymp Erreichbarkeit^a

→ plausibel: löst Erreichbarkeit für jede Marke

- polynomial in Rekursionsmodellgröße

^aErreichbarkeitstest = einfache Form Modellprüfung

Aufwand exakter Wertebereichsanalyse

Komplexität exakter Wertebereichsanalyse @ SPDS

- exakte Wertebereichsanalyse \cong Erreichbarkeit^a
- plausibel: löst Erreichbarkeit für jede Marke
- polynomial in Rekursionsmodellgröße

^aErreichbarkeitstest = einfache Form Modellprüfung

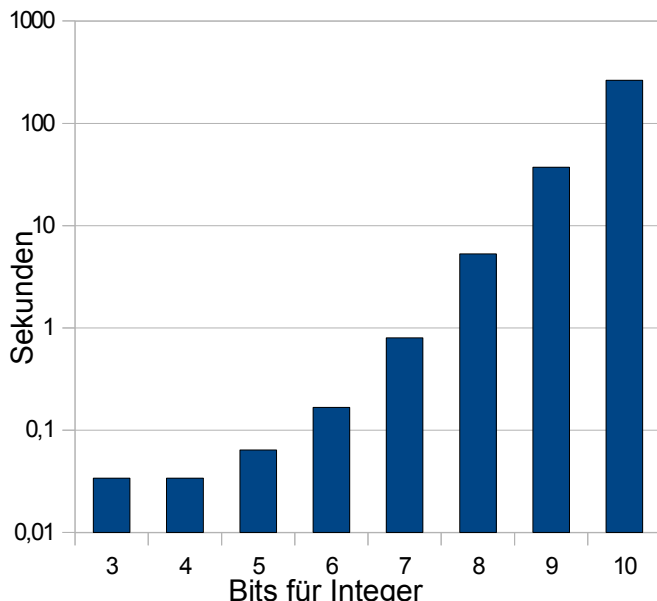
Aufwand exakter Wertebereichsanalyse

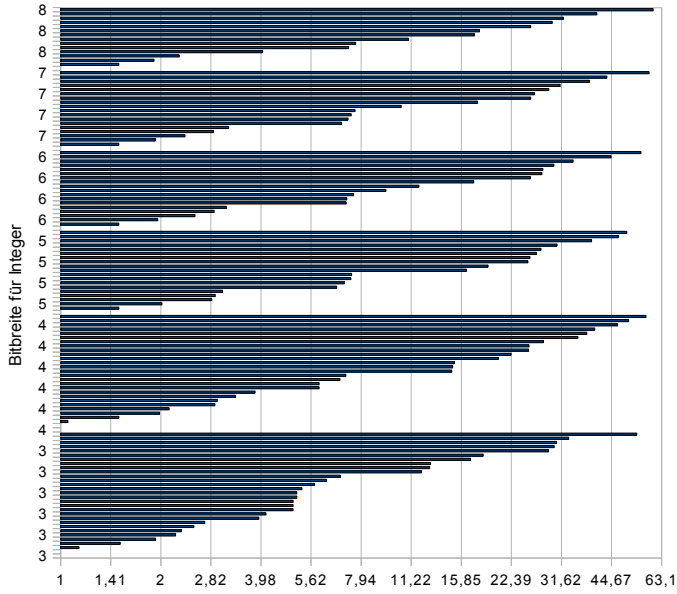
Komplexität exakter Wertebereichsanalyse @ SPDS

- exakte Wertebereichsanalyse \supseteq Erreichbarkeit^a
- plausibel: löst Erreichbarkeit für jede Marke
- **polynomial in Rekursionsmodellgröße**

^aErreichbarkeitstest = einfache Form Modellprüfung

Aufwand exakter Wertebereichsanalyse (Fibonacci rekursiv)



Approximationsfehler konservativer Wertebereichsanalyse (\emptyset 11.65%)

Zusammenfassung

- rekursionspräzise Intervallanalysen

- Problem: Unentscheidbarkeit (Halteproblem)

- 1 Abstraktion zu Rekursionsmodellen

- 2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

- 3 Rückinterpretation

- Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ Modellkompression (HalSPSI)

³HalVre

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: **Unentscheidbarkeit** (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

• Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ Modellkompression (HalSPSI)

³HalVre

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: Unentscheidbarkeit (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

- Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ Modellkompression (HalSPSI)

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: Unentscheidbarkeit (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

- Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ Modellkompression (HalSPSI)

³HalVre

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: Unentscheidbarkeit (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

- Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ Modellkompression (HalSPSI)

³HalVre

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: Unentscheidbarkeit (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

- Ausblick: **praktische Durchführbarkeit** (Konfigurationenraum)

→ Modellkompression (HalSPSI)

³HalVre

Zusammenfassung

- rekursionspräzise Intervallanalysen
- Problem: Unentscheidbarkeit (Halteproblem)

1 Abstraktion zu Rekursionsmodellen

2 Identifikation exakter Wertebereiche in Rekursionsmodellen³

3 Rückinterpretation

- Ausblick: praktische Durchführbarkeit (Konfigurationenraum)

→ **Modellkompression (HalSPSI)**

³HalVre

Fragen?

Gesucht: freie Tools für konservative Intervallanalysen in Java.

Overview of reduction techniques

non-parasitic (remains control/dataflow structure)

- reorganisation of variables
- control flow slicing
- SMT reduction
- constant propagation/-folding
- more trivial transformations

parasitic (changes control/dataflow structure)

- elimination of variables using equivalence information
- flow direction deciding reduction
- stutter reduction
- range reduction using interval analysis