

Theorie der uninterpretierten Funktionen

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

STEPHAN FALKE — INSTITUT FÜR THEORETISCHE INFORMATIK (ITI)

1. Wieso uninterpretierte Funktionen?
2. Entscheidungsverfahren für uninterpretierte Funktionen
3. Gleichheitslogik
4. Von uninterpretierten Funktionen zu Gleichheitslogik
5. Entscheidungsverfahren für Gleichheitslogik

Theorie (UF)

■ $\Sigma = \{=, a, b, c, \dots, f, g, h, \dots\}$

■ Axiome Ax:

1. $\forall x. x = x$

(Reflexivität)

2. $\forall x, y. x = y \rightarrow y = x$

(Symmetrie)

3. $\forall x, y, z. x = y \wedge y = z \rightarrow x = z$

(Transitivität)

4. für jedes n -stellige Funktionssymbol $f \in \Sigma$ mit $n > 0$:

$$\forall x_1, \dots, x_n, y_1, \dots, y_n. \bigwedge_{i=1}^n x_i = y_i \rightarrow f(x_1, \dots, x_n) = f(y_1, \dots, y_n)$$

(Kongruenz)

■ (Kongruenz) ist ein **Axiomschema**

- Uninterpretierte Funktionen **abstrahieren** die Semantik von Funktionen
- Wenn eine Formel UF-unerfüllbar ist so ist sie auch T -unerfüllbar für jede Theorie T die UF erweitert

$$\begin{aligned} & \varphi \text{ ist UF-unerfüllbar} \\ \iff & A_{X_{UF}} \cup \{\varphi\} \text{ ist unerfüllbar} \\ \implies & A_{X_T} \cup \{\varphi\} \text{ ist unerfüllbar} \\ \iff & \varphi \text{ ist } T\text{-unerfüllbar} \end{aligned}$$

- **Vorteil:** UF-Solver können **sehr effizient** implementiert werden, T -Solver können hohe Komplexität haben
- Insbesondere auch interessant wenn T unentscheidbar ist

Beispiel

Sind die folgenden Funktionen äquivalent?

```
uint fma1(uint x, uint y, uint z) {  
    return x + (y * z);  
}
```

```
uint fma2(uint x, uint y, uint z) {  
    uint p = y * z;  
    uint s = x + p;  
    return s;  
}
```

Ist die folgende Formel gültig in Peano Arithmetik?

$$(p = y * z) \wedge (s = x + p) \rightarrow (s = x + (y * z))$$

Ist die folgende Formel unerfüllbar in Peano Arithmetik?

$$(p = y * z) \wedge (s = x + p) \wedge \neg(s = x + (y * z))$$

Ist die folgende Formel UF-unerfüllbar?

$$(p = \text{mul}(y, z)) \wedge (s = \text{add}(x, p)) \wedge \neg(s = \text{add}(x, \text{mul}(y, z)))$$

- Eine binäre Relation \sim auf einer Menge M ist eine **Äquivalenzrelation** falls \sim reflexiv, transitiv und symmetrisch ist
- Der **Äquivalenzabschluß** einer binären Relation R auf einer Menge M ist die “kleinste” Äquivalenzrelation auf M die R enthält
- Eine Äquivalenzrelation \sim auf einer Menge M definiert eine Partition M/\sim bestehend aus **Äquivalenzklassen** $[s]_{\sim} := \{t \in M \mid t \sim s\}$
- Eine Äquivalenzrelation \sim auf einer Menge M ist eine **Kongruenzrelation** falls

$$\bigwedge_{i=1}^n x_i \sim y_i \wedge f(x_1, \dots, x_n) \in M \wedge f(y_1, \dots, y_n) \in M \\ \rightarrow \\ f(x_1, \dots, x_n) \sim f(y_1, \dots, y_n)$$

für alle n -stelligen Funktionssymbole f mit $n > 0$ gilt

- Der **Kongruenzabschluß** einer binären Relation R auf einer Menge M ist die “kleinste” Kongruenzrelation auf M die R enthält

- Sei φ eine Konjunktion von UF-Literalen ($s = t$ oder $\neg(s = t)$)
- Die **Teiltermmenge** S_φ von φ enthält alle Teilterme die in φ auftreten
-

Satz

φ ist UF-erfüllbar gdw. es eine Kongruenzrelation \sim auf S_φ gibt s.d.

- $s \sim t$ für alle UF-Literale $s = t$ in φ
- $s \not\sim t$ für alle UF-Literale $\neg(s = t)$ in φ

Satz

φ ist UF-erfüllbar gdw. es eine Kongruenzrelation \sim auf S_φ gibt s.d.

- $s \sim t$ für alle UF-Literale $s = t$ in φ
- $s \not\sim t$ für alle UF-Literale $\neg(s = t)$ in φ

“ \Leftarrow ” Betrachte die Struktur \mathcal{M} :

- Universum S_φ / \sim
- $=^{\mathcal{M}} := \sim$
- $f^{\mathcal{M}}([s_1]_{\sim}, \dots, [s_n]_{\sim}) := \begin{cases} [f(s_1, \dots, s_n)]_{\sim} & \text{falls } f(s_1, \dots, s_n) \in S_\varphi \\ \text{beliebig} & \text{falls } f(s_1, \dots, s_n) \notin S_\varphi \end{cases}$

Dann ist \mathcal{M} ein UF-Modell von φ

“ \Rightarrow ” Hausaufgabe



Algorithmus

- Eingabe: $\varphi : s_1 = t_1 \wedge \dots \wedge s_n = t_n \wedge \neg(s'_1 = t'_1) \wedge \dots \wedge \neg(s'_m = t'_m)$
- Ausgabe: UF-konsistent oder UF-inkonsistent
- Schritte:
 1. Berechne den Kongruenzabschluß \sim von

$$\{s_1 = t_1, \dots, s_n = t_n\}$$

auf der Teiltermmenge S_φ

2. Falls $s'_i \sim t'_i$ für ein i , gib UF-inkonsistent aus
3. Andernfalls, gib UF-konsistent aus

Der Kongruenzabschluß \sim von

$$\{s_1 = t_1, \dots, s_n = t_n\}$$

auf der Teilmengemenge S_φ kann wie folgt berechnet werden:

1. Sei $L = \{\{s_i, t_i\} \mid 1 \leq i \leq n\} \cup \{\{s\} \mid s \in S_\varphi \setminus \{s_1, \dots, s_n, t_1, \dots, t_n\}\}$
2. Vereinige alle Mengen $M_1, M_2 \in L$ mit $M_1 \cap M_2 \neq \emptyset$
3. Sei $K = L \cup \left\{ \{f(s_1, \dots, s_n), f(t_1, \dots, t_n)\} \mid \begin{array}{l} \text{ex. } M_i \in L \text{ mit } s_i, t_i \in M_i \\ f(s_1, \dots, s_n) \in S_\varphi \\ f(t_1, \dots, t_n) \in S_\varphi \end{array} \right\}$
4. Vereinige alle Mengen $M_1, M_2 \in K$ mit $M_1 \cap M_2 \neq \emptyset$
5. Falls $K \neq L$ so setze $L = K$ und gehe zu 3., sonst höre auf

Konstruktion terminiert da S_φ endlich ist

Beispiel

- $\varphi : f(a, b) = a \wedge \neg(f(f(a, b), b) = a)$
- $S_\varphi = \{a, b, f(a, b), f(f(a, b), b)\}$
- Berechnung des Kongruenzabschluß \sim von

$$\{f(a, b) = a\}$$

auf der Teilmengemenge S_φ :

1. $\{ \{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\} \}$
2. $\{ \{a, f(a, b)\}, \{b\}, \{f(f(a, b), b)\} \} \cup \{ \{f(a, b), f(f(a, b), b)\} \}$
3. $\{ \{a, f(a, b), f(f(a, b), b)\}, \{b\}, \{f(f(a, b), b)\} \}$
4. $\{ \{a, f(a, b), f(f(a, b), b)\}, \{b\} \}$

Kongruenzabschluß \sim

- φ ist UF-inkonsistent da $f(f(a, b), b) \sim a$

- **Ersetzungssystem:** Menge R von Regeln $l \rightarrow r$
- $s \rightarrow_R t$ gdw. $s = C[l]$ und $t = C[r]$ für einen Kontext C
- **Ziel:** Konstruktion eines **Ersetzungssystems** R welches den Kongruenzabschluß beschreibt
 - R ist terminierend und konfluent
 - $s \sim t$ gdw. $s \downarrow_R = t \downarrow_R$
 - Gilt für **beliebige** s, t (nicht eingeschränkt auf die Teiltermmenge!)
- Zusätzlich zu Σ : disjunkte Menge U von Konstanten, geordnet durch \succ
- **C-Regel:** $c \rightarrow d$ für $c, d \in U$
- **D-Regel:** $f(c_1, \dots, c_n) \rightarrow c$ für $f \in \Sigma$ und $c_1, \dots, c_n, c \in U$
- R wird durch ein Inferenzsystem aus $\{s_1 = t_1, \dots, s_n = t_n\}$ erzeugt
 - Zustände: (K, E, R)
 - $K \subseteq U$
 - E Menge von Gleichheiten
 - R Menge von Regeln
 - Startzustand: $(\emptyset, \{s_1 = t_1, \dots, s_n = t_n\}, \emptyset)$

$$\text{Extension} \quad \frac{(K, E[t], R)}{(K \cup \{c\}, E[c], R \cup \{t \rightarrow c\})}$$

$t \rightarrow c$ ist eine D-Regel
und $c \in U \setminus K$

$$\text{Simplification} \quad \frac{(K, E[t], R \cup \{t \rightarrow c\})}{(K, E[c], R \cup \{t \rightarrow c\})}$$

$$\text{Orientation} \quad \frac{(K \cup \{c\}, E \cup \{t = c\}, R)}{(K \cup \{c\}, E, R \cup \{t \rightarrow c\})}$$

$t \in U$ und $t \succ c$ oder
 $t \rightarrow c$ ist eine D-Regel

$$\text{Deletion} \quad \frac{(K, E \cup \{t = t\}, R)}{(K, E, R)}$$

$$\text{Deduction} \quad \frac{(K, E, R \cup \{t \rightarrow c, t \rightarrow d\})}{(K, E \cup \{c = d\}, R \cup \{t \rightarrow d\})}$$

$$\text{Collapse} \quad \frac{(K, E, R \cup \{C[c] \rightarrow c', c \rightarrow d\})}{(K, E, R \cup \{C[d] \rightarrow c', c \rightarrow d\})} \quad C \neq \square$$

Beispiel

- Berechnung des abstrakten Kongruenzabschluß von

$$\{a = b, f(f(a)) = f(b)\}$$

- Anwendung der Inferenzregeln:

| K | E | R | Regel |
|----------------|-------------------------------|--|-----------------------------|
| \emptyset | $\{a = b, f(f(a)) = f(b)\}$ | \emptyset | Extension |
| $\{c_0\}$ | $\{c_0 = b, f(f(a)) = f(b)\}$ | $\{a \rightarrow c_0\}$ | Orientation |
| $\{c_0\}$ | $\{f(f(a)) = f(b)\}$ | $\{a \rightarrow c_0, b \rightarrow c_0\}$ | Simplification ² |
| $\{c_0\}$ | $\{f(f(c_0)) = f(c_0)\}$ | $\{a \rightarrow c_0, b \rightarrow c_0\}$ | Extension |
| $\{c_0, c_1\}$ | $\{f(c_1) = f(c_0)\}$ | $\{a \rightarrow c_0, b \rightarrow c_0, f(c_0) \rightarrow c_1\}$ | Simplification |
| $\{c_0, c_1\}$ | $\{f(c_1) = c_1\}$ | $\{a \rightarrow c_0, b \rightarrow c_0, f(c_0) \rightarrow c_1\}$ | Orientation |
| $\{c_0, c_1\}$ | \emptyset | $\{a \rightarrow c_0, b \rightarrow c_0, f(c_0) \rightarrow c_1, f(c_1) \rightarrow c_1\}$ | — |

- Gilt $f(f(f(a))) \sim f(b)$? Ja, da $f(f(f(a))) \downarrow_R = c_1 = f(b) \downarrow_R$
- Gilt $f(a) \sim a$? Nein, da $f(a) \downarrow_R = c_1 \neq c_0 = a \downarrow_R$

Satz

Alle Ableitungen $(\emptyset, E, \emptyset) \vdash (K_1, E_1, R_1) \vdash \dots$ sind endlich

Satz

Sei $(\emptyset, E, \emptyset) \vdash (K_1, E_1, R_1) \vdash \dots \vdash (K_n, E_n, R_n)$ eine maximale Ableitung

- $E_n = \emptyset$
- R_n beschreibt den Kongruenzabschluß von E

- Gleichheitslogik ist ein **Fragment** von UF
 - Formeln enthalten nur Variablen, **keine** komplizierteren Terme
- Alternativer Ansatz:

Theorie (Gleichheitslogik)

- $\Sigma = \{=\}$
- Axiome Ax :
 1. $\forall x. x = x$ (Reflexivität)
 2. $\forall x, y. x = y \rightarrow y = x$ (Symmetrie)
 3. $\forall x, y, z. x = y \wedge y = z \rightarrow x = z$ (Transitivität)

- Ziel: transformiere UF-Formel φ^{UF} in eine erfüllbarkeitsäquivalente Gleichheitslogik-Formel φ^{E}
- φ^{E} hat die Form $\text{flat}^{\text{E}} \wedge \text{FC}^{\text{E}}$
 - flat^{E} ersetzt maximale Teilterme in φ^{UF} durch Variablen
 - FC^{E} erzwingt Kongruenz

Beispiel

- UF-Formel φ^{UF} : $x = y \wedge \neg(f(x) = f(y))$

- flat^{E} : $x = y \wedge \neg(f_x = f_y)$

f_x und f_y sind neue Variablen

- FC^{E} : $x = y \rightarrow f_x = f_y$

“Falls x und y gleich sind so sind auch $f(x)$ und $f(y)$ gleich”

Algorithmus

- Eingabe: UF-Formal φ^{UF}
- Ausgabe: Erfüllbarkeitsäquivalente Gleichheitslogik-Formel φ^{E}
- Schritte:
 1. Nummeriere Teilterme der Form $f(\dots)$
 2. Sei F_i der i -te Teilterm der Form $f(\dots)$ und sei $\text{arg}_j(F_i)$ sein j -tes Argument
 3. Generiere flat^{E} aus φ^{UF} indem alle Atome der Form $F_i = G_j$ durch $\widehat{F}_i = \widehat{G}_j$ für neue Variablen $\widehat{F}_i, \widehat{G}_j$ ersetzt werden
 4. Generiere FC^{E} als Konjunktion aus Formeln der Form

$$\bigwedge_{j=1}^n \widehat{\text{arg}}_j(F_i) = \widehat{\text{arg}}_j(F_k) \rightarrow \widehat{F}_i = \widehat{F}_k$$

für alle $i < k$ wenn f die Stelligkeit n mit $n > 0$ hat

5. Gib $\text{flat}^{\text{E}} \wedge \text{FC}^{\text{E}}$ aus

Beispiel

■ φ^{UF} :

$$x = y \wedge \underbrace{\underbrace{\underbrace{f(f(\underbrace{g(x)}_{F_1}))}_{F_2}}_{F_3}}_{G_1} = \underbrace{\underbrace{\underbrace{f(f(\underbrace{g(y)}_{F_3}))}_{F_4}}_{F_3}}_{G_2}$$

■ flat^E:

$$x = y \wedge \neg(f_2 = f_4)$$

■ FC^E:

$$\begin{aligned} x = y &\rightarrow g_1 = g_2 \\ \wedge g_1 = f_1 &\rightarrow f_1 = f_2 \\ \wedge g_1 = g_2 &\rightarrow f_1 = f_3 \\ \wedge g_1 = f_3 &\rightarrow f_1 = f_4 \\ \wedge f_1 = g_2 &\rightarrow f_2 = f_3 \\ \wedge f_1 = f_3 &\rightarrow f_2 = f_4 \\ \wedge g_2 = f_3 &\rightarrow f_3 = f_4 \end{aligned}$$

Satz

φ ist Gleichheitslogik-erfüllbar gdw. es eine Äquivalenzrelation \sim auf $V(\varphi)$ gibt so dass

- $x \sim y$ für alle Literale $x = y$ in φ
- $x \not\sim y$ für alle Literale $\neg(x = y)$ in φ

Algorithmus

- Eingabe: $\varphi : x_1 = y_1 \wedge \dots \wedge x_n = y_n \wedge \neg(x'_1 = y'_1) \wedge \dots \wedge \neg(x'_m = y'_m)$
- Ausgabe: konsistent oder inkonsistent
- Schritte:
 1. Berechne den Äquivalenzabschluß \sim von

$$\{x_1 = y_1, \dots, x_n = y_n\}$$

auf $V(\varphi)$

2. Falls $x'_i \sim y'_i$ für ein i , gib inkonsistent aus
3. Andernfalls, gib konsistent aus

Berechnung des Äquivalenzabschluß

Der Äquivalenzabschluß \sim von

$$\{x_1 = y_1, \dots, x_n = y_n\}$$

auf $V(\varphi)$ kann wie folgt berechnet werden:

1. $L = \{\{x_i, y_i\} \mid 1 \leq i \leq n\} \cup \{\{x\} \mid x \in V(\varphi) \setminus \{x_1, \dots, x_n, y_1, \dots, y_n\}\}$
2. Vereinige alle Mengen $M_1, M_2 \in L$ mit $M_1 \cap M_2 \neq \emptyset$

Konstruktion terminiert da $V(\varphi)$ endlich ist

Beispiel

Der Äquivalenzabschluß von $\{x_1 = x_2, x_3 = x_4, x_4 = x_1\}$ auf den Variablen $\{x_1, x_2, x_3, x_4, x_5\}$ wird wie folgt berechnet:

1. $\{\{x_1, x_2\}, \{x_3, x_4\}, \{x_4, x_1\}, \{x_5\}\}$
2. $\{\{x_1, x_2, x_4\}, \{x_3, x_4\}, \{x_5\}\}$
3. $\{\{x_1, x_2, x_3, x_4\}, \{x_5\}\}$

- Idee: Stelle die Äquivalenzklassen als **Bäume** dar
- Wurzel eines Baums ist **Repräsentant** der Äquivalenzklasse
- Pseudocode:

```
function MakeSet(x)
    x.parent := x
```

```
function Find(x)
    if x.parent == x
        return x
    else
        return Find(x.parent)
```

```
function Union(x, y)
    xRoot := Find(x)
    yRoot := Find(y)
    xRoot.parent := yRoot
```

- x und y sind äquivalent gdw. $\text{Find}(x) == \text{Find}(y)$

Der Äquivalenzabschluß \sim von

$$\{x_1 = y_1, \dots, x_n = y_n\}$$

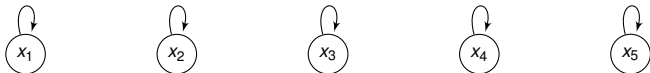
auf $V(\varphi)$ kann wie folgt berechnet werden:

1. führe $\text{MakeSet}(x)$ für alle $x \in V(\varphi)$ aus
2. führe $\text{Union}(x_i, y_i)$ für alle $1 \leq i \leq n$ aus

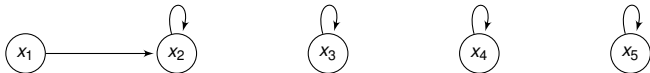
Beispiel

Der Äquivalenzabschluß von $\{x_1 = x_2, x_3 = x_4, x_4 = x_1\}$ auf $\{x_1, x_2, x_3, x_4, x_5\}$ wird wie folgt berechnet:

1. Nach Ausführung von $\text{MakeSet}(x)$ für alle $x \in \{x_1, x_2, x_3, x_4, x_5\}$:



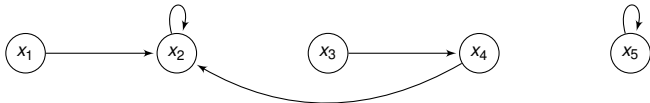
2. Nach Ausführung von $\text{Union}(x_1, x_2)$:



3. Nach Ausführung von $\text{Union}(x_3, x_4)$:



4. Nach Ausführung von $\text{Union}(x_4, x_1)$:



- Der Äquivalenzabschluß erzeugt eine **Partitionierung** der Variablen
- Für n Variablen hat jede Partitionierung höchstens n Elemente
- Formalisierung dieser Beobachtung:

Satz

Eine erfüllbare Gleichheitslogik-Formel mit n Variablen hat ein Modell mit höchstens n Elementen

- **Idee**: Ersetze Variable x durch $N := \lceil \log n \rceil$ **aussagenlogische** Variablen x_1, \dots, x_N ("bits")
- $x = y$ wird ersetzt durch $(x_1 \leftrightarrow y_1) \wedge \dots \wedge (x_N \leftrightarrow y_N)$
- $n \lceil \log n \rceil$ bits, Zustandsraum der Größe $2^{n \lceil \log n \rceil} \approx n^n$
- Verfeinerung des Ansatzes erzeugt Zustandsraum der Größe $\approx n!$

Beispiel

- Gleichheitslogik-Formel:

$$x_1 = x_2 \wedge [x_2 = x_3 \vee \neg(x_1 = x_4)] \wedge x_2 = x_4$$

- 4 Variablen, 2 bits pro Variable
- SAT-Formel:

$$\begin{aligned}
 & ((x_{1,1} \leftrightarrow x_{2,1}) \wedge (x_{1,2} \leftrightarrow x_{2,2})) \\
 \wedge & [((x_{2,1} \leftrightarrow x_{3,1}) \wedge (x_{2,2} \leftrightarrow x_{3,2})) \vee \neg((x_{1,1} \leftrightarrow x_{4,1}) \wedge (x_{1,2} \leftrightarrow x_{4,1}))] \\
 & \wedge ((x_{2,1} \leftrightarrow x_{4,1}) \wedge (x_{2,2} \leftrightarrow x_{4,2}))
 \end{aligned}$$