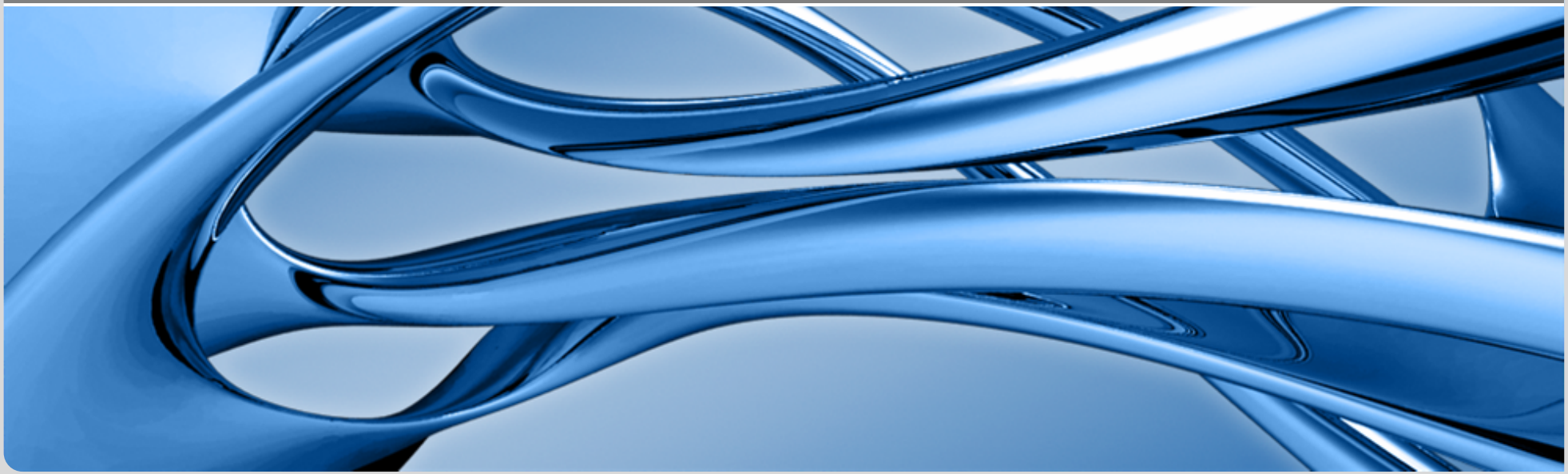
Karlsruher Institut für Technologie

Theorie der uninterpretierten Funktionen

Entscheidungsverfahren mit Anwendungen in der Softwareverifikation

STEPHAN FALKE — INSTITUT FÜR THEORETISCHE INFORMATIK (ITI)



Inhalt

Theorie der uninterpretierten Funktionen

Karlsruher Institut für Technologie

Theorie (UF)

- $\Sigma = \{=, a, b, c, \ldots, f, g, h, \ldots\}$
- 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 $n > 0$:
     $\forall x_1, \ldots, x_n, y_1, \ldots, y_n.\ \bigwedge_{i=1}^{n} x_i = y_i \rightarrow f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$
     (Kongruenz)
- (Kongruenz) ist ein Axiomschema

Benutzung von uninterpretierten Funktionen

- 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

$$
\begin{array}{rl}
 & \varphi \text{ ist UF-unerfüllbar} \\
\iff & Ax_{\mathrm{UF}} \cup \{\varphi\} \text{ ist unerfüllbar} \\
\implies & Ax_T \cup \{\varphi\} \text{ ist unerfüllbar} \\
\iff & \varphi \text{ ist } T\text{-unerfüllbar}
\end{array}
$$

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

Äquivalenz von Programmen

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?

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

Äquivalenz- und Kongruenzrelationen

- 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

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

  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

Kongruenzabschluß als T-Solver für UF

- 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

Beweis

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

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

"$\Rightarrow$" Hausaufgabe □

Kongruenzabschluß als T-Solver für UF

Algorithmus

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

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

     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

Berechnung des Kongruenzabschluß

Der Kongruenzabschluß $\sim$ von

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

auf der Teiltermmenge S_φ kann wie folgt berechnet werden:

1. Sei $L = \{\{s_i, t_i\} \mid 1 \leq i \leq n\} \cup \{\{s\} \mid s \in S_\varphi \setminus \{s_1, \ldots, s_n, t_1, \ldots, t_n\}\}$
2. Vereinige alle Mengen $M_1, M_2 \in L$ mit $M_1 \cap M_2 \neq \emptyset$
3. Sei $K = L \cup \{\{f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)\} \mid$ ex. $M_i \in L$ mit $s_i, t_i \in M_i$, $f(s_1, \ldots, s_n) \in S_\varphi$, $f(t_1, \ldots, t_n) \in S_\varphi \}$
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 Teiltermmenge 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 $f(f(a,b),b) \sim a$

Abstrakter Kongruenzabschluß

- Ersetzungssystem: Menge R von Regeln $l \to r$
- $s \to_R t$ 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: $c \to d$ für $c, d \in U$
- D-Regel: $f(c_1, \ldots, c_n) \to c$ für $f \in \Sigma$ und $c_1, \ldots, c_k, c \in U$
- R wird durch ein Inferenzsystem aus $\{s_1 = t_1, \ldots, s_n = t_n\}$ erzeugt
  - Zustände: (K, E, R)
    - $K \subseteq U$
    - E Menge von Gleichheiten
    - R Menge von Regeln
  - Startzustand: $(\emptyset, \{s_1 = t_1, \ldots, s_n = t_n\}, \emptyset)$

Inferenzregeln 1

Extension $\dfrac{(K,\ E[t],\ R)}{(K \cup \{c\},\ E[c],\ R \cup \{t \to c\})}$ $t \to c$ ist eine D-Regel und $c \in U \setminus K$

Simplification $\dfrac{(K,\ E[t],\ R \cup \{t \to c\})}{(K,\ E[c],\ R \cup \{t \to c\})}$

Orientation $\dfrac{(K \cup \{c\},\ E \cup \{t = c\},\ R)}{(K \cup \{c\},\ E,\ R \cup \{t \to c\})}$ $t \in U$ und $t \succ c$ oder $t \to c$ ist eine D-Regel

Inferenzregeln 2

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

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

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

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
$\{c_0\}$	$\{c_0 = b,\ f(f(a)) = f(b)\}$	$\{a \to c_0\}$	Orientation
$\{c_0\}$	$\{f(f(a)) = f(b)\}$	$\{a \to c_0,\ b \to c_0\}$	Simplification^2
$\{c_0\}$	$\{f(f(c_0)) = f(c_0)\}$	$\{a \to c_0,\ b \to c_0\}$	Extension
$\{c_0, c_1\}$	$\{f(c_1) = f(c_0)\}$	$\{a \to c_0,\ b \to c_0,\ f(c_0) \to c_1\}$	Simplification
$\{c_0, c_1\}$	$\{f(c_1) = c_1\}$	$\{a \to c_0,\ b \to c_0,\ f(c_0) \to c_1\}$	Orientation
$\{c_0, c_1\}$	$\emptyset$	$\{a \to c_0,\ b \to c_0,\ f(c_0) \to c_1,\ f(c_1) \to c_1\}$	—

- 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 $f(a)\downarrow_R = c_1 \neq c_0 = a\downarrow_R$

Eigenschaften

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

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

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

Gleichheitslogik

- 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)

Von UF nach Gleichheitslogik

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

Beispiel

- UF-Formel φ^{UF}: $x = y \wedge \neg(f(x) = f(y))$
- $\mathrm{flat}^{\mathrm{E}}$: $x = y \wedge \neg(f_x = f_y)$

  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”

Ackermanns Reduktion

Algorithmus

- Eingabe: UF-Formal φ^{UF}
- Ausgabe: Erfüllbarkeitsäquivalente Gleichheitslogik-Formel φ^{E}
- Schritte:
  1. Nummeriere Teilterme der Form $f(\ldots)$
  2. Sei F_i der i-te Teilterm der Form $f(\ldots)$ und sei $\arg_j(F_i)$ sein j-tes Argument
  3. Generiere $\mathrm{flat}^{\mathrm{E}}$ 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

$$\bigwedge_{j=1}^{n} \widehat{\arg_j(F_i)} = \widehat{\arg_j(F_k)} \rightarrow \widehat{F_i} = \widehat{F_k}$$

     für alle $i < k$ wenn f die Stelligkeit n mit $n > 0$ hat
  5. Gib $\mathrm{flat}^{\mathrm{E}} \wedge \mathrm{FC}^{\mathrm{E}}$ aus

Beispiel

- φ^{UF}:

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

- $\mathsf{flat}^{\mathsf{E}}$:

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

- FC^{E}:

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

T-Solver für Gleichheitslogik

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

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

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

1. $L = \{\{x_i, y_i\} \mid 1 \leq i \leq n\} \cup \{\{x\} \mid x \in V(\varphi) \setminus \{x_1, \ldots, x_n, y_1, \ldots, y_n\}\}$
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. $\{ \{x_1, x_2, x_3, x_4\}, \{x_5\} \}$

Union-Find Algorithmus

- 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. `Find(`x`) == Find(`y`)`

Äquivalenzabschluß mittels Union-Find

Der Äquivalenzabschluß $\sim$ von

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

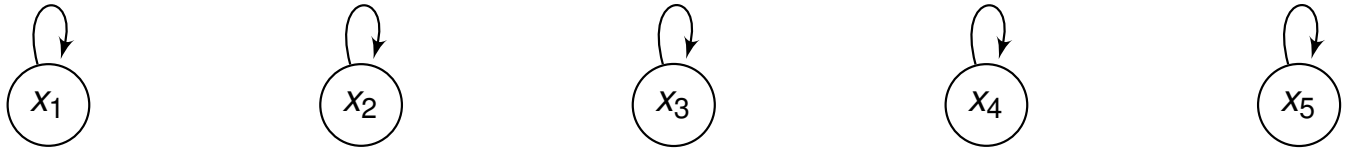
auf $V(\varphi)$ kann wie folgt berechnet werden:

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

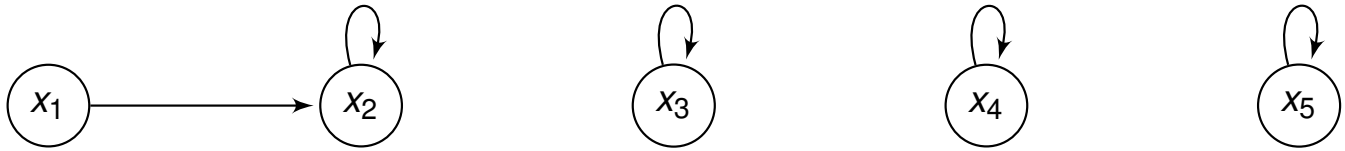
Beispiel

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:

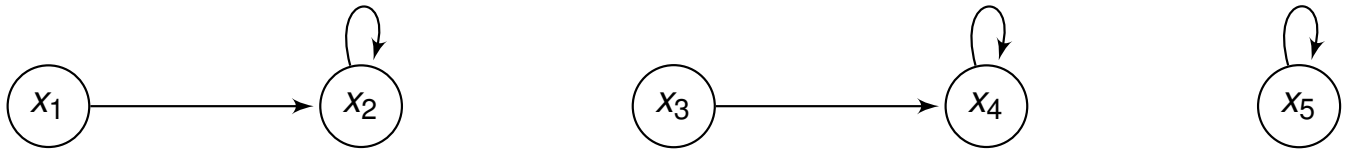
1. Nach Ausführung von `MakeSet(`x`)` für alle $x \in \{x_1, x_2, x_3, x_4, x_5\}$:



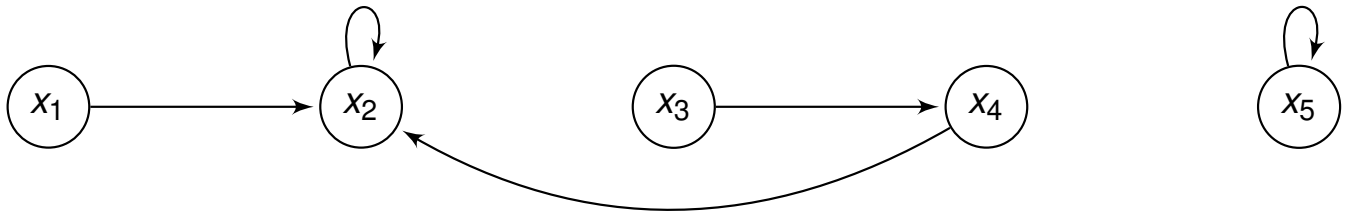
2. Nach Ausführung von `Union(`x_1`, `x_2`)`:



3. Nach Ausführung von `Union(`x_3`, `x_4`)`:



4. Nach Ausführung von `Union(`x_4`, `x_1`)`:



SAT-Reduktion von Gleichheitslogik

- 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 $x_1, \ldots, x_N$ ("bits")
- $x = y$ wird ersetzt durch $(x_1 \leftrightarrow y_1) \wedge \ldots \wedge (x_N \leftrightarrow y_n)$
- $n\lceil \log n \rceil$ 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:

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