Symbolisches SAT-basiertes Model-Checking

Dorian Weber (weber@informatik)

27. Januar 2009

Inhaltsverzeichnis

Vorteile von BDD-basierten Verfahren

- kompakte Repräsentation von Mengen (z. B. Relationen, Funktionen, Zustandsräume, Transitionen . . .)
- erlaubt direkte Manipulation ohne vorherige Dekomprimierung
- effiziente (polynomielle) Algorithmen zum Testen von Erfüllbarkeitseigenschaften existieren

Nachteile von BDD-basierten Verfahren

- Qualität der Kompression ist abhängig von der Reihenfolge der Variablen
- für manche Mengen gibt es keine gute Ordnung
- im Allgemeinen exponentielle Expansion des Zustandsraumes, unabhängig von der Sortierung der Variablen
  - BDDs erreichen dann schnell die Speicherkapazitäten heutiger Rechner für reale Spezifikationen

Stärkere Kompression des Zustandsraumes

- allgemeine aussagenlogische Formeln zur Beschreibung des Zustandsraumes
- Generierung einer aussagenlogischen Formel, die genau dann erfüllbar ist, wenn ein Gegenbeispiel der Länge k zu einer Spezifikation existiert
  - schrittweise Iteration von k
- anschließend Verwendung eines gewöhnlichen SAT-Lösers

Vorteile dieses Ansatzes

- keine exponentielle Speicherentwicklung, da Zustandsraum stets kompakt notiert
- auffinden von Gegenbeispielen minimaler Länge durch schrittweise Iterierung von k
- unabhängig von Variablensortierungen
- potentieller Nutznießer zukünftiger Entwicklungen von SAT-Lösern

2-Bit Zähler

- zwei boolesche Variablen $x[0]$ und $x[1]$
- die Translationsrelation für
  - $x[0]$ ist $x'[0] = \neg x[0]$
  - $x[1]$ ist $x'[1] = (\neg x[0] \wedge x[1]) \vee (x[0] \wedge \neg x[1])$
  - insgesamt also

$$T(s, s') \equiv (x'[0] = \neg x[0]) \wedge (x'[1] = (\neg x[0] \wedge x[1]) \vee (x[0] \wedge \neg x[1]))$$

- mögliche Anfrage: $\neg \mathbf{GF}(x = 2)$ wobei $x := x[0] + 2 \cdot x[1]$
  - intuitiv: „Zählt der 2-Bit Zähler nur endlich oft die 2 auf?“

Generelle Vorgehensweise

- wählen einer natürlichen Zahl k
- expandieren des Transitionssystemes bis auf k Schritte
- konstruieren einer aussagenlogischen Formel, die genau dann erfüllbar ist, wenn es ein Gegenbeispiel der Länge k für die Anfrage gibt
- suchen nach einer erfüllenden Belegung
  - sollte es eine geben, haben wir ein Gegenbeispiel gefunden

Modellexpansion

- wir wählen $k = 3$
- dreimaliges Entfalten der Transitionsrelation ist definiert als $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3)$
- in unserem Fall ergibt das

$$
\begin{aligned}
&(x_1[0] = \neg x_0[0]) && \wedge \\
&(x_1[1] = (\neg x_0[0] \wedge x_0[1]) \vee (x_0[0] \wedge \neg x_0[1])) && \wedge \\
&(x_2[0] = \neg x_1[0]) && \wedge \\
&(x_2[1] = (\neg x_1[0] \wedge x_1[1]) \vee (x_1[0] \wedge \neg x_1[1])) && \wedge \\
&(x_3[0] = \neg x_2[0]) && \wedge \\
&(x_3[1] = (\neg x_2[0] \wedge x_2[1]) \vee (x_2[0] \wedge \neg x_2[1]))
\end{aligned}
$$

Anfrage

- ein endlicher Ausführungspfad, der ein Zeuge für $\mathbf{GF}(x = 2)$ ist, muss eine Schleife enthalten
- demnach muss gelten: $T(s_3, s_0)$ oder $T(s_3, s_1)$ oder $T(s_3, s_2)$ oder $T(s_3, s_3)$
- in Formeln:

$$\bigvee_{l=0}^{3} T(s_3, s_l)$$

- außerdem muss diese Schleife erlauben, dass $\mathbf{F}(x = 2)$ ist

$$\underbrace{\bigwedge_{i=0}^{3} \bigvee_{j=0}^{3} \underbrace{(\neg x_j[0] \wedge x_j[1])}_{x=2}}_{\mathbf{F}(x=2)}$$

2-Bit Zähler

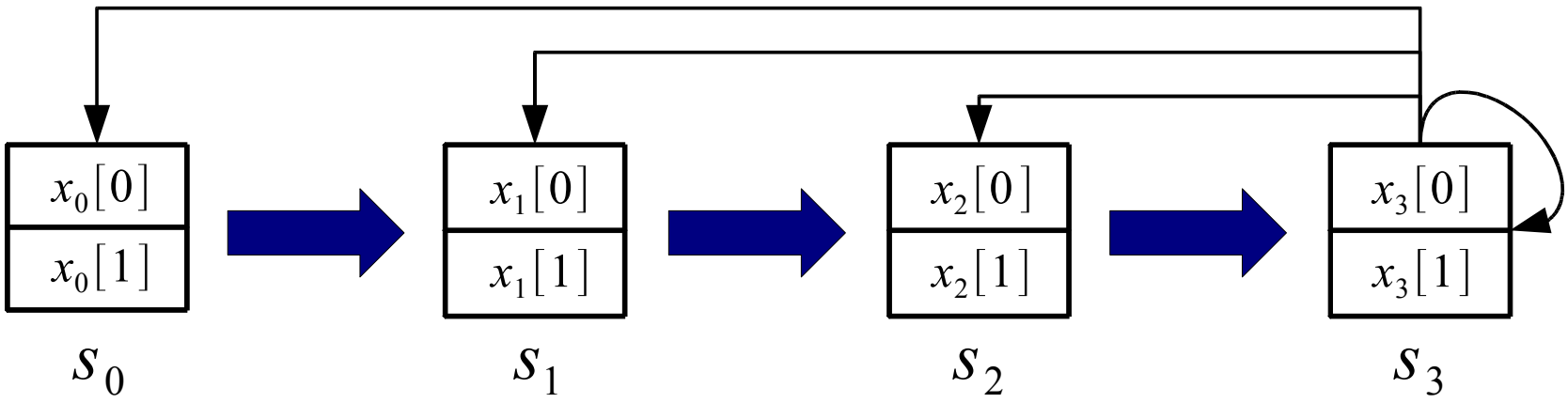


Abbildung: Entfalten der Transitionsrelation und Hinzufügen einer Rückwärtsschleife

Vollständige Formel

$$\bigwedge_{i=0}^{2} T(s_i, s_{i+1}) \wedge \bigvee_{l=0}^{3} \left(T(s_3, s_l) \wedge \bigwedge_{i=0}^{3} \bigvee_{j=0}^{3} (\neg x_j[0] \wedge x_j[1]) \right)$$

- die Formel ist genau dann erfüllbar, wenn es ein Gegenbeispiel zu $\neg \mathbf{GF}(x = 2)$ gibt
- die folgende Belegung erfüllt die Formel:

$x_0[1]$	$x_0[0]$	$x_1[1]$	$x_1[0]$	$x_2[1]$	$x_2[0]$	$x_3[1]$	$x_3[0]$
0	0	0	1	1	0	1	1

Kripke-Struktur

Definition (Kripke-Struktur)

Eine Kripke-Struktur ist ein Tupel $M = (S, I, T, \ell)$ mit

- einer endlichen *Zustandsmenge* S
- einer Menge von *Startzuständen* $I \subseteq S$
- einer *Transitionsrelation* $T \subseteq S \times S$
- und einer *Beschriftungsfunktion* $\ell : S \to \mathcal{P}(A)$ mit *atomaren Aussagen* A

Kripke-Struktur (plakativ)

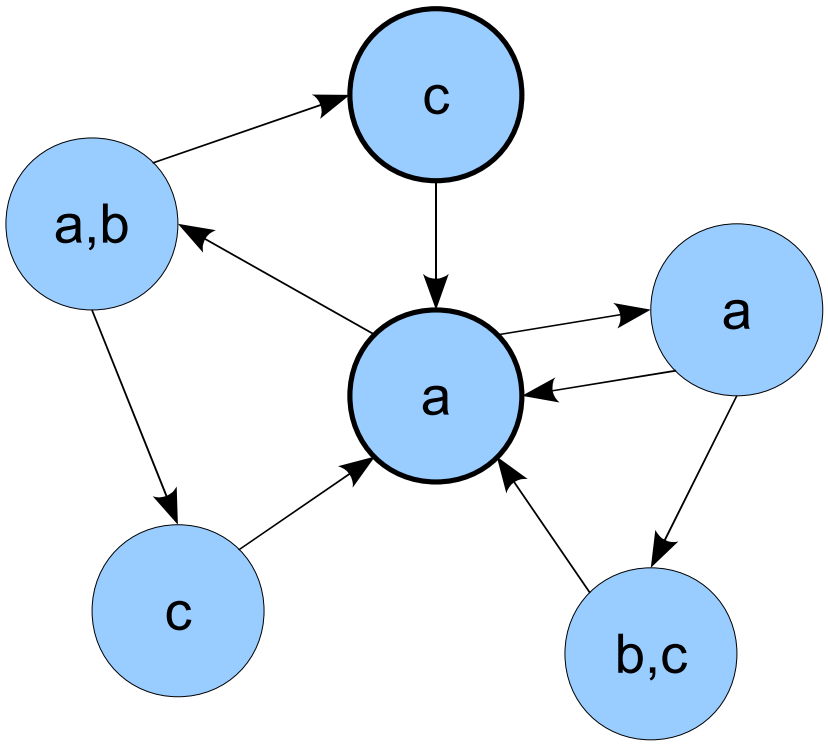


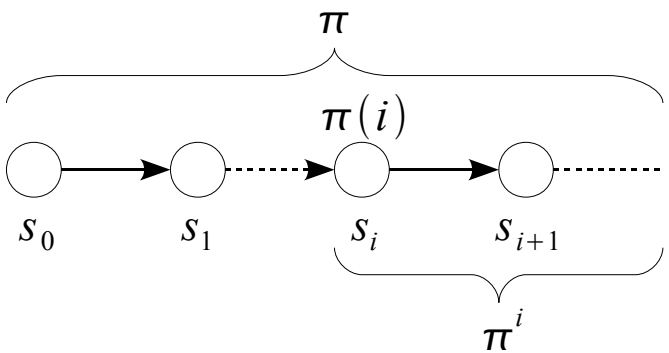
Abbildung: Beispieldarstellung einer Kripke-Struktur

Zusätzliche Hilfsmittel

Definitionen (Notationskonventionen)

- ohne Beschränkung der Allgemeinheit sei $S = \{0, 1\}^n$ und $s = (s(0), \ldots, s(n))$ ein Zustandsvektor von n aussagenlogischen Variablen
- es sei $T(s, t)$ genau dann erfüllt, falls es eine Transition von s nach t gibt
  - wir benutzen im Folgenden auch die Kurznotation $s \rightarrow t$
- es sei $p(s)$ genau dann erfüllt, falls p eine gültige atomare Aussage für s ist

Sequenzen von Zuständen, Pfade



Definitionen

- es sei $\pi = (s_0, s_1, \dots)$ eine unendliche Sequenz von Zuständen
- ferner sei $\pi(i) := s_i$ der i'te Zustand der Sequenz π
- es sei $\pi^i := (s_i, s_{i+1}, \dots)$ das (unendliche) Suffix der Sequenz π
- falls für alle i gilt, dass $\pi(i) \rightarrow \pi(i+1)$, dann nennen wir π einen Ausführungspfad

LTL-Operatoren

next, future, global

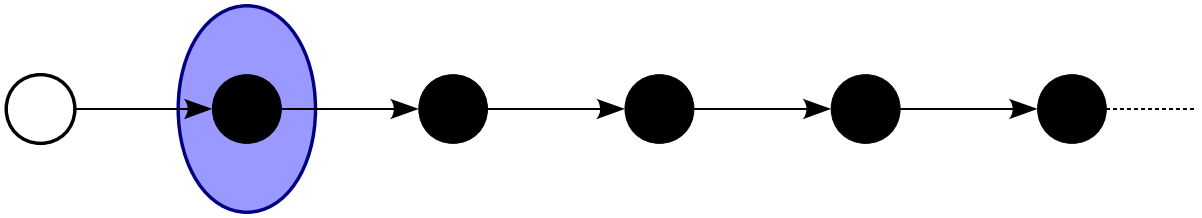


Abbildung: **X** f

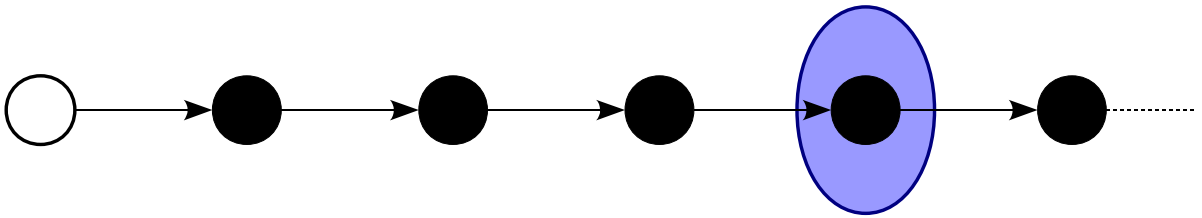


Abbildung: **F** f

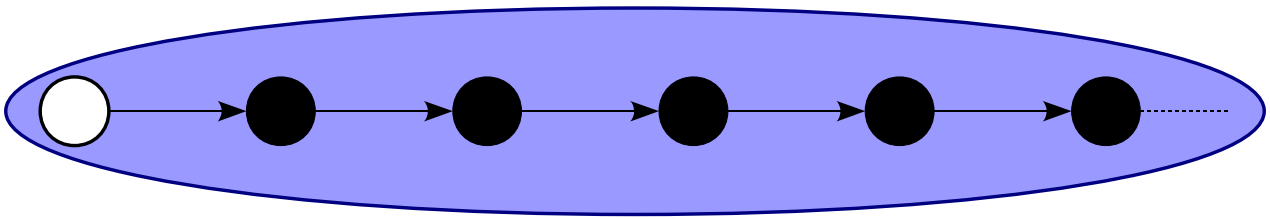


Abbildung: **G** f

LTL-Operatoren
until, release

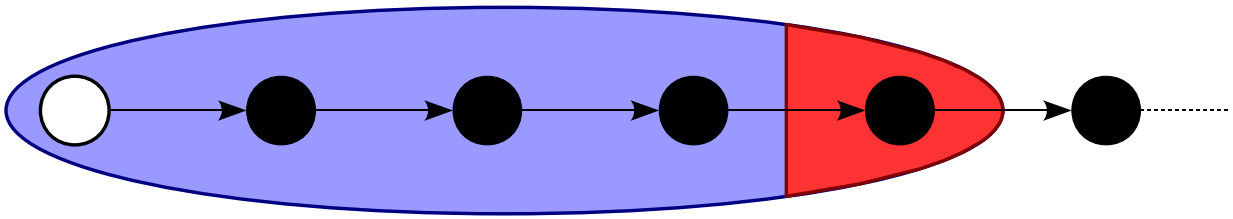


Abbildung: $f \mathbf{U} g$

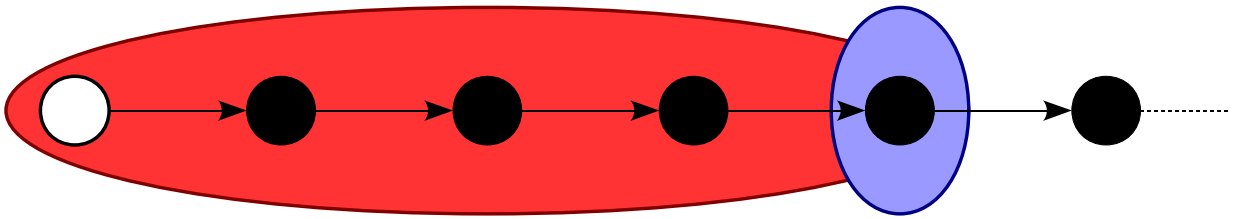


Abbildung: $f \mathbf{R} g$

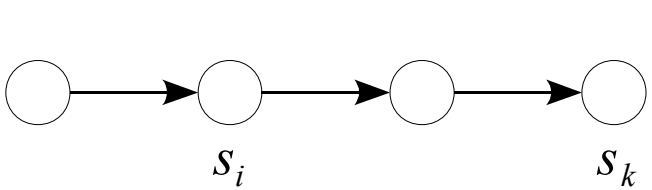
Interpretation einer LTL-Formel

Definition (Semantik)

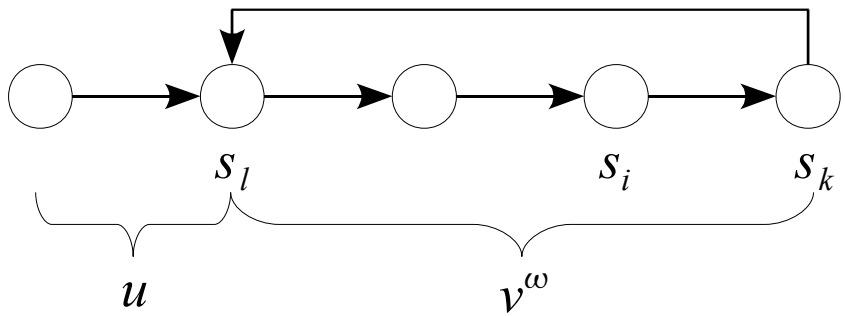
Es sei M eine Kripke-Struktur, π ein Ausführungspfad in M und f eine LTL-Formel. π erfüllt f, falls $\pi \vDash f$, wobei

1. $\pi \vDash p \Leftrightarrow p \in \ell(\pi(0))$
2. $\pi \vDash \neg p \Leftrightarrow p \notin \ell(\pi(0))$
3. $\pi \vDash f \wedge g \Leftrightarrow \pi \vDash f$ und $\pi \vDash g$
4. $\pi \vDash f \vee g \Leftrightarrow \pi \vDash f$ oder $\pi \vDash g$
5. $\pi \vDash \mathbf{G} f \Leftrightarrow \forall i : \pi^i \vDash f$
6. $\pi \vDash \mathbf{F} f \Leftrightarrow \exists i : \pi^i \vDash f$
7. $\pi \vDash \mathbf{X} f \Leftrightarrow \pi^1 \vDash f$
8. $\pi \vDash f \mathbf{U} g \Leftrightarrow \exists i \left(\pi^i \vDash g \text{ und } \forall j, j < i : \pi^j \vDash f\right)$
9. $\pi \vDash f \mathbf{R} g \Leftrightarrow \forall i \left(\pi^i \vDash g \text{ oder } \exists j, j < i : \pi^j \vDash f\right)$

Schleifen



normaler Pfad ohne Schleife



$(k\,,l\,)$-Schleife

Definition (Schleife)

Es sei $l \leq k$ und π ein Ausführungspfad. Wir bezeichnen π als (k, l)-Schleife, falls $\pi(k) \rightarrow \pi(l)$ und $\pi = u \cdot v^\omega$ mit $u = (\pi(0), \ldots, \pi(l-1))$ und $v = (\pi(l), \ldots, \pi(k))$.

Beschränkte Interpretation einer LTL-Formel

Definition (Semantik für Schleifen)

Es sei π eine (k, l)-Schleife. Dann ist eine LTL-Formel f entlang des Ausführungspfades π mit Begrenzung k genau dann gültig (wir schreiben $\pi \models_k f$), falls f in der unbeschränkten Interpretation gilt.

Beschränkte Interpretation einer LTL-Formel

Definition (Semantik für schleifenlose Pfade)

Es sei π ein Pfad, der keine Schleife ist. Für eine LTL-Formel f mit Begrenzung k gilt $\pi \models_k f$ genau dann, wenn $\pi \models_k^0 f$, wobei

1. $\pi \models_k^i p \Leftrightarrow p \in \ell(\pi(i))$
2. $\pi \models_k^i \neg p \Leftrightarrow p \notin \ell(\pi(i))$
3. $\pi \models_k^i f \wedge g \Leftrightarrow \pi \models_k^i f$ und $\pi \models_k^i g$
4. $\pi \models_k^i f \vee g \Leftrightarrow \pi \models_k^i f$ oder $\pi \models_k^i g$
5. $\pi \models_k^i \mathbf{G} f \Leftrightarrow \bot$
6. $\pi \models_k^i \mathbf{F} f \Leftrightarrow \exists j \in [i,k] : \pi \models_k^j f$
7. $\pi \models_k^i \mathbf{X} f \Leftrightarrow i < k$ und $\pi \models_k^{i+1} f$
8. $\pi \models_k^i f \mathbf{U} g \Leftrightarrow \exists j \in [i,k] (\pi \models_k^j g$ und $\forall n \in [i, j-1] : \pi \models_k^n f)$
9. $\pi \models_k^i f \mathbf{R} g \Leftrightarrow \exists j \in [i,k] (\pi \models_k^j f$ und $\forall n \in [i, j-1] : \pi \models_k^n g)$

Beschränkte Interpretation einer LTL-Formel

Fakt

Die Dualität zwischen den Operatoren $\mathbf{G}$ *und* $\mathbf{F}$

$$\neg \mathbf{F} f \equiv \mathbf{G} \neg f$$

sowie $\mathbf{U}$ *und* $\mathbf{R}$

$$\neg (f \mathbf{U} g) \equiv (\neg f) \mathbf{R} (\neg g)$$

gilt in der beschränkten Interpretation nicht mehr.

Korrektheit beschränkter Erfüllbarkeit

Satz

Es sei f eine LTL-Formel und π ein Ausführungspfad. Dann gilt $\pi \models_k f \Rightarrow \pi \models f$.

Vollständigkeit beschränkter Erfüllbarkeit

Satz

Es sei f eine LTL-Formel und M eine Kripke-Struktur. Es gilt $M \models \mathbf{E}f$, wenn ein $k \in \mathbb{N}$ existiert, so dass $M \models_k \mathbf{E}f$.

Konvergenz beschränkter Erfüllbarkeit

Folgerung

Es sei f eine LTL-Formel und M eine Kripke-Struktur. Es gilt $M \models \mathbf{E}f$ genau dann, wenn ein $k \in \mathbb{N}$ existiert, so dass $M \models_k \mathbf{E}f$.

Was nun?

- offenbar reicht die beschränkte Erfüllbarkeit aus, um Spezifikationen in LTL-Form zu testen
- allerdings: im Moment liegen sowohl das Modell, als auch die Anfrage als LTL-Formeln vor
- wir wollen aussagenlogische Erfüllbarkeit testen, also müssen wir eine entsprechende Formel erzeugen

Entpacken der Transitionsrelation

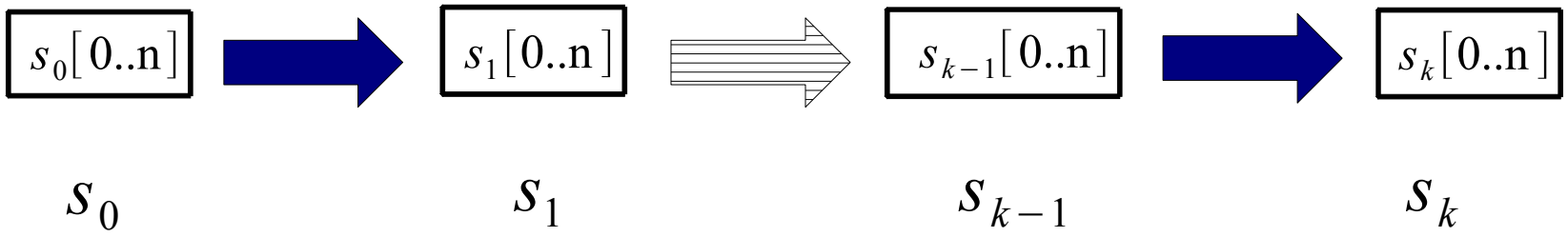


Abbildung: Expansion des Modells

Definition (Modellexpansion)

Sei M eine Kripke-Struktur und $k \in \mathbb{N}$. Das k-fach expandierte Modell ist definiert als

$$[\![M]\!]_k := I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$

Übersetzung der Anfrage
Schleifenfreier Fall

Definition (Schleifenfreie Übersetzung)

Es sei f eine LTL-Formel und $k, i \in \mathbb{N}$ mit $i \leq k$. Dann ist die Übersetzung $[\![f]\!]_k^i$ definiert als

1. $[\![p]\!]_k^i := p(s_i)$
2. $[\![\neg p]\!]_k^i := \neg p(s_i)$
3. $[\![f \wedge g]\!]_k^i := [\![f]\!]_k^i \wedge [\![g]\!]_k^i$
4. $[\![f \vee g]\!]_k^i := [\![f]\!]_k^i \vee [\![g]\!]_k^i$
5. $[\![\mathbf{G} f]\!]_k^i := \bot$
6. $[\![\mathbf{F} f]\!]_k^i := \bigvee_{j=i}^{k} [\![f]\!]_k^j$
7. $[\![\mathbf{X} f]\!]_k^i := \begin{cases} [\![f]\!]_k^{i+1} & \text{falls } i < k \\ \bot & \text{andernfalls} \end{cases}$

Übersetzung der Anfrage

Schleifenfreier Fall (Fortsetzung)

Definition (Schleifenfreie Übersetzung (until, release))

$$[\![f \mathbf{U} g]\!]_k^i := \bigvee_{j=i}^{k} \left([\![g]\!]_k^j \wedge \bigwedge_{n=i}^{j-1} [\![f]\!]_k^n \right)$$

$$[\![f \mathbf{R} g]\!]_k^i := \bigvee_{j=i}^{k} \left([\![f]\!]_k^j \wedge \bigwedge_{n=i}^{j} [\![g]\!]_k^n \right)$$

Übersetzung der Anfrage

Nachfolger in einer Schleife

Definition (Nachfolger)

Wir definieren den Nachfolger $succ(i)$ einer (k, l)-Schleife als

$$succ(i) := \begin{cases} i+1 & \text{falls } i < k \\ l & \text{sonst} \end{cases}$$

Übersetzung der Anfrage

Ausführungspfad mit Schleife

Definition (Übersetzung der Schleife)

Es sei f eine LTL-Formel und $k, l, i \in \mathbb{N}$ mit $l, i \leq k$. Dann ist die Übersetzung ${}_l[\![f]\!]^i_k$ definiert als

1. ${}_l[\![p]\!]^i_k := p(s_i)$
2. ${}_l[\![\neg p]\!]^i_k := \neg p(s_i)$
3. ${}_l[\![f \wedge g]\!]^i_k := {}_l[\![f]\!]^i_k \wedge {}_l[\![g]\!]^i_k$
4. ${}_l[\![f \vee g]\!]^i_k := {}_l[\![f]\!]^i_k \vee {}_l[\![g]\!]^i_k$
5. ${}_l[\![\mathbf{G} f]\!]^i_k := \bigwedge_{j=\min(i,l)}^{k} {}_l[\![f]\!]^j_k$
6. ${}_l[\![\mathbf{F} f]\!]^i_k := \bigvee_{j=\min(i,l)}^{k} {}_l[\![f]\!]^j_k$
7. ${}_l[\![\mathbf{X} f]\!]^i_k := {}_l[\![f]\!]^{succ(i)}_k$

Übersetzung der Anfrage

Ausführungspfad mit Schleife (Fortsetzung)

Definition (Übersetzung der Schleife)

Until-Operator

$$
\begin{aligned}
{}_l[\![f \mathsf{U} g]\!]_k^i := & \bigvee_{j=i}^{k} \left({}_l[\![g]\!]_k^j \wedge \bigwedge_{n=i}^{j-1} {}_l[\![f]\!]_k^n \right) \vee \\
& \bigvee_{j=l}^{i-1} \left({}_l[\![g]\!]_k^j \wedge \bigwedge_{n=i}^{k} {}_l[\![f]\!]_k^n \wedge \bigwedge_{n=l}^{j-1} [\![f]\!]_k^n \right)
\end{aligned}
$$

Übersetzung der Anfrage

Ausführungspfad mit Schleife (Fortsetzung)

Definition (Übersetzung der Schleife)

Release-Operator

$$
\begin{aligned}
{}_l[\![f \,\mathbf{R}\, g]\!]_k^i := & \bigwedge_{j=\min(i,l)}^{k} {}_l[\![g]\!]_k^j \vee \\
& \bigvee_{j=i}^{k} \left({}_l[\![f]\!]_k^j \wedge \bigwedge_{n=i}^{j} {}_l[\![g]\!]_k^n \right) \vee \\
& \bigvee_{j=l}^{i-1} \left({}_l[\![f]\!]_k^j \wedge \bigwedge_{n=i}^{k} {}_l[\![g]\!]_k^n \wedge \bigwedge_{n=l}^{j} [\![g]\!]_k^n \right)
\end{aligned}
$$

Schleifenbedingung

Definition (Schleifenbedingung)

Für $k, l \in \mathbb{N}$ definieren wir ${}_lL_k := T(s_k, s_l)$ und $L_k := \neg \bigvee_{l=0}^{k} {}_lL_k$.

Fertige Formel

Definition

Es sei f eine LTL-Formel, M eine Kripke-Struktur und $k \in \mathbb{N}$. Die SAT-Instanz, die genau dann erfüllbar ist, wenn ein Pfad der Länge k in M existiert, der f erfüllt, ist gegeben durch

$$[\![M, f]\!]_k := [\![M]\!]_k \wedge \left(\left(L_k \wedge [\![f]\!]_k^0 \right) \vee \bigvee_{l=0}^{k} \left({}_l L_k \wedge {}_l[\![f]\!]_k^0 \right) \right)$$

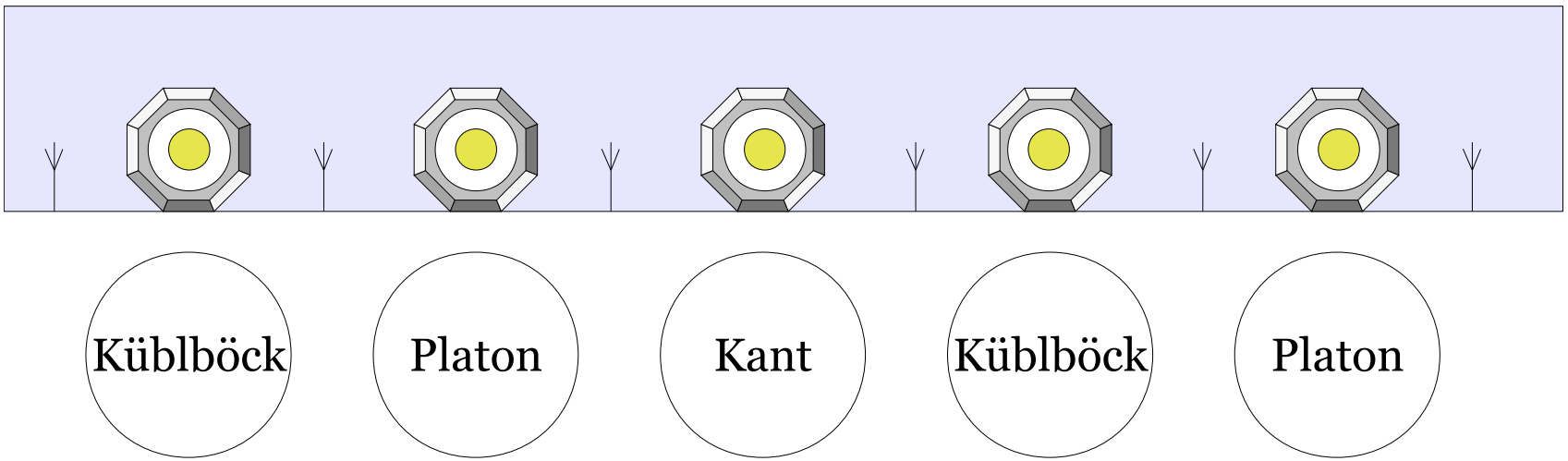
NuSMV

- OpenSource-Software NuSMV *(new symbolic model verification)*
- implementiert das besprochene Verfahren BMC *(bounded model checking)*
- wird in der Industrie zur Verifikation von Hardware eingesetzt

Unser 2-Bit Zähler

```
 1  MODULE main
 2    VAR
 3      bit0 : boolean;
 4      bit1 : boolean;
 5    DEFINE
 6      counter := bit0 + 2*bit1;
 7    ASSIGN
 8      init(bit0) := 0;
 9      init(bit1) := 0;
10
11      next(bit0) := !bit0;
12      next(bit1) := (!bit0 & bit1) | (bit0 & !bit1);
13
14    LTLSPEC
15      !G F (counter = 0);
```

Dining Philosophers



Quellen


Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu.
Symbolic model checking without bdds.
In *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 193–207, London, UK, 1999.


A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella.
NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking.
In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark, July 2002.