Formale Grundlagen der Informatik 3
Kapitel 6

Automatenbasiertes LTL Model Checking

Frank Heitmann
heitmann@informatik.uni-hamburg.de

11. Januar 2016

LTL: Syntax

Definition (Syntax von LTL)

Die (wohlgeformten) Formeln der Linear Temporal Logic (LTL) werden durch die folgende Grammatik definiert:

$$\begin{aligned}\phi \quad ::= \quad & v \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \\ & X\phi \mid F\phi \mid G\phi \mid (\phi U \phi)\end{aligned}$$

wobei $v \in V$ ein aussagenlogisches Atom ist.

Die neuen Operatoren sind ne**X**t, **F**inally, **G**lobally und **U**ntil.

LTL: Syntax (Alternative)

Definition (Syntax von LTL (alternative))

Die *wohlgeformten Ausdrücke/Formeln* von LTL werden induktiv definiert durch

1. Jedes $v \in V$ ist eine (atomare) LTL Formel.
2. Wenn ϕ_1 und ϕ_2 Formeln sind, dann auch $\neg\phi_1, (\phi_1 \wedge \phi_2)$ und $(\phi_1 \vee \phi_2)$.
3. Wenn ϕ_1 und ϕ_2 Formeln sind, dann auch $X\phi_1, F\phi_1, G\phi_1$ and $(\phi_1 U \phi_2)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln von LTL.

Zum Klammersparen binden die unären Junktoren $\neg, X, G$ and F stärker als U und dann $\wedge$ und $\vee$.

Transitionssysteme

LTL und CTL Formeln werden entlang der Pfade eines **Transitionssystems** interpretiert. Das Transitionssystem übernimmt also die Rolle der Belegung in der Aussagenlogik.

Transitionssysteme

Definition (Transitionssystem)

Ein *labelled transition system* (LTS) ist ein Tupel $TS = (S, s_0, R, L)$mit

- einer endlichen Menge von Zuständen S,
- einem Startzustand $s_0 \in S$,
- einer links-totalen Übergangsrelation $R \subseteq S \times S$ und
- einer labelling function $L : S \to \mathcal{P}(V)$, die jedem Zustand s die Menge der atomaren Formeln $L(s) \subseteq V$ zuweist, die in s gelten.

Linkstotal bedeutet, dass es zu jedem $s \in S$ stets ein s' mit $(s, s') \in R$ gibt.

Transitionssysteme

Definition (Pfad im LTS)

Ein *Pfad* π in einem LTS $TS = (S, s_0, R, L)$ ist eine unendliche Sequenz von Zuständen

$$\pi = s_1 s_2 s_3 \dots$$

derart, dass $(s_i, s_{i+1}) \in R$ für alle $i \geq 1$.

- Mit π^i, $i \geq 1$ bezeichnen wir den Suffix, der an s_i startet, d.h. den Pfad $\pi^i = s_i s_{i+1} \dots$
- Mit $\pi(i)$, $i \geq 1$, bezeichnen wir den i-ten Zustand in π, d.h. $\pi(i) = s_i$.
- Wenn s_1 der Startzustand s_0 von TS ist, wird π auch als Rechnung bezeichnet.

LTL: Semantik

Definition (Semantik von LTL (I))

Sei $M = (S, s_0, R, L)$ ein LTS und $\pi = s_1 s_2 \dots$ ein Pfad in M. π erfüllt eine LTL Formel ϕ (in M), wenn $M, \pi \models \phi$ gilt, wobei die Relation $\models$ induktiv definiert ist:

$$
\begin{array}{lll}
M, \pi \models v & \text{gdw.} & v \in L(s_1) \text{ für } v \in V \\
M, \pi \models \neg\phi & \text{gdw.} & M, \pi \not\models \phi \\
M, \pi \models \phi_1 \wedge \phi_2 & \text{gdw.} & M, \pi \models \phi_1 \ \mathit{und}\ M, \pi \models \phi_2 \\
M, \pi \models \phi_1 \vee \phi_2 & \text{gdw.} & M, \pi \models \phi_1 \ \mathit{oder}\ M, \pi \models \phi_2
\end{array}
$$

LTL: Semantik

Definition (Semantik von LTL (II))

$M, \pi \models X\phi$	gdw.	$M, \pi^2 \models \phi$
$M, \pi \models F\phi$	gdw.	$M, \pi^i \models \phi$ für ein $i \geq 1$
$M, \pi \models G\phi$	gdw.	$M, \pi^i \models \phi$ für alle $i \geq 1$
$M, \pi \models \phi_1 U \phi_2$	gdw.	ein $i \geq 1$ existiert mit $M, \pi^i \models \phi_2$ und für alle $j < i$ $M, \pi^j \models \phi_1$ gilt.

LTL: Semantik

Definition (Semantik von LTL (III))

Sei $M = (S, s_0, R, L)$ ein LTS. Sei ϕ eine LTL Formel und $s \in S$ ein Zustand von M.

- $M, s \models \phi$, wenn $M, \pi \models \phi$ gilt für jeden Pfad π in M, der in s startet.
- Wenn $M, s_0 \models \phi$ gilt, schreiben wir $M \models \phi$. Wir sagen: M ist ein *Modell* für ϕ oder ϕ ist in M *erfüllt*.
- Zwei LTL Formeln ϕ und ψ sind *äquivalent*, $\phi \equiv \psi$, wenn für alle Modelle M und alle Pfade π in M auch $M, \pi \models \phi$ gdw. $M, \pi \models \psi$ gilt.

CTL: Syntax

Definition (Syntax von CTL)

Die (wohlgeformten) Formeln der Computation Tree Logic (CTL) werden durch die folgende Grammatik definiert:

$$\begin{aligned}\phi \quad ::= \quad & v \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid \\ & EX\phi \mid EF\phi \mid EG\phi \mid E[\phi U \phi] \mid \\ & AX\phi \mid AF\phi \mid AG\phi \mid A[\phi U \phi]\end{aligned}$$

wobei $v \in V$ ein aussagenlogisches Atom ist.

Kann man natürlich auch wieder mit einer induktiven Definition machen!

CTL: Semantik

Definition (Semantik von CTL (I))

Sei $M = (S, s_0, R, L)$ ein LTS und $s \in S$ ein Zustand. Eine CTL Formel ϕ ist erfüllt in s (in M), wenn $M, s \models \phi$ gilt, wobei die Relation $\models$ induktiv definiert ist:

$$
\begin{array}{lll}
M, s \models v & \text{gdw.} & v \in L(s) \text{ für } v \in V \\
M, s \models \neg\phi & \text{gdw.} & M, s \not\models \phi \\
M, s \models \phi_1 \wedge \phi_2 & \text{gdw.} & M, s \models \phi_1 \textit{ und } M, s \models \phi_2 \\
M, s \models \phi_1 \vee \phi_2 & \text{gdw.} & M, s \models \phi_1 \textit{ oder } M, s \models \phi_2
\end{array}
$$

CTL: Semantik

Definition (Semantik von CTL (II))

$M, s \models EX\phi$	gdw.	ein Zustand $s' \in S$ existiert mit $(s, s') \in R$ und $M, s' \models \phi$
$M, s \models EF\phi$	gdw.	ein Pfad $\pi = s_1 s_2 \ldots$ beginnend bei s $(s_1 = s)$ existiert und ein $i \geq 1$, so dass $M, s_i \models \phi$ gilt.
$M, s \models EG\phi$	gdw.	ein Pfad $\pi = s_1 s_2 \ldots$ beginnend bei s $(s_1 = s)$ existiert und für alle $i \geq 1$ $M, s_i \models \phi$ gilt.
$M, s \models E[\phi_1 U \phi_2]$	gdw.	ein Pfad $\pi = s_1 s_2 \ldots$ beginnend bei s existiert und ein $j \geq 1$, so dass $M, s_j \models \phi_2$ und $M, s_i \models \phi_1$ für alle $i < j$.

CTL: Semantik

Definition (Semantik von CTL (III))

$M, s \models AX\phi$	gdw.	$M, s' \models \phi$ für alle $s' \in S$ mit $(s, s') \in R$.
$M, s \models AF\phi$	gdw.	für alle Pfade $\pi = s_1 s_2 \dots$ beginnend bei s ein $i \geq 1$ existiert mit $M, s_i \models \phi$.
$M, s \models AG\phi$	gdw.	für alle Pfade $\pi = s_1 s_2 \dots$ beginnend bei s $M, s_i \models \phi$ für alle $i \geq 1$ gilt.
$M, s \models A[\phi_1 U \phi_2]$	gdw.	für alle Pfade $\pi = s_1 s_2 \dots$ beginnend bei s ein $j \geq 1$ existiert derart, dass $M, s_j \models \phi_2$ und $M, s_i \models \phi_1$ für alle $i < j$ gilt

CTL: Semantik

Definition (Semantik von CTL (IV))

Sei $M = (S, s_0, R, L)$ ein LTS und ϕ eine CTL Formel.

- Wenn $M, s_0 \models \phi$ gilt, schreiben wir auch $M \models \phi$ und sagen, dass M ein *Modell* für ϕ ist oder dass ϕ *erfüllt ist* in M.
- Zwei CTL Formeln ϕ und ψ sind *äquivalent*, $\phi \equiv \psi$, wenn für alle Modelle M und alle Zustände s in M auch $M, s \models \phi$ gdw. $M, s \models \psi$ gilt.

LTL und CTL: Äquivalenzen

Oft benutzt man ein Set von "adequate connectives", d.h. ein Set von Junktoren, dass Ausdrucksstark genug ist, um jede Formel der Logik auszudrücken.

Für LTL ist ein solches Set z.B.

$$\{\neg, \wedge, X, U\}$$

für CTL z.B.

$$\{\neg, \wedge, EX, EG, EU\}.$$

Z.B. ist (in LTL) $F\phi := \top U\phi$ und $G\phi := \neg F \neg \phi$ und (in CTL) $EF\phi \equiv E[\top U\phi]$ und $AG\phi \equiv \neg EF\neg\phi$.

CTL*: Syntax und Semantik

Definition (CTL* Syntax)

Die Syntax von CTL* ist eine wechselseitig rekursive induktive Definition, die Pfad- und Zustandsformeln beinhaltet:

- Es gibt Zustandsformeln, die in Zuständen ausgewertet werden:

$$\phi ::= \top \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid A[\alpha] \mid E[\alpha]$$

  wobei p eine atomare Formel ist und α eine Pfadformel.

- Es gibt Pfadformeln, die entlang von Pfaden ausgewertet werden:

$$\alpha ::= \phi \mid (\neg\alpha) \mid (\alpha \wedge \alpha) \mid (\alpha U \alpha) \mid (G\alpha) \mid (F\alpha) \mid (X\alpha)$$

  wobei ϕ eine Zustandsformel ist.

CTL*: Syntax und Semantik

- Die Semantik wird dann so wie bei CTL und LTL definiert, je nachdem, ob es eine Zustands- oder eine Pfadformel ist.
- Eine LTL-Formel α ist äquivalent zur CTL*-Formel $A[\alpha]$. LTL kann also als Teillogik von CTL* angesehen werden.
- CTL ist sofort eine Teillogik von CTL*, da man die Pfadformeln auf

  $$\alpha ::= (\phi U \phi) \mid (G\phi) \mid (F\phi) \mid (X\phi)$$

  einschränken kann und dann sofort CTL hat.

LTL, CTL und CTL*

Zusammenhänge der Logiken

1. In CTL, aber nicht in LTL: $\phi_1 := AGEF\ p$.
   - Wann immer nötig, kann ein Zustand erreicht werden, in dem p gilt.
2. In CTL*, aber weder in LTL noch in CTL: $\phi_2 := E[GF\ p]$
   - Es gibt einen Pfad mit unendlich vielen p.
3. In LTL, aber nicht in CTL: $\phi_3 := A[GF\ p \Rightarrow F\ q]$
   - Tritt p entlang eines Pfades unendlich oft auf, dann tritt ein q auf.
4. In LTL und CTL: $\phi_{4,1} := AG(p \Rightarrow AF\ q)$ in CTL bzw. $\phi_{4,2} := G(p \Rightarrow F\ q)$ in LTL.
   - Jedem p folgt irgendwann ein q.

LTL vs. CTL

Take Home Message

LTL und CTL sind beide wichtig, da es jeweils Dinge gibt, die in der anderen nicht ausgedrückt werden können.

Take Home Message 2

LTL kann nicht über Pfade quantifizieren. CTL kann dafür nicht so fein über Pfade argumentieren wie LTL. (Für viele ist LTL einfacher; CTL Formeln wie $AFAX\ p$ erscheinen schwierig...)

Literatur

Zu diesem Teil der Vorlesung siehe Kapitel 3 in *Logic in Computer Science. Modelling and Reasoning about Systems.* Michael Huth und Mark Ryan, 2. Auflage, Cambridge University Press, 2004.

Das Model-Checking-Problem

Das Problem

Das *model checking problem* für LTL oder CTL fragt, gegeben ein LTS M und eine Formel ϕ, ob $M \models \phi$ gilt, d.h. ob M ein Modell für ϕ ist.

Eingabe: Ein LTS M und eine LTL oder CTL Formel ϕ.

Frage: Gilt $M \models \phi$?

Model Checking. Ergebnisse

Satz

Sei M ein LTS.

1. *Sei ϕ eine LTL Formel. Das model checking problem für LTL, d.h. die Frage, ob $M \models \phi$ gilt, ist* PSPACE*-vollständig und kann in $O(|M| \cdot 2^{|\phi|})$ Zeit entschieden werden.*
2. *Sei ϕ eine CTL Formel. Das model checking problem für CTL, d.h. die Frage, ob $M \models \phi$ gilt, kann in $O(|M| \cdot |\phi|)$ Zeit entschieden werden.*

Wichtige Anmerkung

Das Modell M wird allerdings i.A. sehr schnell sehr groß. Daher ist $|M|$ der dominante Faktor, was zu dem berühmten Problem der Zustandsraumexplosion führt.

Die Idee

Sei M ein LTS und ϕ eine LTL Formel.

- Zu $\neg\phi$ (der Negation der Spezifikation!) konstruieren wir einen (Büchi-)Automaten $A_{\neg\phi}$.
- $A_{\neg\phi}$ akzeptiert genau die Wörter w mit $w \models \neg\phi$.
- Bilde den "Produktautomaten" $M \cap A_{\neg\phi}$.
- Prüfe, ob die akzeptierte Sprache von $M \cap A_{\neg\phi}$ leer ist.

Das weitere Vorgehen

Wir benötigen jetzt also:

1. Büchi-Automaten (und drumherum)
2. Eine alternative (aber äquivalente) Semantik für LTL
3. Damit dann die Konstruktion für $A_{\neg\phi}$
4. Den “Produktautomaten”
5. Den Leerheitstest

Büchi-Automaten und ω-Wörter

- *Syntaktisch* sind Büchi-Automaten wie endliche Automaten definiert.
- *Semantisch* lesen sie *unendliche lange* Wörter!

ω-Wörter

Definition (ω-Wörter und -Sprachen)

- Sei Σ ein *endliches* Alphabet. Ein *unendliches Wort über* Σ (oder ω-Wort) ist eine unendliche Folge $w = a_0 a_1 a_2 \dots$ von Buchstaben $a_i \in \Sigma$.
- Die Menge aller unendlichen Wörter über Σ wird mit Σ^ω bezeichnet. Eine Menge $L \subseteq \Sigma^\omega$ wird als ω-Sprache bezeichnet.
- Mit $|w|_a$ ($w \in \Sigma^\omega$, $a \in \Sigma$) wird die Anzahl der Vorkommen des Buchstabens a im Wort w bezeichnet.
- Konkatenation etc. wird erweitert. Es ist allerdings nicht möglich zwei ω-Wörter zu konkatenieren, sondern nur ein endliches Wort v und ein ω-Wort w zu $v \cdot w$ zu machen.
- Ähnlich macht v^ω nur für $v \in \Sigma^*$ Sinn und ist auf L^ω für Sprachen $L \subseteq \Sigma^*$ erweiterbar.

ω-reguläre Sprachen

Definition (ω-reguläre Sprachen)

Sei $L \subseteq \Sigma^\omega$. L ist ω-regulär, wenn ein $n \in \mathbb{N}$ existiert und reguläre Sprachen $U_0, U_1, \ldots, U_{n-1}, V_0, V_1, \ldots, V_{n-1} \subseteq \Sigma^*$ mit $\lambda \notin V_i$ für alle i, so dass

$$L = \cup_{i=0}^{n-1} U_i V_i^\omega$$

gilt.

Satz

Die Klasse der ω-regulären Sprachen ist abgeschlossen unter Vereinigung und Linkskonkatenation mit regulären Sprachen.

Büchi-Automaten

Definition (NBA)

Ein **Büchi-Automat** (NBA) ist ein 5-Tupel

$$A = (Z, \Sigma, \delta, z_0, Z_{end})$$

mit:

- Der endlichen Menge von *Zuständen* Z.
- Dem endlichen Alphabet Σ von *Eingabesymbolen*.
- Der *Überführungsfunktion* $\delta : Z \times \Sigma \to 2^Z$.
- Dem *Startzustand* $z_0 \in Z$.
- Der Menge der *Endzustände* $Z_{end} \subseteq Z$.

Büchi-Automaten

Definition (NBA - Fortsetzung)

- Sei $w = a_0 a_1 a_2 \ldots \in \Sigma^\omega$ ein Wort. Ein Lauf von A auf w ist eine unendliche Folge von Zuständen $\rho = z_0 z_1 z_2 \ldots$, die am Anfangszustand beginnt und die $z_{i+1} \in \delta(z_i, a_i)$ für alle $i \geq 0$ erfüllt.
- Mit $\mathit{inf}(\rho)$ wird die Menge der in ρ unendlich oft vorkommenden Zustände bezeichnet.
- Ein Lauf ist **akzeptierend** wenn $\mathit{inf}(\rho) \cap F \neq \emptyset$ gilt.
- $L(A)$ ist die Menge jener Wörter, für die ein akzeptierender Lauf in A existiert.
- Ist $|\delta(z, a)| = 1$ für alle $z \in Z$ und $a \in \Sigma$, dann ist der NBA *deterministische* (d.h. ein DBA).

Büchi-Automaten

Satz

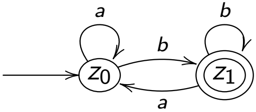
Zu jedem NBA mit mehreren Startzuständen existiert ein äquivalenter NBA mit nur einem Startzustand (und nur einem Zustand mehr).

Beweis.

Wie bei NFAs: Führe einen neuen (einzigen) Startzustand z_{neu} ein und eine a-Kante von z_{neu} zu z, wenn es eine a-Kante von einem früheren Startzustand zu z gab. $\square$

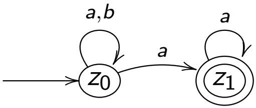
Ein Beispiel

Ein DBA für $L_1 = (a^* b)^\omega$?



Noch ein Beispiel

Ein Büchi-Automat für $L_2 = (a+b)^* a^\omega$?



Der erste Automat war deterministisch, dieser nicht...

DBA $\lneq$ NBA

Satz

NBAs sind echt mächtiger als DBAs, d.h. es gibt ω-Sprachen, die von einem NBA akzeptiert werden können, nicht aber von einem DBA.

Beweis.

Man kann dies gerade an obigem $L_2 = \{w \in \{a,b\}^\omega \mid |w|_b < \infty\}$ zeigen. L_2 kann nach obigem von einem NBA akzeptiert werden. Angenommen $A = (Z, \Sigma, \delta, z_0, Z_{end})$ ist nun ein DBA mit $L(A) = L_2$, dann ... Hausaufgabe! :) $\square$

Leerheitsproblem

Satz

Das Leerheitsproblem für NBA ist in Zeit $O(n)$ lösbar, wobei n die Anzahl der Transitionen des NBA ist.

Beweis.

Sei $A = (Z, \Sigma, \delta, z_0, Z_{end})$ ein NBA und sei außerdem jedes $z \in Z$ erreichbar. Es gilt $L(A) \neq \emptyset$ gdw. es einen Pfad von z_0 zu einem $z \in Z_{end}$ gibt und danach einen (nicht-leeren) Pfad von z nach z.

1. Berechne eine Zerlegung des Zustandsdiagramms in maximale strenge Zusammenhangskomponenten (SCC) in $O(n)$.
2. Prüfe für jedes $z \in Z_{end}$ ob es in einer nicht-trivialen (mindestens eine Kante) SCC liegt.

Ist der zweite Schritt erfolgreich gilt $L(A) \neq \emptyset$, sonst ist die akzeptierte Sprache leer. □

NBA und ω-reguläre Sprachen

Satz

1. Seien A, B zwei NBAs und C ein NFA, dann existieren NBAs D und E mit $L(D) = L(A) \cup L(B)$ und $L(E) = L(C) \cdot L(A)$. Ist außerdem $\lambda \notin L(C)$, dann existiert ein NBA F mit $L(F) = L(C)^{\omega}$.
2. Eine Sprache L ist ω-regulär gdw. ein Büchi-Automat A existiert mit $L(A) = L$.

NBA Schnitt

Satz

Seien A und B NBAs mit n bzw. m Zuständen. Dann existiert ein NBA C mit $L(C) = L(A) \cap L(B)$ und $3 \cdot n \cdot m$ Zuständen.

Beweis

Sei $A = (Z, \Sigma, \delta, z_0, Z_{end})$ und $B = (Z', \Sigma, \delta', z'_0, Z'_{end})$. Wir definieren:

...

NBA Schnitt

Sei $A = (Z, \Sigma, \delta, z_0, Z_{end})$ und $B = (Z', \Sigma, \delta', z_0', Z'_{end})$. Wir definieren:

$$C := (Z \times Z' \times \{0,1,2\}, \Sigma, \delta'', (z_0, z_0', 0), Z \times Z' \times \{2\})$$

mit $\delta''((z,z',i),a) := \{(u,u',j) \mid u \in \delta(z,a), u' \in \delta'(z',a)\}$ wobei

$$j := \begin{cases} 1 & \text{, falls } i = 0 \text{ und } u \in Z_{end} \text{ oder } i = 1 \text{ und } u' \notin Z'_{end} \\ 2 & \text{, falls } i = 1 \text{ und } u' \in Z'_{end} \\ 0 & \text{, sonst.} \end{cases}$$

Weiteres (Korrektheit der Konstruktion) als Hausaufgabe...

Generalisierter NBA

Definition (Generalisierter NBA (GNBA))

- Ein generalisierter NBA (GNBA) ist ein Tupel $A = (Z, \Sigma, \delta, Z_{start}, Z^1_{end}, Z^2_{end}, \ldots, Z^k_{end})$, der wie ein NBA definiert ist mit Ausnahme einer Startzustands*menge* $Z_{start} \subseteq Z$ und mehreren Endzustandsmengen.
- Ein Lauf ist wie beim NBA definiert mit der Ausnahme, dass der Lauf bei einem beliebigen $z \in Z_{start}$ beginnen kann.
- Ein Lauf ρ ist akzeptierend, falls $inf(\rho) \cap Z^i_{end} \neq \emptyset$ für alle i gilt.

Generalisierter NBA

Satz

Zu jedem GNBA $A = (Z, \Sigma, \delta, Z_{start}, Z^0_{end}, \dots, Z^{k-1}_{end})$ lässt sich ein NBA A' konstruieren mit $L(A') = L(A)$ und $|A'| = 1 + |Z| \cdot (k + 1)$.

Beweis

Wir definieren
$A' = (Z \times \{0, \dots, k-1\}, \Sigma, \Delta, Z_{start} \times \{0\}, Z^0_{end} \times \{0\})$ mit
$\Delta((z, i), a) = \{(z', j) \mid z' \in \delta(z, a)\}$ wobei

$$j := \begin{cases} i + 1 \mod k & \text{, falls } z \in F_i \\ i & \text{, sonst.} \end{cases}$$

Generalisierter NBA

Beweis.

- In der ersten Zustandskomponente wird A simuliert.
- In der zweiten Zustandskomponente wird angegeben aus welcher Endzustandsmenge *als nächstes* ein Endzustand besucht werden soll.

Letzteres funktioniert, da in einem akzeptierenden Lauf A aus allen Mengen F_i unendlich oft einen Zustand besucht. Daraus folgt, dass wenn A einen Zustand aus F_i besucht, irgendwann einer aus $F_{i+1 \mod k}$ besucht werden muss (auch wenn dazwischen vielleicht bereits welche aus einem F_j besucht werden). Damit lässt sich dann leicht argumentieren, dass ein akzeptierender Lauf in A auch einer in A' ist und umgekehrt. $\square$

Die Idee

Wie war noch gleich der Plan?!

Sei M ein LTS und ϕ eine LTL Formel.

- Zu $\neg\phi$ (der Negation der Spezifikation!) konstruieren wir einen (Büchi-)Automaten $A_{\neg\phi}$.
- $A_{\neg\phi}$ akzeptiert genau die Wörter w mit $w \models \neg\phi$.
- Bilde den "Produktautomaten" $M \cap A_{\neg\phi}$.
- Prüfe, ob die akzeptierte Sprache von $M \cap A_{\neg\phi}$ leer ist.

Das Vorgehen (Wiederholung)

1. Büchi-Automaten (und drumherum) (erledigt)
2. Eine alternative (aber äquivalente) Semantik für LTL
3. Damit dann die Konstruktion für $A_{\neg\phi}$
4. Den "Produktautomaten" (erledigt, aber ...)
5. Den Leerheitstest (erledigt)

LTL - alternative Definition

Sei $P = \{p_1, p_2, \ldots\}$ eine Menge von atomaren Formeln. Sei

$$\phi ::= p \mid \neg\phi \mid (\phi \vee \phi) \mid X\phi \mid \phi U \phi$$

und als Abkürzungen:

$$\begin{aligned}
\phi R \psi &:= \neg(\neg\phi U \neg\psi) \\
F\phi &:= \top U \phi \\
G\phi &:= \neg F \neg\phi
\end{aligned}$$

Dabei wird $\phi R \psi$ erfüllt, wenn entweder ψ immer gilt oder ψ bis zu einem Moment gilt, in dem sowohl ϕ als auch ψ gelten.

LTL - alternativ

Definition (LTL - alternativ)

Sei $w = a_0 a_1 \ldots \in (2^P)^\omega$. Die Semantik ist induktiv für alle $i \in \mathbb{N}$ definiert durch:

$$
\begin{array}{lll}
w, i \models p & \text{gdw.} & p \in a_i \\
w, i \models \neg\phi & \text{gdw.} & w, i \not\models \phi \\
w, i \models \phi_1 \vee \phi_2 & \text{gdw.} & w, i \models \phi_1 \textit{ oder } w, i \models \phi_2 \\
w, i \models X\phi & \text{gdw.} & w, i+1 \models \phi \\
w, i \models \phi_1 U \phi_2 & \text{gdw.} & \text{ein } k \geq i \text{ existiert mit } w, k \models \phi_2 \text{ und} \\
 & & \text{für alle } j \text{ mit } i \leq j < k \text{ gilt } w, j \models \phi_1
\end{array}
$$

Ein Wort entspricht dabei den Labels jener Zustand, die bei einem Lauf durch ein LTS besucht werden.

LTL - alternativ

Definition

Sei $\Sigma = (2^P)$, $v \in \Sigma^\omega$ und ϕ eine LTL-Formel. Es ist $v \models \phi$, falls $v, 0 \models \phi$ und $L(\phi) = \{u \mid u \models \phi\}$. Zwei Formeln ϕ und ψ sind äquivalent, $\phi \equiv \psi$, falls $L(\phi) = L(\psi)$ gilt.

Ist z.B. $P = \{C, D\}$, dann ist $\Sigma = \{\emptyset, \{C\}, \{D\}, \{C, D\}\}$. Will man nun an ein bestimmtes $a \in \Sigma$ herankommen, so kann man *charakteristische Formeln* χ_a verwenden:

$$\chi_a := (\bigwedge_{p \in a} p) \wedge (\bigwedge_{p \notin a} \neg p)$$

Will man z.B. eine Formel für die Sprache, die nur aus dem Wort $(\{C\}\{D\})^\omega$ besteht, so geht dies mit:

$$\chi_C \wedge G((\chi_C \Rightarrow X\chi_D) \wedge (\chi_D \Rightarrow X\chi_C))$$

Normalform

Definition (Positive Normalform)

Eine LTL-Formel ist in *positiver Normalform*, wenn sie nur aus Literalen $p, \neg p$ (für ein $p \in P$) und den Operatoren $\vee, \wedge, X, U$ und R aufgebaut ist.

Satz

Zu jeder LTL-Formel ϕ gibt es eine äquivalente LTL-Formel ϕ' in positiver Normalform. Ferner ist $|\phi'| \leq 2 \cdot |\phi|$.

Beweis.

Zum Beweis betrachtet man jeden Operator in negierter und nicht-negierter Form und zeigt, dass man ihn wie angegeben ausdrücken kann. Z.B. ist
$Gp \equiv \neg F \neg p \equiv \neg(\top U \neg p) \equiv \neg(\neg \bot U \neg p) \equiv \bot R p$ und
$\neg(pRq) \equiv \neg\neg(\neg p U \neg q) \equiv (\neg p U \neg q)$. $\square$

Abwicklung von U und R

Satz

Es gilt $pUq \equiv q \vee (p \wedge X(pUq))$.

Beweis.

Sei $w, i \models pUq$. Dann gibt es ein $k \geq i$ mit $w, k \models q$ und $w, j \models p$ für alle j mit $i \leq j < k$. Zwei Fälle:

1. $k = i$. Dann gilt $w, i \models q$.
2. $k > i$. Dann ist $w, i \models p$ und $w, i+1 \models pUq$ (Warum?) und daher $w, i \models X(pUq)$.

Damit gilt $w, i \models q \vee (p \wedge X(pUq))$. □

Abwicklung von U und R

Die Rückrichtung zeigt man analog. Ebenso wie die Abwicklung von R:

Satz

Es gilt $pRq \equiv q \wedge (p \vee X(pRq))$.

Beweis.

Zur Übung... $\square$

Von LTL zum NBA - Die Idee

Wir konstruieren nun einen NBA, der genau die Menge aller Modelle für eine LTL Formel ϕ erkennt.

- Die Idee ist als Zustände Hintikka-Mengen zu benutzen. Diese enthalten gerade die (Unter-)Formeln, die an einer bestimmten Stelle im Modell gelten müssen.
- Diese werden in jedem Schritt nichtdeterministisch geraten.
- Durch die Konsistenz der Hintikka-Mengen wird ausgeschlossen, dass etwas geraten wird, was bereits der Aussagenlogik widerspricht.
- U und R Formeln werden entsprechend ihrer Abwicklung behandelt.
- Dass U nicht unendlich lange abgewickelt wird, wird durch die Akzeptanzbedingung sichergestellt.
- Der X Operator wird durch die Übergänge behandelt.

Von LTL zum NBA - Vorarbeiten

Definition (Fischer-Ladner-Abschluss)

Sei ϕ eine LTL-Formel in positiver Normalform. Der Fischer-Ladner-Abschluss von ϕ ist die kleinste Menge $FL(\phi)$, die ϕ enthält und für die folgendes gilt:

1. $p \vee q \in FL(\phi) \Rightarrow \{p, q\} \subseteq FL(\phi)$
2. $p \wedge q \in FL(\phi) \Rightarrow \{p, q\} \subseteq FL(\phi)$
3. $Xp \in FL(\phi) \Rightarrow p \in FL(\phi)$
4. $pUq \in FL(\phi) \Rightarrow \{p, q, q \vee (p \wedge X(pUq)), p \wedge X(pUq), X(pUq)\}$
5. $pRq \in FL(\phi) \Rightarrow \{p, q, q \wedge (p \vee X(pRq)), p \vee X(pRq), X(pRq)\}$

Von LTL zum NBA - Vorarbeiten

Definition (Hintikka-Mengen)

Sei ϕ eine LTL-Formel in positiver Normalform. Eine Hintikka-Menge für ϕ ist eine Menge $M \subseteq FL(\phi)$ mit

1. $p \vee q \in M \Rightarrow p \in M$ oder $q \in M$
2. $p \wedge q \in M \Rightarrow p \in M$ und $q \in M$
3. $pUq \in M \Rightarrow q \in M$ oder $(p \in M$ und $X(pUq) \in M)$
4. $pRq \in M \Rightarrow q \in M$ und $(p \in M$ oder $X(pRq) \in M)$

Von LTL zum NBA - Vorarbeiten

Definition (Hintikka-Mengen (Teil 2))

- Eine Hintikka-Menge M heißt konsistent, falls es kein $p \in P$ mit $\{p, \neg p\} \subseteq M$ gibt.
- Mit $H(\phi)$ wird die Menge aller konsistenten Hintikka-Mengen bezeichnet.
- Mit $P^+(M)$ wird die Menge aller positiven Literale in M bezeichnet (also $P^+(M) = M \cap P$).
- Mit $P^-(M)$ wird die Menge aller negativen Literale in M bezeichnet.

Von LTL zum NBA - Der Satz

Satz

Zu jeder LTL-Formel ϕ in positiver Normalform kann ein NBA A_ϕ konstruiert werden mit $L(A_\phi) = L(\phi)$. Ferner ist $|A_\phi| \leq 2^{2\cdot|\phi|}$.

Korollar

Zu jeder LTL-Formel ϕ kann ein NBA A_ϕ konstruiert werden mit $L(A_\phi) = L(\phi)$. Ferner ist $|A_\phi| \leq 2^{O(|\phi|)}$.

Von LTL zum NBA - Die Konstruktion

Seien $p_1 U q_1, p_2 U q_2, \ldots, p_k U q_k$ alle in $FL(\phi)$ vorkommenden U-Formeln. Wir definieren

$$A := (H(\phi), \Sigma, \delta, Z_{start}, Z^1_{end}, \ldots, Z^k_{end})$$

wobei:

$$\begin{array}{rcl} Z_{start} & := & \{M \mid \phi \in M\} \\ Z^i_{end} & := & \{M \mid p_i U q_i \in M \Rightarrow q_i \in M\} \end{array}$$

Ferner ist

$$\delta(M, a) := \{M' \mid \forall Xq \in M : q \in M'\}$$

im Fall $P^+(M) \subseteq a$ und $P^-(M) \cap a = \emptyset$ und sonst

$$\delta(M, a) := \emptyset.$$

Von LTL zum NBA - Korrektheit

Satz

Zu jeder LTL-Formel ϕ in positiver Normalform kann ein NBA A_ϕ konstruiert werden mit $L(A_\phi) = L(\phi)$. Ferner ist $|A_\phi| \leq 2^{2\cdot|\phi|}$.

Beweis.

Beweis der Korrektheit der Konstruktion
... als Hausaufgabe $\square$

Der Schluss...

Wir nähern uns dem Ende. Das System wird eher mit einem Transitionssystem modelliert (oder mit einem Formalismus, der in dieses übersetzt wird). Daher brauchen wir dafür einen "Produktautomaten". Zur Wiederholung ...

Transitionssysteme

Definition (Transitionssystem)

Ein *labelled transition system* (LTS) ist ein Tupel $TS = (S, s_0, R, L)$mit

- einer endlichen Menge von Zuständen S,
- einem Startzustand $s_0 \in S$,
- einer links-totalen Übergangsrelation $R \subseteq S \times S$ und
- einer labelling function $L : S \to 2^P$, die jedem Zustand s die Menge der atomaren Formeln $L(s) \subseteq P$ zuweist, die in s gelten.

Transitionssysteme

Definition (Pfad im LTS)

- Ein *Pfad* π in einem LTS $TS = (S, s_0, R, L)$ ist eine unendliche Sequenz von Zuständen

$$\pi = s_1 s_2 s_3 \dots$$

  derart, dass $(s_i, s_{i+1}) \in R$ für alle $i \geq 1$.
- Ein *Lauf* in TS ist ein unendliches Wort $a_0 a_1 \dots \in (2^P)^\omega$, so dass ein Pfad $s_0 s_1 \dots$ existiert mit $a_i = L(s_i)$ für alle i. Mit $L(TS)$ wird die Menge der Läufe von TS bezeichnet.

Ein weiterer Produktautomat

Definition

Sei $TS = (S, s_0, R, L)$ ein LTS über P und $A = (Z, 2^P, \delta, z_0, Z_{end})$ ein NBA. Wir definieren deren *Produkt* als NBA $C := (S \times Z, \{\bullet\}, \Delta, (s_0, z_0), S \times Z_{end})$, wobei

$$\Delta((s,z),\bullet) = \{(s',z') \mid (s,s') \in R \wedge z' \in \delta(z, \lambda(s))\}$$

Satz

Ist TS ein LTS, A ein NBA und C der aus obiger Definition hervorgegangener NBA. Es gilt $L(C) = \emptyset$ gdw. $L(TS) \cap L(A) = \emptyset$.

Beweis.

Zur Übung... $\square$

Finale!

Satz

Das Model-Checking-Problem mit LTS TS und LTL-Formel ϕ lässt sich in Zeit $|TS| \cdot 2^{O(|\phi|)}$ entscheiden.

Beweis.

1. Betrachte $\neg\phi$ und konstruiere NBA $A_{\neg\phi}$ mit $L(A_{\neg\phi}) = L(\neg\phi) = \overline{L(A_\phi)}$. Es ist $|A_{\neg\phi}| = 2^{O(|\phi|)}$.
2. Bilde das Produkt C aus LTS TS und $A_{\neg\phi}$. Nach obigem ist $|C| = |TS| \cdot 2^{O(|\phi|)}$.
3. Nun ist nach dem vorherigen Satz $L(C) = \emptyset$ gdw. $L(TS) \cap L(A) = \emptyset$ und wir können das Leerheitsproblem in linearer Zeit, d.h. hier in $O(|TS| \cdot 2^{O(|\phi|)})$ lösen.

□

Zur Lektüre

Literaturhinweis

Der Inhalt der heutigen Vorlesung ist aus *Automatentheorie und Logik* von Martin Hofmann und Martin Lange. Erschienen im Springer-Verlag, 2011.
Dort

- Kapitel 5 (komplett) für Büchi-Automaten
- Satz 9.4 und Korollar 9.5 aus Kapitel 9 zum Leerheitsproblem
- Kapitel 11 (ohne 11.3) für LTL, die Konvertierung zu NBAs und letztendlich für das Model-Checking-Problem für LTL.

Für zu Hause (1/2)

Für zu Hause:

1. DBA $\lneq$ NBA [bisschen knifflig, aber geht]
2. Korrektheit beim Produktautomaten zweier NBAs [einfach]
3. Beweis des Satzes zum Produkt aus NBA und TS [einfach]

Dann bereitet euch noch (mit Kapitel 11 aus dem eben erwähnten Buch) auf

- Beweis der Konstruktion LTL $\rightarrow$ NBA [schwierig]

vor.

Für zu Hause (2/2)

Satz

Sei F eine geschlossene Formel in Skolemform. F ist genau dann erfüllbar, wenn F ein Herbrand-Modell besitzt.

Satz (Gödel-Herbrand-Skolem)

Für jede geschlossene Formel in Skolemform F gilt: F ist genau dann erfüllbar, wenn die Formelmenge $E(F)$ im aussagenlogischen Sinne erfüllbar ist.

Literatur

Erst selbst versuchen, dann bei Bedarf in die Kapitel 2.4 und 2.5 aus dem Buch *Logik für Informatiker* von Uwe Schöning gucken.