Formale Grundlagen der Informatik 3
Kapitel 5
Verification by Model Checking

Frank Heitmann
heitmann@informatik.uni-hamburg.de

4. Januar 2016

Motivation

Ein Beispiel

Mutual Exclusion: Zwei Prozesse, die eine Ressource nutzen wollen, dies aber nicht zur gleichen Zeit tun sollen (z.B. Schreib-Zugriff auf eine Datei).

Schöne Eigenschaften

- Sicherheit/Safety: Nur ein Prozess ist zur Zeit im *kritischen Abschnitt*.
- Lebendigkeit/Liveness: Wenn ein Prozess in den kritischen Abschnitt will (request), dann darf er diesen irgendwann tatsächlich betreten.
- Non-blocking: Ein Prozess kann stets verlangen, in den kritischen Abschnitt zu gelangen.
- ...

Motivation

Ein Beispiel

Mutual Exclusion: Zwei Prozesse, die eine Ressource nutzen wollen, dies aber nicht zur gleichen Zeit tun sollen (z.B. Schreib-Zugriff auf eine Datei).

Schöne Eigenschaften

- Sicherheit/Safety: Nur ein Prozess ist zur Zeit im *kritischen Abschnitt*.
- Lebendigkeit/Liveness: Wenn ein Prozess in den kritischen Abschnitt will (request), dann darf er diesen irgendwann tatsächlich betreten.
- Non-blocking: Ein Prozess kann stets verlangen, in den kritischen Abschnitt zu gelangen.
- ...

Motivation

Weiter an der Tafel ...

Motivation: Zusammenfassung

Was wir jetzt brauchen:

- eine formale Modellierungssprache ($\Rightarrow$ Modell)
- eine Logik ($\Rightarrow$ Spezifikation)
- Algorithmen, die überprüfen, ob das Modell die Spezifikation erfüllt

Wichtige Anmerkung

Entwirft man die Modellierungssprache und die Logik sinnvoll, so hängen Modell M und Formel F zusammen und es macht Sinn von $M \models F$ zu sprechen. Dann ist man beim Erfüllbarkeitsproblem oder beim Model-Checking-Problem. Das Äquivalent in der Aussagenlogik wäre, gegeben eine Belegung $\mathcal{A}$ und eine Formel F, zu prüfen, ob $\mathcal{A} \models F$ gilt, ob also F unter $\mathcal{A}$ wahr ist.

Motivation: Zusammenfassung

Was wir jetzt brauchen:

- eine formale Modellierungssprache ($\Rightarrow$ Modell)
- eine Logik ($\Rightarrow$ Spezifikation)
- Algorithmen, die überprüfen, ob das Modell die Spezifikation erfüllt

Wichtige Anmerkung

Entwirft man die Modellierungssprache und die Logik sinnvoll, so hängen Modell M und Formel F zusammen und es macht Sinn von $M \models F$ zu sprechen. Dann ist man beim Erfüllbarkeitsproblem oder beim Model-Checking-Problem. Das Äquivalent in der Aussagenlogik wäre, gegeben eine Belegung $\mathcal{A}$ und eine Formel F, zu prüfen, ob $\mathcal{A} \models F$ gilt, ob also F unter $\mathcal{A}$ wahr ist.

Motivation: Tool-Support

Ein Tool könnte jetzt:

- Eine Sprache (textuell oder visuell) zum Modellieren anbieten
- Per Knopf-Druck die Algorithmen starten

Anmerkung

Für die Spezifikation erlaubt das Tool meist nur eine einfache Eingabe der Formel der jeweiligen Logik.

Aussagenlogik

Wir haben

- Eine abzählbare Menge $V = \{x_1, x_2, \ldots\}$ von *aussagenlogischen Variablen* oder *Atomen*.
- Das *Alphabet* besteht dann aus V, den *Junktoren* $\wedge$, $\vee$ und $\neg$ (für "and", "or" und "not") und den Klammern (und).

Aussagenlogik

Wir haben

- Eine abzählbare Menge $V = \{x_1, x_2, \ldots\}$ von *aussagenlogischen Variablen* oder *Atomen*.
- Das *Alphabet* besteht dann aus V, den *Junktoren* $\wedge$, $\vee$ und $\neg$ (für "and", "or" und "not") und den Klammern (und).

Aussagenlogik: Syntax

Definition (Syntax der Aussagenlogik)

Die *wohlgeformten Ausdrücke/Formeln* der *Aussagenlogik* (AL) werden induktiv definiert durch

1. Jede Variable $x \in V$ ist eine Formel.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$ und $(\phi \vee \psi)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln der Aussagenlogik.

Aussagenlogik: Syntax

Definition (Syntax der Aussagenlogik)

Die *wohlgeformten Ausdrücke/Formeln* der *Aussagenlogik* (AL) werden induktiv definiert durch

1. Jede Variable $x \in V$ ist eine Formel.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$ und $(\phi \vee \psi)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln der Aussagenlogik.

Aussagenlogik: Syntax

Definition (Syntax der Aussagenlogik)

Die *wohlgeformten Ausdrücke/Formeln* der *Aussagenlogik* (AL) werden induktiv definiert durch

1. Jede Variable $x \in V$ ist eine Formel.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$ und $(\phi \vee \psi)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln der Aussagenlogik.

Aussagenlogik: Syntax

Definition (Syntax der Aussagenlogik)

Die *wohlgeformten Ausdrücke/Formeln* der *Aussagenlogik* (AL) werden induktiv definiert durch

1. Jede Variable $x \in V$ ist eine Formel.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$ und $(\phi \vee \psi)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln der Aussagenlogik.

Aussagenlogik: Syntax

Definition (Syntax der Aussagenlogik)

Die *wohlgeformten Ausdrücke/Formeln* der *Aussagenlogik* (AL) werden induktiv definiert durch

1. Jede Variable $x \in V$ ist eine Formel.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$ und $(\phi \vee \psi)$.
4. Nur Formeln, die durch endliche häufige Anwendungen der Regeln 1-3 entstehen, sind wohlgeformte Formeln der Aussagenlogik.

Aussagenlogik: Syntax (Alternative)

Alternative Definition der Syntax durch die folgende Grammatik, wobei x eine Variable darstellt:

$$\phi ::= x \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi)$$

Aussagenlogik: Abkürzungen etc.

Klammern werden auf übliche Weise eingespart. Ferner haben wir folgende Abkürzungen:

$$\begin{aligned} \phi \Rightarrow \psi &:= \neg\phi \vee \psi \\ \phi \Leftrightarrow \psi &:= (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi) \\ \top &:= (x \vee \neg x) \\ \bot &:= \neg\top \end{aligned}$$

für die *Implikation*, die *Biimplikation*, die *Tautologie* und die *Kontradiktion*.

Aussagenlogik: Begriffe

- Eine Formel, die nur aus einer einzelnen Variable besteht, nennen wir *atomare Formel*.
- Eine Formel der Form $\neg\phi$ nennen wir eine *Negation*.
- $(\phi \wedge \psi)$ ist eine *Konjunktion*.
- $(\phi \vee \psi)$ ist eine *Disjunktion*.
- Ein *Literal* ist eine atomare Formel oder die Negation einer atomaren Formel, also x oder $\neg x$ für ein $x \in V$. Im ersten Fall nenne wir die Formel auch *positives Literal* und im zweiten Fall *negatives Literal*.

Aussagenlogik: Begriffe

- Eine Formel, die nur aus einer einzelnen Variable besteht, nennen wir *atomare Formel*.
- Eine Formel der Form $\neg\phi$ nennen wir eine *Negation*.
- $(\phi \wedge \psi)$ ist eine *Konjunktion*.
- $(\phi \vee \psi)$ ist eine *Disjunktion*.
- Ein *Literal* ist eine atomare Formel oder die Negation einer atomaren Formel, also x oder $\neg x$ für ein $x \in V$. Im ersten Fall nenne wir die Formel auch *positives Literal* und im zweiten Fall *negatives Literal*.

Aussagenlogik: Begriffe

- Eine Formel, die nur aus einer einzelnen Variable besteht, nennen wir *atomare Formel*.
- Eine Formel der Form $\neg\phi$ nennen wir eine *Negation*.
- $(\phi \wedge \psi)$ ist eine *Konjunktion*.
- $(\phi \vee \psi)$ ist eine *Disjunktion*.
- Ein *Literal* ist eine atomare Formel oder die Negation einer atomaren Formel, also x oder $\neg x$ für ein $x \in V$. Im ersten Fall nenne wir die Formel auch *positives Literal* und im zweiten Fall *negatives Literal*.

Aussagenlogik: Begriffe

- Eine Formel, die nur aus einer einzelnen Variable besteht, nennen wir *atomare Formel*.
- Eine Formel der Form $\neg\phi$ nennen wir eine *Negation*.
- $(\phi \wedge \psi)$ ist eine *Konjunktion*.
- $(\phi \vee \psi)$ ist eine *Disjunktion*.
- Ein *Literal* ist eine atomare Formel oder die Negation einer atomaren Formel, also x oder $\neg x$ für ein $x \in V$. Im ersten Fall nenne wir die Formel auch *positives Literal* und im zweiten Fall *negatives Literal*.

Aussagenlogik: Begriffe

- Eine Formel, die nur aus einer einzelnen Variable besteht, nennen wir *atomare Formel*.
- Eine Formel der Form $\neg\phi$ nennen wir eine *Negation*.
- $(\phi \wedge \psi)$ ist eine *Konjunktion*.
- $(\phi \vee \psi)$ ist eine *Disjunktion*.
- Ein *Literal* ist eine atomare Formel oder die Negation einer atomaren Formel, also x oder $\neg x$ für ein $x \in V$. Im ersten Fall nenne wir die Formel auch *positives Literal* und im zweiten Fall *negatives Literal*.

Aussagenlogik: Begriffe

- Eine Formel ist in *konjunktiver Normalform* (KNF) wenn sie eine Konjunktion von Disjunktionen von Literalen ist. Z.B.

$$(x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_4 \vee x_5) \wedge (\neg x_3 \vee x_4)$$

  Eine Disjunktion von Literalen wird *Klausel* genannt. Eine Formel ist in 3-*KNF* wenn jede Klausel genau drei Literale enthält.

- Eine Formel ist in *disjunktiver Normalform* (DNF) wenn sie eine Disjunktion von Konjunktionen von Literalen ist.

Aussagenlogik: Begriffe

- Eine Formel ist in *konjunktiver Normalform* (KNF) wenn sie eine Konjunktion von Disjunktionen von Literalen ist. Z.B.

$$(x_2 \vee \neg x_4) \wedge (x_1 \vee \neg x_4 \vee x_5) \wedge (\neg x_3 \vee x_4)$$

  Eine Disjunktion von Literalen wird *Klausel* genannt. Eine Formel ist in 3-*KNF* wenn jede Klausel genau drei Literale enthält.
- Eine Formel ist in *disjunktiver Normalform* (DNF) wenn sie eine Disjunktion von Konjunktionen von Literalen ist.

Aussagenlogik: Semantik

Um die Semantik zu definieren, benötigen wir ein Modell bzgl. dessen die Wahrheit der Formel definiert werden kann.

Definition (Aussagenlogisches Modell)

Ein *Modell* einer aussagenlogischen Formel ϕ ist eine totale Funktion $\mathcal{A} : V \to \{0, 1\}$.

- Ist $\mathcal{A}(x) = 0$, so ist x *falsch* (in $\mathcal{A}$).
- Ist $\mathcal{A}(x) = 1$, so ist x *wahr* (in $\mathcal{A}$).

Aussagenlogik: Semantik

Um die Semantik zu definieren, benötigen wir ein Modell bzgl. dessen die Wahrheit der Formel definiert werden kann.

Definition (Aussagenlogisches Modell)

Ein *Modell* einer aussagenlogischen Formel ϕ ist eine totale Funktion $\mathcal{A} : V \to \{0, 1\}$.

- Ist $\mathcal{A}(x) = 0$, so ist x *falsch* (in $\mathcal{A}$).
- Ist $\mathcal{A}(x) = 1$, so ist x *wahr* (in $\mathcal{A}$).

Aussagenlogik: Semantik

Definition (Semantik der AL)

Sei ϕ eine Formel und $\mathcal{A}$ ein Modell. $\mathcal{A}$ wird induktiv auf ϕ erweitert:

$$
\begin{array}{lll}
\mathcal{A} \models p & \text{gdw.} & \mathcal{A}(p) = 1, \text{für ein } p \in V \\
\mathcal{A} \models \neg\phi & \text{gdw.} & \mathcal{A} \models \phi \textit{ nicht} \text{ gilt, notiert als } \mathcal{A} \not\models \phi \\
\mathcal{A} \models \phi_1 \wedge \phi_2 & \text{gdw.} & \mathcal{A} \models \phi_1 \textit{ und } \mathcal{A} \models \phi_2 \text{ gilt} \\
\mathcal{A} \models \phi_1 \vee \phi_2 & \text{gdw.} & \mathcal{A} \models \phi_1 \textit{ oder } \mathcal{A} \models \phi_2 \text{ gilt}
\end{array}
$$

Aussagenlogik: Semantik

Definition (Semantik der AL)

Sei ϕ eine Formel und $\mathcal{A}$ ein Modell. $\mathcal{A}$ wird induktiv auf ϕ erweitert:

$\mathcal{A} \models p$	gdw.	$\mathcal{A}(p) = 1$, für ein $p \in V$
$\mathcal{A} \models \neg\phi$	gdw.	$\mathcal{A} \models \phi$ *nicht* gilt, notiert als $\mathcal{A} \not\models \phi$
$\mathcal{A} \models \phi_1 \wedge \phi_2$	gdw.	$\mathcal{A} \models \phi_1$ *und* $\mathcal{A} \models \phi_2$ gilt
$\mathcal{A} \models \phi_1 \vee \phi_2$	gdw.	$\mathcal{A} \models \phi_1$ *oder* $\mathcal{A} \models \phi_2$ gilt

Aussagenlogik: Semantik (Begriffe)

- $\mathcal{A} \models \phi$ bezeichnet, dass ϕ im Modell $\mathcal{A}$ wahr ist, d.h. dass $\mathcal{A}(\phi) = 1$.
- Ist $\mathcal{A}(\phi) = 0$, so schreiben wir $\mathcal{A} \not\models \phi$.
- Gilt $\mathcal{A} \models \phi$, sagen wir $\mathcal{A}$ *erfüllt* ϕ oder ϕ ist *wahr in* $\mathcal{A}$.
- Gilt $\mathcal{A} \not\models \phi$, sagen wir $\mathcal{A}$ *falsifiziert* ϕ oder ϕ ist *falsch* in $\mathcal{A}$.

Aussagenlogik: Semantik (Begriffe)

- $\mathcal{A} \models \phi$ bezeichnet, dass ϕ im Modell $\mathcal{A}$ wahr ist, d.h. dass $\mathcal{A}(\phi) = 1$.
- Ist $\mathcal{A}(\phi) = 0$, so schreiben wir $\mathcal{A} \not\models \phi$.
- Gilt $\mathcal{A} \models \phi$, sagen wir $\mathcal{A}$ *erfüllt* ϕ oder ϕ ist *wahr in* $\mathcal{A}$.
- Gilt $\mathcal{A} \not\models \phi$, sagen wir $\mathcal{A}$ *falsifiziert* ϕ oder ϕ ist *falsch* in $\mathcal{A}$.

Aussagenlogik: Semantik (Begriffe)

- $\mathcal{A} \models \phi$ bezeichnet, dass ϕ im Modell $\mathcal{A}$ wahr ist, d.h. dass $\mathcal{A}(\phi) = 1$.
- Ist $\mathcal{A}(\phi) = 0$, so schreiben wir $\mathcal{A} \not\models \phi$.
- Gilt $\mathcal{A} \models \phi$, sagen wir $\mathcal{A}$ *erfüllt* ϕ oder ϕ ist *wahr in* $\mathcal{A}$.
- Gilt $\mathcal{A} \not\models \phi$, sagen wir $\mathcal{A}$ *falsifiziert* ϕ oder ϕ ist *falsch* in $\mathcal{A}$.

Aussagenlogik: Semantik (Begriffe)

- $\mathcal{A} \models \phi$ bezeichnet, dass ϕ im Modell $\mathcal{A}$ wahr ist, d.h. dass $\mathcal{A}(\phi) = 1$.
- Ist $\mathcal{A}(\phi) = 0$, so schreiben wir $\mathcal{A} \not\models \phi$.
- Gilt $\mathcal{A} \models \phi$, sagen wir $\mathcal{A}$ *erfüllt* ϕ oder ϕ ist *wahr in* $\mathcal{A}$.
- Gilt $\mathcal{A} \not\models \phi$, sagen wir $\mathcal{A}$ *falsifiziert* ϕ oder ϕ ist *falsch* in $\mathcal{A}$.

Aussagenlogik: Semantik (Begriffe)

- Eine Formel ϕ ist *erfüllbar* wenn ein Modell $\mathcal{A}$ existiert mit $\mathcal{A} \models \phi$.
- ϕ ist falsifizierbar wenn ein $\mathcal{A}$ mit $\mathcal{A} \not\models \phi$ existiert.
- ϕ ist *gültig* wenn ϕ in allen Modellen wahr ist. Wir schreiben dafür $\models \phi$. ϕ ist dann eine *Tautologie*.
- ϕ ist eine *Kontradiktion*, notiert durch $\phi \models$, wenn kein Modell ϕ erfüllt.
- Zwei Formeln ϕ und ψ sind *äquivalent*, wenn $\mathcal{A} \models \phi$ gdw. $\mathcal{A} \models \psi$ für alle Modelle $\mathcal{A}$ gilt.

Aussagenlogik: Semantik (Begriffe)

- Eine Formel ϕ ist *erfüllbar* wenn ein Modell $\mathcal{A}$ existiert mit $\mathcal{A} \models \phi$.
- ϕ ist falsifizierbar wenn ein $\mathcal{A}$ mit $\mathcal{A} \not\models \phi$ existiert.
- ϕ ist *gültig* wenn ϕ in allen Modellen wahr ist. Wir schreiben dafür $\models \phi$. ϕ ist dann eine *Tautologie*.
- ϕ ist eine *Kontradiktion*, notiert durch $\phi \models$, wenn kein Modell ϕ erfüllt.
- Zwei Formeln ϕ und ψ sind *äquivalent*, wenn $\mathcal{A} \models \phi$ gdw. $\mathcal{A} \models \psi$ für alle Modelle $\mathcal{A}$ gilt.

Aussagenlogik: Semantik (Begriffe)

- Eine Formel ϕ ist *erfüllbar* wenn ein Modell $\mathcal{A}$ existiert mit $\mathcal{A} \models \phi$.
- ϕ ist falsifizierbar wenn ein $\mathcal{A}$ mit $\mathcal{A} \not\models \phi$ existiert.
- ϕ ist *gültig* wenn ϕ in allen Modellen wahr ist. Wir schreiben dafür $\models \phi$. ϕ ist dann eine *Tautologie*.
- ϕ ist eine *Kontradiktion*, notiert durch $\phi \models$, wenn kein Modell ϕ erfüllt.
- Zwei Formeln ϕ und ψ sind *äquivalent*, wenn $\mathcal{A} \models \phi$ gdw. $\mathcal{A} \models \psi$ für alle Modelle $\mathcal{A}$ gilt.

Aussagenlogik: Semantik (Begriffe)

- Eine Formel ϕ ist *erfüllbar* wenn ein Modell $\mathcal{A}$ existiert mit $\mathcal{A} \models \phi$.
- ϕ ist falsifizierbar wenn ein $\mathcal{A}$ mit $\mathcal{A} \not\models \phi$ existiert.
- ϕ ist *gültig* wenn ϕ in allen Modellen wahr ist. Wir schreiben dafür $\models \phi$. ϕ ist dann eine *Tautologie*.
- ϕ ist eine *Kontradiktion*, notiert durch $\phi \models$, wenn kein Modell ϕ erfüllt.
- Zwei Formeln ϕ und ψ sind *äquivalent*, wenn $\mathcal{A} \models \phi$ gdw. $\mathcal{A} \models \psi$ für alle Modelle $\mathcal{A}$ gilt.

Aussagenlogik: Semantik (Begriffe)

- Eine Formel ϕ ist *erfüllbar* wenn ein Modell $\mathcal{A}$ existiert mit $\mathcal{A} \models \phi$.
- ϕ ist falsifizierbar wenn ein $\mathcal{A}$ mit $\mathcal{A} \not\models \phi$ existiert.
- ϕ ist *gültig* wenn ϕ in allen Modellen wahr ist. Wir schreiben dafür $\models \phi$. ϕ ist dann eine *Tautologie*.
- ϕ ist eine *Kontradiktion*, notiert durch $\phi \models$, wenn kein Modell ϕ erfüllt.
- Zwei Formeln ϕ und ψ sind *äquivalent*, wenn $\mathcal{A} \models \phi$ gdw. $\mathcal{A} \models \psi$ für alle Modelle $\mathcal{A}$ gilt.

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

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 $\vee$.

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 $\vee$.

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 $\vee$.

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 $\vee$.

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 $\vee$.

LTL: Semantik

LTL Formeln werden entlang der Pfade eines Transitionssystems interpretiert. Das Transitionssystem übernimmt also die Rolle des Modells in der Aussagenlogik.

Anmerkung

Gelabelte Transitionssysteme werden zu Ehren von Saul Kripke auch Kripke-Strukturen genannt. Ihre Definition ist in der Literatur leicht verschieden. Oft werden z.B. Kantenbeschriftungen verwendet.

LTL: Semantik

LTL Formeln werden entlang der Pfade eines Transitionssystems interpretiert. Das Transitionssystem übernimmt also die Rolle des Modells in der Aussagenlogik.

Anmerkung

Gelabelte Transitionssysteme werden zu Ehren von Saul Kripke auch Kripke-Strukturen genannt. Ihre Definition ist in der Literatur leicht verschieden. Oft werden z.B. Kantenbeschriftungen verwendet.

LTL: Semantik

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.

LTL: Semantik

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.

LTL: Semantik

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.

LTL: Semantik

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.

LTL: Semantik

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.

LTL: Semantik

Definition (Pfad im LTS)

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

$$\pi = s_1 s_2 s_3 \ldots$$

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 $\pi^i = s_i s_{i+1} \ldots$
- 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 (Pfad im LTS)

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

$$\pi = s_1 s_2 s_3 \ldots$$

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 $\pi^i = s_i s_{i+1} \ldots$
- 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 (Pfad im LTS)

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

$$\pi = s_1 s_2 s_3 \ldots$$

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 $\pi^i = s_i s_{i+1} \ldots$
- 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 (Pfad im LTS)

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

$$\pi = s_1 s_2 s_3 \ldots$$

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 $\pi^i = s_i s_{i+1} \ldots$
- 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 $\pi = s_1 s_2 \ldots$ 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:

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

LTL: Semantik

Definition (Semantik von LTL (I))

Sei $M = (S, s_0, R, L)$ ein LTS und $\pi = s_1 s_2 \ldots$ 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:

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

LTL: Semantik

Definition (Semantik von LTL (II))

$$
\begin{array}{lll}
M, \pi \models X\phi & \text{gdw.} & M, \pi^2 \models \phi \\
M, \pi \models F\phi & \text{gdw.} & M, \pi^i \models \phi \text{ für ein } i \geq 1 \\
M, \pi \models G\phi & \text{gdw.} & M, \pi^i \models \phi \text{ für alle } i \geq 1 \\
M, \pi \models \phi_1 U \phi_2 & \text{gdw.} & \text{ein } i \geq 1 \text{ existiert mit } M, \pi^i \models \phi_2 \\
 & & \text{und für alle } j < i \; M, \pi^j \models \phi_1 \text{ gilt.}
\end{array}
$$

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 gilt: $M, \pi \models \phi$ gdw. $M, \pi \models \psi$.

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 gilt: $M, \pi \models \phi$ gdw. $M, \pi \models \psi$.

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 gilt: $M, \pi \models \phi$ gdw. $M, \pi \models \psi$.

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 gilt: $M, \pi \models \phi$ gdw. $M, \pi \models \psi$.

LTL: Semantik. Beispiel

Beispiel

Beispiel: Siehe Tafel... ;-)

LTL: Spezifikationsmuster

- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $G(req \Rightarrow F\ ack)$
- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $G(F\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $F(G\ deadlock)$
- Tritt ein Ereignis p entlang eines Pfades unendlich oft auf, dann tritt auch das Ereignis q auf: $GF\ p \Rightarrow F\ q$

LTL: Spezifikationsmuster

- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $G(req \Rightarrow F\ ack)$
- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $G(F\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $F(G\ deadlock)$
- Tritt ein Ereignis p entlang eines Pfades unendlich oft auf, dann tritt auch das Ereignis q auf: $GF\ p \Rightarrow F\ q$

LTL: Spezifikationsmuster

- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $G(req \Rightarrow F\ ack)$
- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $G(F\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $F(G\ deadlock)$
- Tritt ein Ereignis p entlang eines Pfades unendlich oft auf, dann tritt auch das Ereignis q auf: $GF\ p \Rightarrow F\ q$

LTL: Spezifikationsmuster

- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $G(req \Rightarrow F\ ack)$
- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $G(F\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $F(G\ deadlock)$
- Tritt ein Ereignis p entlang eines Pfades unendlich oft auf, dann tritt auch das Ereignis q auf: $GF\ p \Rightarrow F\ q$

LTL: Äquivalenzen

Bisweilen werden weitere Junktoren wie z.B. $\Rightarrow$ für die Implikation oder R für "release" benutzt. Diese können durch die Äquivalenzen $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$ und $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$ ausgedrückt werden. Unsere Junktoren bilden ein "adequate set of connectives" für LTL, d.h. alle andern Junktoren können durch sie ausgedrückt werden. Tatsächlich gibt es sogar kleinere Sets.

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

ist ein solches. F und G werden dann durch $F\phi := \top U \phi$ and $G\phi := \neg F \neg\phi$ definiert. Eine kleine Anzahl an Junktoren ist insb. bei Model Checking Algorithmen hilfreich, da man sich um weniger Fälle kümmern muss.

LTL: Äquivalenzen

Bisweilen werden weitere Junktoren wie z.B. $\Rightarrow$ für die Implikation oder R für "release" benutzt. Diese können durch die Äquivalenzen $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$ und $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$ ausgedrückt werden. Unsere Junktoren bilden ein "adequate set of connectives" für LTL, d.h. alle andern Junktoren können durch sie ausgedrückt werden. Tatsächlich gibt es sogar kleinere Sets.

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

ist ein solches. F und G werden dann durch $F\phi := \top U \phi$ and $G\phi := \neg F \neg\phi$ definiert. Eine kleine Anzahl an Junktoren ist insb. bei Model Checking Algorithmen hilfreich, da man sich um weniger Fälle kümmern muss.

LTL: Äquivalenzen

Bisweilen werden weitere Junktoren wie z.B. $\Rightarrow$ für die Implikation oder R für "release" benutzt. Diese können durch die Äquivalenzen $\phi_1 \Rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$ und $\phi_1 R \phi_2 \equiv \neg(\neg\phi_1 U \neg\phi_2)$ ausgedrückt werden. Unsere Junktoren bilden ein "adequate set of connectives" für LTL, d.h. alle andern Junktoren können durch sie ausgedrückt werden. Tatsächlich gibt es sogar kleinere Sets.

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

ist ein solches. F und G werden dann durch $F\phi := \top U \phi$ and $G\phi := \neg F \neg\phi$ definiert. Eine kleine Anzahl an Junktoren ist insb. bei Model Checking Algorithmen hilfreich, da man sich um weniger Fälle kümmern muss.

CTL

In der Computation Tree Logic (CTL) ist es möglich über die Pfade in einem Transistionssystem zu argumentieren. Hierzu wird die Logik um Pfadquantoren ‘A’ und ‘E’ erweitert. Die Semantik wird dann über unendliche, gerichtete Bäume definiert, die man durch ein “unfolding” des Transitionssystems in einen Erreichbarkeitsbaum erhält.

CTL

In der Computation Tree Logic (CTL) ist es möglich über die Pfade in einem Transistionssystem zu argumentieren. Hierzu wird die Logik um Pfadquantoren ‘A’ und ‘E’ erweitert. Die Semantik wird dann über unendliche, gerichtete Bäume definiert, die man durch ein “unfolding” des Transitionssystems in einen Erreichbarkeitsbaum erhält.

CTL

In der Computation Tree Logic (CTL) ist es möglich über die Pfade in einem Transistionssystem zu argumentieren. Hierzu wird die Logik um Pfadquantoren ‘A’ und ‘E’ erweitert. Die Semantik wird dann über unendliche, gerichtete Bäume definiert, die man durch ein “unfolding” des Transitionssystems in einen Erreichbarkeitsbaum erhält.

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: 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: 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 (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 $\pi = s_1 s_2 \ldots$ beginnend bei s ein $i \geq 1$ existiert mit $M, s_i \models \phi$.
$M, s \models AG\phi$	gdw.	für alle Pfade $\pi = s_1 s_2 \ldots$ 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 $\pi = s_1 s_2 \ldots$ 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.

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.

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.

CTL: Semantik. Beispiel

Beispiel

Beispiel: Siehe Tafel... ;-)

CTL: Spezifikationsmuster

- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $AG(AF\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $AF(AG\ deadlock)$
- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $AG(req \Rightarrow AF\ ack)$
- Man kann immer zu einem sicheren Zustand kommen: $AG(EF\ safe)$

CTL: Spezifikationsmuster

- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $AG(AF\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $AF(AG\ deadlock)$
- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $AG(req \Rightarrow AF\ ack)$
- Man kann immer zu einem sicheren Zustand kommen: $AG(EF\ safe)$

CTL: Spezifikationsmuster

- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $AG(AF\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $AF(AG\ deadlock)$
- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $AG(req \Rightarrow AF\ ack)$
- Man kann immer zu einem sicheren Zustand kommen: $AG(EF\ safe)$

CTL: Spezifikationsmuster

- Ein Prozess wird auf jedem Pfad unendlich oft aktiviert: $AG(AF\ act)$
- Was auch immer man tut, ein Prozess wird immer zu einem Deadlock führen: $AF(AG\ deadlock)$
- Für jeden Zustand gilt: Wird eine Ressource angefordert, wird sie irgendwann zur Verfügung gestellt: $AG(req \Rightarrow AF\ ack)$
- Man kann immer zu einem sicheren Zustand kommen: $AG(EF\ safe)$

CTL: Äquivalenzen

Oft - und besonders beim Model Checking - benutzt man ein Set von “adequate connectives”, d.h. ein Set von Junktoren, dass Ausdrucksstark genug ist, um jede Formel der Logik auszudrücken.

Für CTL ist ein solches Set z.B.

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

Alle Formeln, die andere Junktoren benutzen, können stets durch eine äquivalente Formeln ersetzt werden, die nur Junktoren obiger Menge benutzen.

CTL: Äquivalenzen

Oft - und besonders beim Model Checking - benutzt man ein Set von “adequate connectives”, d.h. ein Set von Junktoren, dass Ausdrucksstark genug ist, um jede Formel der Logik auszudrücken.

Für CTL ist ein solches Set z.B.

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

Alle Formeln, die andere Junktoren benutzen, können stets durch eine äquivalente Formeln ersetzt werden, die nur Junktoren obiger Menge benutzen.

CTL: Äquivalenzen

Gängige Abkürzungen bzw. Äquivalenzen:

$$
\begin{aligned}
\phi_1 \vee \phi_2 &\equiv \neg(\neg\phi_1 \wedge \neg\phi_2) \\
\top &\equiv \phi \vee \neg\phi \\
EF\phi &\equiv E[\top U \phi] \\
AG\phi &\equiv \neg EF\neg\phi \\
AF\phi &\equiv \neg EG\neg\phi \\
AX\phi &\equiv \neg EX\neg\phi \\
A[\phi_1 U \phi_2] &\equiv \neg(E[\neg\phi_2 U \neg(\phi_1 \vee \phi_2)] \vee EG\neg\phi_2)
\end{aligned}
$$

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

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

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.

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.

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, 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, 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, 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

Zur heutigen 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.

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

Zur heutigen 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.

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

Zur heutigen 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.

Ausblick: Model Checking

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 $M \models \phi$?

Ausblick: Model Checking

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.

Ausblick: Model Checking

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.