2.2 Linear-Time Logic

Zur Vorlesung

Embedded Systems

WS 14/15

Reiner Kolla

Linear-Zeit Logik

Linear-Zeit Logik ist eine Logik, die auf Sequenzen von Belegungen mit atomaren Eigenschaften basiert. Die Zeit ist diskret und schreitet linear voran. Es gibt nur eine mögliche Zukunft.

Sie hat ihren Ursprung in der Philosophie/Logik und wird seit Ende der 70er Jahre zur formalen Spezifikation von temporalen Korrektheitseigenschaften eingesetzt.

Prominentester Vertreter ist **LTL**.

LTL – eine Einführung

Sei *AP* wieder die Menge der atomaren Aussagen.

Dann korrespondiert das Labeling X⊆AP zu der Tatsache, dass alle Eigenschaften xϵX in dem Zustand gelten und alle yϵAP\X in diesem Zustand nicht gelten.

LTL Formeln werden nun über Mengen von unendlichen Folgen von Labelings interpretiert. D.h. wir betrachten Sequenzen

$$\sigma \in (2^{\mathrm{AP}})^{\omega}$$

wobei für ein Alphabet Σ, Σ^{ω} die Menge aller unendlichen Folgen über Σ sei.

Für $\sigma \in (2^{AP})^{\omega}$ sei

$\sigma(\mathrm{i}) \subseteq$ AP das i-te Element der Folge, und

$\sigma^{i} \in (2^{\mathrm{AP}})^{\omega}$ die unendliche Folge $\sigma(\mathrm{i})\ \sigma(\mathrm{i}+1)\ \sigma(\mathrm{i}+2)$ ...

Syntax von LTL

Sei *AP* eine Menge von atomaren Aussagen. Dann ist die Menge der LTL Formeln über *AP* wie folgt definiert:

1. Jedes $p \in AP$ ist eine LTL Formel.
2. Sind ϕ_1 und ϕ_2 Formeln, dann auch
   $\neg\phi_1$, $\phi_1 \vee \phi_2$, $\boldsymbol{X}\phi_1$, $\phi_1\ \boldsymbol{U}\phi_2$
3. Nichts sonst ist eine LTL Formel.

Dies ist eine sehr minimalistische Definition. ***X*** (next) und ***U*** (until) sind temporale Operatoren.

LTL Formeln werden über (unendlichen) Folgen von Teilmengen atomarer Aussagen interpretiert. Die Semantik einer LTL Formel ϕ ist die Menge aller Folgen $\sigma \in (2^{\mathrm{AP}})^{\omega}$, die ϕ erfüllen, d.h.

$$[\![\phi]\!] = \{\sigma \mid \sigma \vDash \phi\}$$

Erfüllung einer LTL Formel

Sei nun $\sigma \in (2^{AP})^{\omega}$ und sei ϕ eine LTL Formel.

Dann gilt $\sigma \models \phi$ ("σ erfüllt ϕ") nach folgender Fallunterscheidung über die Struktur von ϕ

$\sigma \models p$	falls $p \in AP$ und $p \in \sigma(0)$
$\sigma \models \neg\phi$	falls $\sigma \not\models \phi$
$\sigma \models \phi_1 \vee \phi_2$	falls $\sigma \models \phi_1$ oder $\sigma \models \phi_2$
$\sigma \models \mathbf{X}\phi$	falls $\sigma^1 \models \phi$
$\sigma \models \phi_1 \mathbf{U} \phi_2$	falls $\exists i: (\sigma^i \models \phi_2 \wedge \forall k < i: \sigma^k \models \phi_1)$

Beispiel

Sei $AP = \{p,q,r\}$ und sei $\sigma = \{p\}\{q\}\{p\}^{\omega}$

Welche der folgenden Formeln werden durch σ erfüllt?

p	q
$\mathbf{X}q$	$\mathbf{X}\neg p$
$p\mathbf{U}q$	$q\mathbf{U}p$
$(p \vee q)\mathbf{U}r$	

Nützliche „Abkürzungen“ für LTL Formeln:

Wir erweitern LTL um ein paar Notationen, deren Bedeutung wir aber auf unser Minimal LTL zurückführen. Wenn man Eigenschaften formulieren will, kann man den Komfort nutzen, in Beweisen zu allgemeinen Eigenschaften der LTL, muss man dazu nichts zeigen:

$\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$	$\mathbf{F}\phi \equiv true\ \mathbf{U}\phi$
$\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$	$\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$
$true \equiv a \vee \neg a$	$\phi_1 \mathbf{W}\ \phi_2 \equiv (\phi_1 \mathbf{U}\ \phi_2) \vee \mathbf{G}\ \phi_1$
$false \equiv \neg\ true$	$\phi_1 \mathbf{R}\ \phi_2 \equiv \neg(\neg\phi_1 \mathbf{U}\ \neg\phi_2)$

Dabei steht

F für „finally“ (irgendwann), **G** für „globally“ (immer),

W für „weak until“, und **R** für „releases“

Formulierung von Eigenschaften -- Beispiele

Erreichbarkeit: $\mathbf{G}\neg(\text{HighwayGreen} \wedge \text{FarmroadGreen})$

Dies ist eine spezielle Sicherheitsbedingung, weil man nur die Unerreichbarkeit einer gefährlichen Situation fordert. Manchmal möchte man bei der Sicherheit auch zeitliche Bezüge zwischen Eigenschaften haben:

Sicherheit: $(\neg MotorLäuft)\mathbf{W}\ TürVerschlossen$

Lebendigkeit: $(\neg MotorLäuft)\mathbf{U}\ TürVerschlossen$

Man möchte neben der Sicherheit auch dass die Zieleigenschaft eintritt.

Oder sogar unendlich oft eintritt: $\mathbf{GF}(\text{FarmroadGreen})$

Dass etwas irgendwann für immer gilt: $\mathbf{FG}(\neg\ \text{MotorLäft})$

Oder Fairness herrscht: $\boldsymbol{G}(req_1 \rightarrow \boldsymbol{F}grant_1)$

Tautologie, Unerfüllbarkeit, Äquivalenz

Wir können die Begriffe aus der Aussagenlogik auf temporale Logik übertragen:

Tautologie: Jede Formel ϕ mit $[\![\phi]\!] = (2^{AP})^{\omega}$ heißt Tautologie

Unerfüllbarkeit: Jede Formel ϕ mit $[\![\phi]\!] = \emptyset$ heißt unerfüllbar.

Äquivalenz: zwei Formeln ϕ_1 und ϕ_2 heißen äquivalent,

gdw. $[\![\phi_1]\!] = [\![\phi_2]\!]$

wir schreiben dann auch $\phi_1 \equiv \phi_2$

Einige Äquivalenzen: Beispiele

$$\mathbf{X}(\phi_1 \vee \phi_2) \equiv \mathbf{X}\phi_1 \vee \mathbf{X}\phi_2$$

$$\mathbf{X}(\phi_1 \wedge \phi_2) \equiv \mathbf{X}\phi_1 \wedge \mathbf{X}\phi_2$$

$$\mathbf{X}\neg\phi \equiv \neg\mathbf{X}\phi$$

$$\mathbf{F}(\phi_1 \vee \phi_2) \equiv \mathbf{F}\phi_1 \vee \mathbf{F}\phi_2$$

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

$$\mathbf{G}(\phi_1 \wedge \phi_2) \equiv \mathbf{G}\phi_1 \wedge \mathbf{G}\phi_2$$

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

$$(\phi_1 \wedge \phi_2)\mathbf{U}\psi \equiv \phi_1\mathbf{U}\psi \wedge \phi_2\ \mathbf{U}\psi$$

$$\phi\mathbf{U}(\psi_1 \vee \psi_2) \equiv \phi\mathbf{U}\psi_1 \vee \phi\mathbf{U}\psi_2$$

Idempotenz und Rekursionsgesetze

$$\mathbf{F}\phi \equiv \mathbf{F}\mathbf{F}\phi$$

$$\mathbf{G}\phi \equiv \mathbf{G}\mathbf{G}\phi$$

$$\phi\mathbf{U}\psi \equiv \phi\mathbf{U}(\phi\mathbf{U}\psi)$$

Rekursionsgesetze:

$$\mathrm{F}\phi \equiv \phi \vee \mathbf{X}\mathrm{F}\phi$$

$$\mathbf{G}\phi \equiv \phi \wedge \mathbf{X}\mathbf{G}\phi$$

$$\phi\mathbf{U}\psi \equiv \psi \vee (\phi \wedge \mathbf{X}(\phi\mathbf{U}\psi))$$

$$\phi\mathbf{W}\psi \equiv \psi \vee (\phi \wedge \mathbf{X}(\phi\mathbf{W}\psi))$$

Sicherheitseigenschaften

Eigenschaften sind eigentlich stets Mengen (Sprachen) *L* unendlicher Worte, die Worte die diese Eigenschaften erfüllen. D.h. eine Eigenschaft ist eine Sprache

$$\mathrm{L} \subseteq (2^{\mathrm{AP}})^{\omega}$$

Gibt es für alle $\sigma \in (2^{AP})^{\omega}/L$ einen endlichen Präfix *w*, so dass für alle $\alpha \in (2^{AP})^{\omega}$ schon gilt $w\alpha \notin L$ d.h es gibt keine zulässige Erweiterung mehr, mit der man den Präfix *w* wieder nach *L* bringen kann, dann nennt man *w* einen **schlechten Präfix** und *L* eine **Sicherheitseigenschaft**.

Beispiel: $\mathbf{G}\neg(\mathrm{HighwayGreen} \wedge \mathrm{FarmroadGreen})$

Lebendigkeitseigenschaften

Betrachte wieder eine Eigenschaft

$$\mathrm{L} \subseteq (2^{\mathrm{AP}})^{\omega}$$

Gibt es für jedes $w \in (2^{AP})^*$, ein $\alpha \in (2^{AP})^{\omega}$ so dass wieder

$$w\alpha \in L$$

d.h es gibt stets zulässige Erweiterung, mit der man einen Präfix *w* nach *L* bringen kann, dann nennt man *L* eine **Lebendigkeitseigenschaft**.

Beispiel: $\mathbf{F}\,\mathrm{HighwayGreen}$

Bemerkung:

Jede LTL Eigenschaft lässt sich als Schnitt einer Sicherheitseigenschaft mir einer Lebendigkeitseigenschaft darstellen.

Interpretation von LTL über Kripke Strukturen

Kripke Strukturen sind ja die formalen Modelle, die wir für Implementierungen von Systemen betrachten. Wenn wir nun wissen wollen, ob ein System in LTL formulierte Eigenschaften erfüllt, müssen wir nachprüfen, ob die Eigenschaft über den Menge aller (unendlichen) Ausführungsfolgen der Kripke Struktur gilt.

Sei *K= (S,→,St, AP,L)* eine Kripke Struktur.

Eine Folge $\rho \in S^{\omega}$ mit $\rho(1) \in St$ und $\forall i\colon \rho(i) \to \rho(i+1)$ heißt Ausführung von *K*.

Für eine Ausführung ρ betrachten wir nun $\mathrm{L}(\rho) \in (2^{AP})^{\omega}$ mit

$\forall i\colon L(\rho)(i) := L(\rho(i))$ d.h.die Labellingfolge zur Ausführung ρ.

Dann bezeichnen wir mit $[\![K]\!]$ die Menge aller möglichen Labelingsequenzen zu Ausführungen von *K*.

$$[\![K]\!] = \{\mathrm{L}(\rho) \mid \rho \text{ ist Ausführung von K}\}$$

Das LTL Model Checking Problem

Das Problem ist es, zu entscheiden, ob eine Kripke Struktur, die wir aus der Implementierung abgeleitet haben, ein Modell für die Spezifikation ist. Genauer

Problem: -- LTL Model Checking

Gegeben sei eine Kripke Struktur *K= (S,→,St, AP,L)* und eine LTL Formel ϕ über *AP* . Entscheide, ob $[\![K]\!] \subseteq [\![\phi]\!]$?

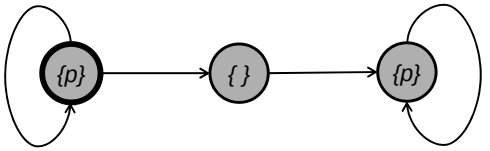
Es muss also für jede Ausführung ρ das Labeling $\mathrm{L}(\rho)$ die Formel ϕ erfüllen. Wir schreiben dann auch

$$K \vDash \phi$$

Vorsicht: Es kann sowohl $K \nvDash \phi$ als auch $K \nvDash \neg\phi$ gelten!

Beispiel:

Wir betrachten folgende einfach Kripke Struktur mit 3 Zuständen:



Dann gibt es zwei Klassen von Ausführungen:

- Das System bleibt ewig im Startzustand d.h. $\mathrm{L}(\rho) = \{p\}^{\omega}$
- Das System läuft nach rechts und bleibt im rechten Zustand, d.h. es gibt ein n so dass $\mathrm{L}(\rho) = \{p\}^{n} \cdot \{\} \cdot \{p\}^{\omega}$

Es gilt:
$\mathrm{K} \vDash \mathbf{FG}\mathrm{p}$ denn alle Abläufe verbleiben irgendwann einmal für immer in einem Zustand, in dem p gilt.
$\mathrm{K} \nvDash \mathbf{G}\mathrm{p}$ denn alle Abläufe der Form $\{p\}^{n} \cdot \{\} \cdot \{p\}^{\omega}$laufen irgendwann mal über den Zustand in dem p nicht gilt. (Es gilt aber auch nicht $\neg\mathbf{G}\mathrm{p} = \mathbf{F}\neg\mathrm{p}$)

Fallstrick endliche Ausführungen

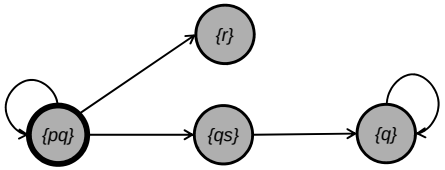
Vorsicht: LTL Formeln werden nur über den unendlichen Ausführungen interpretiert.

Wenn die Kripke Struktur zum Beispiel Deadlocks enthält (Zustände ohne Nachfolger in der Transitionsrelation) , dann werden alle Ausführungen, die auf einem solchen Zustand enden ignoriert, weil sie endliche Länge haben. Auf solchen Ausführungen wird die Formel nicht interpretiert.

Das kann im Extremfall unerwartete Konsequenzen haben:

Angenommen in K führt jeder Weg in einen Deadlock.

Dann ist $[\![K]\!] = \emptyset$. Damit erfüllt K aber jede Formel!



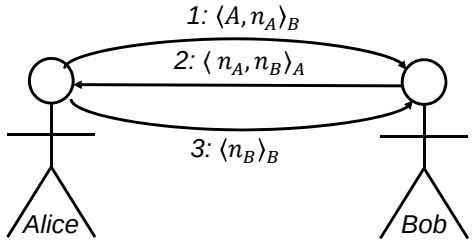
Erfüllt $\mathbf{G}$q

Beispiel: das Needham-Schröder-Protokoll

Dass man mit Modelchecking durchaus Überraschendes entdecken kann, zeigt wohl folgendes Beispiel:

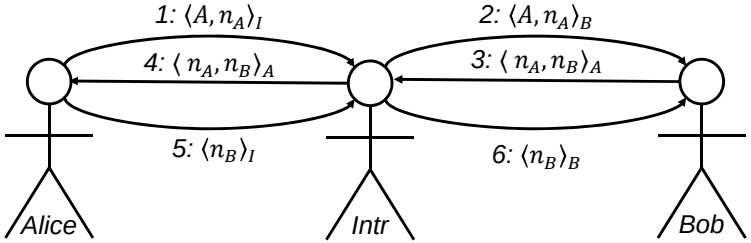
Alice und Bob wollen über ein unsicheres Netzwerk sicher ein gemeinsames Geheimnis erzeugen, das nur sie beide kennen. Dieses Geheimnis sei durch ein Paar von Zahlen $\langle n_A, n_B \rangle$ repräsentiert (sog. Nonces). Zur Kommunikation stehe ein Public Key Verfahren zur Verfügung. $\langle M \rangle_X$ bedeutet, dass Nachricht *M* mit dem public key von *X* verschlüsselt ist. Nur *X* kann sie entschlüsseln.

Folgendes Protokoll galt 18 Jahre lang als sicher:



Beispiel: das Needham-Schröder-Protokoll

Dieses Protokoll wurde implementiert und mit LTL überprüft. Dazu müssen die Akteure (Alice und Bob) als Prozesse implementiert werden, die genau dieses Protokoll nachbilden. Zusätzlich ein Intruder, der Nachrichten entweder unverändert weiterleitet oder eigene Nachrichten erfindet und verschickt. Folgendes Gegenbeispiel wurde bei einem Verifikationsversuch erzeugt:



Bob ist nun fest davon überzeugt, mit Alice ein Geheimnis zu teilen!

LTL Model Checking

Wir wollen nun anschauen, wie man (im Prinzip, schnelle Heuristiken sind nach wie vor Gegenstand der aktuellen Forschung) Model Checking für LTL Formeln algorithmisch entscheidet:

Problem: -- LTL Model Checking

Gegeben sei eine Kripke Struktur *K= (S,→,St, AP,L)* und eine LTL Formel ϕ über *AP* . Entscheide, ob $[\![K]\!] \subseteq [\![\phi]\!]$?

Im Prinzip gibt es zwei Möglichkeiten:

1. Konstruiere eine LTL Formel Ψ mit $[\![K]\!] = [\![\Psi]\!]$ und zeige $\Psi \rightarrow \phi$ Dies ist sehr aufwändig
2. Die Menge der Ausführungen von *K* lässt sich leicht als Sprache eines Büchi Automaten *BA(K)* d.h. $[\![K]\!]$ = *L(BA(K))* auffassen. Konstruiere einen weiteren Büchi Automaten *BA(¬ϕ)* der die Sprache $[\![\neg\phi]\!]$ akzeptiert und entscheide

$$L(BA(K)) \cap L(BA(\neg\phi)) = \varnothing$$

Büchi Automaten

Büchi Automaten sind im Prinzip endliche Automaten mit einem anderen Akzeptanzkriterium:

Ein Tupel $B = (S, S_0, A, \Delta, F)$ heißt **Büchi Automat**, wobei

- S eine endliche Menge von Zuständen
- $S_0 \subseteq S$ eine Menge von Startzuständen
- $\Delta \subseteq S \times A \times S$ die Übergangssrelation
- A ein endliches Alphabet
- $F \subseteq S$ eine Menge von Endzuständen (Akzeptanzzuständen) ist.

Man kann Büchi Automaten ebenso durch Diagramme darstellen, wie man das von endlichen Automaten kennt lediglich die Sprache ist anders definiert:

Sprache eines Büchi Automaten

Sei $B = (S, S_0, S, A, \Delta, F)$ ein Büchi Automat.

Für ein unendliches Wort $\sigma \in A^\omega$ ist $s(1)\sigma(1)s(2)\ \sigma(2)\ldots\ s(i)\sigma(i)\ldots$ ein **Lauf** in B mit Beschriftung σ genau dann wenn

(1) $s(1) \in S_0$ und

(2) für alle i gilt $s(i)\sigma(i)s(i+1) \in \Delta$

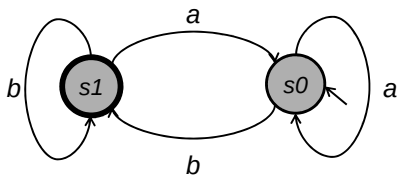
Ein Lauf heißt **akzeptierend**, wenn für unendlich viele i gilt: $s(i) \in F$.

Ein Büchi Automat $B = (S, S_0, S, A, \Delta, F)$ akzeptiert die Sprache

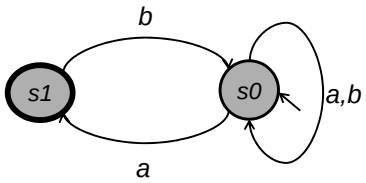
$$L(B) \subseteq A^\omega$$

genau dann wenn es für jedes $\sigma \in L(B)$ einen akzeptierenden Lauf mit Beschriftung σ gibt.

Beispiel zu Büchi Automaten



„Unendlich oft b“



„Unendlich oft ab“

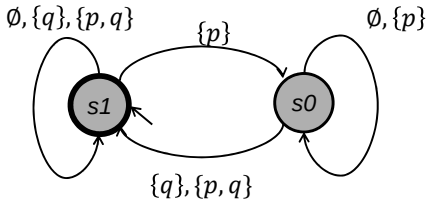
Büchi Automaten und LTL

Sei *AP* wieder eine Menge atomarer Eigenschaften.

Setzt man $A = 2^{AP}$, dann sind Worte aus A^ω wieder unendliche Verhaltensmuster über *AP*.

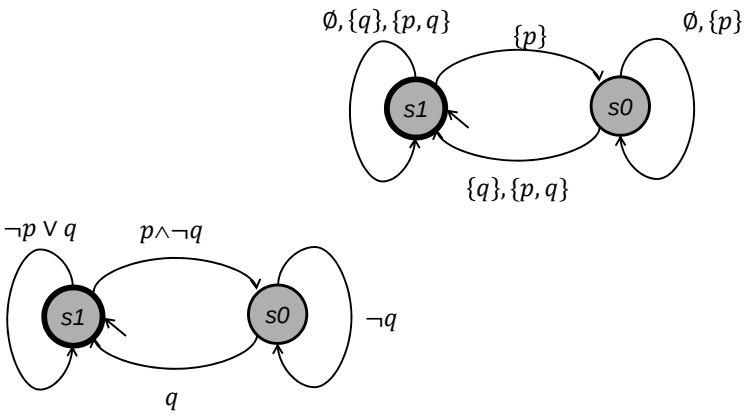
Wir werden sehen, dass man zu jeder LTL Formel ϕ über *AP* einen Büchi Automaten $BA(\phi)$ angeben kann, mit $L(BA(\phi)) = [\![\phi]\!]$ d.h. der nur Läufe akzeptiert auf denen die Formel gilt. Zeigen tun wir das zunächst an einem Beispiel:

Für die Formel $G(p \rightarrow Fq)$ über $AP=\{p,q\}$ akzeptiert folgender Automat die Sprache $[\![G(p \rightarrow Fq)]\!]$



Büchi Automaten und LTL

Wir beschriften die Übergänge, um mehr Übersicht zu erhalten, statt mit Teilmengen von AP mit einer aussagenlogischen Formel, die genau auf diesen Teilmengen gelten, für die der Übergang definiert ist.



Abgeschlossenheit ω regulärer Sprachen

Die von Büchi Automaten akzeptierten Sprachen nennt man auch
ω reguläre Sprachen

Satz:

Sind L_1 und L_2 ω reguläre Sprachen dann sind auch

(1) $L_1 \cup L_2$,

(2) $L_1 \cap L_2$ sowie

(3) $\overline{L_1}$

ω regulär.

Wir zeigen hier nicht alles!

Die Vereinigung ist ziemlich simpel, wie im Falle regulärer Mengen (endliche Automaten) auch (Übung?). Komplement ist ziemlich kompliziert und den Durchschnitt zeigen wir:

Der Produktautomat

Gegeben seien zwei Büchi Automaten

$B_1 = (S_1, S_{1,0}, A, \Delta_1, F_1)$ *und* $B_2 = (S_2, S_{2,0}, A, \Delta_2, F_2)$

als Akzeptoren zweier ω regulärer Sprachen $L(B_1), L(B_2)$:

Wir definieren den **Produktautomaten** $B_{1\times2} = (S_{1\times2}, S_{1\times2,0}, A, \Delta_{1\times2}, F_{1\times2})$ durch

- $S_{1\times2} \coloneqq S_1 \times S_2$
- $S_{1\times2,0} \coloneqq S_{1,0} \times S_{2,0}$
- $\Delta_{1\times2} \coloneqq \{((s_1, s_2), a, (t_1, t_2)) \mid (s_1, a, t_1) \in \Delta_1 \wedge (s_2, a, t_2) \in \Delta_2\}$
- $F_{1\times2} \coloneqq F_1 \times F_2$

Produktautomat und Durchschnitt

Es gilt nun folgende

Beobachtung:

Wenn $F_2 = S_2$, dann ist $L(B_{1\times2}) = L(B_1) \cap L(B_2)$

Beweis:

Nach Konstruktion für jeden Lauf in $B_{1\times2}$

$$\ldots(s_i, t_i), a_i, (s_{i+1}, t_{(i+1)}), a_{i+1}, (s_{i+2}, t_{(i+2)}) \ldots$$

die Folge $\ldots s_i, a_i, s_{i+1} \ldots$ ein Lauf in B_1

und die Folge $\ldots t_i, a_i, t_{i+1} \ldots$ ein Lauf in B_2.

Da $F_{1\times2} \coloneqq F_1 \times S_2$ kommen Elemente aus F_1 unendlich oft in $(s_i, a_i)_{i>0}$ vor, also ist $(s_i, a_i)_{i>0}$ ein akzeptierender Lauf und damit $(a_i)_{i>0} \in L(B_1)$.

Ferner ist mit $F_2 = S_2$ jeder Lauf in B_2 akzeptierend, also $(a_i)_{i>0} \in L(B_2)$.

Demnach ist schon mal $L(B_{1\times2}) \subseteq L(B_1) \cap L(B_2)$ gezeigt.

Produktautomat und Durchschnitt -- ff

Sei nun umgekehrt $(a_i)_{i>0} \in L(B_1) \cap L(B_2)$

Dann gibt es einen akzeptierenden Lauf $\ldots s_i, a_i, s_{i+1} \ldots$ in B_1 der unendlich oft $s_i \in F_1$ erfüllt

und zugleich einen akzeptierenden Lauf $\ldots t_i, a_i, t_{i+1} \ldots$ in B_2

Damit ist aber schon $\ldots(s_i, t_i), a_i, (s_{i+1}, t_{(i+1)}) \ldots$ ein akzeptierender Lauf in $B_{1\times2}$

Also ist auch $L(B_{1\times2}) \supseteq L(B_1) \cap L(B_2)$ und damit gilt

$$L(B_{1\times2}) = L(B_1) \cap L(B_2)$$

Produktautomat und Durchschnitt -- ff

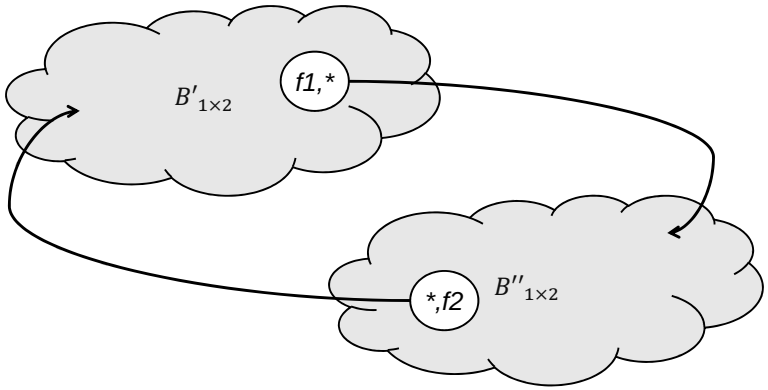
Für den allgemeinen Fall $F_2 \neq S_2$, braucht man noch einen kleinen Trick:

Wir betrachten einen Automaten den wir durch Vereinigung zweier Kopien des Produktautomaten $B'_{1\times2}$ und $B''_{1\times2}$ erzeugen, d.h.wir betrachten $B_{1\cap2} = (S_{1\cap2}, S_{1\cap2,0}, A, \Delta_{1\cap2}, F_{1\cap2})$ durch

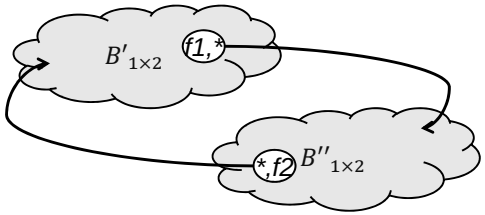
- $S_{1\cap2} := S'_{1\times2} \cup S''_{1\times2}$
- $S_{1\cap2,0} := S'_{1\times2,0}$
- $\Delta_{1\cap2} := \{((s'_1, s'_2), a, (t'_1, t'_2)) | ((s'_1, s'_2), a, (t'_1, t'_2)) \in \Delta'_{1\times2} \wedge s'_1 \notin F'_1\}$
  $\cup \{((s''_1, s''_2), a, (t''_1, t''_2)) | ((s''_1, s''_2), a, (t''_1, t''_2)) \in \Delta''_{1\times2} \wedge s''_2 \notin F''_2\}$
  $\cup \{((s'_1, s'_2), a, (t''_1, t''_2)) | ((s'_1, s'_2), a, (t'_1, t'_2)) \in \Delta'_{1\times2} \wedge s'_1 \in F'_1 \wedge (t''_1, t''_2) \text{ Kopie von } (t'_1, t'_2)\}$
  $\cup \{((s''_1, s''_2), a, (t'_1, t'_2)) | ((s''_1, s''_2), a, (t''_1, t''_2)) \in \Delta'_{1\times2} \wedge s''_2 \in F''_2 \wedge (t''_1, t''_2) \text{ Kopie von } (t'_1, t'_2)\}$
- $F_{1\times2} := F'_1 \times S'_2$

Produktautomat und Durchschnitt -- ff

Anschaulich gesehen läuft man im Produktautomaten $B'_{1\times2}$ bis man einen Endzustand auf der ersten Komponente findet. Von diesem aus muss man dann in die Kopie $B''_{1\times2}$ aus der man wiederum nur über einen Endzustand auf der zweiten Komponente zurückkommt:



Produktautomat und Durchschnitt -- ff



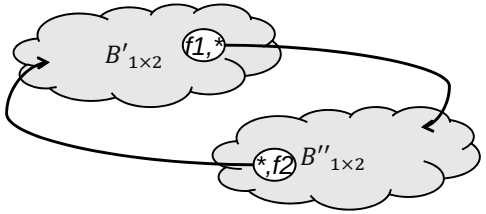
Für jeden Lauf in $B_{1\cap2}$ $\ldots(s_i, t_i), a_i \ldots$ mit unendlich vielen $(s_i, t_i) \in F'_1 \times S'_2$ ist natürlich die Folge $\ldots s_i, a_i \ldots$ ein akzeptierender Lauf in B_1.
Da aber für $(s_i, t_i) \in F'_1 \times S'_2$ der Lauf jedesmal in der Kopie $B''_{1\times2}$ fortgesetzt wird, bis ein Paar $(s_j, t_j) \in S''_1 \times F''_2$ folgt, enthält der Lauf auch unendlich viele $t_j \in F''_2$ auf der zweiten Komponenten also ist auch $\ldots t_i, a_i$ ein akzeptierender Lauf in B_2. Damit ist aber

$$(a_i)_{i>0} \in L(B_1) \cap L(B_2)$$

also

$$L(B_{1\times2}) \subseteq L(B_1) \cap L(B_2)$$

Produktautomat und Durchschnitt -- ff



Sei nun umgekehrt $(a_i)_{i>0} \in L(B_1) \cap L(B_2)$
Dann gibt es einen akzeptierenden Lauf $\ldots s_i, a_i, s_{i+1} \ldots$ in B_1, der unendlich oft $s_i \in F_1$ erfüllt, und zugleich einen akzeptierenden Lauf $\ldots t_i, a_i, t_{i+1} \ldots$ in B_2, der unendlich oft $t_i \in F_2$ erfüllt.
Damit ist aber schon (s'_i, s''_i sind die Kopien von s_i analog t_i)

$(s'_1, t'_1), a_1 \ldots, (s'_{i1}, t'_{i1}), a_{i1} \ldots, (s''_{i2}, t''_{i2}), a_{i2} \ldots, (s'_{i3}, t'_{i3}), a_{i3} \ldots$

ein akzeptierender Lauf in $B_{1\cap2}$ wobei bei den a_{ij} immer ein Umschalten zwischen $B'_{1\times2}$ und $B''_{1\times2}$ erfolgt. Da die „Umschaltzustände“ in beiden Läufen unendlich oft vorkommen, kommen nach wie vor $(s'_i, t'_i) \in F'_1 \times S'_2$ unendlich oft vor.
Also ist auch $L(B_{1\cap2}) \supseteq L(B_1) \cap L(B_2)$ und damit gilt $=$

Der Büchi Automat zur LTL Formel

Wir konstruieren nun einen Büchi Automaten *B(ϕ)* mit *L(B(ϕ))* = $[\![\phi]\!]$

Dazu betrachten wir zunächst einen **erweiterten Büchi Automaten**

BE = (S,S₀,A,Δ,ℱ)

worin $\mathcal{F} = \{F_0, \ldots, F_{k-1}\}$ mit $\forall 0 \leq i < k: F_i \subseteq S$ eine allgemeinere Akzeptanzbedingung angibt:

Ein Lauf … *s(i)*σ_i… heißt **akzeptierend**, wenn es für jedes *j* unendlich viele *i* gibt mit *s(i)*$\in F_j$.

Ein erweiterter Büchi Automat ist grundsätzlich nicht mächtiger als ein Büchi Automat, aber bequemer. Wir überzeugen uns zunächst von

Satz:

Zu jedem erweiterten Büchi Automaten *BE* kann man einen Büchi Automaten *B(BE)* konstruieren mit *L(BE) = L(B(BE))*

VBA ≡ BA

Wir skizzieren hier den Beweis der über eine Konstruktion eines Büchi Automaten *B(BE)* zum erweiterten Büchi Automat läuft. Ähnlich wie schon beim Schnitt arbeitet man die allgemeinere Akzeptanzbedingung über Kopien ab und führt sie so auf die einfache Akzeptanzbedingung zurück: Sei

BE = (S,S₀,A, Δ₁,ℱ) mit $\mathcal{F} = \{F_0, \ldots, F_{k-1}\}$ ein erweiterter Büchi Automat.

Dann betrachte $B(BE)$ = *(S′,S′₀,A, Δ′, F′)* wie folgt:

- $S' := S \times \{0, \ldots, k-1\}$
- $S'_0 := S_0 \times \{0\}$
- $\Delta' := \{((s,i), a, (t,i)) | (s,a,t) \in \Delta \wedge s \notin F_i\} \cup$

  $\{((s,i), a, (t,(i+1) mod\ k)) | (s,a,t) \in \Delta \wedge s \in F_i\}$
- $F' := F_1 \times \{0\}$

Man überzeuge sich, dass jeder akzeptierende Lauf in *BE* auch als akzeptierender Lauf in *B(BE)* gefunden werden kann und umgekehrt!

Der Büchi Automat zur LTL Formel -- ff

Wir konstruieren nun einen Büchi Automaten *B(ϕ)*

Dazu beschränken wir uns auf LTL Formel in **Negationsnormalform (NNF)** d.h. Negationen kommen nur direkt bei den Literalen für die atomaren Eigenschaften vor.:

1. Für jedes p$\in$AP ist *p* und ¬*p* eine LTL Formel in NNF.
2. Sind ϕ_1 und ϕ_2 Formeln in NNF, dann auch

   $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, $\boldsymbol{X}\phi_1$, $\phi_1 \boldsymbol{U} \phi_2$, $\phi_1 \mathbf{R} \phi_2$

Es gilt

Satz:

Zu jeder LTL Formel gibt es ein Äquivalent in NNF.

Der Büchi Automat zur LTL Formel -- ff

Man kann Negationen bei binären Operatoren immer auf die Operanden ziehen, d.h es ist:

- $\neg(\phi_1 \vee \phi_2) = \neg\phi_1 \wedge \neg\phi_2$ und ebenso $\neg(\phi_1 \wedge \phi_2) = \neg\phi_1 \vee \neg\phi_2$
- $\neg \boldsymbol{X}\phi_1 = \boldsymbol{X}\neg\phi_1$
- $\neg(\phi_1 \boldsymbol{U} \phi_2) = \neg\phi_1 \mathbf{R} \neg\phi_2$ und ebenso $\neg(\phi_1 \boldsymbol{R} \phi_2) = \neg\phi_1 \mathbf{U} \neg\phi_2$

Wir können also Negationen über die Operatoren bis zu den Literalen propagieren und dort, wenn nötig, $\neg\neg p = p$ anwenden.

Beispiel: $\boldsymbol{G}(p \rightarrow \boldsymbol{F}q) = \neg\boldsymbol{F}\neg(p \rightarrow \boldsymbol{F}q)$

$= \neg\big(true\ \boldsymbol{U}\ \neg(p\ \rightarrow \boldsymbol{F}q)\big)$

$= \neg true\ \boldsymbol{R}\ (p\ \rightarrow \boldsymbol{F}q)$

$= false\ \boldsymbol{R}\ (\neg p\ \vee \boldsymbol{F}q\ )$

$= false\ \boldsymbol{R}\ (\neg p\ \vee (true\ \boldsymbol{U}\ q)\ )$

Der Büchi Automat zur LTL Formel -- ff

Wir betrachten nun zu einer Formel ϕ in NNF die Menge $Sub(\Phi)$ wie folgt induktiv definiert:

1. $\phi \in Sub(\Phi)$
2. $true \in Sub(\Phi)$
3. falls $\phi_1 \in Sub(\Phi)$ dann auch $\neg\phi_1 \in Sub(\Phi)$ und umgekehrt.
4. falls $\boldsymbol{X}\phi_1 \in Sub(\Phi)$ dann auch $\phi_1 \in Sub(\Phi)$
5. falls $\phi_1 \vee \phi_2 \in Sub(\Phi)$ dann auch $\phi_1, \phi_2 \in Sub(\Phi)$
6. falls $\phi_1 \wedge \phi_2 \in Sub(\Phi)$ dann auch $\phi_1, \phi_2 \in Sub(\Phi)$
7. falls $\phi_1 \boldsymbol{U} \phi_2 \in Sub(\Phi)$ dann auch $\phi_1, \phi_2 \in Sub(\Phi)$
8. falls $\phi_1 \boldsymbol{R} \phi_2 \in Sub(\Phi)$ dann auch $\phi_1, \phi_2 \in Sub(\Phi)$

Es gilt demnach $|Sub(\Phi)\ | = O(|\phi|)$

Der Büchi Automat zur LTL Formel -- ff

Wir betrachten nun zu einer Formel ϕ einen Büchi Automaten, dessen Zustände Teilmengen von $Sub(\Phi)$ sind, d.h. er hat Größe $O(2^{|Sub(\Phi)|})$. Ein Zustand $M \subseteq Sub(\Phi)$ soll alle die Sequenzen akzeptieren, die alle Formeln aus *M* erfüllen und alle die Sequenzen, die alle Formeln aus $Sub(\Phi) \setminus M$ nicht erfüllen. Wir schließen daher Formelmengen aus, die für die leere Menge stehen:

Ein Zustand $M \subseteq Sub(\Phi)$ heißt **konsistent**, wenn er folgende Bedingungen erfüllt:

$true \in M$

falls $\phi_1 \in Sub(\Phi)$ dann ist $\neg\phi_1 \in M \Leftrightarrow \phi_1 \notin M$

falls $\phi_1 \vee \phi_2 \in Sub(\Phi)$ dann ist $\phi_1 \vee \phi_2 \in M \Leftrightarrow \phi_1 \in M\ oder\ \phi_2 \in M$

falls $\phi_1 \wedge \phi_2 \in Sub(\Phi)$ dann ist $\phi_1 \wedge \phi_2 \in M \Leftrightarrow \phi_1 \in M\ und\ \phi_2 \in M$

Wir nennen die Menge der konsistenten Zustände auch

$$CS(\Phi) \subseteq 2^{Sub(\Phi)}$$

Der Büchi Automat zur LTL Formel -- ff

Sei Φ eine LTL Formel. Dann ist *BA(*Φ *) = (*$S, S_0 S, A, \Delta, F$*)* folgender VBA:

$A = 2^{AP}$ (Alphabet ist die Menge aller Belegungen über den AP)

$S = CS(\phi)$ d.h. jeder Zustand ist eine konsistente Menge von Teilformeln

$S_0 = \{M \in S |\ \phi\ \in M\}$ Anfangszustände akzeptieren alles was ϕ erfüllt

Es ist $(M, \sigma, M') \in \Delta$ genau dann, wenn $\sigma = M \cap AP$ und

falls $\boldsymbol{X}\phi_1 \in Sub(\Phi)$ dann ist $\boldsymbol{X}\phi_1 \in M \Leftrightarrow \phi_1 \in M'$

falls $\phi_1 \boldsymbol{U} \phi_2 \in Sub(\Phi)$ dann ist

$\phi_1 \boldsymbol{U} \phi_2 \in M \Leftrightarrow (\phi_2 \in M$ oder $(\phi_1 \in M$ und $\phi_1 \boldsymbol{U} \phi_2 \in M')$

falls $\phi_1 \boldsymbol{R} \phi_2 \in Sub(\Phi)$ dann ist

$\phi_1 \boldsymbol{R} \phi_2 \in M \Leftrightarrow (\phi_1 \wedge \phi_2 \in M$ oder $(\phi_2 \in M$ und $\phi_1 \boldsymbol{R} \phi_2 \in M')$

Für eine Teilformel $\psi = (\phi_1 \boldsymbol{U} \phi_2) \in Sub(\Phi)$ sei

$$F_\psi = \{M \in CS(\Phi) |\ \phi_2 \in M \text{ oder } \neg(\phi_1 \boldsymbol{U} \phi_2) \in M\}$$

Dann setzen wir *F=* $\{F_\psi |\ \psi = (\phi_1 \boldsymbol{U} \phi_2) \in Sub(\Phi)\}$

Der Büchi Automat zur LTL Formel -- Beweis

Es gilt nun sogar folgende

Behauptung:

Sei …M_i, a_i, M_{i+1}… ein Lauf in *BA(Φ)*, dann ist…M_i, a_i, M_{i+1}… akzeptierend, gdw. für alle *i* gilt: a_i ... erfüllt ϕ für $\phi \in M_i$

Beweis:

Strukturelle Induktion nach ϕ

$\phi \epsilon AP$:

Bei eine Lauf …M_i, a_i, M_{i+1}… ist nach Konstruktion stets $a_i = M_i \cap AP$

Also $\phi \in M_i$.und damit erfüllt schon a_i… die Formel ϕ.

$\phi = \neg\theta \; und \; \theta \; \epsilon AP$:

Bei eine Lauf …M_i, a_i, M_{i+1}… ist nach Konstruktion stets $a_i = M_i \cap AP$

Also ist mit $\phi \in M_i, \theta \notin M_i$.und damit erfüllt a_i… die Formel $\phi = \neg\theta$.

Der Büchi Automat zur LTL Formel – Beweis ff

$\phi = \phi_1 \wedge \phi_2 \in M_i$:

Dann gilt wegen der Konsistenz von M_i ist nach Konstruktion stets $\phi_1 \in M_i$ und $\phi_2 \in M_i$.und damit erfüllt schon a_i… nach I.A. die Formeln $\phi_1$und ϕ_2 also auch $\phi_1 \wedge \phi_2$

$\phi = \phi_1 \vee \phi_2 \in M_i$:

Dann gilt wegen der Konsistenz von M_i ist nach Konstruktion stets $\phi_1 \in M_i$ oder $\phi_2 \in M_i$.und damit erfüllt schon a_i… nach I.A. die Formeln ϕ_1 oder ϕ_2 also auch $\phi_1 \vee \phi_2$

$\phi = \boldsymbol{X}\phi \in M_i$:

Dann gilt $\phi \in M_{i+1}$nach Konstruktion der Übergangsfunktion. Mit I.A. erfüllt nun a_{i+1}…die Formel ϕ und damit erfüllt schon a_i… die Formel $\boldsymbol{X}\phi$

Der Büchi Automat zur LTL Formel – Beweis ff

$\phi = \phi_1 \boldsymbol{R} \phi_2 \in M_i$:

Dann gilt gemäß der Definition der Übergangsfunktion

$\phi_1 \boldsymbol{R} \phi_2 \in M_i \Leftrightarrow (\phi_1 \wedge \phi_2 \in M_i$ oder $(\phi_2 \in M_i$ und $\phi_1 \boldsymbol{R} \phi_2 \in M_{i+1})$

Ist $\phi_1 \wedge \phi_2 \in M_i$ dann erfüllt mit I.A. a_i …die Formel $\phi_1 \wedge \phi_2$ und damit auch schon $\phi_1 \boldsymbol{R} \phi_2$

Ist $\phi_2 \in M_i$ und $\phi_1 \boldsymbol{R} \phi_2 \in M_{i+1}$denn erfüllt mit I.A. a_i … die Formel ϕ_2 und induktiv erfüllt dann a_{i+1} … die Formel $\phi_1 \boldsymbol{R} \phi_2$

Also erfüllt a_i … die Formel

$$(\phi_1 \wedge \phi_2) \vee (\phi_2 \wedge \boldsymbol{X}(\phi_1 \boldsymbol{R} \phi_2)) = \phi_1 \boldsymbol{R} \phi_2$$

Der Büchi Automat zur LTL Formel – Beweis ff

$\phi = \phi_1 \boldsymbol{U} \phi_2 \in M_i$:

Dann gilt gemäß der Definition der Übergangsfunktion

$\phi_1 \boldsymbol{U} \phi_2 \in M_i \Leftrightarrow (\phi_2 \in M_i$ oder $(\phi_1 \in M_i$ und $\phi_1 \boldsymbol{U} \phi_2 \in M_{i+1})$

Ist $\phi_2 \in M_i$ dann erfüllt mit I.A. a_i …die Formel ϕ_2 und damit auch schon $\phi_1 \boldsymbol{U} \phi_2$

Ist $\phi_1 \in M_i$ und $\phi_1 \boldsymbol{U} \phi_2 \in M_{i+1}$

Annahme: für alle $j \geq i+1$ wäre $\phi_1 \boldsymbol{U} \phi_2 \in M_j$

Dann ist …M_i, a_i, M_{i+1}… kein akzeptierender Lauf weil kein Zustand aus $F_{\phi_1 \boldsymbol{U} \phi_2}$mehr vorkommt. Also muss für ein $j \geq i+1$ $\phi_2 \in M_j$ sein.

Dann erfüllt mit I.A. a_i …a_{j-1} die Formel ϕ_1 und a_j … die Formel $\phi_1 \boldsymbol{U} \phi_2$

Also erfüllt a_i … die Formel $\phi_1 \boldsymbol{U} \phi_2$ ■

Bemerkung: Die Umkehrung gilt nicht, d.h. man kann nicht die Sprache jedes beliebigen Büchi-Automaten durch eine LTL Formel fassen.

LTL Model Checking

Wir haben nun eine Möglichkeit entwickelt, mit der man (im Prinzip, schnelle Heuristiken sind nach wie vor Gegenstand der aktuellen Forschung) Model Checking für LTL Formeln algorithmisch durchführen kann:

Problem: -- LTL Model Checking

Gegeben sei eine Kripke Struktur $K= (S,\rightarrow,St, AP,L)$ und eine LTL Formel ϕ über AP . Entscheide, ob $[\![K]\!] \subseteq [\![\phi]\!]$?

1. Die Kripke Struktur K stammt in der Regel von einer Spezifikation des Systems in Form eines Automaten. Bei unendlichem Betrieb lässt sich diese leicht durch einen Büchi- Automaten $BA(K)$ ausdrücken.
2. Übersetze die LTL Formel ϕ in einen Büchi Automaten $BA(\neg\phi)$ der die Sprache $[\![\neg\phi]\!]$ akzeptiert.
3. Bilde den Produktautomaten $BA(L(BA(K)) \cap L(BA(\neg\phi))$ und entscheide, ob die Menge der akzeptierenden Läufe leer ist.
4. Gibt es einen akzeptierenden Lauf, ist dieser ein Gegenbeispiel.