

Model Checking mit Hilfe regulärer Sprachen

Seminar Beyond First Order Logic bei Andreas Schäfer

Eike Möhlmann

15. September 2007

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer
- 5 Verifikation
- 6 Fazit

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer
- 5 Verifikation
- 6 Fazit

Einleitung

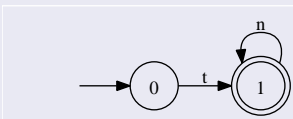
- Logik LTL(MSO)
 - Kombination von linear-time-logic (LTL) und monadische Logik zweiter Stufe (MSO)
 - formale Beschreibung von Sprachen
 - Sprachen als Beschreibung der Zustandsmengen komplexer Systemen (unendlichen Anzahl von Zuständen)
 - zu Formeln können Automaten konstruiert werden (hier Büchi-Automaten)
- Büchi-Automaten
 - Model-Checking möglich, daruch Verifikation möglich

Agenda

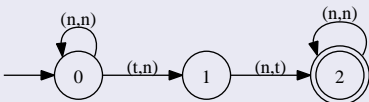
- 1 Einleitung
- 2 Beispiel**
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer
- 5 Verifikation
- 6 Fazit

Beispiel - Token-passing-protocol

Der Automat I



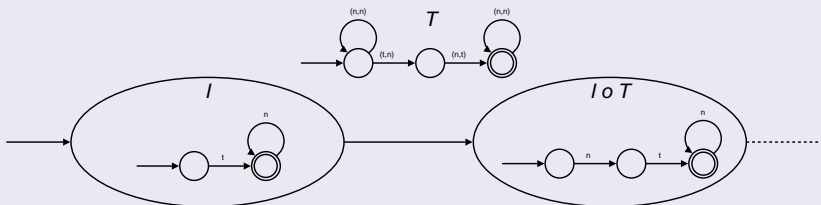
Der Transducer T



- endliche viele Prozesse
- Token t
- nicht Token n
- Automat I ist Menge der initialen Zustände
- Automat T stellt Übergänge (Weiterreichen) dar

Beispiel - Token-passing-protocol

Systemübersicht



- Stellt Schar von Systemen dar
- Automaten stellen die möglichen Zustände mit Sprachen dar
- Transducer stellen die Übergänge durch Transformation der Sprachen dar
- Möglichkeit parametrisierte Systeme darzustellen

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)**
- 4 Büchi-Transducer
- 5 Verifikation
- 6 Fazit

LTL - Linear-time Logic

Syntax

$$\begin{aligned} \varphi ::= & \text{true} \mid \text{false} \mid \\ & \Box\varphi \mid \Diamond\varphi \mid \bigcirc\varphi \mid \varphi_1 \cup \varphi_2 \mid \\ & \varphi_1 * \varphi_2, \text{ wobei } * \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \mid \neg\varphi \end{aligned}$$

Semantik

- $\Box\varphi$ bedeutet, dass φ immer (engl. always) gilt
- $\Diamond\varphi$ bedeutet, dass φ irgendwann (engl. eventually) gilt

MSO - Monadic Second Order Logic

Syntax

$$Form ::= \top \mid \perp \mid \neg F \mid (F) \mid F * G \mid p(t_1, \dots, t_n)$$
$$Term ::= v \mid V \mid f(t_1, \dots, t_n)$$
$$\varphi ::= \exists x \varphi \mid \forall x \varphi \mid F$$

- $F, G \in Form$,
- $t_1, \dots, t_n \in Term$,
- $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$,
- $v \in Var_{FO}, V \in Var_{SO}$,
- $x \in Var_{FO} \cup Var_{SO}$,
- $p \in Präd$ und $f \in Func$.

- Var_{FO} und Var_{SO} Variablen erster bzw. zweiter Ordnung
- $Präd$ und $Func$ Mengen von Prädikaten bzw. Funktionen
- $Präd, Func, Var_{FO}, Var_{SO}$ paarweise disjunkt

LTL(MSO)

Syntax

$$\varphi ::= i \in I \mid I \subseteq J \mid i = j + 1 \mid j = j \mid x[i] \mid x'[i] \mid$$
$$true \mid false \mid \Box \varphi \mid \Diamond \varphi \mid \varphi_1 * \varphi_2 \mid \neg \varphi$$

Interpretation von LTL(MSO)

Semantik

$$i \in I \quad \text{gdw.} \quad \mathfrak{I}(i) \in \mathfrak{I}(I)$$

$$I \subseteq J \quad \text{gdw.} \quad \mathfrak{I}(I) \subseteq \mathfrak{I}(J)$$

$$i = j + 1 \quad \text{gdw.} \quad \mathfrak{I}(i) = \mathfrak{I}(j) + 1$$

$$x[i] \quad \text{gdw.} \quad x \in M(t, \mathfrak{I}(i))$$

$$x'[i] \quad \text{gdw.} \quad x \in M(t + 1, \mathfrak{I}(i))$$

Interpretation von LTL(MSO)

Semantik

$$i \in I \quad \text{gdw.} \quad \mathfrak{I}(i) \in \mathfrak{I}(I)$$

$$I \subseteq J \quad \text{gdw.} \quad \mathfrak{I}(I) \subseteq \mathfrak{I}(J)$$

$$i = j + 1 \quad \text{gdw.} \quad \mathfrak{I}(i) = \mathfrak{I}(j) + 1$$

$$x[i] \quad \text{gdw.} \quad x \in M(t, \mathfrak{I}(i))$$

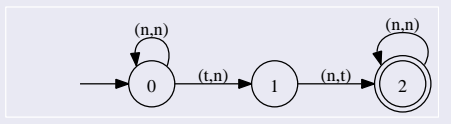
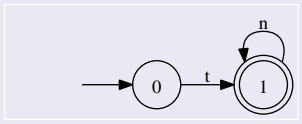
$$x'[i] \quad \text{gdw.} \quad x \in M(t + 1, \mathfrak{I}(i))$$

Matrizen

- Interpretation über Matrizen M
- Dimension $\infty \times n$ mit $n \in \mathbb{Z}_+$
 - die Zeit (engl. Time) und der Raum (engl. Space)

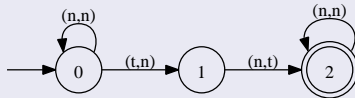
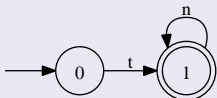
Beispiel - Formale Beschreibung von Systemen

Automaten



Beispiel - Formale Beschreibung von Systemen

Automaten



Beschreibung des Token-passing-protocols

$$\begin{aligned}
 \mathbf{initial} &= \forall i (t[i] \leftrightarrow i = 0) \\
 \mathbf{idle}(i) &= t[i] \leftrightarrow t'[i] \\
 \mathbf{pass}(i) &= (t[i] \wedge \neg t'[i]) \wedge (\neg t[i+1] \wedge t'[i+1]) \\
 &\quad \wedge \forall j ((j \neq i+1 \wedge j \neq i) \rightarrow \mathbf{idle}(j)) \\
 \mathbf{system} &= \mathbf{initial} \wedge \square (\exists i \mathbf{pass}(i) \vee \forall i \mathbf{idle}(i))
 \end{aligned}$$

Interpretation mit Matrizen

Beschreibung des Token-passing-protocols

$$\begin{aligned}
 \mathbf{initial} &= \forall i (t[i] \leftrightarrow i = 0) \\
 \mathbf{idle}(i) &= t[i] \leftrightarrow t'[i] \\
 \mathbf{pass}(i) &= (t[i] \wedge \neg t'[i]) \wedge (\neg t[i+1] \wedge t'[i+1]) \\
 &\quad \wedge \forall j ((j \neq i+1 \wedge j \neq i) \rightarrow \mathbf{idle}(j)) \\
 \mathbf{system} &= \mathbf{initial} \wedge \square (\exists i \mathbf{pass}(i) \vee \forall i \mathbf{idle}(i))
 \end{aligned}$$

Interpretation mit Matrizen

Beschreibung des Token-passing-protocols

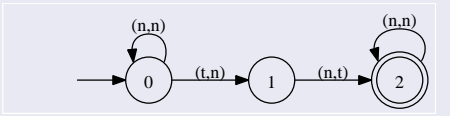
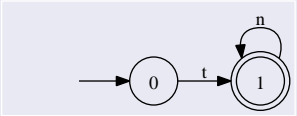
$$\begin{aligned}
 \mathbf{initial} &= \forall i (t[i] \leftrightarrow i = 0) \\
 \mathbf{idle}(i) &= t[i] \leftrightarrow t'[i] \\
 \mathbf{pass}(i) &= (t[i] \wedge \neg t'[i]) \wedge (\neg t[i+1] \wedge t'[i+1]) \\
 &\quad \wedge \forall j ((j \neq i+1 \wedge j \neq i) \rightarrow \mathbf{idle}(j)) \\
 \mathbf{system} &= \mathbf{initial} \wedge \square (\exists i \mathbf{pass}(i) \vee \forall i \mathbf{idle}(i))
 \end{aligned}$$

Matrix

0	t	n	n	n	...	n
1	n	t	n	n	...	n
2	n	n	t	n	...	n
...	n

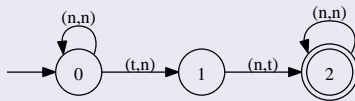
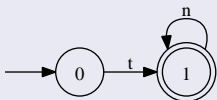
Beispiel - Formale Beschreibung von Anforderungen

Automaten



Beispiel - Formale Beschreibung von Anforderungen

Automaten



Anforderung

$$\text{**safety**} = \Box \neg \exists i, j (i \neq j \wedge t[i] \wedge t[j])$$

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer**
- 5 Verifikation
- 6 Fazit

Büchi-Automaten

Struktur

$$\mathcal{A} = (Q, \Sigma, q_0, \Delta, Acc)$$

- Q ist die Menge der Zustände
- Σ ist das Eingabealphabet
- q_0 der Initialzustand
- Δ die Menge der Transitionen der Form $Q \times \Sigma \times Q$
- Acc eine Akzeptanzbedingung

Büchi-Automaten

Struktur

$$\mathcal{A} = (Q, \Sigma, q_0, \Delta, Acc)$$

- Q ist die Menge der Zustände
- Σ ist das Eingabealphabet
- q_0 der Initialzustand
- Δ die Menge der Transitionen der Form $Q \times \Sigma \times Q$
- Acc eine Akzeptanzbedingung

Akzeptanzbedingung Acc

$$Inf(\rho) \cap F \neq \emptyset$$

$$Inf(\rho) = \{q \in Q \mid \exists_i^\omega : \rho(i) = q\}$$

Der Vorteil von Büchi-Automaten

Büchi Automaten sind abgeschlossen gegen:

- Vereinigung
- Schnitt
- Komplement

Der Vorteil von Büchi-Automaten

Büchi Automaten sind abgeschlossen gegen:

- Vereinigung
- Schnitt
- Komplement

Entscheidbarkeit:

- Leerheitsproblem entscheidbar

Büchi-Transducer

Büchi-Transducer sind

- spezielle Büchi-Automaten.
- nur mit Ausgabealphabet Γ

$$\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \Delta, Acc)$$

Anwendung

Da Zustände des Systems durch Sprachen beschrieben werden, können Transducer nun durch Transformation auf Sprachen den Übergang eines Systems beschreiben.

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer
- 5 Verifikation**
- 6 Fazit

Problemstellung

Die böse Menge

Stellen wir uns nun eine Menge B vor, die die Zustände beinhaltet, welche die Safty-Eigenschaft

$$\mathbf{safety} = \square \neg \exists i, j (i \neq j \wedge t[i] \wedge t[j])$$

nicht erfüllen. Also mindestens zwei Prozesse den Token haben.

Problemstellung

Die böse Menge

Stellen wir uns nun eine Menge B vor, die die Zustände beinhaltet, welche die Safty-Eigenschaft

$$\mathbf{safety} = \square \neg \exists i, j (i \neq j \wedge t[i] \wedge t[j])$$

nicht erfüllen. Also mindestens zwei Prozesse den Token haben.

$$n^* t (t^* n^*)^* t n^*$$

Weitere Benennungen

- Die Menge der initialen Zustände I
- Der Transducer T

Problemstellung

Es gilt heraus zu finden ob,

$$(I \circ T^*) \cap B = \emptyset$$

Problemstellung

Es gilt heraus zu finden ob,

$$(I \circ T^*) \cap B = \emptyset$$

Problem: Bestimmung von $Inv = I \circ T^*$

Problemstellung

Es gilt heraus zu finden ob,

$$(I \circ T^*) \cap B = \emptyset$$

Problem: Bestimmung von $Inv = I \circ T^*$

Zwei Verfahren zum bestimmen von T^*

- Quotientenbildung
- Abstraktion

Lösung durch Quotientenbildung - Idee

Die Idee

- 1 Produktkonstruktion von T^n für $n = 1, 2, 3, \dots$

Lösung durch Quotientenbildung - Idee

Die Idee

- 1 Produktkonstruktion von T^n für $n = 1, 2, 3, \dots$
 - Zustandsnamen für T^n sind die durchlaufenen Zustände des Transducers T in den verschiedenen Iterationen

Lösung durch Quotientenbildung - Idee

Die Idee

- 1 Produktkonstruktion von T^n für $n = 1, 2, 3, \dots$
 - Zustandsnamen für T^n sind die durchlaufenen Zustände des Transducers T in den verschiedenen Iterationen
- 2 Vereinigung der Produktautomaten zu einem History-Transducer
 - Problem: unendlich viele Zustände

Lösung durch Quotientenbildung - Idee

Die Idee

- 1 Produktkonstruktion von T^n für $n = 1, 2, 3, \dots$
 - Zustandsnamen für T^n sind die durchlaufenen Zustände des Transducers T in den verschiedenen Iterationen
- 2 Vereinigung der Produktautomaten zu einem History-Transducer
 - Problem: unendlich viele Zustände
 - Lösung: Quotientenbildung
- 3 Finden geeigneter Äquivalenzrelation

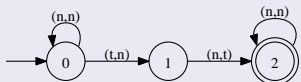
Lösung durch Quotientenbildung - Idee

Die Idee

- 1 Produktkonstruktion von T^n für $n = 1, 2, 3, \dots$
 - Zustandsnamen für T^n sind die durchlaufenen Zustände des Transducers T in den verschiedenen Iterationen
- 2 Vereinigung der Produktautomaten zu einem History-Transducer
 - Problem: unendlich viele Zustände
 - Lösung: Quotientenbildung
- 3 Finden geeigneter Äquivalenzrelation
- 4 Restklassenautomat T^+ Model-Checken

Lösung durch Quotientenbildung - Beispiel T^2

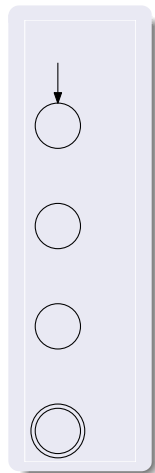
Eingabe:



Zwischenschritt:

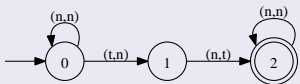


Ausgabe:

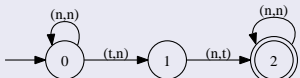


Lösung durch Quotientenbildung - Beispiel T^2

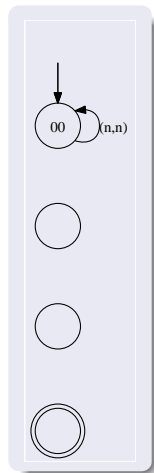
Eingabe: n



Zwischenschritt: n

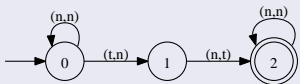


Ausgabe: n

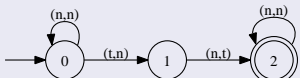


Lösung durch Quotientenbildung - Beispiel T^2

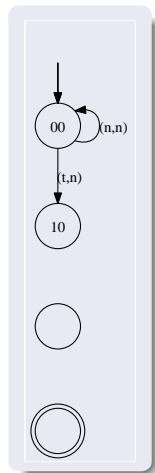
Eingabe: n t



Zwischenschritt: n n

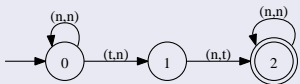


Ausgabe: n n

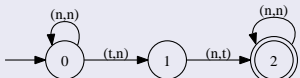


Lösung durch Quotientenbildung - Beispiel T^2

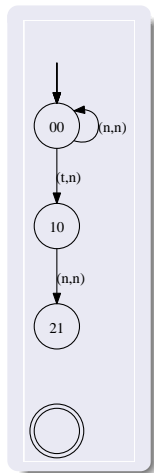
Eingabe: n t n



Zwischenschritt: n n t

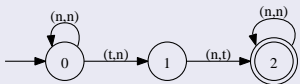


Ausgabe: n n n

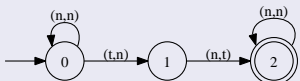


Lösung durch Quotientenbildung - Beispiel T^2

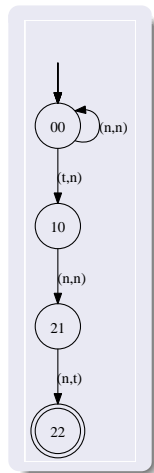
Eingabe: n t n n



Zwischenschritt: n n t n

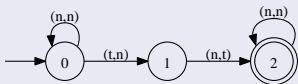


Ausgabe: n n n t

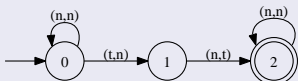


Lösung durch Quotientenbildung - Beispiel T^2

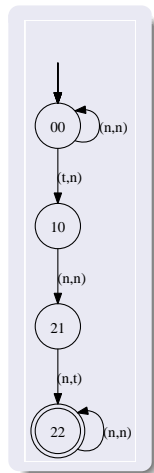
Eingabe: n t n n n



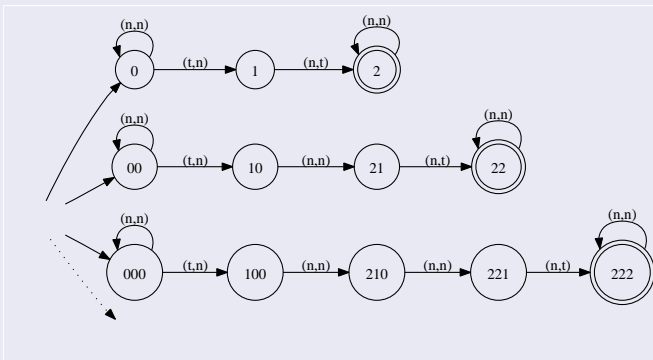
Zwischenschritt: n n t n n



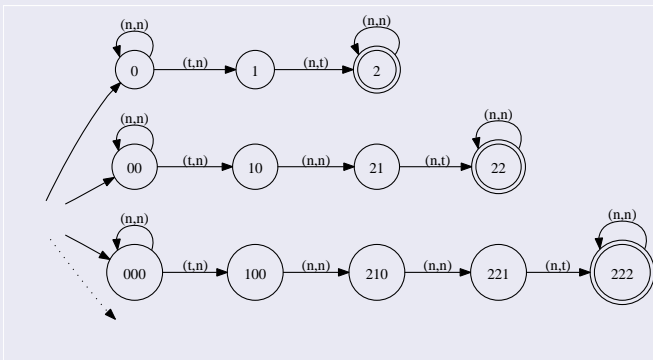
Ausgabe: n n n t n



Lösung durch Quotientenbildung - Produktkonstruktion

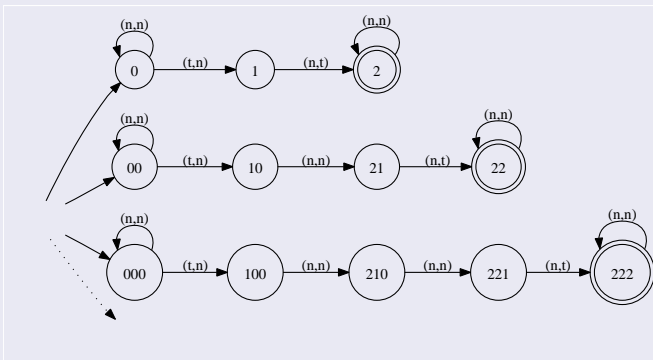


Lösung durch Quotientenbildung - Produktkonstruktion



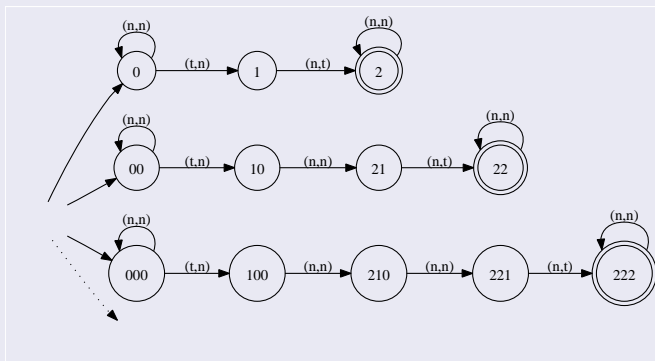
Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen

Lösung durch Quotientenbildung - Produktkonstruktion



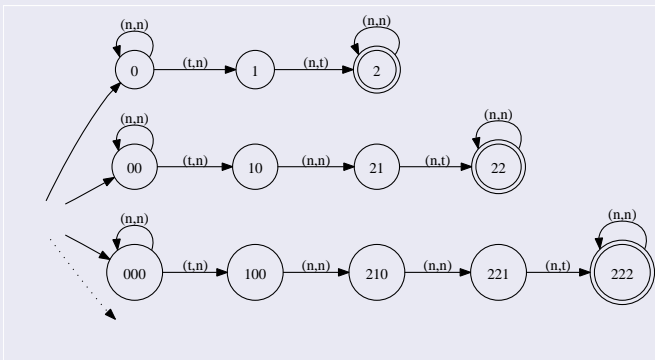
Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$

Lösung durch Quotientenbildung - Produktkonstruktion



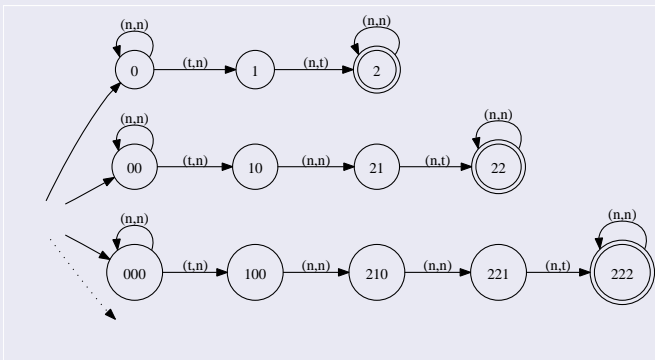
Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$, $[10^+]$

Lösung durch Quotientenbildung - Produktkonstruktion



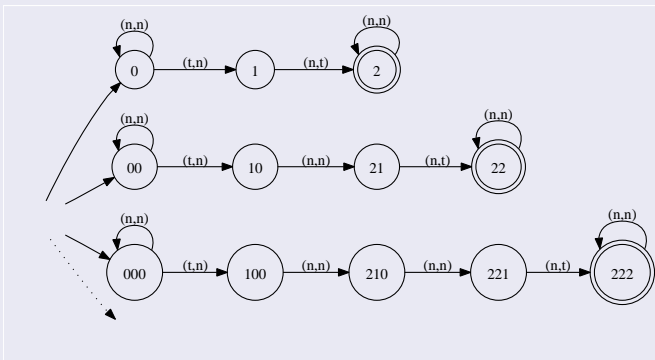
Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$, $[10^+]$, $[2^+10^+]$

Lösung durch Quotientenbildung - Produktkonstruktion



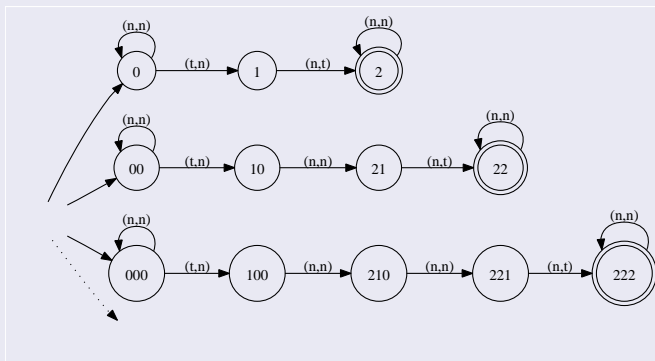
Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$, $[10^+]$, $[2^+10^+]$, $[1]$

Lösung durch Quotientenbildung - Produktkonstruktion



Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$, $[10^+]$, $[2^+10^+]$, $[1]$, $[2^+1]$

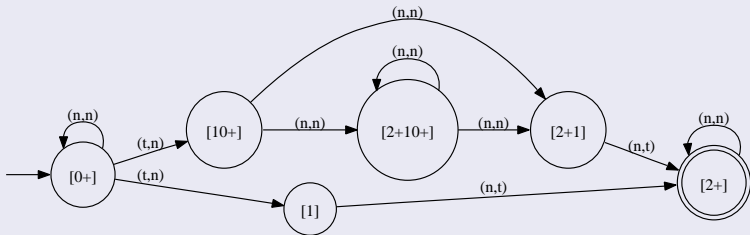
Lösung durch Quotientenbildung - Produktkonstruktion



Durch Vorwärts- und Rückwärts-Bisimulation Äquivalenzklassen bestimmen $[0^+]$, $[10^+]$, $[2^+10^+]$, $[1]$, $[2^+1]$, $[2^+]$

Lösung durch Quotientenbildung - Restklassenautomat

Ergebnis T^+ mit endlich vielen Zuständen



Lösung durch Abstraktion - Idee

Die Idee

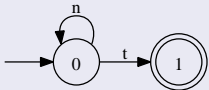
- 1 Finden der Äquivalenzrelation durch Post-Sprachen $\mathcal{L}(\mathcal{A}, q)$
- 2 Bestimmung einer abstrakten Sprache $\mathcal{L}(T^{lim})$

Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

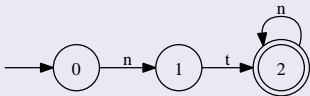
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) =$

Der Automat $\mathcal{A}: (I \circ T)$

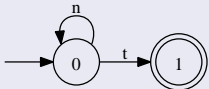


Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

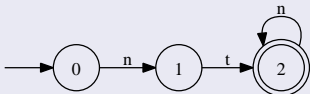
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) =$

Der Automat $\mathcal{A}: (I \circ T)$

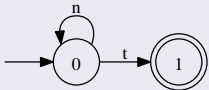


Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

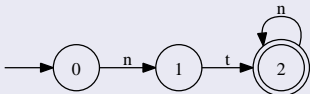
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) =$

Der Automat $\mathcal{A}: (I \circ T)$

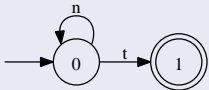


Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

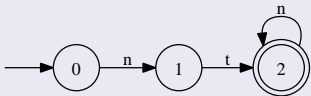
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) = ntn^*$
- $\mathcal{L}(A, 1) =$

Der Automat \mathcal{A} : ($I \circ T$)

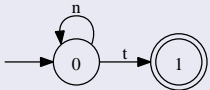


Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

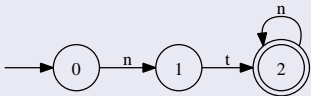
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) = ntn^*$
- $\mathcal{L}(A, 1) = tn^*$
- $\mathcal{L}(A, 2) =$

Der Automat \mathcal{A} : ($I \circ T$)

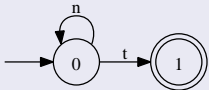


Lösung durch Abstraktion - Post-Sprachen

 $\mathcal{L}(A, q)$

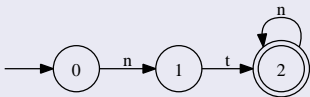
Die Post-Sprache $\mathcal{L}(A, q)$ beinhaltet alle Wörter, die in einem Automaten A ab dem Zustand q akzeptiert werden.

Der Automat \mathcal{B}



- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) = ntn^*$
- $\mathcal{L}(A, 1) = tn^*$
- $\mathcal{L}(A, 2) = n^*$

Der Automat \mathcal{A} : ($I \circ T$)



Lösung durch Abstraktion - Äquivalenzrelation

Äquivalenzrelation $q \simeq q'$

Zwei Zustände q, q' eines Automaten \mathcal{A} sind äquivalent, wenn für alle Zustände r eines Automaten \mathcal{B} gilt, dass

$\mathcal{L}(\mathcal{A}, q) \cap \mathcal{L}(\mathcal{B}, r) = \emptyset$ genau dann wenn $\mathcal{L}(\mathcal{A}, q') \cap \mathcal{L}(\mathcal{B}, r) = \emptyset$.

- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) = ntn^*$
- $\mathcal{L}(A, 1) = tn^*$
- $\mathcal{L}(A, 2) = n^*$

- $\mathcal{L}(A, 0) \cap \mathcal{L}(B, 0) = \{nt\}$
- $\mathcal{L}(A, 1) \cap \mathcal{L}(B, 0) = \{t\}$
- $\mathcal{L}(A, 2) \cap \mathcal{L}(B, 0) = \emptyset$
- $\mathcal{L}(A, 0) \cap \mathcal{L}(B, 1) = \emptyset$
- $\mathcal{L}(A, 1) \cap \mathcal{L}(B, 1) = \emptyset$
- $\mathcal{L}(A, 2) \cap \mathcal{L}(B, 1) = \{\varepsilon\}$

Lösung durch Abstraktion - Äquivalenzrelation

Äquivalenzrelation $q \simeq q'$

Zwei Zustände q, q' eines Automaten \mathcal{A} sind äquivalent, wenn für alle Zustände r eines Automaten \mathcal{B} gilt, dass

$\mathcal{L}(\mathcal{A}, q) \cap \mathcal{L}(\mathcal{B}, r) = \emptyset$ genau dann wenn $\mathcal{L}(\mathcal{A}, q') \cap \mathcal{L}(\mathcal{B}, r) = \emptyset$.

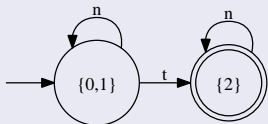
- $\mathcal{L}(B, 0) = n^*t$
- $\mathcal{L}(B, 1) = \{\varepsilon\}$
- $\mathcal{L}(A, 0) = ntn^*$
- $\mathcal{L}(A, 1) = tn^*$
- $\mathcal{L}(A, 2) = n^*$

Äquivalenzklassen: $[0, 1]$ und $[2]$.

- $\mathcal{L}(A, 0) \cap \mathcal{L}(B, 0) = \{nt\}$
- $\mathcal{L}(A, 1) \cap \mathcal{L}(B, 0) = \{t\}$
- $\mathcal{L}(A, 2) \cap \mathcal{L}(B, 0) = \emptyset$
- $\mathcal{L}(A, 0) \cap \mathcal{L}(B, 1) = \emptyset$
- $\mathcal{L}(A, 1) \cap \mathcal{L}(B, 1) = \emptyset$
- $\mathcal{L}(A, 2) \cap \mathcal{L}(B, 1) = \{\varepsilon\}$

Lösung durch Abstraktion - \mathcal{T}^{lim}

Der Restklassenautomat



Bilden von \mathcal{T}^{lim}

- Sequenz $((((I \circ T) / \simeq) \circ T) / \simeq \dots$
- Konvergenz der Sequenz ist \mathcal{T}^{lim}

Lösung durch Abstraktion - T^{lim}

Eigenschaften von $\mathcal{L}(T^{lim})$

$$\mathcal{L}(I \circ T^*) \subseteq \mathcal{L}(T^{lim})$$

Aussagen über T^*

- Wenn $T^{lim} \cap B = \emptyset$, dann $(I \circ T^*) \cap B = \emptyset$.
- Sonst testen ob Gegenbeispiel auch im Original Automaten nachvollzogen werden kann, ist dies der Fall, ist ein Gegenbeispiel gefunden.
- Andernfalls muss die Abstraktion mit einer feineren Äquivalenzklasse wiederholt werden.

Agenda

- 1 Einleitung
- 2 Beispiel
- 3 LTL, MSO, LTL(MSO)
- 4 Büchi-Transducer
- 5 Verifikation
- 6 Fazit**

Fazit

Was passiert ist

- die Haupttechniken beim Model-Checking mit regulären Sprachen
- Beschreibung des Verhaltens von Systemen deren Konfigurationen sich durch endliche Worte darstellen lassen
- Übergänge dieser Systeme durch Transducer (Transformation auf Sprache)

Was benutzt wurde

- Kombination von LTL und MSO
- reguläre Ausdrücke
- Automaten
- Sprachen