Seminar Model Checking

mu-Kalkül

Andreas Tonder

27. Januar 2005

Einführung

- der mu-Kalkül ist eine ausdrucksstarke Fixpunkt-Logik, um Eigenschaften von z.B. Transitions-Systemen zu beschreiben
- viele temporale Logiken, wie z.B. CTL,LTL und CTL^*, lassen sich in Formeln des mu-Kalkül übersetzen

$\Rightarrow$ Model Checking interessant für den mu-Kalkül !

- für gewisse Teilmengen des mu-Kalkül existieren effiziente Model Checking Algorithmen

Grundlagen

Definition

Ein *labeled transition system(LTS)* ist ein Tupel $M = (S, Act, \rightarrow)$, wobei

- S eine nicht-leere, endliche Zustandsmenge,
- Act eine Menge von Aktionen und
- $\rightarrow \subseteq S \times Act \times S$ eine Transitionsrelation ist.

Bemerkung: statt $(s, a, t) \in \rightarrow$ schreibt man $s \xrightarrow{a} t$

Grundlagen

Formeln im mu-Kalkül werden (z.B.) interpretiert bezüglich

- eines LTS $M = (S, Act, \rightarrow)$,
- einer Menge Var von Variablen($Var = \{X, Y, Z, \ldots\}$)
- einer Umgebung $e : Var \xrightarrow{part.} 2^S$ zur Interpretation von freien Variablen in Formeln

Syntax von mu-Kalkül -Formeln

Die Formeln werden wie folgt konstruiert:

- *true* und *false* sind Formeln
- Wenn $X \in Var$ ist, dann ist X eine Formel
- Wenn Φ und Ψ Formeln sind, dann sind $\Phi \vee \Psi$ und $\Phi \wedge \Psi$ Formeln
- Wenn Φ eine Formel ist, dann sind $\langle a \rangle \Phi$ und $[a]\Phi$ Formeln
- Wenn Φ eine Formel und $X \in Var$ ist, dann sind $\mu X.\Phi$ und $\nu X.\Phi$ Formeln

L_μ =Menge aller Formeln

Syntax von mu-Kalkül -Formeln

Syntax in Backus-Naur-Form

$$\Phi ::= \mathit{true} \mid \mathit{false} \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid [a]\Phi \mid \langle a \rangle \Phi \mid \mu X.\Phi \mid \nu X.\Phi$$

Semantik von mu-Kalkül -Formeln(informell)

Allgemein: Φ wird interpretiert als eine Menge von Zuständen, in denen Φ gilt

- *true* bzw. *false* entspricht der Menge aller Zustände bzw. der leeren Zustandsmenge
- $\vee$ und $\wedge$ wie üblich
- $\langle a \rangle \Phi$: „es ist möglich - mittels einer a-Transition - in einen Zustand zu gelangen, in dem Φ gilt“
- $[a]\Phi$: „Φ gilt in allen durch eine a-Transition erreichbaren Zuständen“
- $\mu X.\Phi$: „kleinster Fixpunkt“
- $\nu X.\Phi$: „größter Fixpunkt“

Semantik von mu-Kalkül -Formeln(formal)

Definition der sem. Funktion $[\![-]\!]_M : L_\mu \to (Var \to 2^S) \to 2^S$

- $[\![true]\!]_M e = S$
- $[\![false]\!]_M e = \emptyset$
- $[\![X]\!]_M e = e(X)$
- $[\![\Phi \vee \Psi]\!]_M e = [\![\Phi]\!]_M e \cup [\![\Psi]\!]_M e$
- $[\![\Phi \wedge \Psi]\!]_M e = [\![\Phi]\!]_M e \cap [\![\Psi]\!]_M e$
- $[\![\langle a \rangle \Phi]\!]_M e = \{s \mid \exists s' : s \xrightarrow{a} s' \wedge s' \in [\![\Phi]\!]_M e\}$
- $[\![[a]\Phi]\!]_M e = \{s \mid \forall s' : s \xrightarrow{a} s' \Rightarrow s' \in [\![\Phi]\!]_M e\}$

Semantik von Fixpunkt-Formeln

Sei $\tau : 2^S \rightarrow 2^S$ eine Abbildung.

- man nennt τ *monoton*, wenn gilt: $X_1 \subseteq X_2 \Rightarrow \tau(X_1) \subseteq \tau(X_2)$
- man nennt X' einen *Fixpunkt* von τ, wenn gilt: $\tau(X') = X'$
- Fixpukt X' ist der kleinste Fixpunkt, wenn gilt: $X' \subseteq X''$ für alle Fixpunkte X''
  Bezeichnung: $\mu X.\tau(X)$
- Fixpukt X' ist der größte Fixpunkt, wenn gilt: $X'' \subseteq X'$ für alle Fixpunkte X''
  Bezeichnung: $\nu X.\tau(X)$

Semantik von Fixpunkt-Formeln

Theorem

Sei $\tau : 2^S \rightarrow 2^S$ *eine monotone Abbildung. Dann gilt:*

$$\mu X.\tau(X) = \bigcup\{X \mid \tau(X) \subseteq X\}$$
$$\nu X.\tau(X) = \bigcap\{X \mid \tau(X) \supseteq X\}$$

Wenn $|S|$ *endlich gilt:*

$$\mu X.\tau(X) = \bigcup_{i \in \mathbb{N}_0} \tau^i(\mathit{false})$$
$$\nu X.\tau(X) = \bigcap_{i \in \mathbb{N}_0} \tau^i(\mathit{true})$$

Semantik von Fixpunkt-Formeln

Die semantische Funktion der Fixpunkte ist definiert durch:

- $[\![\mu X.\Phi]\!]_M e$ ist der kleinste Fixpunkt der Funktion $\tau : 2^S \to 2^S, \tau(Y) = [\![\Phi]\!]_M e[X \mapsto Y]$
- $[\![\nu X.\Phi]\!]_M e$ ist der größte Fixpunkt der Funktion $\tau : 2^S \to 2^S, \tau(Y) = [\![\Phi]\!]_M e[X \mapsto Y]$

Bemerkung

τ ist monoton, da $\vee, \wedge, [-]$ und $\langle - \rangle$ monoton sind.

$\Rightarrow$ die Fixpunkte sind wohldefiniert

Beispiele für mu-Kalkül -Formeln

Es sei $M = (S, \{a\}, \rightarrow)$.

- „Φ gilt auf allen a-Pfaden“
  $\nu X.(\Phi \wedge [a]X)$
- „Φ gilt schließlich auf einem a-Pfad“
  $\mu X.(\Phi \vee (\langle a \rangle \mathit{true} \wedge [a]X))$

Illustration des Unterschiedes von μ und ν

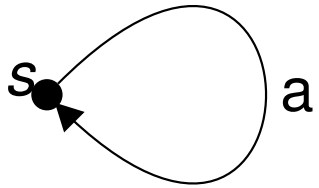
Intuitiv kann man sagen:

- größte Fixpunkte entsprechen Eigenschaften, die für immer gelten
- kleinste Fixpunkte entsprechen Eigenschaften, die schliesslich gelten

Illustration des Unterschiedes von μ und ν(Forts.)

Man betrachte folgende Formeln und folgendes LTS:

1. $\Phi_\mu = \mu X.(\langle a \rangle X \vee \langle b \rangle true)$
2. $\Phi_\nu = \nu X.(\langle a \rangle X \vee \langle b \rangle true)$



Auswertung der Formeln durch Bestimmung des kl. bzw. gr. Fixpunktes von $\tau(Y) = [\![\langle a \rangle X \vee \langle b \rangle true]\!]_M e[X \mapsto Y]$
Man erhält:

1. $[\![\Phi_\mu]\!]_M e = \emptyset$
2. $[\![\Phi_\nu]\!]_M e = \{s\}$

Globales und lokales Model Checking

Globales Model Checking Problem

Gegeben: endliches Modell M, Formel Φ
Aufgabe: bestimme die Menge der Zustände in M, die Φ erfüllen

Lokales Model Checking Problem

Gegeben: endliches Modell M, Formel Φ, Zustand s in M
Aufgabe: finde heraus, ob s die Formel Φ erfüllt

Ein naiver globaler Model Checking Algorithmus

```
1  function eval(Φ, e)
     if Φ = true then return S;
3    if Φ = false then return ∅;
     if Φ = X then return e(X);
5    if Φ = Ψ1 ∧ Ψ2 then
       return eval(Ψ1, e) ∩ eval(Ψ2, e);
7    if Φ = Ψ1 ∨ Ψ2 then
       return eval(Ψ1, e) ∪ eval(Ψ2, e);
9    if Φ = ⟨a⟩Ψ then
       return {s | ∃s' : s -a-> s' ∧ s' ∈ eval(Ψ, e)};
11   if Φ = [a]Ψ then
       return {s | ∀s' : s -a-> s' ⇒ s' ∈ eval(Ψ, e)};
```

Ein naiver Model Checking Algorithmus (Forts. 1)

```
if Φ = μX.Ψ(X) then
    X_val := false;
    repeat
        X_old = X_val;
        X_val = eval(Ψ, e[X ↦ X_val]);
    until X_val = X_old;
    return X_val;
end if
```

Ein naiver Model Checking Algorithmus (Forts. 2)

```
    if Φ = νX.Ψ(X) then
2       X_val := true;
        repeat
4           X_old = X_val;
            X_val = eval(Ψ, e[X ↦ X_val]);
6       until X_val = X_old;
        return X_val;
8   end if
end function
```

Komplexität des naiven Algorithmus

Es bezeichne

- k die maximale Verschachtelungstiefe der Fixpunkt-Operatoren,
- $|M| = |S| + |\rightarrow|$

Bemerkung

Der naive Algorithmus wertet eine mu-Kalkül -Formel Φ in $O((|M| \cdot |\Phi|)^k)$ Schritten aus.

Komplexität des naiven Algorithmus

- Problem: Algorithmus hat exponentielle Laufzeit, Exponent ist die max. Verschachtelungstiefe der Fixpunkt-Operatoren
- Grund: die Fixpunkt-Iteration wird zu Beginn immer mit *true* bzw. *false* initialisiert, d.h. bei verschachtelten Fixpunkten ist die Auswertung sehr aufwändig
- Abhilfe: bei der Auswertung von verschachtelten Fixpunkten können in manchen Fällen Zwischenergebnisse weiterverwendet werden

$\longrightarrow$ Emerson-Lei Algorithmus

Der Model Checking Algorithmus von Emerson-Lei

- Reduktion der Komplexität durch Wiederverwendung von Zwischenergebnissen: möglich bei direkt aufeinanderfolgenden Sequenzen von Fixpunkten gleichen Typs
- bei *alternierenden* Fixpunkten muss die Iteration immer *true* bzw. *false* beginnen

Beispiel

Man betrachte die folgenden Formeln:

1. $\mu X_1.\Phi_1(X_1, \mu X_2.\Phi_2(X_1, X_2))$
2. $\mu X_1.\Phi_1(X_1, \nu X_2.\Phi_2(X_1, X_2))$
3. $\mu X_1.\Phi_1(X_1, \nu X_2.\Phi_2(X_2))$

Der Model Checking Algorithmus von Emerson-Lei

- man nutzt aus, dass es zur Berechnung eines kl. bzw. gr. Fixpunktes genügt, die Iteration mit einer Approximation unterhalb bzw. oberhalb des gesuchten Fixpunktes zu starten
- zum Algorithmus:
  - der Alg. unterscheidet sich vom naiven nur in der Berechnung der Fixpunkte
  - verwendet ein Array A, um die Fixpunkt-Approximationen zu speichern
    $A[i]$ ist die Approximation der i-ten Fixpunktformel
  - Initial: $A[i] = \mathit{false}$,falls i-ter Fixpunktoperator μ ist
    $A[i] = \mathit{true}$, falls i-ter Fixpunktoperator ν ist

```
1  function eval(Φ, e)
     ... same as naive algorithm ...
3
     if Φ = μX_i.Ψ(X_i) then
5      forall toplevel greatest fixpoints
                 subformulas νX_j.Ψ'(X_j) of Ψ
7        do A[j] := true;
       repeat
9        X_old = A[i];
         A[i] := eval(Ψ, e[X_i ↦ A[i]]);
11     until A[i] = X_old;
       return A[i];
13   end if
```

Der Model Checking Algorithmus von Emerson-Lei(Forts.)

```
1   if Φ = νX_i.Ψ(X_i) then
      forall toplevel least fixpoint
3             subformulas μX_j.Ψ'(X_j) of Ψ
        do A[j] := false;
5     repeat
        X_old = A[i];
7       A[i] := eval(Ψ, e[X_i ↦ A[i]]);
      until A[i] = X_old;
9     return A[i];
    end if
11 end function
```

Komplexität des Algorithmus von Emerson-Lei

Bemerkung

Der Emerson-Lei Algorithmus wertet eine Formel Φ in $O((|\Phi| \cdot |M|)^d)$ Schritten aus, wobei d die *Alternierungs-Tiefe* von Φ ist.

Vergleich von naivem und Emerson-Lei Algorithmus

- naiver Alg.: exponentiell, wobei der Exponent die maximale Verschachtelungstiefe der Fixpunkt-Operatoren ist
- Emerson-Lei: exponentiell, wobei der Exponent die Alternierungstiefe der Fixpunkt-Operatoren ist

Model Checking mit Tableaus

- Tableau-Methode dient zur Lösung des lokalen Model Checking Problems:
  zu einem Modell M, einer Formel Φ und *einem* Zustand s finde heraus, ob $s \models^M \Phi$
- globale Information unnötig, da lediglich die Zustände, die von s aus erreichbar sind, betrachtet werden
- die Beweissuche für $s \models^M \Phi$ wird „von oben nach unten“ durchgeführt, d.h. eine Formel wird durch Anwendung von „subgoaling rules“ in ihre Bestandteile zerlegt
- bei der Beweissuche ensteht dann ein sog. *Tableau*

Subgoaling rules (Teil 1)

Initiale Anfrage: $s \vdash_\Delta \Phi$, wobei Δ eine Umgebung ist

$$\frac{s \vdash_\Delta \Phi_1 \wedge \Phi_2}{s \vdash_\Delta \Phi_1 \qquad s \vdash_\Delta \Phi_2}$$

$$\frac{s \vdash_\Delta \Phi_1 \vee \Phi_2}{s \vdash_\Delta \Phi_1}$$

$$\frac{s \vdash_\Delta \Phi_1 \vee \Phi_2}{s \vdash_\Delta \Phi_2}$$

$$\frac{s \vdash_\Delta [a]\Phi}{s_1 \vdash_\Delta \Phi \cdots s_n \vdash_\Delta \Phi} \quad \textit{if} \quad \{s_1, \cdots, s_n\} = \{s' \mid s \xrightarrow{a} s'\}$$

$$\frac{s \vdash_\Delta \langle a \rangle \Phi}{s \vdash_\Delta \Phi} \quad \textit{if} \quad s \xrightarrow{a} s'$$

$$\frac{s \vdash_\Delta \textit{true}}{}$$

Erfolgreiche Tableaus

Definition

Man nennt ein Tableau *erfolgreich*, wenn an allen seinen Blätter nichts steht.

Satz (Vollständigkeit)

1. *Die Existenz eines erfolgreichen Tableaus für* $s \vdash_\Delta \Phi$ *impliziert* $s \models_M \Phi$.
2. *Existiert kein erfolgreiches Tableau für* $s \vdash_\Delta \Phi$, *so gilt* $s \not\models_M \Phi$.

Subgoaling rules (Teil 2)

- Fixpunkt-Formeln werden mittels „unfolding rules“ analysiert
- da wir endliche Modelle betrachten existiert ein Abbruchkriterium für die Suche:
  - Wenn eine Anfrage $s \vdash_\Delta \mu X.\Phi(X)$ das Teilziel $s \vdash_\Delta X$ hat, so folgert man, dass das Teilziel *nicht erfolgreich* ist
  - Grund:

$$s \models^M \mu X.\Phi(X) \quad \text{iff} \quad s \models^M \bigvee_{i \geq 0} X_i$$

$$\text{wobei } X_0 = \mathit{false},\ X_{i+1} = \Phi(X_i)$$

Subgoaling rules (Teil 2)

-
  - Duale Anfrage: Wenn eine Anfrage $s \vdash_\Delta \nu X.\Phi(X)$ das Teilziel $s \vdash_\Delta X$ hat, so folgert man, dass das Teilziel *erfolgreich* ist
  - Grund:

$$s \models^M \nu X.\Phi(X) \quad \text{iff} \quad s \models^M \bigwedge_{i \geq 0} X_i$$

$$\text{wobei } X_0 = \mathit{true},\ X_{i+1} = \Phi(X_i)$$

Subgoaling rules (Teil 2)

$$\frac{s \vdash_\Delta \mu X.\Phi(X)}{s \vdash_{\Delta'} \mathcal{U}} \quad \text{where } \Delta' = \Delta + [\mathcal{U} \mapsto \mu X.\Phi(X)] \text{ and } \mathcal{U} \text{ fresh for } \Delta$$

$$\frac{s \vdash_\Delta \mathcal{U}}{s \vdash_\Delta \Phi(\mathcal{U})} \quad \text{where } \Delta(\mathcal{U}) = \mu X.\Phi(X)$$

$$\frac{s \vdash_\Delta \nu X.\Phi(X)}{s \vdash_{\Delta'} \mathcal{U}} \quad \text{where } \Delta' = \Delta + [\mathcal{U} \mapsto \nu X.\Phi(X)] \text{ and } \mathcal{U} \text{ fresh for } \Delta$$

$$\frac{s \vdash_\Delta \mathcal{U}}{s \vdash_\Delta \Phi(\mathcal{U})} \quad \text{where } \Delta(\mathcal{U}) = \nu X.\Phi(X)$$

Erfolgreiche Tableaus, Erweiterung

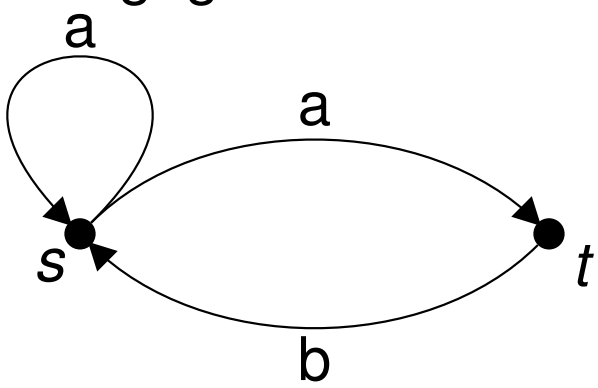
Definition

Man nennt ein Tableau *erfolgreich*

1. wenn für Blätter der Form $s \vdash_\Delta \mathcal{U}$ gilt: $\Delta(\mathcal{U}) = \nu X.\Phi(X)$ und
2. wenn an allen anderen Blättern nichts steht.

Beispiel: Model Checking mit Tableaus

Es sei folgendes LTS M gegeben:



Bestimme Tableau für die folgende Anfrage:
$s \vdash_{\emptyset} \nu X.(\mu Y.\langle a \rangle true \vee \langle b \rangle Y) \wedge [b]X$

Übersetzung von CTL-Formel in den mu-Kalkül

- die CTL-Formeln lassen sich in mu-Kalkül -Formeln der Alternierungs-Tiefe 1 übersetzen

Beispiel

$$
\begin{aligned}
AU(\Phi,\Psi) &\equiv \mu X.(\Psi \vee (\Phi \wedge []X \wedge \langle\rangle true)) \\
EU(\Phi,\Psi) &\equiv \mu X.(\Psi \vee (\Phi \wedge \langle\rangle X)) \\
AG(\Phi) &\equiv \nu X.\Phi \wedge []X \\
AF(\Phi) &\equiv \mu X.\Phi \vee []X
\end{aligned}
$$

Ende

Fragen?