Kapitel IV

Wiederholung

Symbolisches Model Checking

Gegeben:

- CTL-Interpretation $\mathcal{I}$ mit Transitionssystem $(S, \rightarrow)$
- CTL-Formel φ

Ziel: Berechne die Menge aller Zustände, in denen φ gilt, als BDD.

Berechnung der Menge aller Zustände, in denen φ gilt:
Labelling-Algorithmus.
In Mengennotation: Berechnung der Menge $\mathcal{L}(\varphi)$ aller Zustände, die φ erfüllen.

- $\mathcal{L}(\top) := S$
- $\mathcal{L}(p) := \mathcal{I}(p)$ für aussagenlogische Variable p
- $\mathcal{L}(\neg\varphi_1) := S \backslash \mathcal{L}(\varphi_1)$
- $\mathcal{L}(\varphi_1 \wedge \varphi_2) := \mathcal{L}(\varphi_1) \cap \mathcal{L}(\varphi_2)$

Symbolisches Model Checking

- $\mathcal{L}(\mathsf{E}[\varphi_1 \mathsf{U} \varphi_2])$: Berechne $B_0 \subseteq B_1 \subseteq B_2 \subseteq \ldots$, bis irgendwann B_n mit $B_n = B_{n+1}$ erreicht ist. Das Ergebnis ist dann B_n.

$$B_0 := \mathcal{L}(\varphi_2)$$
$$B_{i+1} := \mathcal{L}(\varphi_2) \cup (\mathcal{L}(\varphi_1) \cap \mathsf{EX}(B_i))$$

  Dabei ist $\mathsf{EX}(B) := \{s \in S \mid \exists s'.\, s \to s' \wedge s \in B\}$ die Menge aller Vorgänger von B.

  Man überlegt sich: B_i ist die Menge der Zustände s, für die ein Pfad $s = s_0 \to s_1 \to \cdots \to s_k$ mit $k \leqslant i$ existiert, so dass $s_k \models \varphi_2$ und $s_j \models \varphi_1$ für alle $j < k$.

- …

Symbolisches Model Checking

Beispiel: Aufgabe 4-2 c): $E[\top \mathsf{U} q]$

- $\mathcal{L}(\top) = S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$
- $\mathcal{L}(q) = \{s_1\}$
- $\mathcal{L}(E[\top \mathsf{U} q]) = \{s_0, s_1, s_2, s_3, s_4\}$

$$
\begin{aligned}
B_0 &= \mathcal{B}(q) = \{s_1\} \\
B_1 &= \mathcal{B}(q) \cup (S \cap \mathsf{EX}(B_0)) = \{s_0, s_1, s_2\} \\
B_2 &= \mathcal{B}(q) \cup (S \cap \mathsf{EX}(B_1)) = \{s_0, s_1, s_2, s_3, s_4\} \\
B_3 &= \mathcal{B}(q) \cup (S \cap \mathsf{EX}(B_2)) = \{s_0, s_1, s_2, s_3, s_4\}
\end{aligned}
$$

Symbolisches Model Checking

Die Mengenoperationen werden mit BDDs implementiert.

Die gegebenen Daten werden als BDDs kodiert.

- Kodierung der Zustände durch Variablen festlegen (Folie 93).
- Die Menge $\mathcal{L}(\varphi)$ soll durch ein BDD kodiert (Folie 93) berechnet werden.
- Transitionssystem kodieren: `sanity` repräsentiert die Menge S, `next` repräsentiert die Relation $\rightarrow$ wie auf Folie 96 festgelegt.
- Interpretation kodieren, d.h. für jede Menge $\mathcal{I}(p)$ durch ein BDD kodieren.

Implementiert man die Mengenberechnungen der vorangegangenen Folie mit den Operationen für BDDs (siehe auch Folie 95), so erhält man das symbolische Model-Checking von Folien 200–208.

CTL mit Fairness

- Motivation: In nebenläufigen Systemen wollen wir nur Ausführungen betrachten, in denen jeder Prozess immer wieder dran kommt.
- Semantik unterscheidet sich von der für CTL nur darin, dass immer über faire Pfade quantifiziert wird.
- Ein Zustand s erfüllt EGφ wenn es einen fairen unendlichen Pfad gibt, auf dem überall φ gilt.
- Ein Zustand s erfüllt E$[\varphi_1$U$\varphi_2]$ wenn es einen fairen unendlichen Pfad gibt, auf dem irgendwann φ_2 gilt und bis dahin φ_1.

Labelling-Algorithmus mit Fairness

- $\bot$: Markiere keinen Zustand mit $\bot$.
- p: Markiere Zustände mit aussagenlogischen Variablen, wie von der Interpretation vorgegeben.
- $\neg\varphi$: Führe den Algorithmus rekursiv für φ aus. Markiere danach alle Zustände mit $\neg\varphi$, die nicht mit φ beschriftet sind.
- $\varphi \wedge \psi$: Führe den Algorithmus rekursiv für φ und ψ aus. Markiere danach alle Zustände mit $\varphi \wedge \psi$, die sowohl mit φ als auch mit ψ beschriftet sind.
- EXφ: Führe den Algorithmus rekursiv für φ aus. Markiere dann alle Zustände mit EXφ, die einen unmittelbaren Nachfolger haben, der schon mit φ markiert ist.

(Diese Fälle sind genau wie im normalen Labelling-Algorithmus. Nur die Fälle für EG und E[$-$U$-$] ändern sich.)

Labelling-Algorithmus mit Fairness

- EGφ:
  - Führe den Algorithmus rekursiv für φ aus.
  - Betrachte den Teilgraphen G des Transitionsystems, der aus den mit φ markierten Zuständen besteht. Zwischen diesen Knoten hat G die gleichen Kanten wie das Transitonssystem.
  - Markiere in G alle Knoten, von denen aus eine nichttriviale faire SCC in G erreichbar ist.
  - Markiere im ursprünglichen Transitionssystem alle Zustände, die in G markiert sind.
- E$[\varphi$U$\psi]$:
  1. Führe den Algorithmus rekursiv für φ, ψ und EG$\top$ aus.
  2. Markiere alle Zustände mit E$[\varphi$U$\psi]$, die schon mit ψ und EG$\top$ markiert sind.
  3. Ist *ein* unmittelbarer Folgezustände eines Zustands s bereits mit E$[\varphi$U$\psi]$ markiert und ist s selbst mit φ markiert, so markiere s auch mit E$[\varphi$U$\psi]$. Wiederhole Schritt 3 bis keine neuen Markierungen mehr hinzukommen.

Typinferenz im polymorphen Typsystem

$$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) =$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) =$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f f}) =$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f f}) =$
    - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_2 \to \alpha_2)$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\mathsf{x} \mapsto \alpha_1], \mathsf{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\mathsf{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \mathsf{f\ f}) =$
    - $\mathcal{W}([\mathsf{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \mathsf{f}) = (\emptyset, \alpha_2 \to \alpha_2)$
    - $\mathcal{W}([\mathsf{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \mathsf{f}) = (\emptyset, \alpha_3 \to \alpha_3)$

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1.\, \alpha_1 \to \alpha_1], \texttt{f f}) =$
  - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1.\, \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_2 \to \alpha_2)$
  - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1.\, \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_3 \to \alpha_3)$
  - $\mathcal{U}(\alpha_2 \to \alpha_2, (\alpha_3 \to \alpha_3) \to \alpha_4)$ liefert die Substitution $\theta := [\alpha_2 \mapsto (\alpha_3 \to \alpha_3), \alpha_4 \mapsto (\alpha_3 \to \alpha_3)]$.

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\varnothing, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\varnothing, \texttt{fun x -> x}) = (\varnothing, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\varnothing, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 .\, \alpha_1 \to \alpha_1], \texttt{f f}) = (\theta, (\alpha_3 \to \alpha_3))$
    - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 .\, \alpha_1 \to \alpha_1], \texttt{f}) = (\varnothing, \alpha_2 \to \alpha_2)$
    - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 .\, \alpha_1 \to \alpha_1], \texttt{f}) = (\varnothing, \alpha_3 \to \alpha_3)$
    - $\mathcal{U}(\alpha_2 \to \alpha_2, (\alpha_3 \to \alpha_3) \to \alpha_4)$ liefert die Substitution $\theta := [\alpha_2 \mapsto (\alpha_3 \to \alpha_3), \alpha_4 \mapsto (\alpha_3 \to \alpha_3)]$.

Typinferenz im polymorphen Typsystem

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) = (\theta, (\alpha_3 \to \alpha_3))$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f f}) = (\theta, (\alpha_3 \to \alpha_3))$
  - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_2 \to \alpha_2)$
  - $\mathcal{W}([\texttt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_3 \to \alpha_3)$
  - $\mathcal{U}(\alpha_2 \to \alpha_2, (\alpha_3 \to \alpha_3) \to \alpha_4)$ liefert die Substitution $\theta := [\alpha_2 \mapsto (\alpha_3 \to \alpha_3), \alpha_4 \mapsto (\alpha_3 \to \alpha_3)]$.

Herleitung im polymorphen Typsystem

$$
\dfrac{
\text{(GEN)}\ \dfrac{\text{(FN)}\ \dfrac{\text{(VAR)}\ \dfrac{}{[\mathtt{x} \mapsto \alpha_1] \vdash \mathtt{x} : \alpha_1 \to \alpha_1}}{\vdash \mathtt{fun\ x\ ->\ x} : \alpha_1 \to \alpha_1}}{\vdash \mathtt{fun\ x\ ->\ x} : \forall \alpha_1 . \alpha_1 \to \alpha_1}
\qquad
\text{(APP)}\ \dfrac{(1) \qquad (2)}{[\mathtt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1] \vdash \mathtt{f\ f} : \alpha_3 \to \alpha_3}
}{\vdash \mathtt{let\ f\ =\ fun\ x\ ->\ x\ in\ f\ f} : \alpha_3 \to \alpha_3}
$$

Teilherleitung (1) ist:

$$
\text{(INST)}\ \dfrac{\text{(VAR)}\ \dfrac{}{[\mathtt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1] \vdash \mathtt{f} : \forall \alpha_1 . \alpha_1 \to \alpha_1}}{[\mathtt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1] \vdash \mathtt{f} : (\alpha_3 \to \alpha_3) \to (\alpha_3 \to \alpha_3)}
$$

Teilherleitung (2) ist:

$$
\text{(INST)}\ \dfrac{\text{(VAR)}\ \dfrac{}{[\mathtt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1] \vdash \mathtt{f} : \forall \alpha_1 . \alpha_1 \to \alpha_1}}{[\mathtt{f} \mapsto \forall \alpha_1 . \alpha_1 \to \alpha_1] \vdash \mathtt{f} : \alpha_3 \to \alpha_3}
$$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:
$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) =$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) =$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f f}) =$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f} = \texttt{fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f f}) =$
  - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \rightarrow \alpha_1)$
  - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \rightarrow \alpha_1], \texttt{f f}) =$
  - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \rightarrow \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \rightarrow \alpha_1)$
  - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \rightarrow \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \rightarrow \alpha_1)$

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f f}) =$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{U}(\alpha_1 \to \alpha_1, (\alpha_1 \to \alpha_1) \to \alpha_2)$ liefert `fail`!

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) =$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f f}) = \texttt{fail}$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{U}(\alpha_1 \to \alpha_1, (\alpha_1 \to \alpha_1) \to \alpha_2)$ liefert `fail`!

Was passiert im einfachen Typsystem?

Im einfachen Typsystem ohne Polymorphie:

$\mathcal{W}(\emptyset, \texttt{let f = fun x -> x in f f}) = \texttt{fail}$

- $\mathcal{W}(\emptyset, \texttt{fun x -> x}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{x} \mapsto \alpha_1], \texttt{x}) = (\emptyset, \alpha_1)$
- $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f f}) = \texttt{fail}$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{W}([\texttt{f} \mapsto \alpha_1 \to \alpha_1], \texttt{f}) = (\emptyset, \alpha_1 \to \alpha_1)$
    - $\mathcal{U}(\alpha_1 \to \alpha_1, (\alpha_1 \to \alpha_1) \to \alpha_2)$ liefert `fail`!

Ansatz zur Herleitung

$$\dfrac{(\textsc{App})\ \dfrac{(1) \qquad (2)}{[\mathtt{f} \mapsto (\alpha \to \alpha)] \vdash \mathtt{f\ f} : ?} \qquad (\textsc{Fn})\ \dfrac{(\textsc{Var})\ \dfrac{}{[\mathtt{x} \mapsto \alpha] \vdash \mathtt{x} : \alpha \to \alpha}}{\vdash \mathtt{fun\ x\ \text{->}\ x} : \alpha \to \alpha}}{\vdash \mathtt{let\ f} = \mathtt{fun\ x\ \text{->}\ x\ in\ f\ f} : ?}$$

Es gibt keinen Typ τ, so dass

$$[\mathtt{f} \mapsto (\alpha \to \alpha)] \vdash \mathtt{f\ f} : \ \tau$$

herleitbar ist (weder im einfachen, noch im polymorphen Typsystem).