6 – SYMBOLIC MODEL CHECKING

Sommersemester 2009

Dr. Carsten Sinz, Universität Karlsruhe

Verbände

- **Def.:** Eine Menge M mit zwei binären Operatoren $\sqcap$ und $\sqcup$ heißt *Verband*, wenn:
  - $\sqcap$ und $\sqcup$ sind kommutativ und assoziativ,
  - für $\sqcap$ und $\sqcup$ gelten die Absorptionsgesetze, d.h. $a\sqcap(a\sqcup b)=a$ und $a\sqcup(a\sqcap b)=a$
- **Def.:** Ein Verband $(M, \sqcap, \sqcup)$ heißt *vollständig*, falls jede Teilmenge von M ein Supremum und Infimum besitzt.
- Über $a\sqsubseteq b$ gdw. $a\sqcap b=a$ ist eine Verbandsordnung (Halbordnung) definiert.
- Für jede Menge M ist $(\mathbf{P}(M), \cap, \cup)$ ist ein vollständiger Verband mit der Verbandsordnung $\subseteq$.

Prädikat-Transformer

- Gegeben Kripke-Struktur M=(S,T,L)
- Potenzmenge **P**(S) bildet Verband
- Ein Prädikat über Zuständen (z.B. Zustände, in denen p gilt), kann als Teilmenge von **P**(S) aufgefasst werden.
- **Def.:** Ein *Prädikat-Transformer* (PT) ist eine Funktion τ: **P**(S)$\rightarrow$**P**(S).

Fixpunkte

- **Def.:** Sei τ: **P**(S)$\rightarrow$**P**(S) ein Prädikat-Transformer.
  - τ ist *monoton*, wenn aus $P \subseteq Q$ folgt, dass $\tau(P) \subseteq \tau(Q)$
  - τ ist *$\cup$-stetig*, falls aus $P_1 \subseteq P_2 \subseteq \ldots$ folgt, dass $\tau(\cup_i P_i) = \cup_i \tau(Pi)$.
  - τ ist *$\cap$-stetig*, falls aus $P_1 \supseteq P_2 \supseteq \ldots$ folgt, dass $\tau(\cap_i P_i) = \cap_i \tau(Pi)$.
- **Def.:** Eine Menge $S' \subseteq S$ ist ein *Fixpunkt* von τ, falls $\tau(S')=S'$.
- **Satz:** Jeder monotone Prädikat-Transformer besitzt einen kleinsten ($\mu Z.\ \tau(Z)$) und größten Fixpunkt ($\nu Z.\ \tau(Z)$).

Fixpunktberechnung

- **Satz (Tarski / Kleene):**
  1. Falls τ monoton ist, so ist $\mu Z.\ \tau(Z) = \bigcap\{Z \mid \tau(Z) \subseteq Z\}$ und $\nu Z.\ \tau(Z) = \bigcup\{Z \mid \tau(Z) \supseteq Z\}$.
  2. Falls τ monoton und $\bigcup$-stetig ist, so ist $\mu Z.\ \tau(Z) = \bigcup_i \tau^i(\varnothing)$.
  3. Falls τ monoton und $\bigcap$-stetig ist, so ist $\nu Z.\ \tau(Z) = \bigcup_i \tau^i(S)$.
  4. Falls τ monoton ist, so ist $\tau^{i+1}(\varnothing) \supseteq \tau^i(\varnothing)$ und $\tau^{i+1}(S) \subseteq \tau^i(S)$.
- **Lemma:** Falls S endlich und τ monoton ist, so ist τ immer auch $\bigcup$-stetig und $\bigcap$-stetig.
- **Anmerkung:** Wir werden nur endliche Mengen S betrachten, so dass Fixpunkte immer mittels 2. und 3. berechnet werden können.
- Fixpunkte können also über die folgenden Sequenzen angenähert werden:

  $\mu Z.\ \tau(Z)$: $\varnothing, \tau(\varnothing), \tau^2(\varnothing), \tau^3(\varnothing), \ldots$

  $\nu Z.\ \tau(Z)$: $S, \tau(S), \tau^2(S), \tau^3(S), \ldots$

  Sobald $\tau^{i+1}(Z) = \tau^i(Z)$ $(Z \in \{\varnothing, S\})$, kann die Berechnungssequenz abgebrochen werden, der Fixpunkt ist gefunden.

Fixpunktberechnung – Algorithmen


```
Predicate
LFP(PredicateTransformer tau)
{
    Q := false;
    Q' := tau(Q);
    while(Q ≠ Q') do {
        Q := Q';
        Q' := tau(Q');
    }
    return Q;
}
```

```
Predicate
GFP(PredicateTransformer tau)
{
    Q := true;
    Q' := tau(Q);
    while(Q ≠ Q') do {
        Q := Q';
        Q' := tau(Q');
    }
    return Q;
}
```

Beschreibung von Mengen durch Formeln

- Sei S eine (endliche) Menge mit 2^k Elementen und $P = \{p_1,\ldots,p_k\}$ eine Menge von aussagenlogischen Variablen.
  - Jedes Element $a \in S$ lässt sich eineindeutig einer Variablenbelegung α_a: $P \rightarrow \{0,1\}$ zuordnen.
  - Damit lässt sich jede Teilmenge S' von S über eine aussagenlogische Formel $F_{S'}$ beschreiben, für die gilt: $\alpha_a \models F_{S'}$ genau dann, wenn $a \in S'$.
- **Beispiel:** $S = \{0,1,2,3\}$, $P=\{p_1,p_2\}$, α_a sei so definiert, dass p_2p_1 die Binärdarstellung von a ist.
  - $S' = \varnothing$ ➠ $F_{\varnothing} = \mathit{false}$
  - $S' = \{0,1\}$ ➠ $F_{\{0,1\}} = \neg p_2$
  - $S' = \{0,1,2\}$ ➠ $F_{\{0,1,2\}} = \neg(p_1 \wedge p_2)$
- **Anmerkungen:**
  - Die Formeldarstellung ist nicht eindeutig.
  - Für Mengen mit einer Mächtigkeit ungleich 2^k lassen sich "unzulässige Elemente" durch eine zusätzliche Annahme bei Beweisen ausschließen.

Fixpunkt-Repräsentation von CTL-Operatoren

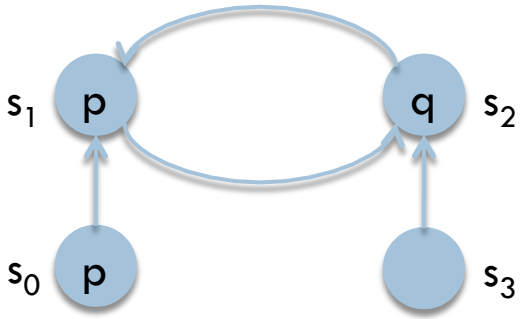
- Im Folgenden identifizieren wir Zustandsmengen mit Formeln; Fixpunktberechnungen sind dann auch auf Formeln definiert
- CTL-Operatoren lassen sich als Fixpunkte beschreiben:

$$\mathbf{AF}\, f = \mu Z \,.\, f \vee \mathbf{AX}\, Z \qquad \mathbf{A}[f\, \mathbf{U}\, g] = \mu Z \,.\, g \vee (f \wedge \mathbf{AX}\, Z)$$
$$\mathbf{EF}\, f = \mu Z \,.\, f \vee \mathbf{EX}\, Z \qquad \mathbf{E}[f\, \mathbf{U}\, g] = \mu Z \,.\, g \vee (f \wedge \mathbf{EX}\, Z)$$
$$\mathbf{AG}\, f = \nu Z \,.\, f \wedge \mathbf{AX}\, Z \qquad \mathbf{A}[f\, \mathbf{R}\, g] = \nu Z \,.\, g \wedge (f \vee \mathbf{EX}\, Z)$$
$$\mathbf{EG}\, f = \nu Z \,.\, f \wedge \mathbf{EX}\, Z \qquad \mathbf{E}[f\, \mathbf{R}\, g] = \nu Z \,.\, g \wedge (f \vee \mathbf{EX}\, Z)$$

- **Hinweise:**
  - Die „Mengen“ Z in diesen Fixpunktberechnungen sind Formeln!
  - Die nach Anwendung der Operatoren **EX** und **AX** entstehenden Zustandsmengen müssen in Formeln zurückübersetzt werden

Fixpunktberechnung: Beispiel

Kripke-Struktur:



Zu prüfende Eigenschaft F:

E[p **U** q]

Prädikat-Transformer für F:

$\tau(Z) = q \vee (p \wedge \mathbf{EX}\ Z)$

Codierung der Zustände:

$s_0 = \neg a \wedge \neg b$, $s_1 = \neg a \wedge b$

$s_2 = a \wedge \neg b$, $s_3 = a \wedge b$

Fixpunktberechnung:

$Q_0 = \text{false}$ $\quad \varnothing$

$Q_1 = \tau(Q_0) = (a \wedge \neg b) \vee (\neg a \wedge \mathbf{EX}\ \text{false}) = (a \wedge \neg b)$ $\quad \{s_2\}$

$Q_2 = \tau(Q_1) = (a \wedge \neg b) \vee (\neg a \wedge \mathbf{EX}\ (a \wedge \neg b))$

$= (a \wedge \neg b) \vee (\neg a \wedge b) = (a \oplus b)$ $\quad \{s_1, s_2\}$

$Q_3 = \tau(Q_2) = (a \wedge \neg b) \vee (\neg a \wedge \mathbf{EX}\ (a \oplus b))$

$= (a \wedge \neg b) \vee (\neg a \wedge \text{true}) = \neg a \vee \neg b$ $\quad \{s_0, s_1, s_2\}$

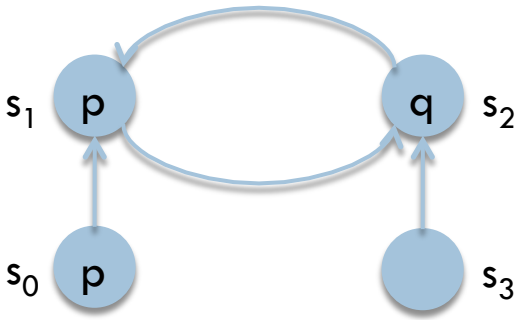
$Q_4 = Q_3$

Symbolic Model Checking (I)

- **Idee:** Verwende BDDs zur Darstellung von sowohl Zustandsmengen als auch Übergangsfunktion.
  - Darstellung von Mengen als BDDs: klar (Menge ➠ Formel ➠ BDD)
  - Darstellung der Übergangsrelation:
    - Verwende Variablen P zur Darstellung des Zustands vor Übergang und P' nach Übergang
    - Generiere dann Formel R(x,x'), die genau dann wahr ist, wenn ein Übergang von Zustand x in Zustand x' möglich ist.

Fortsetzung Beispiel

Kripke-Struktur:



Codierung der Zustände:

$s_0 = \neg a \wedge \neg b$, $s_1 = \neg a \wedge b$

$s_2 = a \wedge \neg b$, $s_3 = a \wedge b$

Darstellung der Übergangsrelation als Formel:

- Variablen zur Darstellung von Zuständen:
  a, b
- Demnach Variablen für Übergangsrelation:
  a, b, a', b'
- Übersetze Übergangsrelation in Formel R(a,b,a',b'):

$(\neg a \wedge \neg b \wedge \neg a' \wedge b') \vee$	"$s_0 \rightarrow s_1$"
$(\neg a \wedge b \wedge a' \wedge \neg b') \vee$	"$s_1 \rightarrow s_2$"
$(a \wedge \neg b \wedge \neg a' \wedge b') \vee$	"$s_2 \rightarrow s_1$"
$(a \wedge b \wedge a' \wedge \neg b')$	"$s_3 \rightarrow s_2$"

- Eventuell Vereinfachung von R möglich (wird durch BDD "automatisch" vorgenommen)

Symbolic Model Checking (II)

- Für gegebene Kripke-Struktur Prüfung einer Formel F mittels check-Funktionen:
  - check: CTL → BDD
  - checkEX: BDD → BDD
  - checkEU: BDD → BDD
  - checkEG: BDD → BDD
- check(F) gibt die Zustände an, die Formel F wahr machen
  - Eine CTL Formel F gilt in einer Kripke-Struktur, wenn check(F)=1

SMC: Funktion *check*

- check(F):
  - Falls F eine atomare Aussage a ist, dann ist F der BDD, der alle Zustände, in denen a gilt, repräsentiert.
  - Falls $F = F_1 \wedge F_2$, $F = F_1 \wedge F_2$ oder $F = \neg F_1$: berechne BDDs für Subformeln mittels check(F_1) und check(F_2), verwende dann BDD-Operation (*Apply*) zur Berechnung von F.
  - Falls F mit Temporaloperator beginnt:
    - check(**EX** f) = checkEX(check(f))
    - check(**E**[f **U** g]) = checkEU(check(f), check(g))
    - check(**EG** f) = checkEG(check(f))
    - Andere Temporaloperatoren können in die vorigen drei umgeschrieben werden.

SMC: Funktion *checkEX*

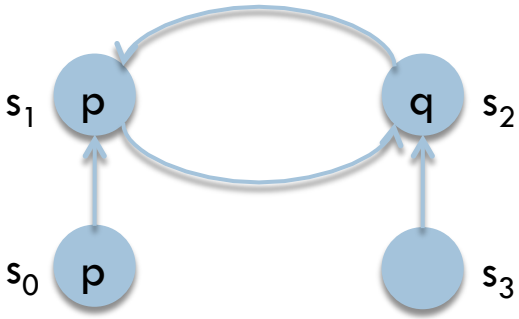
- checkEX(f(**x**)) = $\exists$ **x**' . f(**x**') $\wedge$ R(**x**,**x**')
  - **x** = (x_1,…,x_n) sind Zustandsvariablen
  - R(**x**,**x**') ist BDD der Übergangsrelation
  - f(**x**) ist BDD, der die Zustandsmenge f (in Abhängigkeit von den Zustandsvariablen **x**) beschreibt
  - $\exists$ **x** ist existentielle Abstraktion auf BDDs, d.h. $\exists$ **x** . f = $\exists$ x_1… $\exists$ x_n . f, wobei $\exists$ x_i . f = $f|_{xi=0} \vee f|_{xi=1}$

SMC: Funktionen *checkEU*, *checkEG*

- checkEU(f, g) = μZ . g $\vee$ (f $\wedge$ **EX** Z)
  - Mittels des Algorithmus LFP wird Fixpunkt berechnet.
  - Fixpunktberechnung durch Approximationssequenz $Q_0, Q_1, \ldots, Q_k$ mit Q_0 = false.
  - Dabei wird Q_{i+1} anhand von f, g und Q_i unter Zuhilfenahme der Funktion *checkEX* berechnet.
- checkEG(f) = νZ . f $\wedge$ **EX** Z
  - Berechnung analog zu *checkEU*.

Fortsetzung Beispiel

Kripke-Struktur:



Codierung der Zustände:

$s_0 = \neg a \wedge \neg b$, $s_1 = \neg a \wedge b$

$s_2 = a \wedge \neg b$, $s_3 = a \wedge b$

Zu prüfende Eigenschaft F: **E**[p **U** q]

Berechnung von *check*(F):

check(F) = checkEU(check(p), check(q))

= checkEU($\neg a$, $a \wedge \neg b$)

= $\mu Z . (a \wedge \neg b) \vee (\neg a \wedge$ **EX** Z)

Fixpunktberechnung:

Q_0 = false

Q_1 = $(a \wedge \neg b) \vee (\neg a \wedge$ checkEX(false))

= $a \wedge \neg b$

Q_2 = $(a \wedge \neg b) \vee (\neg a \wedge$ checkEX($a \wedge \neg b$))

= $(a \wedge \neg b) \vee (\neg a \wedge \exists a', b' . (a' \wedge \neg b') \wedge R(a,b,a',b'))$

= $(a \wedge \neg b) \vee (\neg a \wedge b) = a \oplus b$

Q_3 = ...

Vergleich mit *Explicit State Model Checking*

- BDDs können deutlich kompaktere Repräsentationen von Zustandsmengen und der Übergangsrelation ermöglichen.
  - Grober Richtwert: BDDs bis ca. 200-400 Zustandsvariablen erfolgreich (siehe auch Paper von Burch *et al.*: Symbolic Model Checking: 10^{20} States and Beyond)
  - Beispiel: n-Bit-Zähler
- SMC erfolgreich eingesetzt in der Hardware-Verifikation