Freie Universität Berlin

Embedding Imperative Synchronous Languages in Interactive Theorem Provers

J. Sydow C. Tietz
Freie Universität Berlin

12. Februar 2013

Outline

- Einbetten von Quartz in Theorembeweiser
- Programme werden Formeln
- Beweise über Programme/Eigenschaften
- Modelchecking

HOL

- HOL = Higher order logic
- Theorembeweiser
- Hardwareverifikation
- Logik in ML implementiert

Quartz

- entwickelt an der Universität Karlsruhe
- Esterel-Variante
- Teil des Averest Systems
- für reaktive Systeme
- kompiliert zu C, Verilog

Quartz

Unterschiede zu Esterel:

- Verzögerte Zuweisung
- Asynchrone Nebenläufigkeit
- Nichtdeterministisches Verhalten

Quartz

- Semantikbeschreibung mit primitiver Rekursion
- Beweiser kann Induktionsregeln ableiten
- Schleifen nicht modellierbar
- Trick: Aufteilung in Kontrollfluss und Datenfluss

Outline

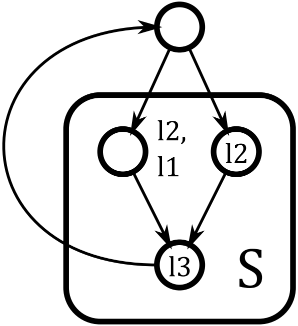
Kontrollfluss

- Programmablauf immer an pause-Statement
- Zeit modelliert als $t \in \mathbb{N}$
- Kontrollfluss modelliert mit endlichem Automaten
- Kontrollflussbeschreibung mit Label-Menge
- pause-Label werten zu *true* oder *false* aus

Kontrollfluss

Beispiel 1

```
   while x do
      y := y + 1;
      l1 : pause
   end
||
   suspend
      l2 : pause;
      emit a;
   when x;
   l3 : pause
```



Beispiel 2

```
   while x do
      y := y + 1;
      l1 : pause
   end
||
   suspend
      l2 : pause;
      emit a;
   when x;
   l3 : pause
```

Kontrollflusszustand: $\{l1, l2\}$

Notation

- Anweisungen: S, S_1, S_2
- *labels*(*S*) Menge aller Label in S
- $X\alpha$ bezeichnet Wert von α zum nächsten Zeitpunkt

Kontrollfluss

Definition über Prädikate:

- $in(S)$
- $stutter(S)$
- $inst(S)$
- $enter(S)$
- $move(S)$
- $term(S)$

in(S)

- $in(S) = true \Leftrightarrow$ Kontrollfluss ist in S

Definition

$$in(S) :\equiv \bigvee_{l \in labels(S)} l$$

Beispiel: Xin(l3 : pause)

```
l1 : pause;
l2 : pause;
l3 : pause
```

stutter(S)

- Kontrollflusszustand ändert sich nicht

Definition

$$stutter(S) :\equiv \bigwedge_{l \in labels(S)} (l = Xl)$$

Beispiel: stutter(S)

```
while true do
   l1 : pause
end
```

inst(S)

- *inst*antaneous = augenblicklich
- Ausführung erreicht kein pause

Beispiel: inst(S)$= \neg\alpha$

```
while α do
   l1 : pause;
end
```

enter(S)

- Kontrollfluss betritt Anweisung
- erstes pause-Statement als nächstes aktiv
- enter(S) ⇒ Xin(S)

Beispiel: enter(S)

```
l1 : pause;
l2 : pause;
```

term(S)

- Kontrollfluss verlässt Anweisung
- in(S) gilt
- kann S trotzdem wieder betreten

Beispiel: term(S)

```
l1 : pause;
l2 : pause;
```

move(S)

- Kontrollfluss innerhalb Anweisung
- in(S), Xin(S), ¬term(S)

Beispiel: move(S)

```
l1 : pause;
l2 : pause;
```

Finaler Kontrollfluss

- Startbedingung st
- Initialzustand $\mathcal{I}_{cf}(st, S) :\equiv \neg in(\mathrm{S})$
- Übergansrelation $\mathcal{R}_{cf}(st, S)$ (7 Fälle)

Kontrollflussfälle

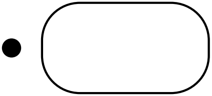


Abbildung: ¬st

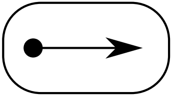


Abbildung: move(S)

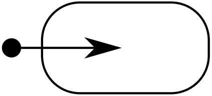


Abbildung: enter(S)

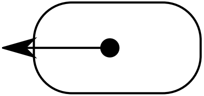


Abbildung: term(S)

Kontrollflussfälle

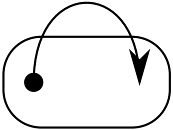


Abbildung: term(S) ∧ enter(S)

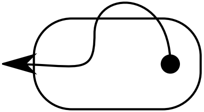


Abbildung: term(S) ∧ inst(S)

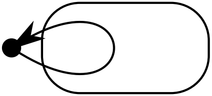


Abbildung: inst(S)

Freie Universität Berlin

Lemma

Für alle Quartz-Anweisungen gilt:

- $enter(S) \to Xin(S)$
- $enter(S) \to \neg inst(S)$
- $term(S) \to in(S)$
- $move(S) \to in(S) \land Xin(S)$
- $move(S) \to \neg term(S)$
- $stutter(S) \to in(S) = Xin(S)$
- $\neg in(S) \to stutter(S) = \neg Xin(S)$

Outline

Datenfluss

- Menge von guarded Commands
- (γ,C)
- C ∈ {emit x, emit delayed x, y := τ, y := delayed τ, now σ }
- γ → exec C ∨ γ → σ
- Berechnung: $gcmd(\varphi, S)$
- $gcmd(\varphi, \text{emit x}) :\equiv \{(\varphi, \text{emit x})\}$
- $gcmd(\varphi, S_1; S_2) :\equiv gcmd(\varphi, S_1) \cup gcmd(inst(S_1) \wedge \varphi \vee term(S_1), S_2)$
- φ enthält immer den aktuellen Kontrollfluss und die aktuelle Variablenbelegung

Event-Variablen

Definition für eine Event-Variable x

- Teilmenge von gcmd = {(α_1, emit x),...,(α_m, emit x),(β_1, emit delayed x),...,(β_n, emit delayed x)}
- Initialzustand: prüfe alle Bedingungen α_i
- $x = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m$
- Übergang: Wert von x zum Zeitpunkt t+1 gilt, wenn ein α_i zu t+1 gilt oder ein β_i zum Zeitpunkt t
- $\mathsf{X}x = \mathsf{X}\alpha_1 \vee \ldots \vee \mathsf{X}\alpha_m \vee \beta_1 \vee \cdots \vee \beta_n$

Event-Variablen

Definition

$$\mathcal{I}_{df}(st, x, S) :\equiv \left(x = \bigvee_{i=1}^{m} \alpha_i\right)$$

$$\mathcal{R}_{df}(st, x, S) :\equiv \left(\mathsf{X}x = \left(\bigvee_{i=1}^{n} \beta_i\right) \vee \mathsf{X}\left(\bigvee_{i=1}^{m} \alpha_i\right)\right)$$

Zustands-Variablen

Definition für eine Zustands-Variable y

- Teilmenge von gcmd = {(α_1, y := τ_1),...,(α_m, y := τ_m),(β_1, y := delayed π_1),...,(β_n, y := delayed π_n)}
- Initialzustand: prüfe alle Bedingungen α_i
- y = $(\alpha_1 \rightarrow y = \tau_1) \wedge \ldots \wedge (\alpha_m \rightarrow y = \tau_m)$
- Übergang: Wert von x zum Zeitpunkt t+1 ergibt sich, wenn ein α_i zu t+1 gilt oder ein β_i zum Zeitpunkt t
- $\mathrm{X}(\alpha_i \rightarrow y = \tau_i)$
- $\beta_i \rightarrow \mathrm{X}y = \pi_i$
- gilt keins davon $\rightarrow$ $\mathrm{X}y = y$

Zustands-Variablen

Definition

$$\mathcal{I}_{df}(st, y, S) :\equiv \bigwedge_{i=1}^{m} (\alpha_i \rightarrow [y = \tau_i])$$

$$\mathcal{R}_{df}(st, y, S) :\equiv \begin{pmatrix} (\bigwedge_{i=1}^{m} \mathsf{X}[\alpha_i \rightarrow (y = \tau_i)]) \land \\ (\bigwedge_{i=1}^{n} [\beta_i \rightarrow (\mathsf{X}y = \pi_i)]) \land \\ ([\bigwedge_{i=1}^{m} \neg \mathsf{X}\alpha_i \land \bigwedge_{i=1}^{n} \neg \beta_i] \rightarrow [\mathsf{X}y = y]) \end{pmatrix}$$

Probleme bei Zustands-Variablen

Es gibt zwei Probleme bei dieser Definition

Betrachte $\bigwedge_{i=1}^{m} \mathsf{X}[\alpha_i \rightarrow (y = \tau_i)]$

- S = y:=5||y:=3
- zu diesen Zeitpunkt muss gelten (y=5)$\wedge$(y=3)
- $\rightarrow$ Widerspruch, Schreibkonflikt
- Bedingungen für einen Schreibkonflikt können formuliert werden, Problem ist jedoch nicht entscheidbar

- kein α_i gilt zu Beginn; y ist undefiniert.
- mehrere Möglichkeiten
- 1. Wähle beliebigen Wert für y aus dessen Datentypbereich
- **2. Wähle einen festen Defaultwert, der für alle Datentypen gilt**

Alternative Definition von Zustands-Variablen

Definition

$$\mathcal{I}_{df}(st, y, S) \equiv \left(y = \left(\begin{array}{c} \text{if } \alpha_1 \text{ then } \tau_1 \\ . \\ . \\ \text{elsif } \alpha_m \text{ then } \tau_n \\ \text{else } \tau_0 \end{array} \right) \right)$$

$$\mathcal{R}_{df}(st, y, S) \equiv \left(\mathsf{X}y = \left(\begin{array}{c} \text{if } \mathsf{X}\alpha_1 \text{ then } \mathsf{X}\tau_1 \\ . \\ . \\ \text{elsif } \mathsf{X}\alpha_m \text{ then } \mathsf{X}\tau_m \\ \text{elsif } \beta_1 \text{ then } \pi_1 \\ . \\ . \\ \text{elsif } \beta_n \text{ then } \pi_n \\ \text{else } y \end{array} \right) \right)$$

Gesamter Datenfluss

Definiere den Datenfluss über alle Variablen durch Konjunktion

Definition

$$\mathcal{I}_{df}(st, S) :\equiv \bigwedge_{i=1}^{m} \mathcal{I}_{df}(st, y_i, S)$$
$$\mathcal{R}_{df}(st, S) :\equiv \bigwedge_{i=1}^{m} \mathcal{R}_{df}(st, y_i, S) \wedge \bigwedge_{i=1}^{p} (\varphi_i \rightarrow \sigma_i)$$

Vereinige Kontrollfluss und Datenfluss zur Gesamtsemantik

Definition

$$\mathcal{I}(st, S) :\equiv \mathcal{I}_{cf}(st, S) \wedge \mathcal{I}_{df}(st, S)$$
$$\mathcal{R}(st, S) :\equiv \mathcal{R}_{cf}(st, S) \wedge \mathcal{R}_{df}(st, S)$$

Ergebnisse

Ergebnisse, was kann man damit machen Beispielformel?

- Transformation eines Quartz Programmes in eine logische Formel
- Benutzbar in Theorem Beweisern wie HOL
- High level Implementierung → Low Level Implementierung
- Theorem Beweise über die Sprache selbst :
  $\forall P, Q : Quartz(\alpha).(P||Q) = (Q||P)$
- Beweis von Invarianten
- Induktion über Anzahl an Threads oder Datentypen

Quellen I

K. Schneider.
Embedding Imperative Synchronous Languages in Interactive Theorem Provers
2010 10th International Conference on Application of Concurrency to System Design, 143–154, 2001.

Thomas Melham.
Automating Recursive Type Definitions in Higher Order Logic.
Current Trends in Hardware Verification and Automated Theorem Proving, 341–386, 1988.

Zusammenfassung

- Einbetten von Quartz in HOL
- Aufteilung in Kontroll- und Datenfluss
- Kontrollfluss:
  - in, stutter, inst, enter, term, move
  - Zustandsübergangsrelation
- Datenfluss
  - Wächterkommandos
  - Schreibkonflikte