Freie Universität Berlin

SIGNAL

Seminar Programmiersprachen: synchrone Programmiersprachen

Florian Freudenberg

Dozentinnen: Prof. Dr. E. Fehr, Lilit Hakobyan

Freie Universität Berlin

Gliederung

1. Übersicht
2. Zeitannahmen
3. SIGNAL
   - Operatoren
   - Operationskomposition
   - Prozesse
4. Clock-Kalkül
5. Conditional-Dependency-Graph
6. Zusammenfassung
7. Hinweise

Übersicht

- synchrone Programmiersprache
- deklarativ
- data-flow-orientiert
- überschaubare Syntax
- Kalkül zur statische Analyse
- garantierte Termination
- polychron

Zeitannahmen

- Aussagen auf Basis einer *logischen Zeitebene*
- SIGNAL behandelt *nicht* die *physikalische Zeitebene*

Annahme: *Relative globale Ordnung* über alle Signale durch speziellen *Mechanismus* (nicht genau definiert), *Abstraktion* der nichtdeterministischen, asynchronen *Kommunikation mit Außenwelt*

- Betrachtung der temporalen Abfolge der Kommunikationsereignisse
- *Operationen* haben *Ausführungszeit* von null
- Ausführung erfolgt sobald alle Daten verfügbar sind

Zeitannahmen

- Ausführung nur abhängig von den deterministischen Abhängigkeiten der Operationen von den Daten
- Parallelität der Ausführung auf die direkten Abhängigkeiten beschränkt

Signal

- Basisobjekt der Sprache
- Besteht aus geordneter Sequenz von getypten Werten und einer Uhr
- Uhr definiert relative Zeitpunkte der Verfügbarkeit der Werte
- Signale x und y:

$$x_1, y_2, (x_3, y_3), x_4, \ldots$$

- Jedes Signal kann eine andere Uhr besitzen

SIGNAL - Operatoren

- Elementare Operatoren
- Delay-Operator
- Undersampling-Operator
- Event-Operator
- Merge-Operator
- Synchro-Operator

SIGNAL - Elementare Operatoren

- Arithmetische und logische Operationen
- Beispiel:

$$z := x + y$$

- Als Gleichung:

$$\forall t: z_t = x_t + y_t$$

- Zum Zeitpunkt t muss jedes Eingangssignal verfügbar sein.

  $\Longrightarrow$ alle Signale besitzen die gleiche Uhr

SIGNAL - Delay-Operator

Notation:

$$< signale > := < signal > \ \$k$$

Beispiel:

$$y \ := x \ \$1$$

Ausführung:

x:	2	5	1	0	4	1	3	7
y:	0	2	5	1	0	4	1	3

Uhren:

Uhr von Signal x = Uhr von Signal y

SIGNAL - Undersampling-Operator

Notation:

$$< signal >:=< signal > \ when < signal >$$

Beispiel:

$$y \ := x \ when \ b$$

Ausführung:

x:	2	5	⊥	0	4	⊥	3	7
b:	t	f	f	f	⊥	t	f	t
y:	2	⊥	⊥	⊥	⊥	⊥	⊥	7

Uhren:

Uhr von Signal y <= Uhr von Signal x
<= Uhr von booleschen Signal b

SIGNAL - Event-Operator

Notation:

$$< signal >:= event < signal >$$

Beispiel:

$$y := event\ x$$

Ausführung:

x:	2	⊥	1	⊥	⊥	1	3	7
y:	t	⊥	t	⊥	⊥	t	t	t

Uhren:

Uhr von Signal y = Uhr von Signal x

SIGNAL - Merge-Operator

Notation:

$$< signal >:=< signal > default < signal >$$

Beispiel:

$$y := x \; default \; z$$

Ausführung:

x:	2	⊥	⊥	⊥	4	⊥	⊥	7
z:	1	4	⊥	5	⊥	2	4	5
y:	2	4	⊥	5	4	2	4	7

Uhren:

Uhr von Signal y >= Uhr von Signal x

>= Uhr von Signal z

SIGNAL - Synchro-Operator

Notation:

$$synchro < signal >, < signal >$$

Beispiel:

$$synchro\ x, y$$

Uhren:

Uhr von Signal x = Uhr von Signal y

$$\Leftrightarrow$$

Explizite Synchronisation

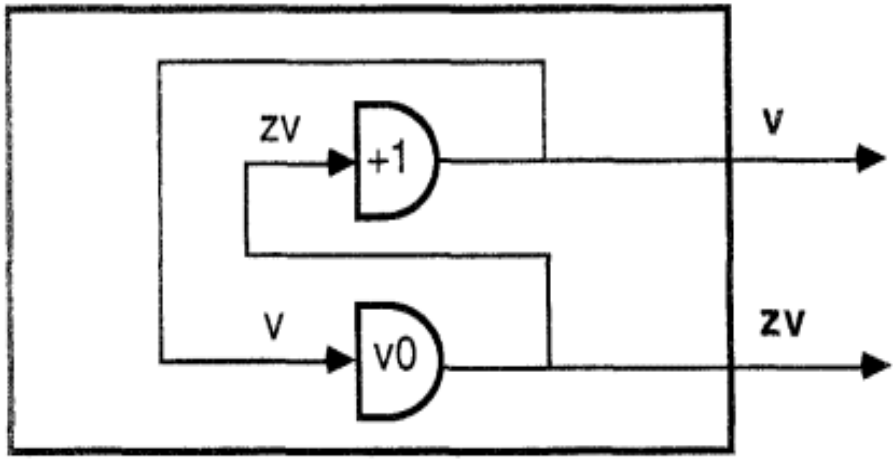
SIGNAL - Komposition

Notation:

$$(|P1| \ldots |Pn|)$$

Beispiel:

$$(|v := zv + 1|zv := v\ \$1|\ )$$



Quelle: [1]

SIGNAL - Komposition

Cell-Operator:

$$y \coloneqq x \; cell \; b$$

Definition:

$$(|y \coloneqq x \; default \; (y \; \$1) | synchro \; a, y \\ |a \coloneqq (event \; x) \; default \; (when \; b))$$

Ausführung:

x:	2	⊥	⊥	⊥	4	⊥	⊥	7
b:	t	f	⊥	t	t	t	t	⊥
y:	2	⊥	⊥	2	4	4	4	7

SIGNAL - Prozesse

Notation:

$$Name\,\{\,Konstante\,?\;Eingabeports\,!\;Ausgabeports\,\}\;=\;P\,/\,a_1\,,\ldots,a_n$$

```
topmod { integer v0 ? logical hreset,iev ! integer v; logical oev}
        = (         |synchro iev,zv
                    |zv := v $1
                    |v := (0 when hreset) default (zv+1)
                    |oev := true when zv>=v0|)/zv
          where integer zv init v
end
```

Aufruf:

```
topmod(n-1) ? iev:signal ! oev:hreset @ hreset
```

Clock-Kalkül

Idee:

Abbildung der Deklarationen auf Gleichungen, die Aussagen zu den jeweiligen Uhren der Signale treffen.

Behandlung von booleschen Signalen durch Undersampling-Operator notwendig!

Allgemein bei Signalen von Interesse:

nicht verfügbar, verfügbar

Bei booleschen Signalen:

nicht verfügbar, true, false

Abbildung mit Gleichungen in $\mathbb{Z}/3\mathbb{Z}$

Clock-Kalkül

Restklassenring: $\mathbb{Z}/3\mathbb{Z}$

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

x	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Polynome in $\mathbb{Z}/3\mathbb{Z}$ haben maximal Grad 2, da:

$$x^3 = x$$

Clock-Kalkül

Abbildung eines booleschen Signals auf $\mathbb{Z}/3\mathbb{Z}$:

- 0 = nicht verfügbar
- 1 = true
- 2 = false

Abbildung eines Signals auf $\mathbb{Z}/3\mathbb{Z}$:

- 0 = nicht verfügbar
- 1 = verfügbar

Nicht für alle Operationen ist die Differenzierung notwendig. Abbildung:

$$x \to x^2$$

konvertiert das boolesche Signale.

Clock-Kalkül

expressions	boolean result (values)	non boolean result (clocks)
y := *not* x	$y = -x$	$y^2 = x^2$
y := a *or* b	$y = ab\,(1 - (a + b + ab))$ $a^2 = b^2$	$y^2 = a^2 = b^2$
y := a *and* b	$y = ab\,(ab - (a + b + 1))$ $a^2 = b^2$	$y^2 = a^2 = b^2$
y := f(a_1, ..., a_n)	$y^2 = a_1{}^2 = \ldots = a_n{}^2$	$y^2 = a_1{}^2 = \ldots = a_n{}^2$
y := x $1	$y^2 = x^2$	$y^2 = x^2$
synchro a_1, ..., a_n	$a_1{}^2 = \ldots = a_n{}^2$	$a_1{}^2 = \ldots = a_n{}^2$
c := *event* a	$c = a^2$	
y := a *when* c	$y = a\,(-c - c^2)$	$y^2 = a^2\,(-c - c^2)$
y := a *default* b	$y = a + b - a^2b$	$y^2 = a^2 + b^2 - a^2b^2$

Quelle: [1]

Clock-Kalkül

Verfahren:

Gleichungssystem für jede Variable lösen, keine Signal-Variable darf zwingend den Wert null haben. Wenn ein Signal zwingend null ist, ist eine Berechnung nicht möglich.

Beispiel:

$$\big(|x \coloneqq a\ when\ (a > 0)|y \coloneqq a\ when\ \big(not(a > 0)\big)|z \coloneqq x + y|\big)$$

Gleichungen:

$$x^2 = a^2(-c - c^2)$$
$$y^2 = a^2(-(-c) - (-c)^2) = a^2(c - c^2)$$
$$z^2 = x^2 = y^2$$

Auflösen:

$$a^2(-c - c^2) = a^2(c - c^2)$$
$$-c - c^2 = c - c^2$$
$$-c = c \qquad \Rightarrow c = 0$$

Conditional-Dependency-Graph

- Temporale Korrektheit ist jedoch nicht ausreichen da keine Abhängigkeiten der Signale voneinander beachtet wird
- Conditional-Dependency-Graph
  - Knoten: Uhren und Signale
  - Kanten: Gleichungen des Clock-Kalküls
- Kreis können blockieren
- Kriterium: Produkt alle Kantengleichungen = 0

Bei Verstoß gibt es eine zyklische Abhängigkeit und es können nicht alle Signale gleichzeitig verfügbar sein.

Conditional-Dependency-Graph

Beispiel:

$$(|x \coloneqq \sin\{y\} + b | y \coloneqq a\ default\ x|)$$

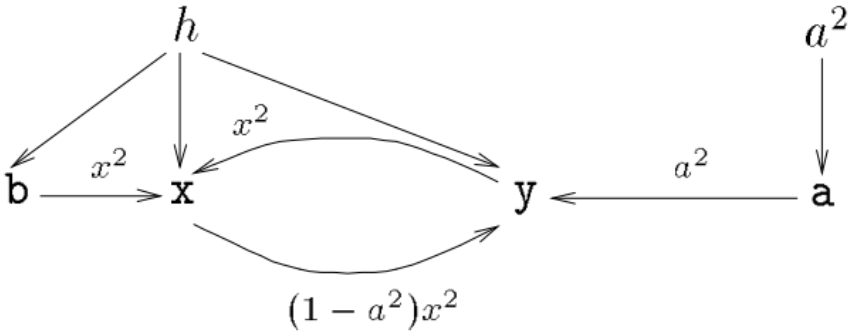
Clock-Kalkül:

$$x^2 = y^2 = b^2$$
$$x^2 = a^2 + b^2 + a^2 b^2$$

Uhr:

$$h = x^2 = b^2 = y^2 = a^2 + (1 - a^2)b^2$$

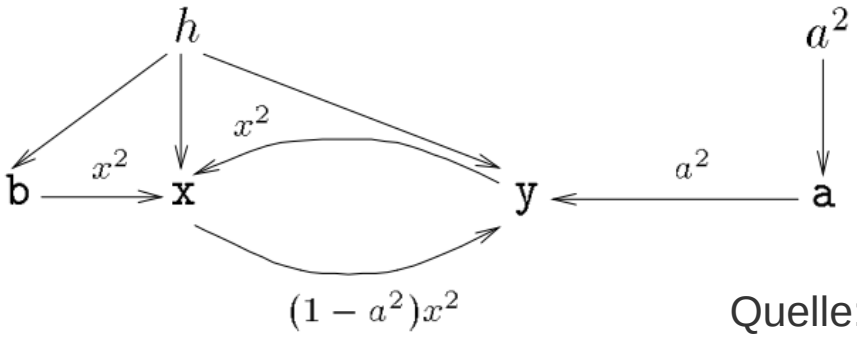
Graph:



Quelle: [3]

Conditional-Dependency-Graph

Graph:



Quelle: [3]

Produkt: $(1 - a^2)x^2 \neq 0$

nur berechenbar wenn Signal a verfügbar

Entspricht:

$$(|y \coloneqq a|x \coloneqq \sin\{a\} + b|)$$

Zusammenfassung

- Logische Zeit
- Sechse Basisoperationen
- Komposition von Operationen
- Konstruktion und Verwendung von Prozessen
- Clock-Kalkül und Conditional-Dependency-Graphen als statische Analysewerkezeuge

Hinweise

Stand des neusten Papers was betrachtet wurde: 1991

Es gibt eine Toolbox mit Compiler für Signal namens Polychrony:
http://www.irisa.fr/espresso/Polychrony/

(Enthält einige komplexe Beispiele)

Formale Definition für SIGNAL und das Clock-Kalkül ist in [2] zu finden.

Quellen

[1] Gautier, T.; Le Guernic, P.; Besnard, L.: SIGNAL: A declarative language for synchronous programming of real-time systems; In Functional programming languages and computer architecture; Springer Berlin/Heidelberg, 1987; S. 257-277.

[2] Le Guernic, P.; Benveniste, A.: Real-Time, Synchronous, Data-Flow Programming: the Language SIGNAL and its Mathematical Semantics; In INRIA, Rennes, Research Report n° 533, 1986.

[3] Le Guernic, P; Gautier, T.; Le Borgne, M.; Le Maire, C.: Programming real time applications with SIGNAL; In Proceedings of the IEEE 79.9, 1991; S. 1321-1336.

Freie Universität Berlin

DANKE