Programmierung und Verifikation reaktiver Systeme mittels funktionaler Programmierung

Wolfgang Jeltsch

Brandenburgische Technische Universität Cottbus
Institut für Informatik sowie Informations- und Medientechnik

Institutskolloquium am
18. November 2009

Überblick

Überblick

Funktionale Reaktive Programmierung

- Idealvorstellung von einem reaktiven System:
  - kontinuierliche Zustandsänderungen
  - sofortige, atomare Reaktion auf Ereignisse
- Abweichungen vom Ideal bei herkömmlicher Programmierung reaktiver Systeme:
  - Diskretisierung sichtbar
  - inkonsistente Zwischenzustände sichtbar
  - zyklische Event-Kaskaden
- Programmierer mit technischen Details konfrontiert:
  - Polling-Schleifen
  - Event-Handler
- Ziel der funktionale Programmierung: Problembeschreibung statt Ablaufplan
- Funktionale Reaktive Programmierung (FRP): Anwendung dieses Prinzips auf reaktive Systeme
- in diesem Vortrag: Haskell-Bibliothek Grapefruit

Signale

- das Herzstück der FRP
- beschreiben zeitliches Verhalten
- drei Arten:

  DSignal Werte, die mit diskreten Zeitpunkten assoziiert sind (diskret)

  CSignal zeitveränderliche Werte (kontinuierlich)

  SSignal zeitveränderliche Werte, die sich nur zu diskreten Zeitpunkten ändern (segmentiert)
- Beispiele für Signale und deren Typen:
  - eingehende Netzwerkpakete (*DSignal Packet*)
  - bisheriges Datenvolumen (*SSignal Int*)
  - Zeit seit dem Programmstart (*CSignal DiffTime*)

Signalkombinatoren (1)

- einige Kombinatoren:

$$\begin{aligned} \mathit{union} &:: \mathit{DSignal}\ \mathit{val} \to \mathit{DSignal}\ \mathit{val} \to \mathit{DSignal}\ \mathit{val} \\ \mathit{filter} &:: (\mathit{val} \to \mathit{Bool}) \to \\ &\quad (\mathit{DSignal}\ \mathit{val} \to \mathit{DSignal}\ \mathit{val}) \\ \mathit{scan} &:: \mathit{accu} \to (\mathit{accu} \to \mathit{val} \to \mathit{accu}) \to \\ &\quad (\mathit{DSignal}\ \mathit{val} \to \mathit{SSignal}\ \mathit{accu}) \end{aligned}$$

- Anwendung der Kombinatoren:

$$\begin{aligned} &\mathit{packets} :: \mathit{DSignal}\ \mathit{Packet} \\ &\mathit{packets} = \mathit{union}\ \mathit{inPackets}\ \mathit{outPackets} \\ &\mathit{tcpPackets} :: \mathit{DSignal}\ \mathit{Packet} \to \mathit{DSignal}\ \mathit{Packet} \\ &\mathit{tcpPackets}\ \mathit{packets} = \mathit{filter}\ \mathit{isTCPPacket}\ \mathit{packets} \\ &\mathit{amounts} :: \mathit{DSignal}\ \mathit{Packet} \to \mathit{SSignal}\ \mathit{Int} \\ &\mathit{amounts}\ \mathit{packets} = \mathit{scan}\ 0\ \mathit{next}\ \mathit{packets}\ \textbf{where} \\ &\quad \mathit{next}\ \mathit{amount}\ \mathit{packet} = \mathit{amount} + \mathit{size}\ \mathit{packet} \end{aligned}$$

Signalkombinatoren (2)

- Umschalten zwischen verschiedenen Signalen:

$$\begin{aligned} &\mathit{switch} :: (\mathit{Signal}\ \mathit{signal}) \Rightarrow \\ &\qquad \mathit{SSignal}\ (\mathit{signal}\ \mathit{val}) \rightarrow \mathit{signal}\ \mathit{val} \end{aligned}$$

instance *Signal DSignal* **where** ...

instance *Signal SSignal* **where** ...

instance *Signal CSignal* **where** ...

- Anwendung: je nach Wahl des Benutzers Anzeige des Datenvolumens eingegangener oder ausgegangener Pakete

Überblick

Überblick

Bedarfsgesteuerte Ausführung

- ereignisgetriebene Zustandsänderungen
- Signalkonsumenten registrieren Event-Handler
- typische Implementierung von Signalen:

  DSignal v Signal ist Registrierungsaktion:

  $$(v \to \mathit{IO}\ ()) \to \mathit{IO}\ (\mathit{IO}\ ())$$

  SSignal v Signal ist Initialwert plus Aktualisierungssignal:

  $$(v, \mathit{DSignal}\ v)$$

Keine Signale, sondern Generatoren

- Registrierungsaktionen werden einmal pro Konsument ausgeführt
- Folge bei Benutzung von *scan*:
  - jeder Konsument erzeugt veränderliche Variable, die akkumulierten Wert speichert
  - jeder Konsument registriert Event-Handler, der diese Variable aktualisiert
- zwei Probleme:
  1. Mehrfachberechnungen
  2. Signalwerte können vom Zeitpunkt des Konsumierens abhängen
- Erklärung: keine Signale, sondern Signalgeneratoren

Überblick

Nutzung nativer Memoization

- berechnete Werte werden gespeichert, wenn Datenstruktur an Variable gebunden ist
- Problem: Datenstruktur diskreter Signale enthält die Signalwerte nicht
- Änderung der Datenstruktur:

$$([(\mathit{Time}, v)], \ldots)$$

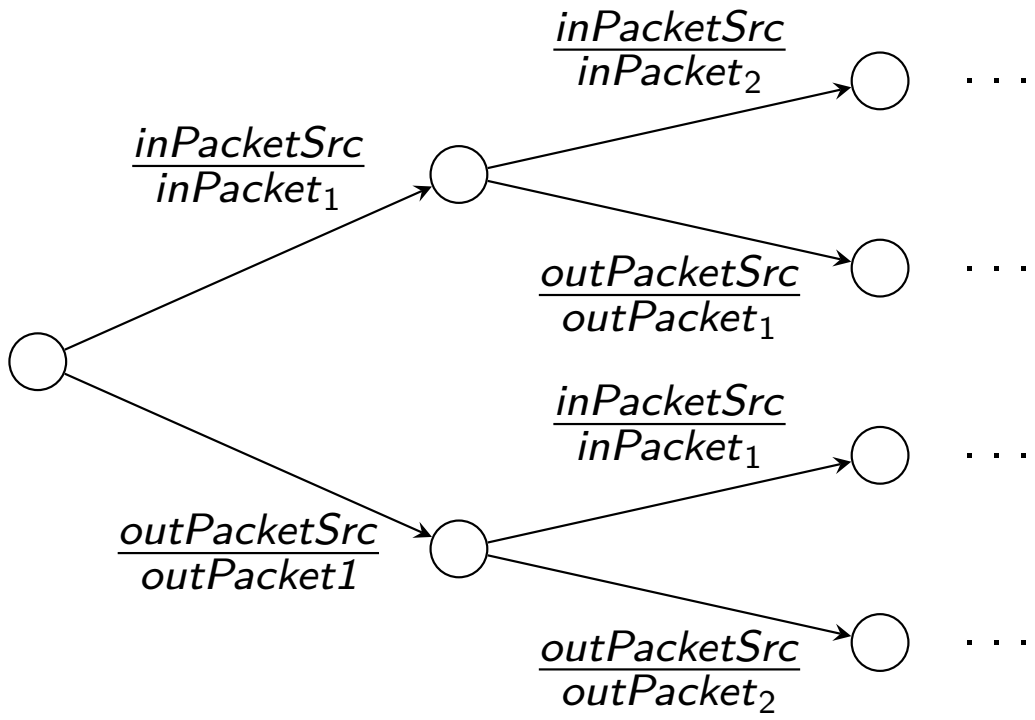
- Vereiningung von Listen von Zeit-Wert-Paaren:

$$\begin{aligned} &\mathit{occsUnion}\ ((\mathit{time}_1, \mathit{val}_1) : \mathit{occs}_1) \\ &\qquad ((\mathit{time}_2, \mathit{val}_2) : \mathit{occs}_2) = \mathit{occs}' \ \textbf{where} \\ &\mathit{occs}' = \textbf{case}\ \mathit{compare}\ \mathit{time}_1\ \mathit{time}_2\ \textbf{of}\ \ldots \end{aligned}$$

- Problem: Vergleich von Zeiten geschieht zu zeitig
- Lösung: Entscheidung vertagen

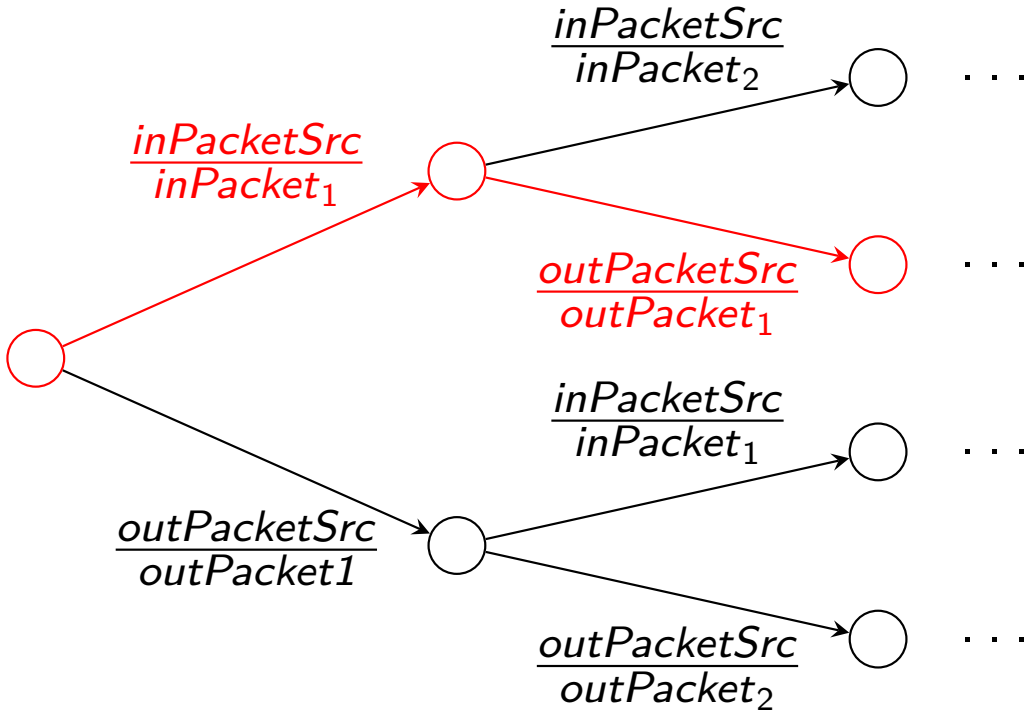
Vistas

- Vista enthält für jeden Zeitpunkt mehrere mögliche „Zukünfte“ in Abhängigkeit davon, welche Ereignisquelle als nächstes feuert
- Vista für *union inPackets outPackets*:



Konsumieren von Vistas

- Konsument wertet nur relevanten Pfad aus:



Überblick

Fixieren von Startzeiten (1)

- Signaltyp erhält einen zusätzlichen (Phantom-)Typparameter, der die Lebenszeit („Ära“) des Signals repräsentiert
- Signalkombinatoren erzwingen Gleichheit von Ären:

$$\begin{aligned} \mathit{union} &:: \mathit{DSignal\ era\ val} \to \\ &\quad \mathit{DSignal\ era\ val} \to \\ &\quad \mathit{DSignal\ era\ val} \\ \mathit{scan} &:: \mathit{accu} \to (\mathit{accu} \to \mathit{val} \to \mathit{accu}) \to \\ &\quad (\mathit{DSignal\ era\ val} \to \mathit{SSignal\ era\ accu}) \end{aligned}$$

Fixieren von Startzeiten (2)

- reaktive Aktionen mit Ära-Parametern:

$$\textbf{newtype}\ \mathit{Reactive}\ \mathit{era}\ \mathit{val} = \mathit{Reactive}\ (\mathit{IO}\ \mathit{val})$$

- Signalproduktion und -konsumierung erzwingt Gleichheit von Signalära mit der Ära der reaktiven Aktion
- Ausführung von reaktiven Aktionen nur bei Äraunabhängigkeit möglich:

$$\mathit{toIO} :: (\forall \mathit{era}.\mathit{Reactive}\ \mathit{era}\ \mathit{val}) \to \mathit{IO}\ \mathit{val}$$

Sicheres Umschalten zwischen Signalen

- sicherer Kombinator:

$$\begin{aligned} &\mathit{switch} :: (\mathit{Signal}\ \mathit{signal}) \Rightarrow \\ &\qquad \mathit{SSignal}\ \mathit{era}\ (\forall \mathit{era}'.\mathit{signal}\ \mathit{era}'\ \mathit{val}) \rightarrow \\ &\qquad \mathit{signal}\ \mathit{era}\ \mathit{val} \end{aligned}$$

- schaltet nur zu Signalen um, die nicht von externen Werten abhängen:
  - diskrete Signale müssen leer sein
  - segmentierte Signale müssen konstant sein
- Folge: nicht sehr sinnvoll
- Idee: Umschalten zwischen Signalfunktionen statt Signalen

Umschalten zwischen Signalfunktionen

- Funktionen über Signalen mit identischer Ära:

$$\begin{aligned} &\mathit{SignalFun}\ \mathit{era}\ (\mathit{signal}_1\ \text{`}\mathit{Of}\text{`}\ \mathit{val}_1 \mapsto \\ &\quad \ldots \qquad\qquad\qquad\quad \mapsto \\ &\quad \mathit{signal}_n\ \text{`}\mathit{Of}\text{`}\ \mathit{val}_n \mapsto \\ &\quad \mathit{signal}'\ \text{`}\mathit{Of}\text{`}\ \mathit{val}') \end{aligned}$$

- Kombinatortyp:

$$\begin{aligned} &\mathit{SSignal}\ \mathit{era}\ (\forall \mathit{era}'.\mathit{SignalFun}\ \mathit{era}'\ \mathit{shape}) \rightarrow \\ &\mathit{SignalFun}\ \mathit{era}\ \mathit{shape} \end{aligned}$$

- Arbeitsweise des Kombinators:
  - Argumente der Resultatfunktion werden auf die Segmentintervalle „zurechtgestutzt“
  - jede Funktion des Argumentsignals wird auf entsprechend gestutzte Argumente angewendet
  - Resultate werden aneinander gehängt

Überblick

Überblick

Intuitionistische Logik

- zwei grundsätzliche Arten von Logik:

  klassisch Wahrheit

  konstruktiv Konstruierbarkeit

- intuitionistische Logik ist spezielle Form von konstruktiver Logik
- wichtige Eigenschaften intuitionistischer Logik:
  - Aussagen gelten nur, wenn sie beweisbar sind
  - Existenz kann nur durch Konstruktion bewiesen werden
  - Disjunktionsbeweis enthält Angabe einer gültigen Alternative
- Folge: Satz vom ausgeschlossenen Dritten ($P \vee \neg P$) gilt nicht

Rolle der intuitionistischen Logik

- Grundlagenstreit der Mathematik in den 1920er Jahren mit „Sieg“ für den klassischen Ansatz
- intuitionistische Logik aber gerade für Informatik sehr interessant:
  - Typsysteme
  - computergestütztes Beweisen
- Gründe für die Beschäftigung mit intuitionistischer Logik:
  - Ansicht, dass intuitionistische Logik die „richtige Logik“ und klassische Logik die „falsche Logik“ ist (G. Sambin)
  - theoretisches Interesse
  - Anwendbarkeit intuitionistischer Logik

Brouwer-Heyting-Kolmogorov-Interpretation

- Struktur eines Beweises:

  $P_1 \wedge P_2$ ein Paar (p_1, p_2), wobei gilt:
  - p_1 ist Beweis von P_1
  - p_2 ist Beweis von P_2

  $P_1 \vee P_2$ ein Paar (i, p), wobei gilt:
  - $i \in \{1, 2\}$
  - p ist Beweis von P_i

  $P_1 \to P_2$ eine berechenbare Funktion f, wobei gilt:
  - für jeden Beweis p_1 von P_1 ist $f(p_1)$ Beweis von P_2

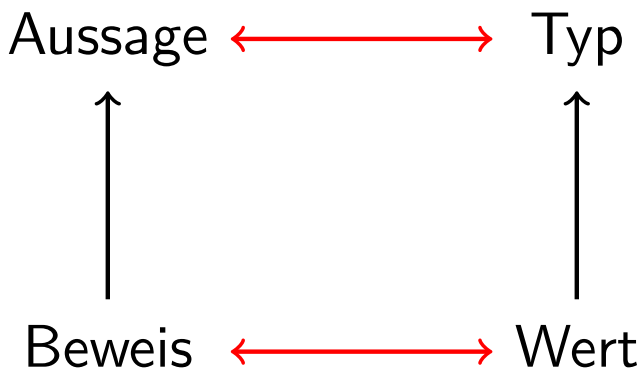
  $\top$ ein beliebiger, aber fester Wert
- kein Beweis für $\bot$
- Definitionen:

$$\neg P := P \to \bot$$

$$P_1 \leftrightarrow P_2 := (P_1 \to P_2) \wedge (P_2 \to P_1)$$

Curry-Howard-Beziehung

- logische Aussage entspricht dem Typ ihrer Beweise:



- Typ $\langle P \rangle$ zu einer Aussage P:

$$\langle P_1 \wedge P_2 \rangle = \langle P_1 \rangle \times \langle P_2 \rangle$$
$$\langle P_1 \vee P_2 \rangle = \langle P_1 \rangle + \langle P_2 \rangle$$
$$\langle P_1 \to P_2 \rangle = \langle P_1 \rangle \to \langle P_2 \rangle$$
$$\langle \top \rangle = 1$$
$$\langle \bot \rangle = 0$$

Abhängige Typen

- Typen können von Werten abhängen
- Beispiel: $\mathit{List}\ \eta\ \ell$ als Typ aller Listen der Länge ℓ mit Elementen vom Typ η
- verallgemeinerte Produkte und Summen:

  $\Pi\alpha : \tau_1.\tau_2$ Funktionen, die bei Anwendung auf ein υ vom Typ τ_1 ein Resultat vom Typ $\tau_2[\upsilon/\alpha]$ liefern

  $\Sigma\alpha : \tau_1.\tau_2$ Paare aus einem υ vom Typ τ_1 und einem Wert vom Typ $\tau_2[\upsilon/\alpha]$
- Beispiele:

  Typ einer Funktion zur Konstruktion von Nullvektoren

  $\Pi \mathit{dim} : \mathit{Nat}.\mathit{List}\ \mathit{Real}\ \mathit{dim}$

  Typ aller Zeichenketten

  $\Sigma \mathit{len} : \mathit{Nat}.\mathit{List}\ \mathit{Char}\ \mathit{len}$
- Curry-Howard-Beziehung zu Prädikatenlogik

Intuitionistische Prädikatenlogik

- Prädikatenlogik:
  - Aussagen können sich auf Werte (Individuen) beziehen
  - All- und Existenzquantifizierung
- Brouwer-Heyting-Kolmogorov-Interpretation für Aussagen mit Quantoren:

  $\forall \alpha : \tau . P$ eine berechenbare Funktion f, wobei gilt:
  - für jedes v vom Typ τ ist $f(v)$ Beweis von $P[v/\alpha]$

  $\exists \alpha : \tau . P$ ein Paar (v, p), wobei gilt:
  - v ist vom Typ τ
  - p ist Beweis von $P[v/\alpha]$
- Curry-Howard-Beziehung:

$$\langle \forall \alpha : \tau . P \rangle = \Pi \alpha : \tau . \langle P \rangle$$

$$\langle \exists \alpha : \tau . P \rangle = \Sigma \alpha : \tau . \langle P \rangle$$

Verifizierte Software mit abhängigen Typen

- Typ einer Sortierfunktion (ohne Längenangaben):

$$(el \to el \to Bool) \to List\ el \to List\ el$$

- Darstellung von totalen Ordnungen:
  - Aussage „ξ ist totale Ordnung über Elementtyp η“ dargestellt als Typ $IsTotalOrder\ \eta\ \xi$
  - Werte von $IsTotalOrder\ \eta\ \xi$ sind Beweise dieser Aussage
  - Typ aller totalen Ordnungen:

$$TotalOrder\ el = \Sigma comp\ : el \to el \to Bool.\ IsTotalOrder\ el\ comp$$

- in analoger Weise Typ $SortedList\ \eta\ \omega$ aller gemäß Ordnung ω sortierten Listen
- Typ einer Sortierfunktion mit integrierten Vor- und Nachbedingungen:

$$\Pi order : TotalOrder\ el.List\ el \to SortedList\ el\ order$$

Überblick

Temporallogik

- klassische Temporallogik:
  - Wahrheit von Aussagen hängt von betrachtetem Zeitpunkt ab
  - Aussagen über die Zukunft mittels modaler Operatoren:
    - $\Box P$ P ist ab jetzt zu jedem Zeitpunkt wahr
    - $\Diamond P$ P ist jetzt oder zu einem zukünftigen Zeitpunkt wahr
- intuitionistische Temporallogik:
  - Beweismenge einer Aussage hängt vom betrachteten Zeitpunkt ab
  - Beweise von Aussagen mit modalen Operatoren:
    - $\Box P$ enthält für den gegenwärtigen und jeden zukünftigen Zeitpunkt einen Beweis für P
    - $\Diamond P$ enthält für den gegenwärtigen oder einen zukünftigen Zeitpunkt einen Beweis für P
- Curry-Howard-Beziehung für intuitionistische Temporallogik?

Curry-Howard-Beziehung für Temporallogik

- „temporales Typsystem“:
  - Wertemenge eines Typs hängt von betrachtetem Zeitpunkt ab
  - Werte von Typen mit „modalen Typkonstruktoren“:
    - $\Box\tau$ enthält für den gegenwärtigen und jeden zukünftigen Zeitpunkt einen Wert von τ
    - $\Diamond\tau$ enthält für den gegenwärtigen oder einen zukünftigen Zeitpunkt einen Wert von τ

Curry-Howard-Beziehung für Temporallogik

- „temporales Typsystem“:
  - Wertemenge eines Typs hängt von betrachtetem Zeitpunkt ab
  - Werte von Typen mit „modalen Typkonstruktoren“:
    - $\Box\tau$ enthält für den gegenwärtigen und jeden zukünftigen Zeitpunkt einen Wert von τ
    - $\Diamond\tau$ enthält für den gegenwärtigen oder einen zukünftigen Zeitpunkt einen Wert von τ
- Überraschung: Das ist FRP!
- „modale Typkonstruktoren“:
  - $\Box$ *CSignal*
  - $\Diamond$ *Event*

    (diskrete Signale mit genau einem Wert)
- *DSignal* und *SSignal* aus *Event* erzeugbar
- gegenseitige „Befruchtung“ zwischen Temporallogik und FRP

Von FRP zu Temporallogik (1)

- kategorientheoretische Strukturen in FRP:

  CSignal ist Komonade

  Event ist Monade

- übertragen sich auf modale Operatoren der Temporallogik:

  $\Box$ ist Komonade

  $\Diamond$ ist Monade

- führt zu Ableitungsregeln für modale Operatoren, die Dualität widerspiegeln:

$$\frac{P_2 \vdash P_1}{\Box P_2 \vdash \Box P_1} \qquad \frac{}{\Box P \vdash P} \qquad \frac{}{\Box P \vdash \Box\Box P}$$

$$\frac{P_1 \vdash P_2}{\Diamond P_1 \vdash \Diamond P_2} \qquad \frac{}{P \vdash \Diamond P} \qquad \frac{}{\Diamond\Diamond P \vdash \Diamond P}$$

Von FRP zu Temporallogik (2)

- kein Wissen über zukünftige Werte in FRP
- analog kein Wissen über zukünftige Beweise in intuitionistischer Temporallogik
- Folge: zeitabhängiges Wissen kann ausgedrückt werden
- folgende klassische Regeln gelten nicht in intuitionistischer Logik:

$$\frac{}{\Diamond(P_1 \vee P_2) \vdash \Diamond P_1 \vee \Diamond P_2} \qquad \frac{}{\Diamond \bot \vdash \bot}$$

- für Modallogik bereits bekannt

Von Temporallogik zu FRP (1)

- „Until“-Operatoren in Temporallogik:

  $P_1 \triangleright P_2$ P_1 gilt ab jetzt zu jedem Zeitpunkt bis unmittelbar vor einem Zeitpunkt, an dem P_2 gilt

  $P_1 \blacktriangleright P_2$ wie $P_1 \triangleright P_2$ oder P_1 gilt für immer

- $\Box$ und $\Diamond$ können auf $\triangleright$ und $\blacktriangleright$ zurückgeführt werden:

$$\Box P = P \blacktriangleright \bot$$

$$\Diamond P = \top \triangleright P$$

Von Temporallogik zu FRP (2)

- „Until“-Operatoren der Temporallogik führen zu „Until“-Typkonstruktoren für FRP:

  $\tau_1 \triangleright \tau_2$ zeitlich begrenzte kontinuierliche Signale über τ_1, jeweils gefolgt von einem einzelnen Wert vom Typ τ_2

  $\tau_1 \blacktriangleright \tau_2$ alle Werte vom Typ $\tau_1 \triangleright \tau_2$ sowie alle kontinuierlichen Signale über τ_1

- Anwendungsbeispiele:
  - Signale mit kontinuierlichen und diskreten Anteilen
  - endlich lange kontinuierliche Signale
  - kontinuierliche Signale, deren Wertetyp sich zu diskreten Zeitpunkten ändert

Weitere interessante Punkte

- Zeitabhängigkeit von „temporalen Typen“ vs. Zeitabhängigkeit durch Ära-Parameter
- kategorientheoretische Fundierung von kompletter Temporallogik
- Definition intuitionistischer Temporallogik auf Basis eines „Next“-Operators und Induktion/Koinduktion:
  - kategorientheoretische Struktur hinter dem „Next“-Operator
  - Zeitabhängigkeit des Wissens folgt automatisch
- zeitliches „Verschieben“ von Werten nur explizit, dadurch eventuell Möglichkeiten zur statischen Ressourcenbeschränkung

Überblick

Grapefruit-Themen für Abschlussarbeiten

- Model-View-Unterstützung
- Unterstützung unterschiedlicher GUI-Backends
- Unterstützung grafischer Animationen
- Echtzeitfähigkeit
- inkrementelle und bidirektionale Programmierung

Programmierung und Verifikation reaktiver Systeme mittels funktionaler Programmierung

Wolfgang Jeltsch

Brandenburgische Technische Universität Cottbus
Institut für Informatik sowie Informations- und Medientechnik

Institutskolloquium am
18. November 2009

Äquivalenzen für abhängige Produkte und Summen

- Äquivalenzen:

$$\Pi i : \{1, 2\}.\tau \cong \tau[1/i] \times \tau[2/i]$$
$$\Sigma i : \{1, 2\}.\tau \cong \tau[1/i] + \tau[2/i]$$
$$\Pi \alpha : \tau_1.\tau_2 \cong \tau_1 \to \tau_2$$
$$\Sigma \alpha : \tau_1.\tau_2 \cong \tau_1 \times \tau_2 \ ,$$

wobei $\alpha \notin \mathrm{FV}(\tau_2)$