

# Using Logic for Programming Reactive Systems

Wolfgang Jeltsch

TTÜ Küberneetika Instituut

TTÜ Küberneetika Institudi sügisseminar

November 13, 2011

- 1 Logic and programming
- 2 The Temporal Curry–Howard Correspondence
- 3 Conclusions and outlook

# Propositional logic

- basic operators:

$\alpha \wedge \beta$   $\alpha$  and  $\beta$

$\alpha \vee \beta$   $\alpha$  or  $\beta$

$\alpha \rightarrow \beta$  if  $\alpha$  then  $\beta$

$\perp$  false

$\top$  true

- derived operators:

$\alpha \leftrightarrow \beta$   $\alpha$  if and only if  $\beta$

$$\alpha \leftrightarrow \beta := (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

$\neg\alpha$  not  $\alpha$

$$\neg\alpha := \alpha \rightarrow \perp$$

# Simple types

- operators:

$\alpha \times \beta$  type of pairs that consist of an  $\alpha$ -value  
and a  $\beta$ -value

$\alpha + \beta$  type of variants that are either an  $\alpha$ -value  
or a  $\beta$ -value

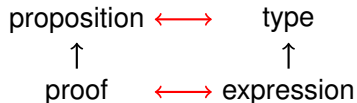
$\alpha \rightarrow \beta$  type of functions from  $\alpha$  to  $\beta$

0 empty type

1 singleton type

# Curry–Howard correspondence

- correspondence between logic and type system:



- well-known correspondences:

- propositional logic  $\longleftrightarrow$  simple types:

$$\langle \alpha \wedge \beta \rangle = \langle \alpha \rangle \times \langle \beta \rangle$$

$$\langle \alpha \vee \beta \rangle = \langle \alpha \rangle + \langle \beta \rangle$$

$$\langle \alpha \rightarrow \beta \rangle = \langle \alpha \rangle \rightarrow \langle \beta \rangle$$

$$\langle \perp \rangle = 0$$

$$\langle \top \rangle = 1$$

- predicate logic  $\longleftrightarrow$  dependent types:
  - almost any specification expressible as a type
  - specifications checked by the compiler

# Temporal logic

- lies between propositional and predicate logic:
  - not as expressible as predicate logic
  - easier to handle than predicate logic
- trueness of a proposition depends on the time
- all operators from propositional logic plus a few new ones:
  - $\Box\alpha$   $\alpha$  will always hold
  - $\Diamond\alpha$   $\alpha$  will eventually hold
  - $\alpha \triangleright \beta$   $\alpha$  will hold for some time, and then  $\beta$  will hold
  - $\bigcirc\alpha$   $\alpha$  will hold at the next point in time
- discrete vs. continuous time:
  - $\bigcirc$  requires a discrete time scale
  - by dropping  $\bigcirc$ , we can also handle continuous time

# A Curry–Howard correspondence for temporal logic

- type membership depends on the time
- all simple type operators plus a few new ones:



- a type member denotes a proof of the corresponding logical formula:

- $\alpha$  a family of  $\alpha$ -values, one for each future time
- ◆  $\alpha$  a future time together with an  $\alpha$ -value
- $\alpha \blacktriangleright \beta$  a tuple of the following:
  - a future time (the “stop time”)
  - a family of  $\alpha$ -values, one for each future time before the stop time
  - a  $\beta$ -value, belonging to the stop time
- $\alpha$  an  $\alpha$ -value, belonging to the next time

# Does this look familiar?

- ■ and ◆ are essentially type operators from FRP (Functional Reactive Programming)
- FRP:
  - a declarative way of programming reactive systems
- two core concepts:
  - **behaviors** time-varying values
  - **events** values occurring at a certain point in time
- examples:
  - **behavior** position of the mouse
  - **event** a key press
- executable system descriptions built by composing behaviors and events
- ■ and ◆ construct behavior and event types, respectively



# What about the other operators?

- $\blacktriangleright$ -values are finite behaviors with a terminating event
- examples:

- an audio signal that terminates at some point:

$$(\mathbb{R} \times \mathbb{R}) \blacktriangleright 1$$

- an audio signal that switches between stereo and mono:

$$(\mathbb{R} \times \mathbb{R}) \blacktriangleright \mathbb{R} \blacktriangleright (\mathbb{R} \times \mathbb{R}) \blacktriangleright \mathbb{R} \blacktriangleright \dots$$

(assuming  $\blacktriangleright$  is right-associative)

- an audio signal that is sometimes interrupted:

$$(\mathbb{R} \times \mathbb{R}) \blacktriangleright 1 \blacktriangleright (\mathbb{R} \times \mathbb{R}) \blacktriangleright 1 \blacktriangleright \dots$$

etc.

- ●-values for “Functional Reactive Dataflow Programming”
- dataflow programming:  
working with streams of values associated  
with discrete times

## Conclusions and outlook

- Curry–Howard correspondence as a one-to-one mapping between concepts from logic and concepts from programming
- programming can benefit from logic and vice versa
- discovery of the Temporal Curry–Howard Correspondence leads to improvements of FRP
- examples shown in this talk:
  - ▶-types for specifying advanced temporal behavior
  - ●-types for bringing the glory of FRP to dataflow programming
- further examples:
  - a principled way of ensuring start time consistency for avoiding problems with semantics and performance
  - a better structured interface to FRP
  - new techniques for implementing FRP