# An Introduction to Functional Reactive Programming Lecture 2 (of 2)

Neil Sculthorpe

Functional Programming Group
Information and Telecommunication Technology Center
University of Kansas
neil@ittc.ku.edu

EECS 776
Lawrence, Kansas
5th November 2012

## Yampa Implementation

### *SF* Implementation (simplified)

$$\textbf{data}\ SF\ a\ b\ =\ SF\ (DTime\ \rightarrow\ a\ \rightarrow\ (SF\ a\ b, b))$$

$DTime$ = amount of time since the previous sample

## Yampa Implementation

**_SF_ Implementation (simplified)**

$$\textbf{data}\ SF\ a\ b\ =\ SF\ (DTime\ \rightarrow\ a\ \rightarrow\ (SF\ a\ b, b))$$

$DTime =$ amount of time since the previous sample

- The latest implementation:
  - uses a GADT with multiple constructors
  - dynamically applies domain-specific optimisations

## Yampa Implementation

### *SF* Implementation (simplified)

**data** $SF\ a\ b\ =\ SF\ (DTime\ \rightarrow\ a\ \rightarrow\ (SF\ a\ b, b))$

$DTime$ = amount of time since the previous sample

- The latest implementation:
  - uses a GADT with multiple constructors
  - dynamically applies domain-specific optimisations
- Yampa is a ??? embedding.

## Yampa Implementation

### *SF* Implementation (simplified)

$$\textbf{data}\ \mathit{SF}\ a\ b\ =\ \mathit{SF}\ (\mathit{DTime}\ \rightarrow\ a\ \rightarrow\ (\mathit{SF}\ a\ b, b))$$

$\mathit{DTime}$ = amount of time since the previous sample

- The latest implementation:
  - uses a GADT with multiple constructors
  - dynamically applies domain-specific optimisations
- Yampa is a shallow embedding.

## Extra Signal Functions

### One time-step delay

$iPre :: a \rightarrow SF\ a\ a$

## Extra Signal Functions

**One time-step delay**

$iPre :: a \rightarrow SF\ a\ a$

**Suppress events at the first time step**

$notYet :: SF\ (Event\ a)\ (Event\ a)$

## Extra Signal Functions

**One time-step delay**

$iPre :: a \rightarrow SF\ a\ a$

**Suppress events at the first time step**

$notYet :: SF\ (Event\ a)\ (Event\ a)$

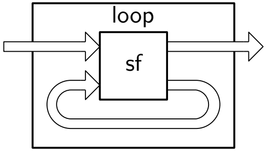
**An edge detector with a specific event value**

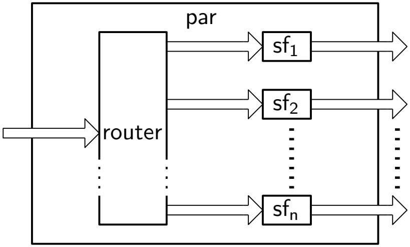
$edgeTag :: a \rightarrow SF\ Bool\ (Event\ a)$
$edgeTag\ a = edge \ggg arr\ (tagWith\ a)$

## Example: The Second Dimension

See accompanying code. . .

## Advanced Yampa Routing Combinators

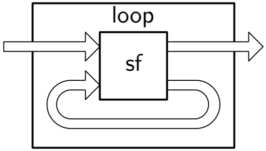


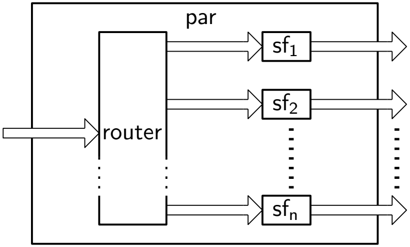


$loop :: SF\ (a, c)\ (b, c) \rightarrow SF\ a\ b$

$par :: (\forall\ sf\ .\ a \rightarrow [sf] \rightarrow [(b, sf)]) \rightarrow [SF\ b\ c] \rightarrow SF\ a\ [c]$

## Advanced Yampa Routing Combinators





$$loop :: SF\ (a, c)\ (b, c) \rightarrow SF\ a\ b$$

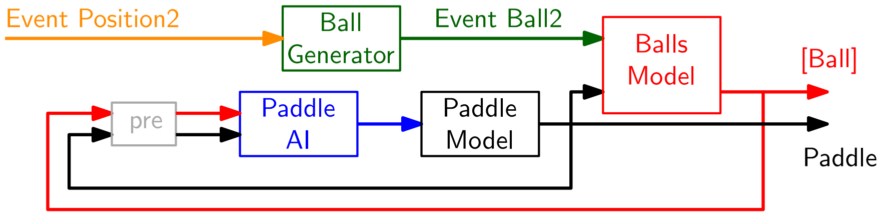
$$par :: (\forall\ sf\ .\ a \rightarrow [sf] \rightarrow [(b, sf)]) \rightarrow [SF\ b\ c] \rightarrow SF\ a\ [c]$$

$$\begin{aligned} pSwitch :: \ & (\forall\ sf\ .\ a \rightarrow [sf] \rightarrow [(b, sf)]) \\ \rightarrow\ & [SF\ b\ c] \\ \rightarrow\ & SF\ (a, [c])\ (Event\ e) \\ \rightarrow\ & ([SF\ b\ c] \rightarrow e \rightarrow SF\ a\ [c]) \\ \rightarrow\ & SF\ a\ [c] \end{aligned}$$

## Example: Adding and Deleting Signal Functions

See accompanying code. . .

## Pong: High-level View



## Conclusion

- Yampa is just one FRP language among many.
- If you want to learn more about Yampa, I'd recommend Henrik Nilsson's recent mini-course:
  `http://www.cs.nott.ac.uk/~nhn/ITU-FRP2010/`
- Exercise: Add an alien spaceship to the Pong game, with at least one of the following features:
  - balls bounce off its sides; [hint: see the ball/paddle interaction]
  - it flies towards mouse clicks; [hint: use *hold* or *switch*]
  - a gun that fires new balls at regular intervals; [hint: use *repeatedly*]
  - a second (slower) spaceship that chases the first.

  Email scripts to me by Friday 16th November.