Exploiting Signal Functions and Signal Kinds in Functional Reactive Programming

Neil Sculthorpe

(supervised by Henrik Nilsson)

Functional Programming Laboratory
School of Computer Science
University of Nottingham
United Kingdom
nas@cs.nott.ac.uk

Functional Programming Laboratory Away Day
Castleton, England
21st July 2010

Reactive Programming

- Reactive Program: one that continually interacts with its environment, interleaving input and output in a timely manner.

Reactive Programming

- Reactive Program: one that continually interacts with its environment, interleaving input and output in a timely manner.
- Examples: MP3 players, robot controllers, video games, aeroplane control systems. . .

Reactive Programming

- Reactive Program: one that continually interacts with its environment, interleaving input and output in a timely manner.
- Examples: MP3 players, robot controllers, video games, aeroplane control systems. . .
- Contrast with transformational programs, which take all input at the start of execution and produce all output at the end (e.g. a compiler).

Functional Reactive Programming (FRP)

Functional Reactive Programming (FRP)

- Time is considered to be continuous.

Functional Reactive Programming (FRP)

- Time is considered to be continuous.
- Based around time-varying values called signals.

Functional Reactive Programming (FRP)

- Time is considered to be continuous.
- Based around time-varying values called signals.

Conceptual Model of Signals

$$
\begin{aligned}
\text{Time} &= \mathbb{R} \\
\text{Signal A} &= \text{Time} \rightarrow \text{A}
\end{aligned}
$$

Functional Reactive Programming (FRP)

- Time is considered to be continuous.
- Based around time-varying values called signals.

Conceptual Model of Signals

$$\begin{aligned} \text{Time} &= \mathbb{R} \\ \text{Signal A} &= \text{Time} \to \text{A} \end{aligned}$$

- Usually implemented as an embedded language.

Functional Reactive Programming (FRP)

- Time is considered to be continuous.
- Based around time-varying values called signals.

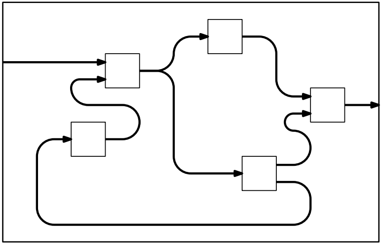
Conceptual Model of Signals

$$\begin{aligned} \text{Time} &= \mathbb{R} \\ \text{Signal A} &= \text{Time} \rightarrow \text{A} \end{aligned}$$

- Usually implemented as an embedded language.
- Compared to most other reactive languages, FRP:
  - is more expressive;
  - lacks performance and safety guarantees.

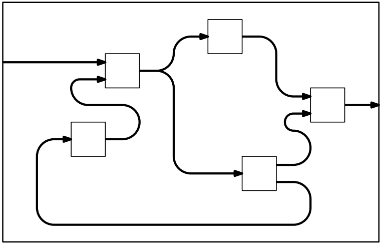
Dynamic Hybrid Synchronous Data-flow

- FRP programs are synchronous data-flow networks.



Dynamic Hybrid Synchronous Data-flow

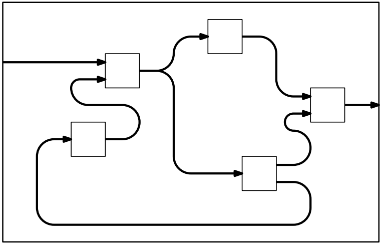
- FRP programs are synchronous data-flow networks.



- Hybrid: Continuous-time and discrete-time aspects.
  - Continuous-time signals.
  - Discrete-time events.

Dynamic Hybrid Synchronous Data-flow

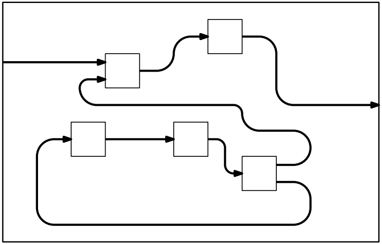
- FRP programs are synchronous data-flow networks.



- Hybrid: Continuous-time and discrete-time aspects.
  - Continuous-time signals.
  - Discrete-time events.
- Dynamic: Network structure can change at run-time in arbitrary ways.

Dynamic Hybrid Synchronous Data-flow

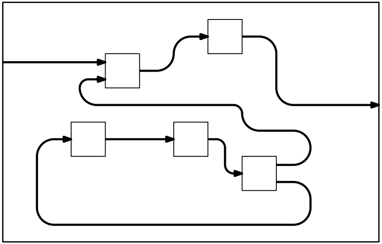
- FRP programs are synchronous data-flow networks.



- Hybrid: Continuous-time and discrete-time aspects.
  - Continuous-time signals.
  - Discrete-time events.
- Dynamic: Network structure can change at run-time in arbitrary ways.

Dynamic Hybrid Synchronous Data-flow

- FRP programs are synchronous data-flow networks.



- Hybrid: Continuous-time and discrete-time aspects.
  - Continuous-time signals.
  - Discrete-time events.
- Dynamic: Network structure can change at run-time in arbitrary ways.
- Optimisation and safety guarantees are much harder than for static networks.

Signal Functions

- FRP programming involves applying functions to signals.

Signal Functions

- FRP programming involves applying functions to signals.

Example: Functions on Signals

lift : (A → B) → Signal A → Signal B

integrate : Signal $\mathbb{R}$ → Signal $\mathbb{R}$

Signal Functions

- FRP programming involves applying functions to signals.

Example: Functions on Signals

$$\text{lift} : (A \to B) \to \text{Signal } A \to \text{Signal } B$$

$$\text{integrate} : \text{Signal } \mathbb{R} \to \text{Signal } \mathbb{R}$$

- Rather than signals, we could take such signal functions as the primary first-class abstraction.

Conceptual Model of Signal Functions

$$\text{SF } A\ B \;=\; \text{Signal } A \to \text{Signal } B$$

Signal Functions

- FRP programming involves applying functions to signals.

Example: Functions on Signals

lift : (A → B) → Signal A → Signal B

integrate : Signal $\mathbb{R}$ → Signal $\mathbb{R}$

- Rather than signals, we could take such signal functions as the primary first-class abstraction.

Conceptual Model of Signal Functions

SF A B = Signal A → Signal B

Example: Signal Functions

lift : (A → B) → SF A B

integrate : SF $\mathbb{R}$ $\mathbb{R}$

Advantages of Signal Functions

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
    - Practical implementations of first-class signals have to resort to hidden IO to solve this.

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
    - Practical implementations of first-class signals have to resort to hidden IO to solve this.
    - Whereas signal functions can be implemented purely and simply.

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
    - Practical implementations of first-class signals have to resort to hidden IO to solve this.
    - Whereas signal functions can be implemented purely and simply.
- The network structure can be stored in the signal function abstraction, not hidden in host language functions (useful for optimisation).

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
  - Practical implementations of first-class signals have to resort to hidden IO to solve this.
  - Whereas signal functions can be implemented purely and simply.
- The network structure can be stored in the signal function abstraction, not hidden in host language functions (useful for optimisation).
- Additional information can be associated with signal functions. Useful for:

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
  - Practical implementations of first-class signals have to resort to hidden IO to solve this.
  - Whereas signal functions can be implemented purely and simply.
- The network structure can be stored in the signal function abstraction, not hidden in host language functions (useful for optimisation).
- Additional information can be associated with signal functions. Useful for:
  - Optimisation (e.g. fusing lifted pure functions, change propagation)

Advantages of Signal Functions

- First-class signals often lead to space and time leaks.
  - Practical implementations of first-class signals have to resort to hidden IO to solve this.
  - Whereas signal functions can be implemented purely and simply.
- The network structure can be stored in the signal function abstraction, not hidden in host language functions (useful for optimisation).
- Additional information can be associated with signal functions. Useful for:
  - Optimisation (e.g. fusing lifted pure functions, change propagation)
  - Safety Guarantees (e.g. ensuring an absence of instantaneous feedback, even with dynamic network structure)

Different Kinds of Signals

Different Kinds of Signals

Three kinds of signal:

- Continuous Signals
- Event Signals
- Step Signals

Different Kinds of Signals

Three kinds of signal:

- Continuous Signals
- Event Signals
- Step Signals

Conceptual Model of Signal Kinds

$$\begin{aligned}\mathbf{C}\ \mathsf{A} &= \mathsf{Time} \to \mathsf{A}\\ \mathbf{E}\ \mathsf{A} &= \mathsf{List}\ (\mathsf{Time} \times \mathsf{A})\\ \mathbf{S}\ \mathsf{A} &= \mathsf{A} \times \mathsf{List}\ (\mathsf{Time} \times \mathsf{A})\end{aligned}$$

Different Kinds of Signals

Three kinds of signal:

- Continuous Signals
- Event Signals
- Step Signals

Conceptual Model of Signal Kinds

$$\begin{aligned}\mathbf{C}\ \mathsf{A} &= \mathsf{Time} \rightarrow \mathsf{A}\\ \mathbf{E}\ \mathsf{A} &= \mathsf{List}\ (\mathsf{Time} \times \mathsf{A})\\ \mathbf{S}\ \mathsf{A} &= \mathsf{A} \times \mathsf{List}\ (\mathsf{Time} \times \mathsf{A})\end{aligned}$$

Distinguishing between signal kinds allows for more efficient implementation strategies.

Leaky Abstractions

- In some FRP implementations, there is only one signal kind.

Leaky Abstractions

- In some FRP implementations, there is only one signal kind.
- Events are embedded in continuous signals as an option type.

Embedding Events in Continuous Signals

$$\text{Event A} = \text{Signal (Maybe A)}$$

Leaky Abstractions

- In some FRP implementations, there is only one signal kind.
- Events are embedded in continuous signals as an option type.

Embedding Events in Continuous Signals

$$\text{Event A} \;=\; \text{Signal (Maybe A)}$$

- In a sampled implementation, this can be a leaky abstraction.

Leaky Abstractions

- In some FRP implementations, there is only one signal kind.
- Events are embedded in continuous signals as an option type.

Embedding Events in Continuous Signals

$$\text{Event A} = \text{Signal (Maybe A)}$$

- In a sampled implementation, this can be a leaky abstraction.
- Sampled implementations:
  - Approximate signals over a discrete sequence of time steps.
  - The sampling rate is not specified, and can vary.
  - If the sampling rate varies, it is reasonable to eliminate or duplicate samples of a continuous signal.

Leaky Abstractions

- In some FRP implementations, there is only one signal kind.
- Events are embedded in continuous signals as an option type.

Embedding Events in Continuous Signals

$$\text{Event A} = \text{Signal (Maybe A)}$$

- In a sampled implementation, this can be a leaky abstraction.
- Sampled implementations:
  - Approximate signals over a discrete sequence of time steps.
  - The sampling rate is not specified, and can vary.
  - If the sampling rate varies, it is reasonable to eliminate or duplicate samples of a continuous signal.
- Duplicating or eliminating events could be disastrous!

Summary

- FRP languages choose between signals or signal functions as the primary abstraction.
- First-class signal functions provide opportunities for safety guarantees and optimisation.
- Identifying distinct kinds of signals allows for more efficient implementation strategies, and avoids some leaky abstractions.