ReactiveML in the Landscape of Synchronous and Functional Reactive Languages

Louis Mandel   Marc Pouzet

ICFP
Saint Louis
Sept. 2018

The begining

The first implem. starts in 2003. [PPDP'05; Mandel's PhD. thesis, 2006]

Objective

Provide high level constructs in ML to program reactive applications.

E.g., deterministic parallelism, instantaneous broadcast, suspension, pre-emption, etc.

Build on the principles of synchronous languages, with an Esterel style.

Do not impose that program must run in statically known bounded space and time.

Some novel questions to address

A semantics that highlights the interaction with the base language, e.g., higher-order, recursion.

Define an efficient implementation that avoid busy waiting.

ReactiveML in the "synchronous" landscape

A programming style *a la Esterel* [Berry & Gonthier, SCP 1992]

Differences

It is possible to react instantaneously to the presence of an event.

But not instantaneously to the absence of an event. Hence:

The Esterel program:

```
present s1 else emit s2
```

is interpreted as:

```
present s2 else (pause; emit s2)
```

This makes all programs "causal-by-construction".

This idea was introduced by Frédéric Boussinot in the early 90's.

Computations are dynamically scheduled vs statically scheduled.

Differences w.r.t data-flow synchronous languages.

Lustre [Caspi et al., POPL 1987],
Lucid Synchrone [Caspi and Pouzet, ICFP 1996]

The data-flow programming style

A signal is a stream; a system is a stream function; streams are synchronous.

X	1	2	1	4	5	6	...
Y	2	4	2	1	1	2	...
$X + Y$	3	6	3	5	6	8	...
$pre\ X$	*nil*	1	2	1	4	5	...
$Y \text{->} X$	2	2	1	4	5	6	...

The equation $Z = X + Y$ means $\forall n. Z_n = X_n + Y_n$.

Time is logical: inputs X and Y arrivent "at the same time"; output Z is produced "at the same time"

Is-it real-time?

Reason in wort case: check that the generated code produces the output before a new input arrives.

Differences w.r.t Lustre

The programming style of RML is different.

The compiler statically ensure that the program can be compiled into statically scheduled code that run in bounded time and space.

RML has features that are absent in Lustre: higher order, ML polymorphism, type inference.

Computations are dynamically scheduled.

No static guaranty that the generated code runs in bounded time and space.

Differences w.r.t Lucid Synchrone

The programming style of RML is different.

Lucid Synchrone extends Lustre with ML-like features.

It adds higher-order, ML polymorphism and several dedicated type systems.

Code is statically scheduled. The compiler can ensure that the generated code runs in bounded time and space.

Differenes w.r.t FRP

The programming style of RML is more imperative.

It can call any OCaml function, possibly with side-effects.

RML focuses on control-structures: suspension, pre-emption.

RML provides a type system to separate instantaneous computations from the others.

A static analysis that check the absence of instantaneous recursion.