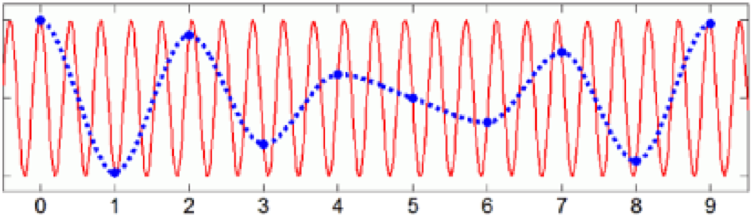
Verifying Faust in Coq

Progress report

Emilio J. Gallego Arias, Pierre Jouvelot, Olivier Hermant, Arnaud Spiwack

Mines-ParisTech, PSL Research University

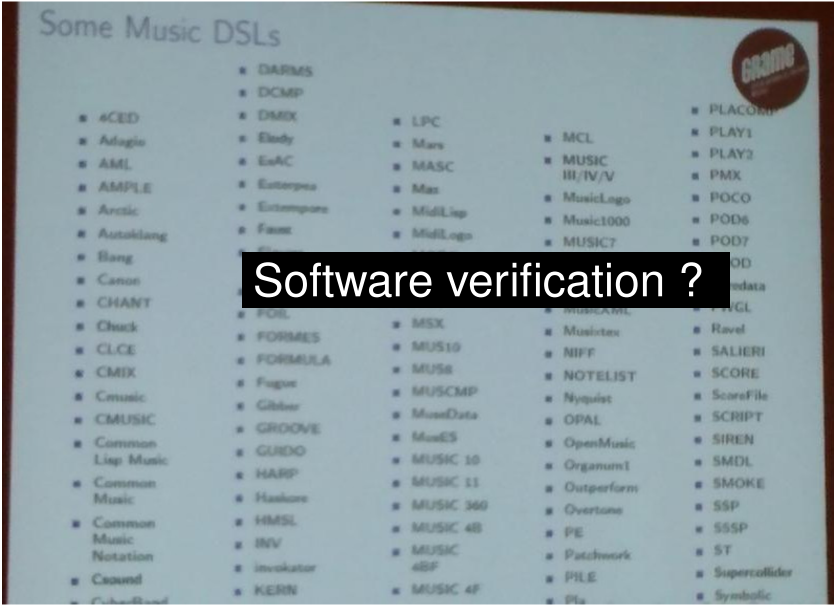
CoqPL 2015



Music and PL?

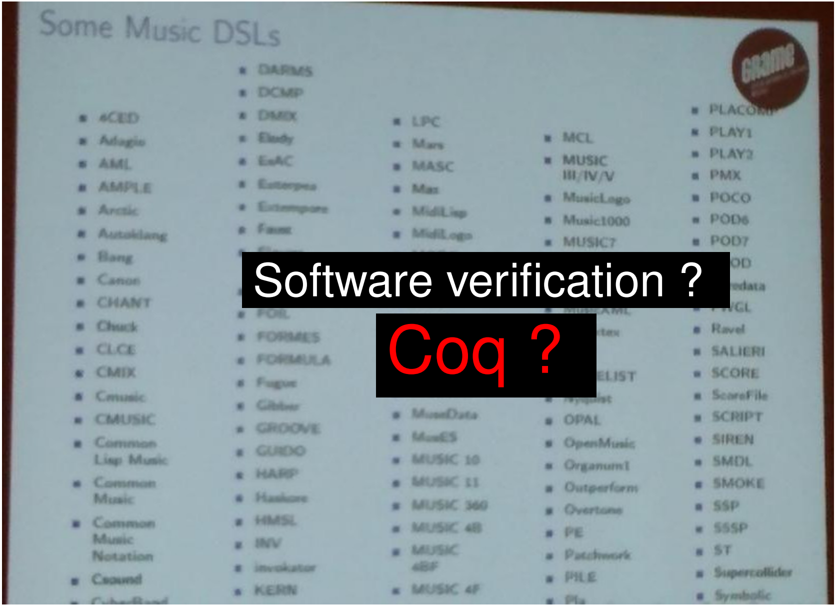

Some Music DSLs
4CED
Adagio
AML
AMPLE
Arctic
Autoklang
Bang
Canon
CHANT
Chuck
CLCE
CMIX
Cmusic
CMUSIC
Common Lisp Music
Common Music
Common Music Notation
Csound
DARMS
DCMP
DMIX
Elody
EsAC
Euterpea
Extempore
Faust
Flavors Band
Fluxus
FOIL
FORMES
FORMULA
Fugue
Gibber
GROOVE
GUIDO
HARP
Haskore
HMSL
INV
invokator
KERN
LPC
Mars
MASC
Max
MidiLisp
MidiLogo
MODE
MOM
MSX
MUS10
MUS8
MUSCMP
MuseData
MusES
MUSIC 10
MUSIC 11
MUSIC 360
MUSIC 4B
MUSIC 4BF
MUSIC 4F
MCL
MUSIC III/IV/V
MusicLogo
Music1000
MUSIC7
Musictex
MUSIGOL
MusicXML
Musixtex
NIFF
NOTELIST
Nyquist
OPAL
OpenMusic
Organum1
Outperform
Overtone
PE
Patchwork
PILE
PLACOMP
PLAY1
PLAY2
PMX
POCO
POD6
POD7
PROD
Puredata
PWGL
Ravel
SALIERI
SCORE
ScoreFile
SCRIPT
SIREN
SMDL
SMOKE
SSP
SSSP
ST
Supercollider

Music and PL?



Software verification ?

Music and PL?



Software verification ?

Coq ?

Faust

- Functional PL for digital signal processing.
- Synchronous paradigm, geared towards audio.
- Programs: circuits/block diagrams + feedbacks.
- Semantics: streams of samples.
- *Efficiency is crucial*.
- Created in 2000 by Yann Orlarey et al. at GRAME.
- Mature, compiles to more than 14 platforms.

Faust's Ecosystem

Users:

- Grame: Multiple projects, main developer.
- Stanford: Class/books on signal processing, STK instrument toolkit, Faust2android, Mephisto…
- Ircam: Acoustic libraries, effects libraries,…
- Guitarix, moForte guitar, etc...

Faust's Ecosystem

Users:

- Grame: Multiple projects, main developer.
- Stanford: Class/books on signal processing, STK instrument toolkit, Faust2android, Mephisto…
- Ircam: Acoustic libraries, effects libraries,…
- Guitarix, moForte guitar, etc...

It has its market! Much easier than dwelling into C.

Faust's Ecosystem

Users:

- Grame: Multiple projects, main developer.
- Stanford: Class/books on signal processing, STK instrument toolkit, Faust2android, Mephisto…
- Ircam: Acoustic libraries, effects libraries,…
- Guitarix, moForte guitar, etc...

It has its market! Much easier than dwelling into C.

Recent Events:

- Faust day at Stanford happened yesterday.
- Ongoing Faust program competition (2000€ in prices).
- FEEVER project :)

Syntax and Well-Formedness

$$\text{TERM}\ \frac{}{\vdash ! : 1 \to 0} \qquad \text{ID}\ \frac{}{\vdash \_ : 1 \to 1}$$

$$\text{PAR}\ \frac{\vdash f_1 : i_1 \to o_1 \qquad \cdots \qquad \vdash f_n : i_n \to o_n}{\vdash (f_1, \ldots, f_n) : \sum_j^n i_j \to \sum_j^n o_j}$$

$$\text{COMP}\ \frac{\vdash f : i \to k \qquad \vdash g : k \to o}{\vdash (f : g) : i \to o}$$

$$\text{PAN}\ \frac{\vdash f : i \to k \qquad \vdash g : k * n \to o \qquad 0 < k \land 0 < n}{\vdash f <: g : i \to o}$$

Syntax and Typing

PL standard practice vs what the musicians want/imagine:

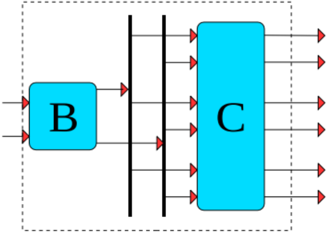


Figure 2: (B:C) sequential composition of B and C

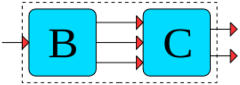
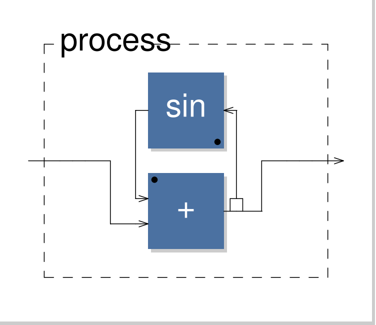


Figure 3: sequential composition of B and C when $k = 1$

Feedbacks

$$\textsc{Feed}\ \frac{\vdash f : g_o + f_i \to g_i + f_o \qquad \vdash g : g_i \to g_o}{\vdash f \sim g : f_i \to f_o}$$

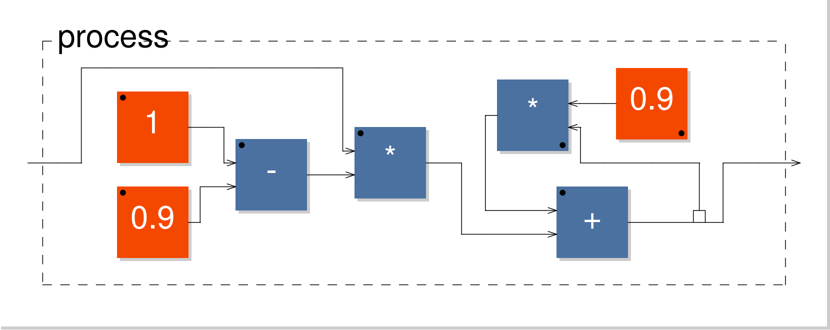
Diagram for `+` $\sim$ `sin`:



Synchronous semantics: execution in “ticks” + state.

Simple Low-pass Filter

```
smooth(c) = *(1−c) : + *( c);
process = smooth(0.9);
```

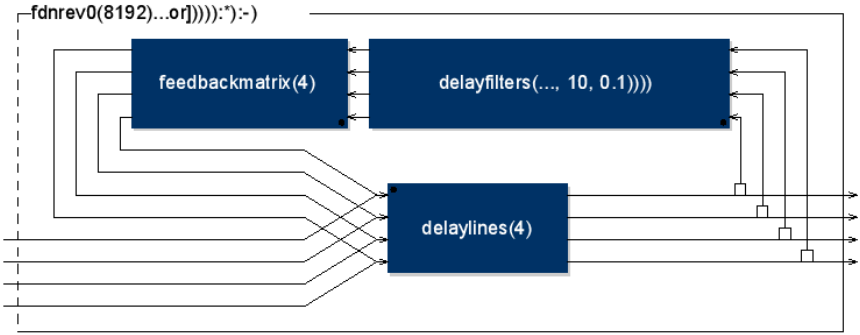


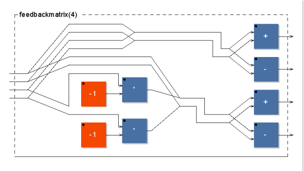
T:	1	2	3	4	5	6	7	8
I:	1.00	1.05	1.10	1.15	1.20	1.25	1.20	1.25
O:	0.10	0.19	0.28	0.37	0.45	0.53	0.61	0.68

A More Real Example:

```
fdnrev0(delays, BBSO, freqs, durs, loopgainmax, nonl)
  = (bus(2*N) :> bus(N) : delaylines(N))
    (delayfilters(N,freqs,durs) : feedbackmatrix(N))
with {
  delayval(i) = take(i+1,delays);
  delaylines(N) = par(i,N,(delay(dlmax(i),(delayval(i)−1))));
  delayfilters(N,freqs,durs) = par(i,N,filter(i,freqs,durs));
  feedbackmatrix(N) = bhadamard(N);
  vbutterfly(n) = bus(n) <: (bus(n):>bus(n/2)) ,  ...)
  ...
};
```

A More Real Example:





Why Coq?

Why Coq?

Does there exist any other programming language?

Why Coq? Motivations and Goals:

Philosophical

- Manual proofs starting to feel odd in PL.
- Motto: use Coq from the start.
- Try to develop in reusable way: both for the Faust/DSP and Coq communities.

Why Coq? Motivations and Goals:

PHILOSOPHICAL — MATHEMATICAL

- Prove programs correct, reason about them in new ways. Current testing process is to compare output with MatLab's.

Why Coq? Motivations and Goals:

PHILOSOPHICAL — MATHEMATICAL

- Prove programs correct, reason about them in new ways. Current testing process is to compare output with MatLab's.
- Optimizations performed by the compiler are not well understood. *Semantics trickier than it looks to the eye*

Why Coq? Motivations and Goals:

PHILOSOPHICAL — MATHEMATICAL

- Prove programs correct, reason about them in new ways. Current testing process is to compare output with MatLab's.
- Optimizations performed by the compiler are not well understood. *Semantics trickier than it looks to the eye*
- Explore the formalization of concepts from the signal processing community: Finite Impulse Response (FIR) filters, LTI theory, spectral analysis, Nyquist...

Why Coq? Motivations and Goals:

PHILOSOPHICAL — MATHEMATICAL
PRACTICAL

Less effort than to build a custom analysis tool.

Applications:

Why Coq? Motivations and Goals:

PHILOSOPHICAL — MATHEMATICAL
PRACTICAL

Less effort than to build a custom analysis tool.

Applications:

IMHO: **Robust Definitions and Standards** are crucial.

Don't repeat the mistakes of the past

Some Properties

- Stability properties: bound input produces bounded output.

Some Properties

- Stability properties: bound input produces bounded output. This will be our example.

Some Properties

- Stability properties: bound input produces bounded output. This will be our example.
- Linearity/Time invariance. [Note: relational!]
- Stabilization: Zero input eventually produces zero output.

Relating Programs:

Impulse response (two poles filter):

$$H(z) = \frac{1 - z^{-2}}{1 - 2R\cos(\Theta_c)z^{-1} + R^2 z^{-2}}$$

Relating Programs:

Impulse response (two poles filter):

$$H(z) = \frac{1 - z^{-2}}{1 - 2R\cos(\Theta_c)z^{-1} + R^2 z^{-2}}$$

```
process = firpart :+  feedback
with {
  bw = 100; fr = 1000; g = 1;  //  parameters − see caption
  SR = fconstant(int fSamplingFreq, <math.h>); // Faust fn
  pi = 4*atan(1.0);      //  circumference over diameter
  R = exp(0−pi*bw/SR); // pole radius [0 required]
  A = 2*pi*fr/SR;        //  pole angle (radians)
  RR = R*R;
  firpart(x) = (x − x'')  *  g  *  ((1−RR)/2);
  feedback(v) = 0 + 2*R*cos(A)*v − RR*v';
};
```

Finally! Let's Talk About Coq!

So far:

- Mathcomp library allowed us to do a prototype in two weeks.
- New feedback reasoning rule: proved sound.
- *Motivated by real use cases*.
- Defined a one-state logic, proved it sound.
- Again, mathcomp was key.

Finally! Let's Talk About Coq!

So far:

- Mathcomp library allowed us to do a prototype in two weeks.
- New feedback reasoning rule: proved sound.
- *Motivated by real use cases*.
- Defined a one-state logic, proved it sound.
- Again, mathcomp was key.

Currently:

- Investigating more complex logics.
- New semantics needed, based on guarded recursion.

The Pieces of the Puzzle

The First Piece: Streams

- We ported [Boulmé, Hamon and Pouzet], some problems with CoInductives.
- Like in C. Auger Lustre certified compiler, we choose to work with sequences (for now).
- Didn't look into PACO and more advanced co-reasoning tools.

The First Piece: Streams

- We ported [Boulmé, Hamon and Pouzet], some problems with CoInductives.
- Like in C. Auger Lustre certified compiler, we choose to work with sequences (for now).
- Didn't look into PACO and more advanced co-reasoning tools.

The current solution: a realizability semantics in guarded recursion style. Suggested simultaneously by A. Spiwak and A. Guatto:

$$[\![\vdash f : i \to o]\!]^n_W : [\![i]\!]^n \to [\![o]\!]^n$$

The Second Piece: Analysis

- Not in Mathcomp. rcfType good enough for now.
- How hard is to prove Euler's identity:

todo

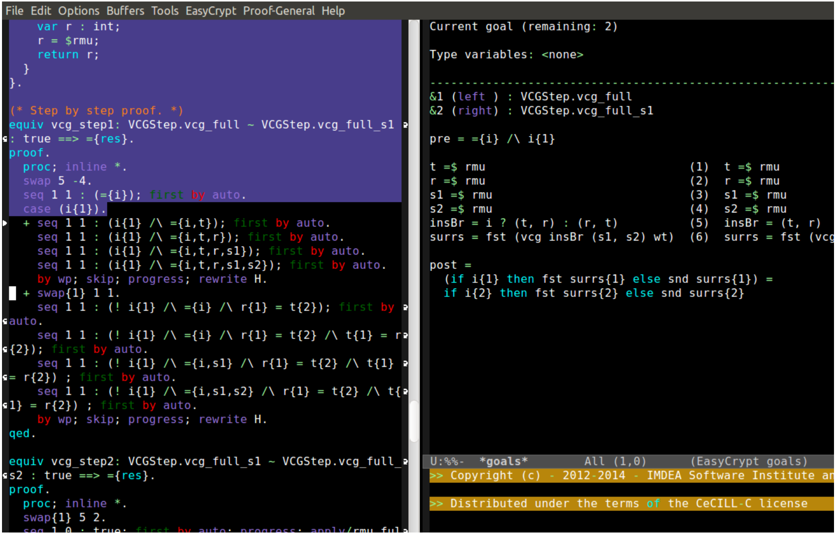
- Difficult to chose. C-CorN? The standard library? Coquelicot?
- Our feeling is that given the amount of analysis going on our life is going to be very painful.

[We ignore precision issues and machine floats for now]

The Third Piece: Coq as a Tool

- Is building a verification tool on top of Coq feasible? Does it even make sense?
- We got some inspiration from EasyCrypt.
- Would our tool mature, we would certainly need to plug deeply into Coq's parsing/display routines.
- We still think this may be better than rewriting everything from scratch.
- Our approach to automation: last thing to worry about.

The Third Piece: Coq as a Tool



Verification of the Smooth Filter:

Recall the smooth filter.

```
smooth(c) = *(1−c) : + *( c);
```

We want to prove stability, that is, bounded inputs produce bounded outputs, provided the coefficient *c* is in [0, 1]. Three significant cases:

```
by rewrite ?ler_wpmul2r ?ler_subr_addr ?add0r.

have Ha: a = a * c + a * (1 − c)
  by rewrite −mulrDr addrC addrNK mulr1.
have Hb: b = b * c + b * (1 − c)
  by rewrite −mulrDr addrC addrNK mulr1.
by rewrite Ha Hb !ler_add.

by rewrite ?ler_wpmul2r.
```

We pushed the VC to Why3 with success. Technique ready for incorporation into the main compiler.

Conclusions:

- Young project, highly positive so far.
- First alpha release very near.
- Tons of related work, difficult to get a good perspective.
- Most challenging topic: real/complex analysis.
- Certified audio/dsp processing? (Do we need it?)
- All of the usual Coq caveats apply to us.
- What do *you* think?

Thanks!