A Taste of (formal) Sound Reasoning

A Tutorial

Emilio J. Gallego Arias, Pierre Jouvelot, Olivier Hermant

MINES ParisTech, PSL Research University, France

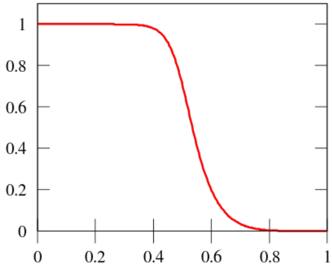
Linux Audio Conf 2015

e+lac@x80.org @ejgallego

https://github.com/ejgallego/mini-faust-coq

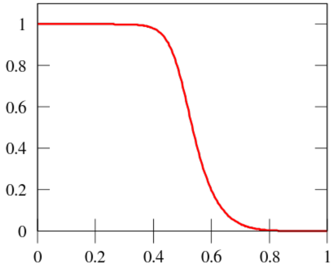
Let's start with a simple IIR filter:

$$\text{smooth}_n = (1 - c) \cdot x_n + c \cdot \text{smooth}_{n-1}$$



Let's start with a simple IIR filter:

$$\text{smooth}_n = (1 - c) \cdot x_n + c \cdot \text{smooth}_{n-1}$$



What can we know about it?

$$\text{smooth}_n = (1 - c) \cdot x_n + c \cdot \text{smooth}_{n-1}$$

Natural questions are:

- Frequency response;
- Stability;
- Linearity/Time Invariance.

Answers given by standard DSP theory.

$$\text{smooth}_n = (1 - c) \cdot x_n + c \cdot \text{smooth}_{n-1}$$

Natural questions are:

- Frequency response;
- Stability;
- Linearity/Time Invariance.

Answers given by standard DSP theory.

What about the implementation of the filter?

We dive into the realm of PL theory!

$$\text{smooth}_n = (1 - c) \cdot x_n + c \cdot \text{smooth}_{n-1}$$

Natural questions are:

- Frequency response;
- Stability;
- Linearity/Time Invariance.

Answers given by standard DSP theory.

What about the implementation of the filter?

We dive into the realm of PL theory!

Paradigm shift!

Certainty

Certainty

“Absolute” confidence in something we believe.

Certainty

"Absolute" confidence in something we believe.
How do we know something is "absolutely" true?

Many possible answers

Many possible answers

In the Programming Languages field, we want computers to check knowledge for us!

How does it work?

Welcome to Evidence!

How does it work?

Welcome to Evidence!

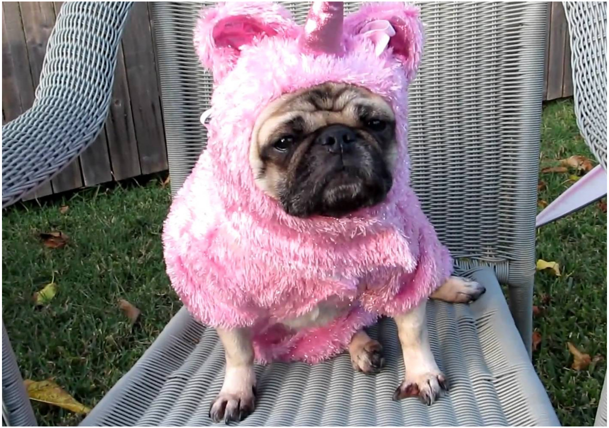
We will build a particular kind of evidence for a property of our filter, then use the computer to validate it.

Types of Evidence

Bob	Hi Alice, my dog is feeling weird!
Alice	I don’t believe you!

Types of Evidence

Bob	Hi Alice, my dog is feeling weird!
Alice	I don't believe you!



Logical Evidence: Proofs

We want to agree on a convention to produce and check evidence.

A logic is a language and a set of rules geared towards the production of symbolic evidence.

Logical Evidence: Proofs

We want to agree on a convention to produce and check evidence.

A logic is a language and a set of rules geared towards the production of symbolic evidence.

Proposition Examples

"Every even number is not prime."
"Every complex polynomial has a root."
"Every finite impulse filter is stable."

Checking Validity: Inference

To check when a proposition holds, we need rules.

Rule Examples

"If A and B hold, B holds."

"If P holds for 0, and assuming P holds for n we can prove that P holds for n+1, then P holds for all n."

The Theory of Forms

Truth

Truth lives in the idealistic, infinite universe.

$$\Gamma \models \varphi \qquad \text{if } \Gamma \text{ is true, then } \varphi \text{ is true}$$

Proof

Reasoning lives in the concrete, syntactic universe.

$$\Gamma \vdash \varphi \qquad \varphi \text{ can be proved from } \Gamma$$

using a valid application of the rules.

Linking the Worlds

$\Gamma \models \varphi$	if Γ is true, then φ is true
$\Gamma \vdash \varphi$	φ can be proved from Γ

Main Properties

- Soundness: $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$.

Linking the Worlds

$\Gamma \models \varphi$	if Γ is true, then φ is true
$\Gamma \vdash \varphi$	φ can be proved from Γ

Main Properties

- Soundness: $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$.
- Completeness: $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.

Linking the Worlds

$\Gamma \models \varphi$	if Γ is true, then φ is true
$\Gamma \vdash \varphi$	φ can be proved from Γ

Main Properties

- Soundness: $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$.
- Completeness: $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.
- Consistency: $\nvdash A \wedge \neg A$.

Linking the Worlds

$\Gamma \models \varphi$	if Γ is true, then φ is true
$\Gamma \vdash \varphi$	φ can be proved from Γ

Main Properties

- Soundness: $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$.
- Completeness: $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.
- Consistency: $\nvdash A \wedge \neg A$.

We are liberated from the complexity of the ideal, infinite world; we can now use mechanical, finitary rules to reason about it!

Computational Evidence

Assume we want our computer to check our deductions.

Computational Evidence

Assume we want our computer to check our deductions.

We could write a rule checker. But how do we know the rule checker is correct?

Computational Evidence

Assume we want our computer to check our deductions.

We could write a rule checker. But how do we know the rule checker is correct?

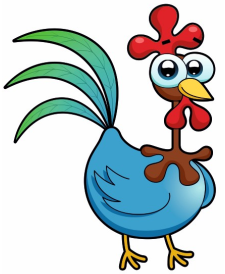
A crucial, fundamental idea:

Programs are Proofs!
Types are Propositions!

Computational Evidence

Welcome to Coq!

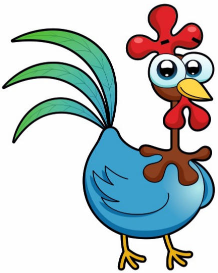
```
aptitude install coq
```



Computational Evidence

Welcome to Coq!

```
aptitude install coq
```



In Coq, proofs are precisely the well-typed functional programs. Type-checking validates our logical deductions!

BHK-Interpretation

Computational interpretation of logic

Type	**Proof / Program**
$P \wedge Q$	Record with proofs of P and Q.
$P \rightarrow Q$	Program that takes a proof of P, then produces a proof of Q.
$\forall(x : P), Q(x)$	Program that takes p, a proof of P, then produces a proof of $Q(p)$
$\exists(x : P), Q(x)$	Pair (p, q) of p, a proof of P, and q, a proof of $Q(p)$.
$P \vee Q$	????

Let’s Move Back to Audio

Use Coq to reason about audio programs; in particular, we’ll use a toy version of Faust!

Let's Move Back to Audio

Use Coq to reason about audio programs; in particular, we'll use a toy version of Faust!

What's the plan?

Let's Move Back to Audio

Use Coq to reason about audio programs; in particular, we'll use a toy version of Faust!

What's the plan?

1. Define the syntax of Faust inside Coq.

Let’s Move Back to Audio

Use Coq to reason about audio programs; in particular, we’ll use a toy version of Faust!

What’s the plan?

1. Define the syntax of Faust inside Coq.
2. Define a representation for (sampled) sound.

Let's Move Back to Audio

Use Coq to reason about audio programs; in particular, we'll use a toy version of Faust!

What's the plan?

1. Define the syntax of Faust inside Coq.
2. Define a representation for (sampled) sound.
3. Link the two.

Let's Move Back to Audio

Use Coq to reason about audio programs; in particular, we'll use a toy version of Faust!

What's the plan?

1. Define the syntax of Faust inside Coq.
2. Define a representation for (sampled) sound.
3. Link the two.
4. ?

Let’s Move Back to Audio

Use Coq to reason about audio programs; in particular, we’ll use a toy version of Faust!

What’s the plan?

1. Define the syntax of Faust inside Coq.
2. Define a representation for (sampled) sound.
3. Link the two.
4. ?
5. Profit!

Let's Move Back to Audio

Use Coq to reason about audio programs; in particular, we'll use a toy version of Faust!

What's the plan?

1. Define the syntax of Faust inside Coq.
2. Define a representation for (sampled) sound.
3. Link the two.
4. ?
5. Profit!

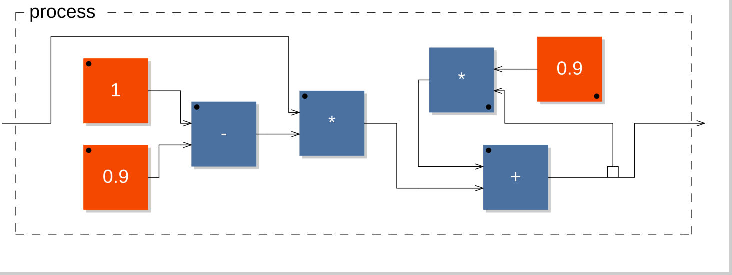
[Seriously, we'd love to hear about 4!]

Back to the Filter

$$\text{smooth}_n = (1 - c)x_n + c \cdot \text{smooth}_{n-1}$$

Using Faust:

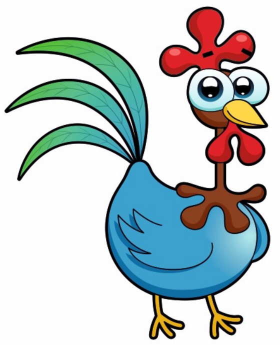
```
smooth(c) = *(1-c) : + ~ *(c)
```



[For c = 0.9]

Let's do it!

```
smooth(c) = *(1-c) : + ~ *(c)
```



Semantics

We can “write” Faust programs inside Coq. Now we want to run them.

Semantics

We can “write” Faust programs inside Coq. Now we want to run them.

Output of Smooth

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

What is Sound? Choices…

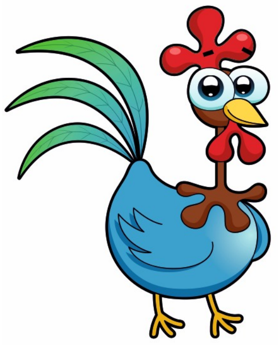
We need to choose how to represent sound in Coq? In the formal world, we *pay* for every detail.

- Conceptual representations? ($\mathbb{R} \to \mathbb{R}$).
- Infinite representations? ($\mathbb{N} \to \mathbb{R}$)
- Finite representation? (seq $\mathbb{R}$)

We'll use the last one.

Let's do it!

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



When is Smooth Stable?

We are in good shape; now, when is smooth stable?

$$\text{smooth}_n = (1 - c)x_n + c \cdot \text{smooth}_{n-1}$$

When is Smooth Stable?

We are in good shape; now, when is smooth stable?

$$\text{smooth}_n = (1 - c)x_n + c \cdot \text{smooth}_{n-1}$$

Smooth is stable when $c \in (0, 1)$. Formally:

$$\forall i \in [a, b],\ c \in [0, 1] \rightarrow smooth(c)\ i \in [a, b]$$

Proving Stability

We can do the proof directly in Coq; it is not difficult but cumbersome in general.

Proving Stability

We can do the proof directly in Coq; it is not difficult but cumbersome in general.

But we can use better, higher-level reasoning principles: Use *program logics* and target global properties over all samples.

Sampled Logic

Definition

A sample-level property φ holds for a signal *s* if it holds for all samples: $\forall n.\varphi(s[n])$.

Boundedness is a sample-level property!

Sampled Logic

Definition

A sample-level property φ holds for a signal *s* if it holds for all samples: $\forall n. \varphi(s[n])$.

Boundedness is a sample-level property!

Definition

Assume a program *f*, then we write $\{\varphi\}\ f\ \{\psi\}$ for "for all inputs satisfying φ, the output of *f* satisfies ψ."

Stability for smooth is written:

$$\{x \in [a, b]\} \text{ smooth } \{x \in [a, b]\}$$

Sampled Logic

$$\frac{\forall i_1, i_2, (\varphi_1(i_1) \wedge \varphi_1(i_2)) \implies \psi(i_1 + i_2)}{\{\varphi_1, \varphi_2\} + \{\psi\}}\,\textit{Prim}$$

$$\frac{\{\varphi\}\ f\ \{\theta\} \qquad \{\theta\}\ g\ \{\psi\}}{\{\varphi\}\ f : g\ \{\psi\}}\,\textit{Comp}$$

$$\frac{\models \psi(x_0) \qquad \{\theta, \varphi\}\ f\ \{\psi\} \qquad \{\psi\}\ g\ \{\theta\}}{\{\varphi\}\ f \sim g\ \{\psi\}}\,\textit{Feed}$$

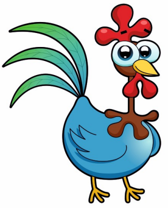
Stability Proof

$$
\dfrac{\dfrac{\Box}{\{I_{ab}\} \ast(1-c) \ \{I_{ab\bar{c}}\}} \qquad \dfrac{\dfrac{\Box}{\{I_{abc}, I_{ab\bar{c}}\} + \{I_{ab}\}} \qquad \dfrac{\Box}{\{I_{ab}\} \ast(c) \ \{I_{abc}\}}}{\{I_{ab\bar{c}}\} + \sim \ast(c) \ \{I_{ab}\}}}{\{i \in [a,b]\} \ast(1-c) : + \sim \ast(c) \ \{o \in [a,b]\}}
$$

with:

$$
\begin{aligned}
I_{ab}(x) &\equiv x \in [a, b] \\
I_{abc}(x) &\equiv x \in [a * c, b * c] \\
I_{ab\bar{c}}(x) &\equiv x \in [a * (1-c), b * (1-c)]
\end{aligned}
$$

Stability Proof



Conclusions

- Interesting exercise; we learned a lot!
- The full language is basically done.
- We need your help! Let us know what would be interesting to check!
- Most complaints about plugins cannot be solved by verification.
- We are investigating a slightly different approach.
- Working on linear systems theory and frequency domain properties.

A Taste of (formal) Sound Reasoning

A Tutorial

Emilio J. Gallego Arias, Pierre Jouvelot, Olivier Hermant

MINES ParisTech, PSL Research University, France

Linux Audio Conf 2015