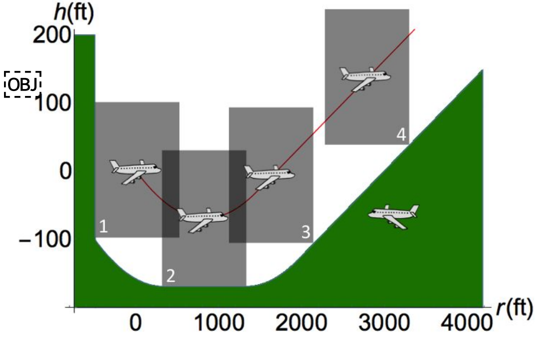
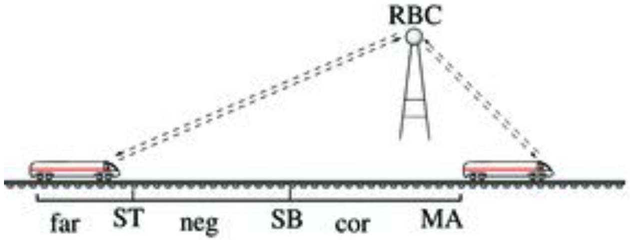
Formally Verified Differential Dynamic Logic (in Isabelle/HOL and Coq)

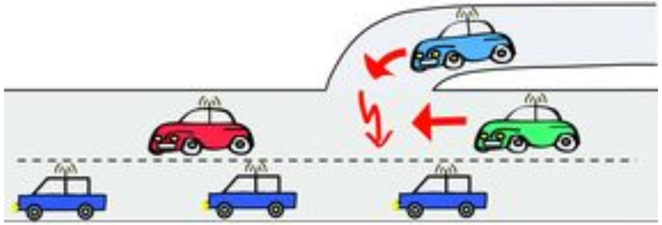
Brandon Bohrer[1], Vincent Rahli[2], Ivana Vukotic[2], Marcus Völp[2], André Platzer[1]
Thanks to: Johannes Hölzl, Fabian Immler, Tobias Nipkow, et. al.[3]
+ Coquelicot team

1 Carnegie Mellon University
2 University of Luxembourg
3 Technical University Munich

Safety-Critical Control Software is Everywhere







How can we design cyber-physical systems people can bet their lives on?

– Jeanette Wing

Formal Modeling and Verification Provide Safety

- *Differential Dynamic Logic* enables constructing and verifying hybrid models
- *KeYmaera X* theorem prover implements differential dynamic logic

  Example theorem: $v \geq 0 \ \& \ A() \geq 0 \rightarrow [a := A(); \{v' = a, x' = v \ \& \ true\}]v \geq 0$

Formal Modeling and Verification Provide Safety

- *Differential Dynamic Logic* enables constructing and verifying hybrid models
- *KeYmaera X* theorem prover implements differential dynamic logic

  Example theorem: $v \geq 0 \; \& \; A() \geq 0 \rightarrow [a := A(); \{v' = a, x' = v \; \& \; true\}]v \geq 0$

- Correct proof requires correct prover

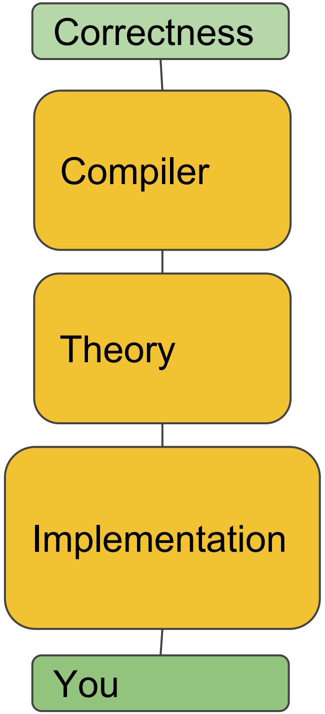
Formal Modeling and Verification Provide Safety

- *Differential Dynamic Logic* enables constructing and verifying hybrid models
- *KeYmaera X* theorem prover implements differential dynamic logic

  Example theorem: $v \geq 0 \,\&\, A() \geq 0 \rightarrow [a := A(); \{v' = a, x' = v \,\&\, true\}]v \geq 0$

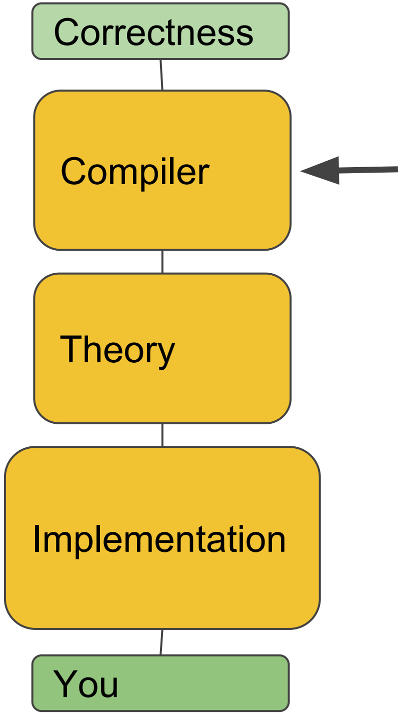
- Correct proof requires correct prover
- **Goal: <u>Ensure correctness of KeYmaera X</u>**

KeYmaera X Depends on a Big Stack



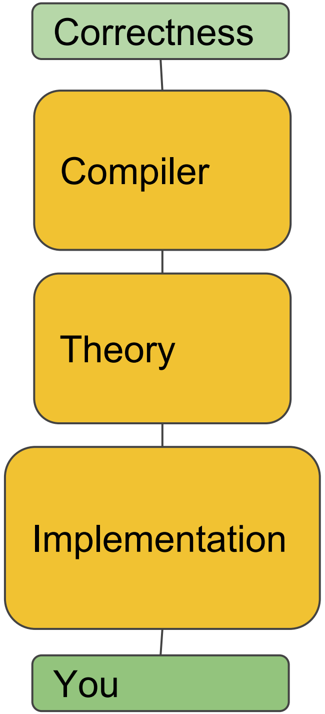
Pick part of the stack, try to improve it

KeYmaera X Depends on a Big Stack



Implemented in Scala: Large stack
(Not specific to hybrid systems proving)

KeYmaera X Depends on a Big Stack

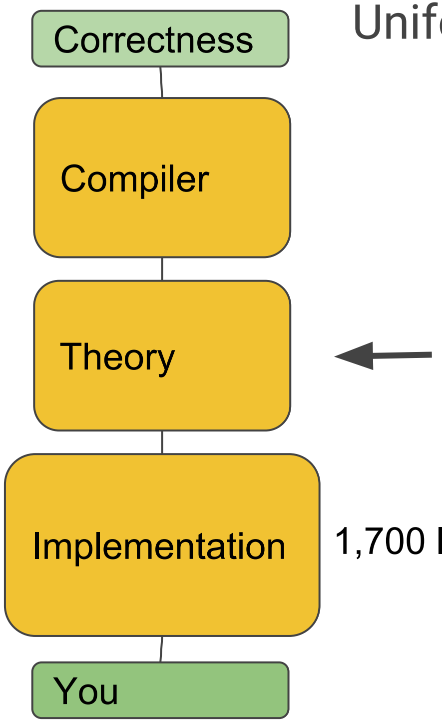


LCF Approach:

- Abstract type of proofs isolates critical code
- Majority of code untrusted

Prover	Lines of Code
HOL Light	400
KeYmaera 3	66,000
KeYmaera X	1,700 (2%)
Isabelle/Pure	8,000
Coq	20,000
NuPrl	15,000 + 50,000
PHAVer	30,000
SpaceEx	100,000

Small Core Requires Simple Proof Calculus



1,700 lines

Uniform Substitution (US) Simplifies Provers

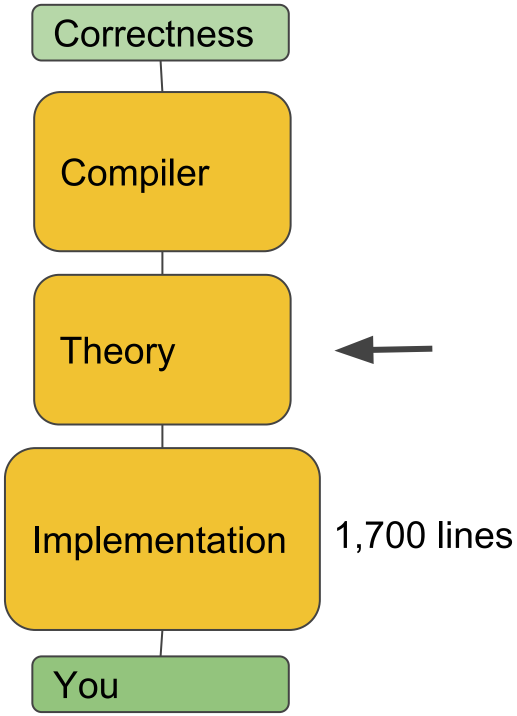
$[t := 0; t' = 1]\, t \geq 0 \leftrightarrow [t := 0][t' = 1] t \geq 0$ Axiom Instance

$[a ; b]P \leftrightarrow [a][b]P$ Axiom

Axioms are just data

Substitution $\sigma = \{a \mapsto t := 0,\ b \mapsto t' = 1,\ P \mapsto t \geq 0\}$

Uniform Substitution is Easy



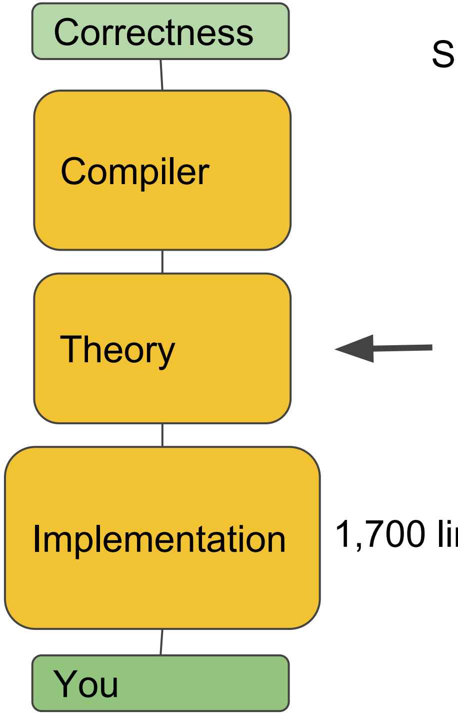
Substitution σ maps symbols to replacements
Replace recursively (Some cases **not** primitive recursive!)

Example: $\sigma=\{f \mapsto x + 1,\ p(y) \mapsto y \neq x\}$

$$\sigma(p(f))$$
$$= \sigma(p(x + 1))$$
$$= x + 1 \neq x$$

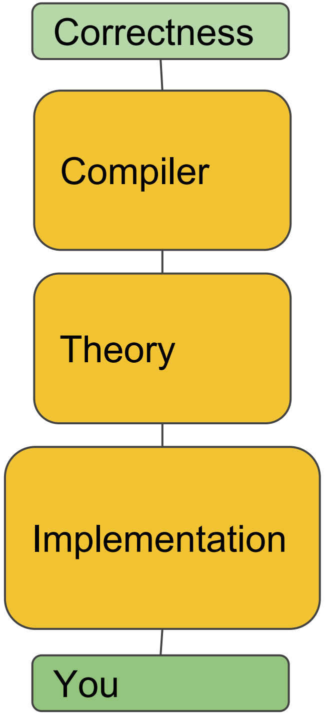
Uniform Substitution is Easy

Simple Proof Rule: $\text{US}\ \dfrac{\varphi}{\sigma(\varphi)}$



1,700 lines

Uniform Substitution is Easy



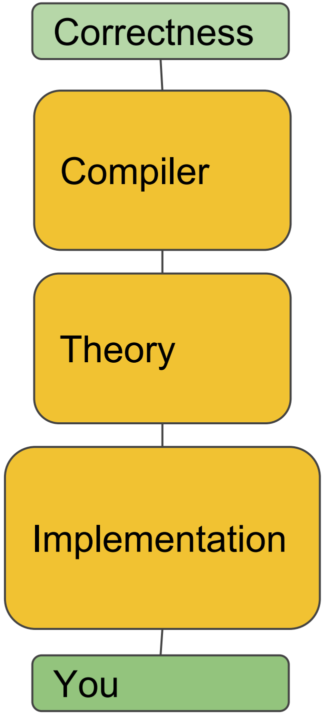
1,700 lines

Simple Proof Rule: $\text{US}\ \dfrac{\varphi}{\sigma(\varphi)}$

Example: $\text{US}\ \dfrac{[x := f]p(x) \leftrightarrow p(f)}{[x := \_]\_\_\_ \leftrightarrow \_\_\_}$

Substitution $\sigma=\{f \mapsto x + 1,\ p(y) \mapsto y \neq x\}$

Uniform Substitution is Easy



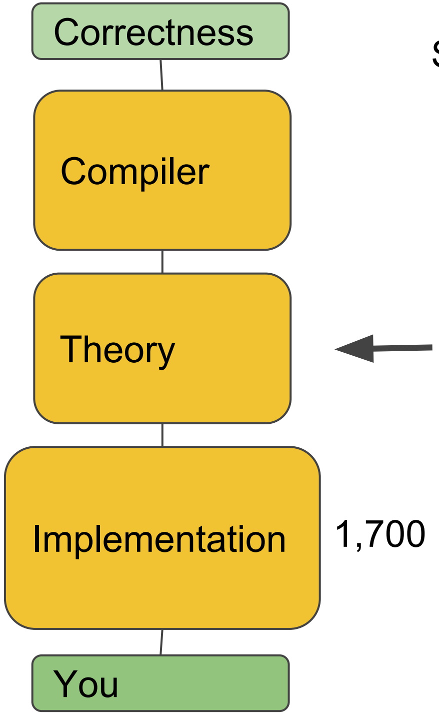
1,700 lines

Simple Proof Rule: $$\text{US}\ \frac{\varphi}{\sigma(\varphi)}$$

Example: $$\text{US}\ \frac{[x := f]p(x) \leftrightarrow p(f)}{[x := x + 1]\_\_\_ \leftrightarrow \_(x + 1)}$$

Substitution $\sigma = \{f \mapsto x + 1,\ p(y) \mapsto y \neq x\}$

Uniform Substitution is Easy



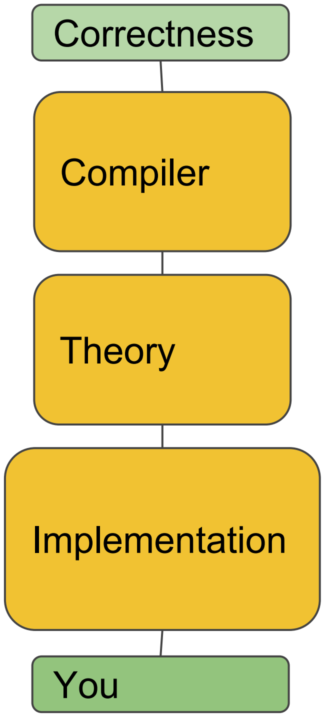
1,700 lines

Simple Proof Rule: $$\text{US}\ \frac{\varphi}{\sigma(\varphi)}$$

Example: $$\text{US}\ \frac{[x := f]p(x) \leftrightarrow p(f)}{[x := x + 1]x \neq x \leftrightarrow \_(x + 1)}$$

Substitution $\sigma = \{f \mapsto x + 1, p(y) \mapsto y \neq x\}$

Uniform Substitution is Easy



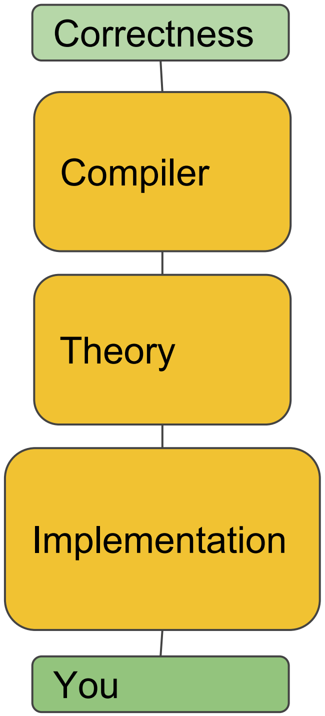
1,700 lines

Simple Proof Rule: $$\text{US}\ \frac{\varphi}{\sigma(\varphi)}$$

Example: $$\text{US}\ \frac{[x := f]p(x) \leftrightarrow p(f)}{[x := x + 1]x \neq x \leftrightarrow x + 1 \neq x}$$

Substitution $\sigma=\{f \mapsto x + 1,\ p(y) \mapsto y \neq x\}$

Uniform Substitution is Hard



Simple Proof Rule: $$\text{US}\ \frac{\varphi}{\sigma(\varphi)}$$

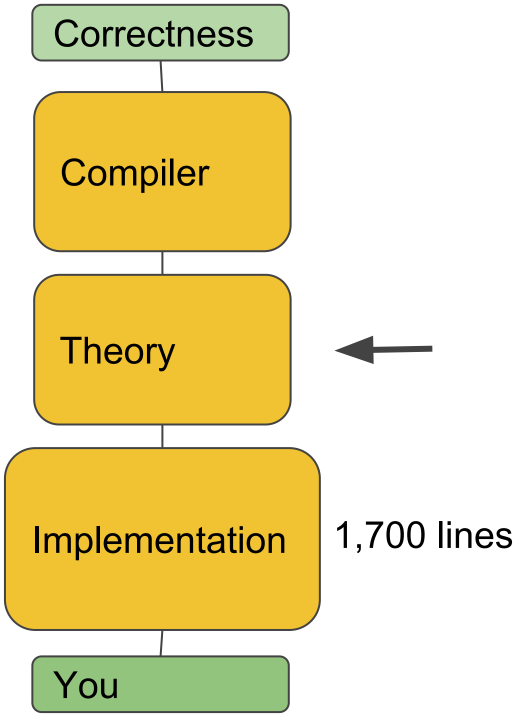
Example: $$\text{US}\ \cfrac{[x := f]p(x) \leftrightarrow p(f)}{\cfrac{[x := x + 1]x \neq x \leftrightarrow x + 1 \neq x}{\text{False} \leftrightarrow \text{True}}}$$

1,700 lines

Substitution $\sigma=\{f\mapsto x + 1,\ p(y)\mapsto y \neq x\}$

Naive Substitution: UNSOUND!

Uniform Substitution is Hard



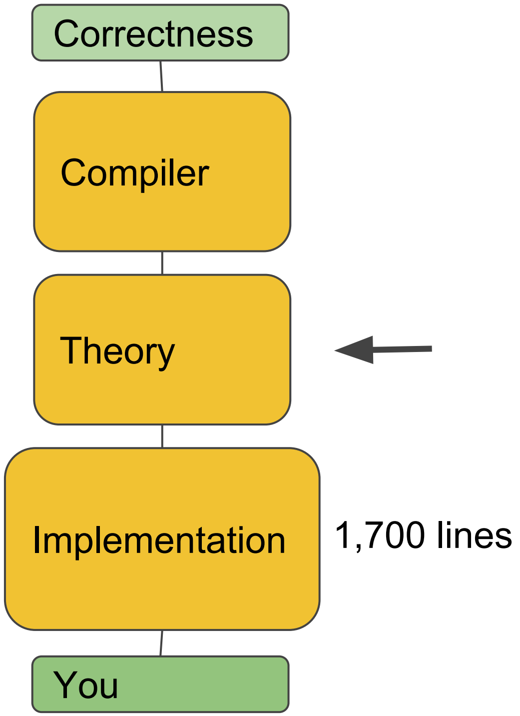
Admissibility checks determine when substitution is sound

Example: $\sigma([\alpha]\varphi) = [\sigma(\alpha)]\sigma(\varphi)$
If $FV(\sigma) \cap BV(\alpha) = \varnothing$

$FV(\sigma) = \{x\}$
$BV(x := f) = \{x\}$

$FV(\sigma) \cap BV(x := f) = \{x\} \neq \varnothing$
Clash Detected

Justify the Theory with Formal Verification

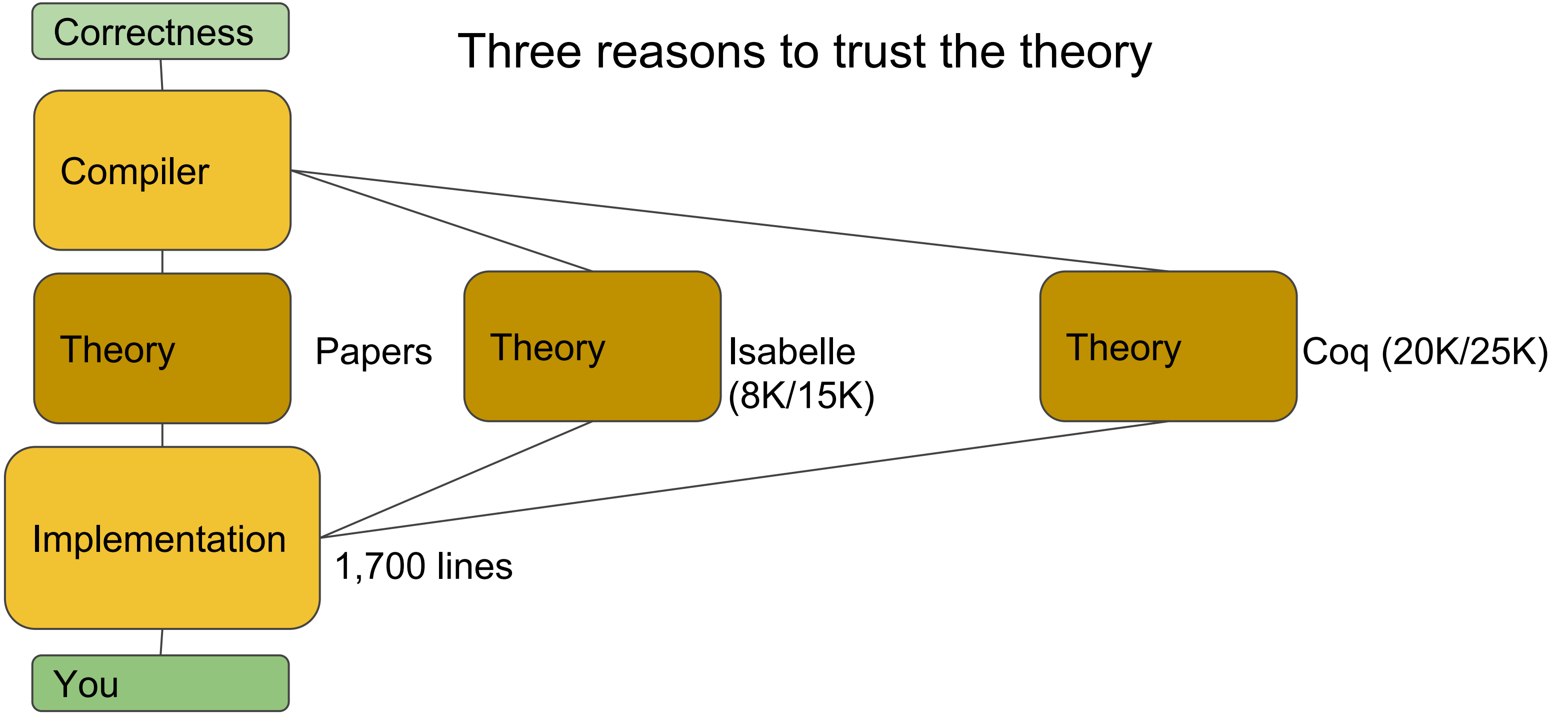


Related Work:

Verified Theorem Prover	Verified Using
HOL Light	Self*-verified
HOL Light	HOL 4
Milawa	HOL 4
Subset of Coq	Coq
Theory of NuPRL	Coq

Let's do it for hybrid systems!

Verification: Independent, Trustworthy Justification



Three reasons to trust the theory

Formalization of dL in Isabelle/HOL

(Also applies to Coq version)

What’s in the Formalization?

- **Syntax** ←
- Dynamics
- Axioms
- Statics
- Substitution
- Renaming
- Proof Checker

$\alpha, \beta ::=$
a
$\mid x := \theta$
$\mid x' := \theta$
$\mid ?\varphi$
$\mid (ODE \;\&\; \varphi)$
$\mid (\alpha \cup \beta)$
$\mid (\alpha; \beta)$
$\mid \alpha^*$

$\Rightarrow$

```
datatype hp =
  Pvar        id
| Assign      id trm
| DiffAssign  id trm
| Test        formula
| EvolveODE   ODE formula
| Choice      hp hp
| Sequence    hp hp
| Loop        hp
```

Syntax ⇒ Datatypes

What's in the Formalization?

- Syntax
- **Dynamics** ←
- Axioms
- Statics
- Substitution
- Renaming
- Proof Checker

[[α ∪ β]]Iv = [[α]]Iv ∪ [[β]]Iv

....

↓

fun HPsem :: "interp → hp → (state * state) set"
where
| "HPsem I (Pvar p) = Programs I p"
| "HPsem I (Assign x t) = {(v, ω). ω = v (x:= (θsem I t v))}"
| "HPsem I (DiffAssign x t) = {(v, ω). ω = v (x':= (θsem I t v))}"
| "HPsem I (Test φ) = {(v, v) | v. v ∈ fml_sem I φ}"
| "HPsem I (Choice α β) = HPsem I α ∪ HPsem I β"
| "HPsem I (Sequence α β) = HPsem I α O HPsem I β"
| "HPsem I (Loop α) = (HPsem I α)*"

Semantic Functions ⇒ Isabelle Functions

What's in the Formalization?

- Syntax
- Dynamics
- **Axioms** ←
- Statics
- Substitution
- Renaming
- Proof Checker

p() -> [a]p()

↓

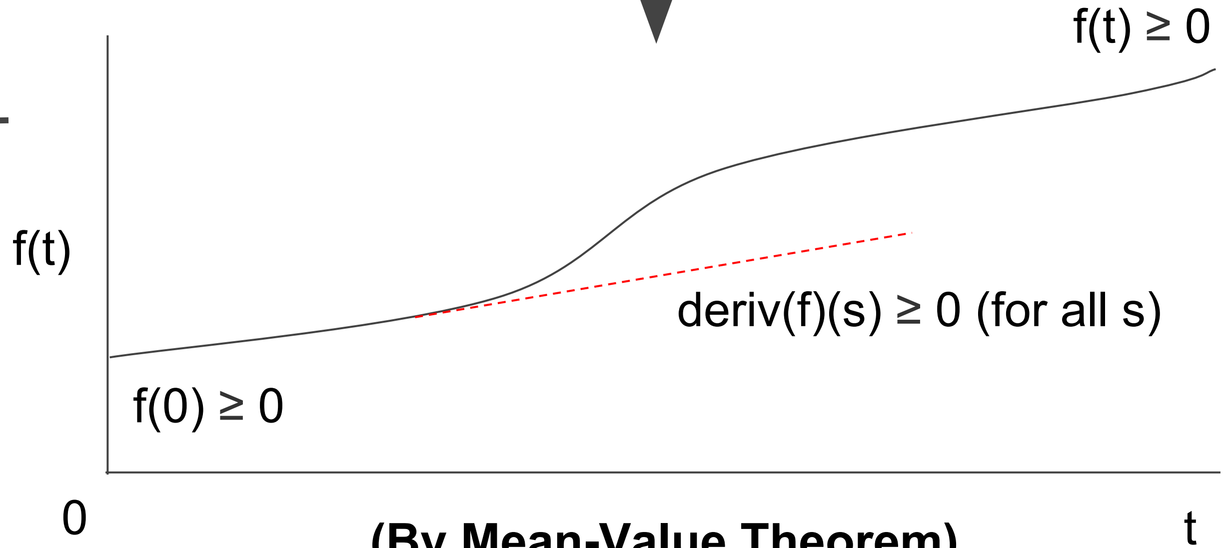
```
definition Vaxiom ::formula
  where "Vaxiom ≡ ($φ p ()) → ([[$α a]]($φ p ()))"

theorem V_valid: "valid Vaxiom"
  by(auto simp: valid_def Vaxiom_def)
```

Axiom ⇒ Definition + Validity Lemma

Differential Induction is Harder

- Syntax
- Dynamics
- **Axioms**
- Statics
- Substitution
- Renaming
- Proof Checker

$$f \geq 0 \rightarrow [x' = \theta \ \& \ \varphi]\mathrm{deriv}(f) \geq 0 \rightarrow [x' = \theta \ \& \ \varphi]f \geq 0$$



(By Mean-Value Theorem)

Static Semantics Enable Substitution

- Syntax
- Dynamics
- Axioms
- Statics ←
- Substitution
- Renaming
- Proof Checker

Compute FV(e), BV(e) by structural recursion

Theorem(Coincidence)**:** Expressions depend only free variables

Theorem(Bound Effect)**:** Programs affect only bound variables

Definitions verified, not trusted

Static Semantics Enable Substitution

Syntax

Dynamics

Axioms

Statics

Substitution ←

Renaming

Proof Checker

$$\sigma(\theta \geq \eta) = \sigma(\theta) \geq \sigma(\eta)$$

$$\sigma(\forall x\, \phi) = \forall x\, \sigma(\phi) \quad \text{if } \sigma\ \{x\}\text{-admissible for } \phi$$

$$\sigma([\alpha]\phi) = [\sigma(\alpha)]\sigma(\phi) \quad \text{if } \sigma\ BV(\sigma(\alpha))\text{-admissible for } \phi$$

```
primrec FOTsubst::"trm ⇒ FOsubst ⇒ trm"
primrec Tsubst::"trm ⇒ subst ⇒ trm"
```

```
inductive FOTadmit :: "FOsubst ⇒ trm ⇒ bool"
inductive Tadmit :: "subst ⇒ trm ⇒ bool"
```

Formalize primitive recursive variant instead
(First-order substitution for arguments)

Static Semantics Enable Substitution

- Syntax
- Dynamics
- Axioms
- Statics
- **Substitution** ←
- Renaming
- Proof Checker

$$\text{US}\ \frac{\varphi}{\sigma(\varphi)}\ (\sigma \text{ admissible for } \varphi)$$

↓

```
lemma subst_fml_valid:
assumes valid:"valid φ"
assumes Fadmit:"Fadmit σ φ"
shows "valid (Fsubst φ σ)"
```

New: Uniform Renaming

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- **Renaming** ←
- Proof Checker

$$\text{UR}\ \frac{\varphi}{\varphi\{x \leftrightarrow y\}}$$

↓

```
lemma URename_sound:
assumes "valid φ"
shows "valid (FUrename x y φ)"
```

No Admissibility!

New: Bound Renaming Renames Destinations

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- **Renaming**
- Proof Checker

$$\text{BR}\ \frac{[x{:=}\ \theta]\varphi}{[y{:=}\ \theta]\varphi\{x \leftrightarrow y\}}$$

```
lemma BRename_sound:
assumes valid:"valid ([[Assign x θ]]φ)"
assumes FVF:"{y, y'} ∩ FVF φ = {}"
shows "valid([[Assign y θ]]FUrename x y φ)"
```

Yes Admissibility!

KeYmaera X Bound Rename Unsound

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- **Renaming**
- Proof Checker

```
     [x:= x'](x=x')
BR  ────────────────  BUG!
     [y:= x'](y=y')
```

```
lemma BRename_sound:
assumes valid:"valid ([[Assign x θ]]φ)"
assumes FVF:"{y, y', x'} ∩ FV φ = {}"
shows "valid([[Assign y θ]]FUrename x y φ)"
```

Proof Checker - Step Toward Implementation

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- Renaming
- **Proof Checker**

KeYmaera X

$\Gamma \vdash \Delta$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}$$

```
fun seq2fml :: " sequent ⇒ formula"
where
  "seq2fml (ante,succ) = Implies (foldr And ante TT) (foldr Or succ FF)"

type pf = "sequent * derivation"

type rule = "sequent list * sequent"
```

Examples Validate Proof Checker

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- Renaming
- **Proof Checker** ←

Example 1: A minimal hybrid system example (~ 100 proof steps)

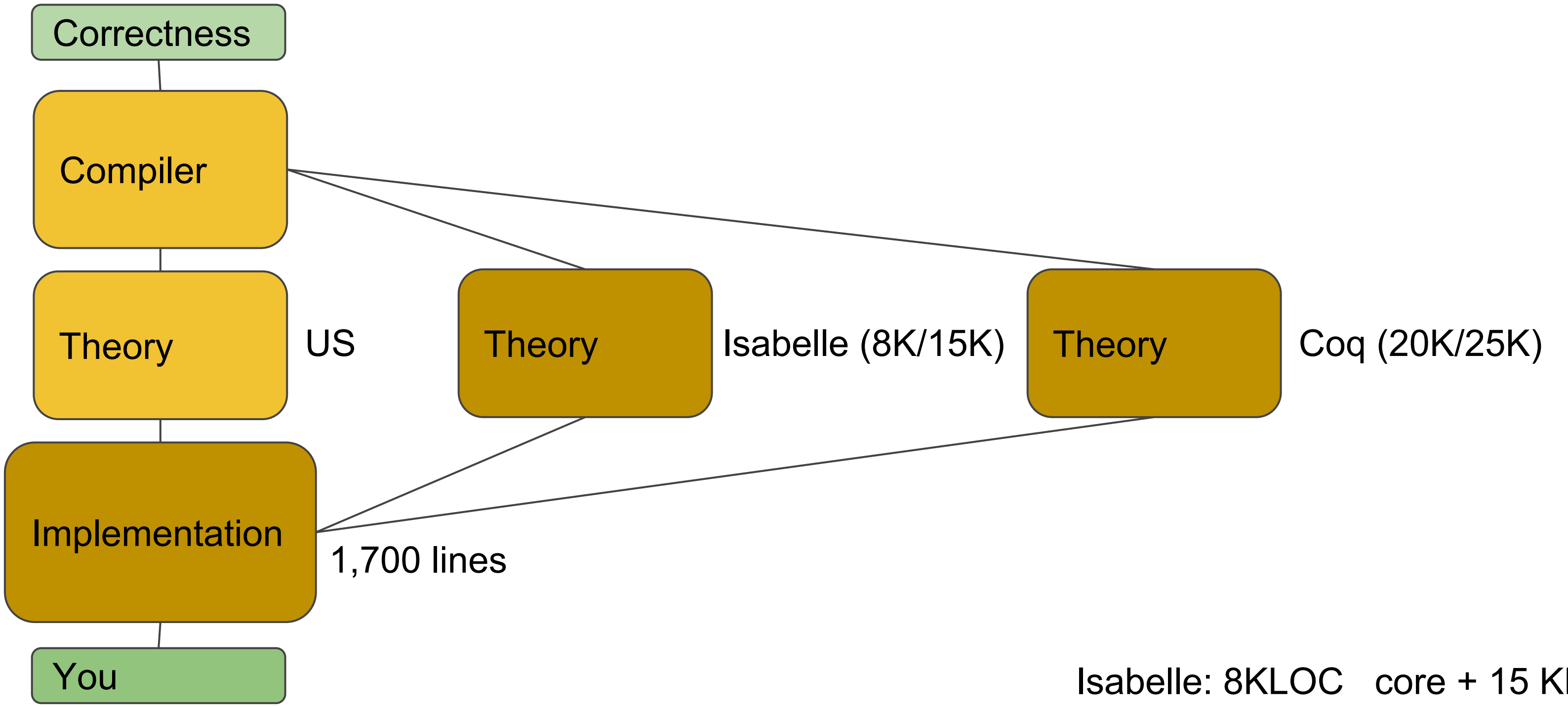
$$v \geq 0 \ \&\ A() \geq 0 \rightarrow [\{v' = A(),\ x' = v\ \&\ true\}]v \geq 0$$

Example 2: A derived case of differential induction (~ 60 proof steps)

$$\frac{P \rightarrow [a]P' \rightarrow [a]P \qquad Q \rightarrow [a]Q' \rightarrow [a]Q}{P\ \&\ Q \rightarrow [a](P'\ \&\ Q') \rightarrow [a](P\ \&\ Q)}$$

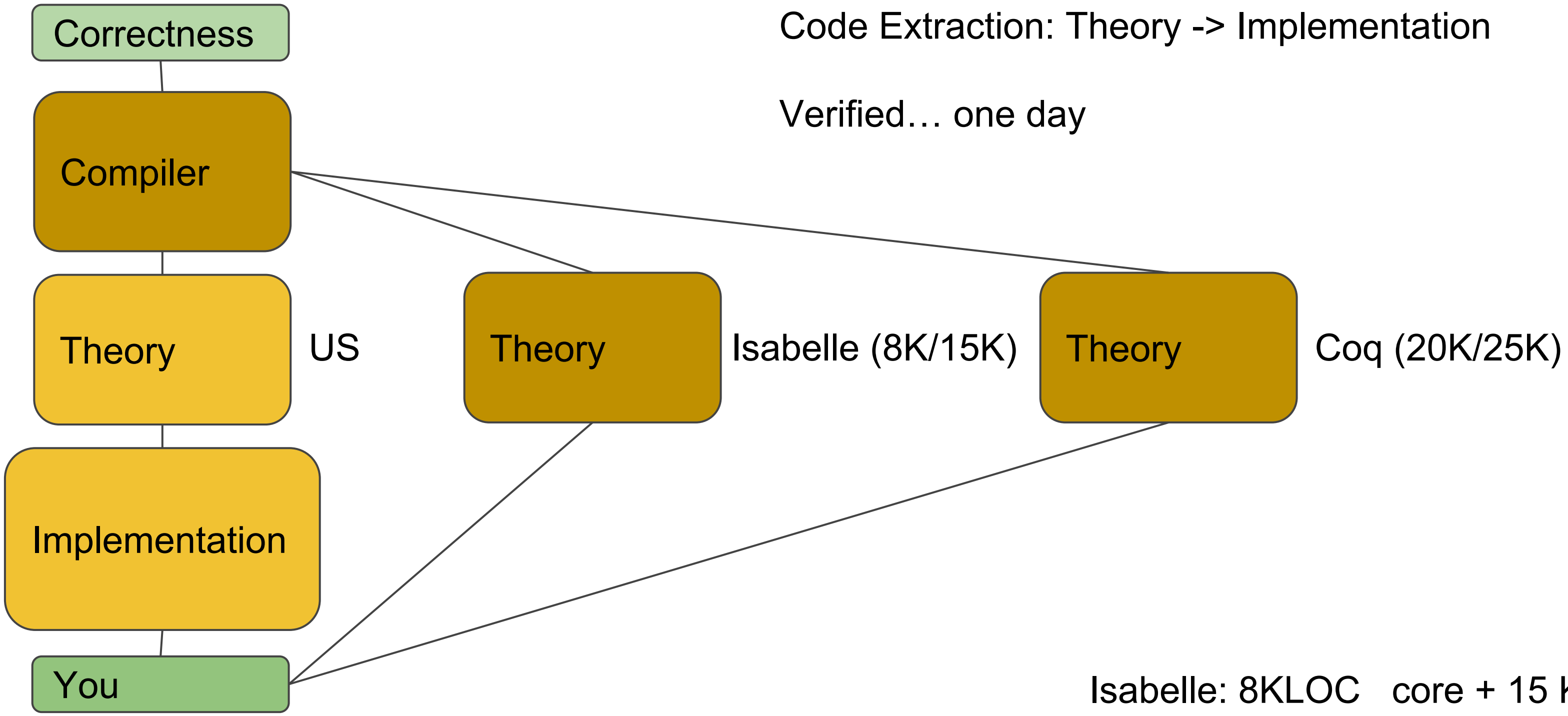
Future Work

Future Work: Further Reduce Stack



Isabelle: 8KLOC core + 15 KLOC proof
Coq: 20KLOC core + 25 KLOC proof

Future Work: Further Reduce Stack

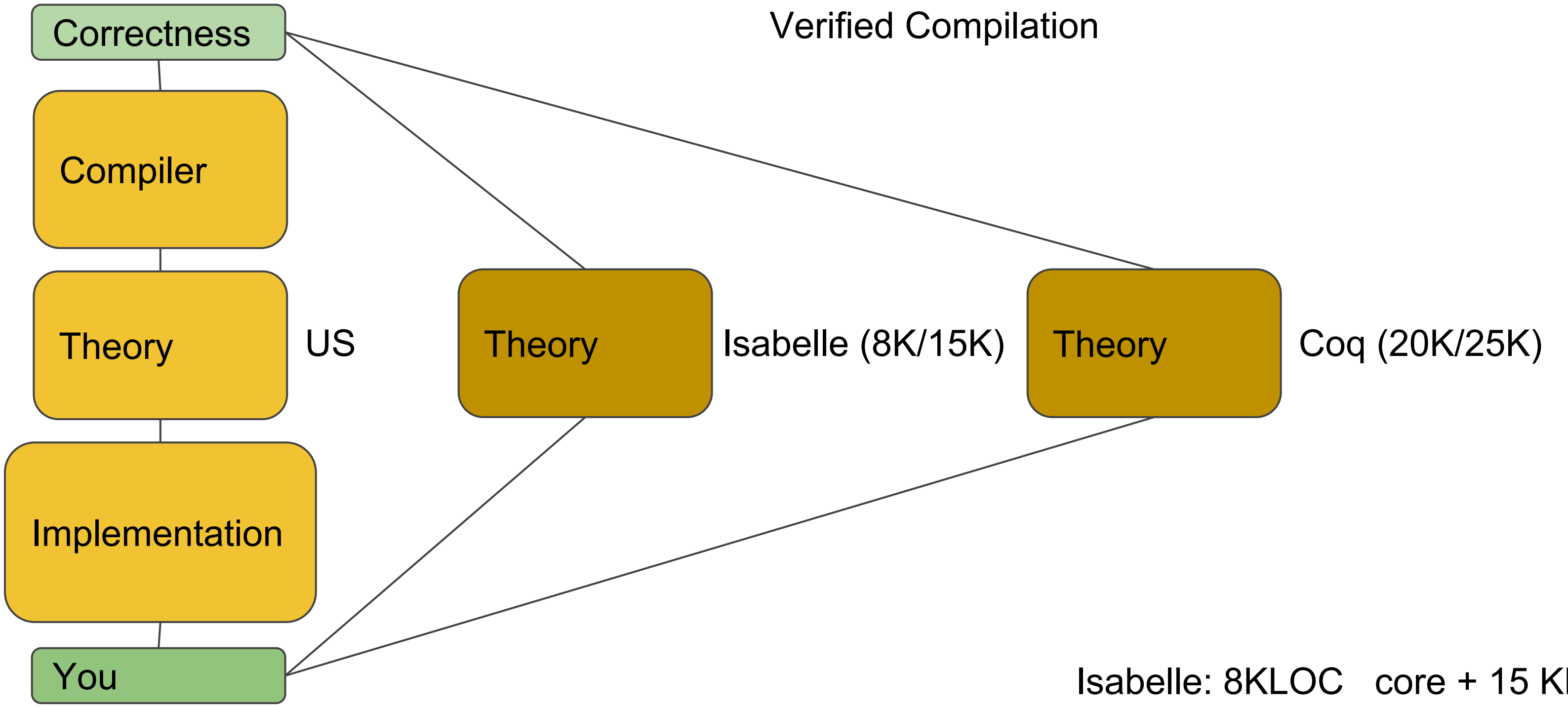


Code Extraction: Theory -> Implementation

Verified… one day

Isabelle: 8KLOC core + 15 KLOC proof
Coq: 20KLOC core + 25 KLOC proof

Future Work: Further Reduce Stack



Isabelle: 8KLOC core + 15 KLOC proof
Coq: 20KLOC core + 25 KLOC proof

Future Work: KeYmaera X Integration

- Syntax
- Dynamics
- Axioms
- Statics
- Substitution
- Renaming
- Proof Checker

NEXT?

KeYmaera X Integration

- Extract prover core from proof
- Export proof terms from KeYmaera X
- Import and check proof
- Verify theorems of real arithmetic
  - E.g. $\forall x.\ x^2 \geq 0$

Questions?

	CADE'15 [38]	JAR'16 [39]	KeYmaera X	**Isabelle**	**Coq**
systems of ODEs	[36]		✓	✓	✓
multiple argument functions	✗	✗	✓	✓	✓
infinite number of identifiers	✓	✓	✓	✗	✓
explicit set representation	✗	✗	✓	✗	✓
higher-order differentials	✗	✓	✗	✗	✓
uniform substitution defns.	1	1	1	2	2
uniform variable renaming	✗	✗	✓	✓	✓
bound variable renaming	✗	✗	✓	✓	✓
sequent calculus	[36]		✓	(✓)	(✓)
DG	✓	✓	✓	✓	✗
DG+DE: ODE systems	[37]		✓	✗	✗

Comparison of Formalizations/Presentations