Calculating Certified Compilers for Non-Deterministic Languages

Patrick Bahr

University of Copenhagen
paba@diku.dk

MPC 2015

Overview

Goal: derive compiler from high-level specification

- by calculation
- formal
- systematic

Overview

Goal: derive compiler from high-level specification

- by calculation
- formal
- systematic

This paper

- Challenges for non-deterministic languages
- proof system for non-determinism
- proof automation $\rightsquigarrow$ Coq

Calculating Correct Compilers

It works like this:

1. write semantics of object language
2. formulate compiler correctness property
3. prove correctness property

Calculating Correct Compilers

It works like this:

1. write semantics of object language
2. formulate compiler correctness property
3. prove correctness property

Output

- compiler implementation
- target language + virtual machine
- compiler correctness proof

Toy Example: Arithmetic Expressions

Syntax

data *Expr* = *Val Int* | *Add Expr Expr*

Toy Example: Arithmetic Expressions

Syntax

$$\textbf{data}\ \mathit{Expr} = \mathit{Val}\ \mathit{Int} \mid \mathit{Add}\ \mathit{Expr}\ \mathit{Expr}$$

Semantics

$$\frac{}{\mathit{Val}\ n \ \Downarrow\ n}\ \textsc{Val} \qquad \frac{e_1 \ \Downarrow\ n_1 \qquad e_2 \ \Downarrow\ n_2}{\mathit{Add}\ e_1\ e_2 \ \Downarrow\ (n_1 + n_2)}\ \textsc{Add}$$

The Setup

Stack-based VM

$$\begin{aligned}&\textbf{type}\ \mathit{Stack} = [\mathit{Int}]\\&\textbf{type}\ \mathit{Conf}\ \ = (\mathit{Code}, \mathit{Stack})\end{aligned}$$

$$\Longrightarrow \subseteq \mathit{Conf} \times \mathit{Conf}$$

The Setup

Stack-based VM

$$\textbf{type}\ \mathit{Stack} = [\mathit{Int}]$$
$$\textbf{type}\ \mathit{Conf} = (\mathit{Code}, \mathit{Stack})$$

$$\Longrightarrow\ \subseteq \mathit{Conf} \times \mathit{Conf}$$

Code is a sequence of instructions

$$\textbf{type}\ \mathit{Code} = [\mathit{Instr}]$$
$$\textbf{data}\ \mathit{Instr} = ??$$

The Setup

Stack-based VM

$$\textbf{type}\ \mathit{Stack} = [\mathit{Int}]$$
$$\textbf{type}\ \mathit{Conf} = (\mathit{Code}, \mathit{Stack})$$

$$\Longrightarrow \subseteq \mathit{Conf} \times \mathit{Conf}$$

Code is a sequence of instructions

$$\textbf{type}\ \mathit{Code} = [\mathit{Instr}]$$
$$\textbf{data}\ \mathit{Instr} = ??$$

compiler is formulated in CPS

$$\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code}$$
$$\mathit{comp}' = ??$$
$$\mathit{comp} :: \mathit{Expr} \to \mathit{Code}$$
$$\mathit{comp}\ e = \mathit{comp}'\ e\ [\,]$$

Step 2: Compiler Correctness Property

$$e \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}\ e,\quad [\,]) \overset{*}{\Longrightarrow} ([\,],[n])$$

Step 2: Compiler Correctness Property

$$e \Downarrow n \qquad \text{implies} \qquad (comp' e\ c, [\,]) \stackrel{*}{\Longrightarrow} (c,\ [n])$$

Step 2: Compiler Correctness Property

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\, e\ c, s) \stackrel{*}{\Longrightarrow} (c,\ n : s)$$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp' \ e \ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{array}{lcl} (c, n : s) & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \end{array}$$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{array}{rcl} (c, n : s) & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & (c', s) \end{array}$$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{array}{rcl} (c, n : s) & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & \ldots \\ & \overset{*}{\Longleftarrow} & (c', s) \end{array}$$

Result:

1. definition $comp'\ e\ c = c'$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \stackrel{*}{\Longrightarrow} (c, n : s)$$

$$\begin{array}{rcl} (c, n : s) & \stackrel{*}{\Longleftarrow} & \ldots \\ & \stackrel{*}{\Longleftarrow} & \ldots \\ & \stackrel{*}{\Longleftarrow} & \ldots \\ & \stackrel{*}{\Longleftarrow} & (c', s) \end{array}$$

Result:

1. definition $comp'\ e\ c = c'$
2. new rules for $\Longrightarrow$

Step 3: Prove correctness property

We prove by induction on e:

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{aligned} (c, n : s) \quad & \overset{*}{\Longleftarrow} \quad \ldots \\ & \overset{*}{\Longleftarrow} \quad \ldots \\ & \overset{*}{\Longleftarrow} \quad \ldots \\ & \overset{*}{\Longleftarrow} \quad (c', s) \end{aligned}$$

Result:

1. definition $comp'\ e\ c = c'$
2. new rules for $\Longrightarrow$
3. new instructions

Let's Calculate!

$$e \quad \Downarrow n \qquad \text{implies} \qquad (comp' \quad e \qquad c, s) \stackrel{*}{\Longrightarrow} (c, n : s)$$

Let's Calculate!

$$Val\ m \Downarrow n \qquad \text{implies} \qquad (comp'\,(Val\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

Let's Calculate!

$$Val\ m \Downarrow n \qquad \text{implies} \qquad (comp'\,(Val\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$(c, n : s)$$

Let's Calculate!

$$\mathit{Val}\ m \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}'\,(\mathit{Val}\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{aligned} & (c, n : s) \\ = & \quad \{\ \text{by } \textsc{Val}\ \} \\ & (c, m : s) \end{aligned}$$

Let's Calculate!

$Val\ m \Downarrow n$ implies $(comp'\,(Val\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$

$$\frac{}{Val\ m \Downarrow m}\ \textsc{Val}$$

$(c, n : s)$

$=$ { by $\textsc{Val}$ }

$(c, m : s)$

Let's Calculate!

$$\mathit{Val}\ m \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}'\,(\mathit{Val}\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{aligned}
& (c, n : s) \\
= \quad & \{ \text{ by } \textsc{Val} \ \} \\
& (c, m : s) \\
\Longleftarrow \quad & \{ \text{ define } \textsc{VM-Push} \ \} \\
& (\mathit{PUSH}\ m : c, s)
\end{aligned}$$

Let's Calculate!

$$\mathit{Val}\ m \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}'\,(\mathit{Val}\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$(c, n : s)$$

$=$ { by VAL

$(\mathit{PUSH}\ m : c, s) \Longrightarrow (c, m : s)$

$$(c, m : s)$$

$\Longleftarrow$ { define VM-PUSH }

$$(\mathit{PUSH}\ m : c, s)$$

Let's Calculate!

$$Val\ m \Downarrow n \qquad \text{implies} \qquad (comp'\,(Val\ m)\,c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

$$\begin{array}{ll}
& (c, n : s) \\
= & \{ \text{ by } \textsc{Val} \ \} \\
& (c, m : s) \\
\Longleftarrow & \{ \text{ define } \text{VM-}\textsc{Push} \ \} \\
& (PUSH\ m : c, s) \\
= & \{ \text{ define } comp'\ (Val\ m)\ c = PUSH\ m : c \ \} \\
& (comp'\ (Val\ m)\ c, s)
\end{array}$$

Let's Calculate!

$$\mathit{Add}\ e_1\ e_2 \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c, s) \overset{*}{\Longrightarrow} (c, n{:}s)$$

Induction Hypotheses

$$e_i \Downarrow n_i \qquad \text{implies} \qquad (\mathit{comp}'\ e_i\ c', s') \overset{*}{\Longrightarrow} (c', n_i : s')$$

Let's Calculate!

$$\textit{Add}\ e_1\ e_2 \Downarrow n \qquad \text{implies} \qquad (\textit{comp}'\ (\textit{Add}\ e_1\ e_2)\ c, s) \overset{*}{\Longrightarrow} (c, n{:}s)$$

$$(c, n : s)$$

Induction Hypotheses

$$e_i \Downarrow n_i \qquad \text{implies} \qquad (\textit{comp}'\ e_i\ c', s') \overset{*}{\Longrightarrow} (c', n_i : s')$$

Let's Calculate!

$Add\ e_1\ e_2 \Downarrow n$ ... $, s) \stackrel{*}{\Longrightarrow} (c, n{:}s)$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{Add\ e_1\ e_2 \Downarrow (n_1 + n_2)}\ \textsc{Add}$$

$$\begin{array}{ll} & (c, n : s) \\ = & \{\ \text{by } \textsc{Add}\ \} \\ & (c, (n_1 + n_2) : s) \end{array}$$

Induction Hypotheses

$e_i \Downarrow n_i$ implies $(comp'\ e_i\ c', s') \stackrel{*}{\Longrightarrow} (c', n_i : s')$

Let's Calculate!

$$Add\ e_1\ e_2 \Downarrow n \quad \text{implies} \quad (comp'\ (Add\ e_1\ e_2)\ c, s) \overset{*}{\Longrightarrow} (c, n{:}s)$$

$$
\begin{aligned}
& (c, n : s) \\
= \quad & \{ \text{ by A}\text{D}\ldots \} \\
& (c, (n_1 + n_2) : s) \\
\Longleftarrow \quad & \{ \text{ define VM-ADD } \} \\
& (ADD : c, n_2 : n_1 : s)
\end{aligned}
$$

$$(ADD : c, n_2 : n_1 : s) \Longrightarrow (c, (n_1 + n_2) : s)$$

Induction Hypotheses

$$e_i \Downarrow n_i \quad \text{implies} \quad (comp'\ e_i\ c', s') \overset{*}{\Longrightarrow} (c', n_i : s')$$

Let's Calculate!

$$Add\ e_1\ e_2 \Downarrow n \quad \text{implies} \quad (comp'\ (Add\ e_1\ e_2)\ c, s) \overset{*}{\Longrightarrow} (c, n{:}s)$$

$$\begin{aligned}
& (c, n : s) \\
= \quad & \{ \text{ by } \textsc{Add} \ \} \\
& (c, (n_1 + n_2) : s) \\
\Longleftarrow \quad & \{ \text{ define } \text{VM-}\textsc{Add} \ \} \\
& (ADD : c, n_2 : n_1 : s) \\
\overset{*}{\Longleftarrow} \quad & \{ \text{ induction hypothesis for } e_2 \ \} \\
& (comp'\ e_2\ (ADD : c), n_1 : s) \\
\overset{*}{\Longleftarrow} \quad & \{ \text{ induction hypothesis for } e_1 \ \} \\
& (comp'\ e_1\ (comp'\ e_2\ (ADD : c)), s)
\end{aligned}$$

Induction Hypotheses

$$e_i \Downarrow n_i \quad \text{implies} \quad (comp'\ e_i\ c', s') \overset{*}{\Longrightarrow} (c', n_i : s')$$

Let's Calculate!

$$Add\ e_1\ e_2 \Downarrow n \quad \text{implies} \quad (comp'\ (Add\ e_1\ e_2)\ c, s) \overset{*}{\Longrightarrow} (c, n{:}s)$$

$$
\begin{aligned}
& (c, n : s) \\
= \quad & \{ \text{ by } \textsc{Add} \ \} \\
& (c, (n_1 + n_2) : s) \\
\Longleftarrow \quad & \{ \text{ define } \text{VM-}\textsc{Add} \ \} \\
& (ADD : c, n_2 : n_1 : s) \\
\overset{*}{\Longleftarrow} \quad & \{ \text{ induction hypothesis for } e_2 \ \} \\
& (comp'\ e_2\ (ADD : c), n_1 : s) \\
\overset{*}{\Longleftarrow} \quad & \{ \text{ induction hypothesis for } e_1 \ \} \\
& (comp'\ e_1\ (comp'\ e_2\ (ADD : c)), s) \\
= \quad & \{ \text{ definition of } comp'\ (Add\ e_1\ e_2)\ c = \ldots \ \} \\
& (comp'\ (Add\ e_1\ e_2)\ c, s)
\end{aligned}
$$

Induction Hypotheses

$$e_i \Downarrow n_i \quad \text{implies} \quad (comp'\ e_i\ c', s') \overset{*}{\Longrightarrow} (c', n_i : s')$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD}$$

$$\begin{aligned}
&\mathit{comp}' :: \mathit{Expr} \rightarrow \mathit{Code} \rightarrow \mathit{Code} \\
&\mathit{comp}'\ (\mathit{Val}\ m)\qquad\ \ c = \mathit{PUSH}\ m : c \\
&\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c))
\end{aligned}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD}$$

$$\begin{aligned}
&\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code} \\
&\mathit{comp}'\ (\mathit{Val}\ m)\qquad\ \ c = \mathit{PUSH}\ m : c \\
&\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c))
\end{aligned}$$

$$(\mathit{PUSH}\ m : c, s) \Longrightarrow (c, m : s) \qquad \text{(VM-Push)}$$

$$(\mathit{ADD} : c, n_2 : n_1 : s) \Longrightarrow (c, (n_1 + n_2) : s) \qquad \text{(VM-Add)}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD}$$

$$\begin{aligned}
&\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code} \\
&\mathit{comp}'\ (\mathit{Val}\ m)\qquad\ \ c = \mathit{PUSH}\ m : c \\
&\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c))
\end{aligned}$$

$$(\mathit{PUSH}\ m : c, s) \Longrightarrow (c, m : s) \qquad \text{(VM-Push)}$$

$$(\mathit{ADD} : c, n_2 : n_1 : s) \Longrightarrow (c, (n_1 + n_2) : s) \qquad \text{(VM-Add)}$$

$$e \Downarrow n \qquad \text{implies} \qquad (\mathit{comp}'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

Non-determinism

What about non-determinism?

Non-determinism

What about non-determinism?

Compiler "correctness" property

$$e \Downarrow n \qquad \text{implies} \qquad (comp' \; e \; c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

Non-determinism

What about non-determinism?

Compiler "correctness" property

$$e \Downarrow n \qquad \text{implies} \qquad (comp' \; e \; c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

Non-determinism

What about non-determinism?

Compiler "correctness" property

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

This is only the completeness property!

What about soundness?

Non-determinism

What about non-determinism?

Compiler "correctness" property

$$e \Downarrow n \qquad \text{implies} \qquad (comp'\ e\ c, s) \overset{*}{\Longrightarrow} (c, n : s)$$

This is only the completeness property!

What about soundness?

We need to prove soundness and completeness in one go!

A Holistic View

"$\stackrel{*}{\Longrightarrow}$" provides focused view

A Holistic View

"$\stackrel{*}{\Longrightarrow}$" provides focused view

- one execution path
- only successful executions

A Holistic View

"$\stackrel{*}{\Longrightarrow}$" provides focused view

"$\Rrightarrow$" provides holistic view

- one execution path
- only successful executions

A Holistic View

"$\overset{*}{\Longrightarrow}$" provides focused view

- one execution path
- only successful executions

"$\Rrightarrow$" provides holistic view

Characteristics

1. $\Rrightarrow$ works on sets of configurations
2. $\Rrightarrow$ is "exhaustive" $\rightsquigarrow$ soundness property
3. proof rules for $\Rrightarrow$

A Holistic View

"$\overset{*}{\Longrightarrow}$" provides focused view

- one execution path
- only successful executions

"$\Rightarrow\!\!\rhd$" provides holistic view

Characteristics

1. $\Rightarrow\!\!\rhd$ works on sets of configurations
2. $\Rightarrow\!\!\rhd$ is "exhaustive" $\rightsquigarrow$ soundness property
3. proof rules for $\Rightarrow\!\!\rhd$

(items 1–2:) Hutton & Wright JFP 2007

Definition of $\Rrightarrow$

$S \Rrightarrow T$ iff

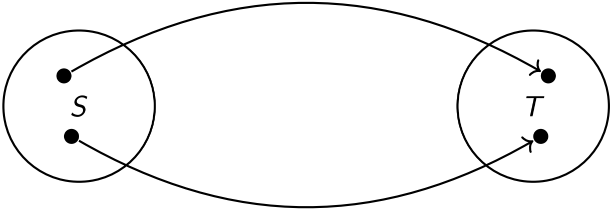
1. $\forall s \in S \quad \exists t \in T : \quad s \stackrel{*}{\Longrightarrow} t$, and

2. $\forall s \in S : \quad$ every execution from s goes through T

Definition of $\Rrightarrow$

$S \Rrightarrow T$ iff

1. $\forall s \in S \quad \exists t \in T : \quad s \stackrel{*}{\Longrightarrow} t$, and

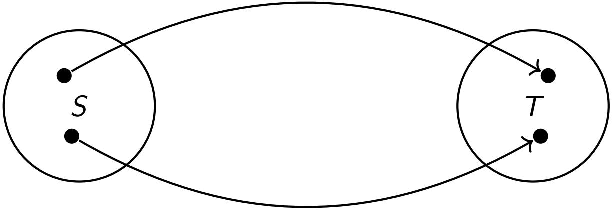


2. $\forall s \in S :$ every execution from s goes through T

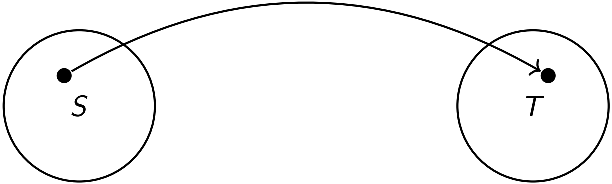
Definition of $\Rightarrow\!\!\!\triangleright$

$S \Rightarrow\!\!\!\triangleright T$ iff

1. $\forall s \in S \quad \exists t \in T : \quad s \overset{*}{\Longrightarrow} t$, and



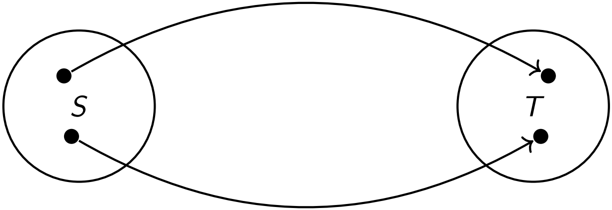
2. $\forall s \in S :$ every execution from s goes through T



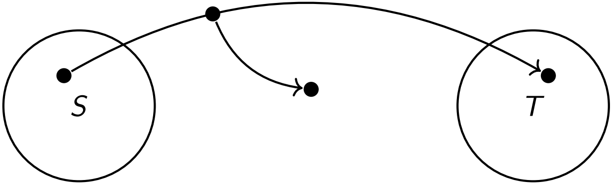
Definition of $\Rightarrow\!\!\rhd$

$S \Rightarrow\!\!\rhd T$ iff

1. $\forall s \in S \quad \exists t \in T : \quad s \stackrel{*}{\Longrightarrow} t$, and



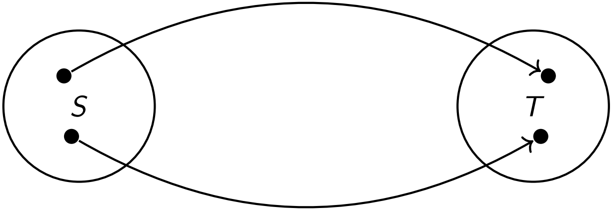
2. $\forall s \in S :$ every execution from s goes through T



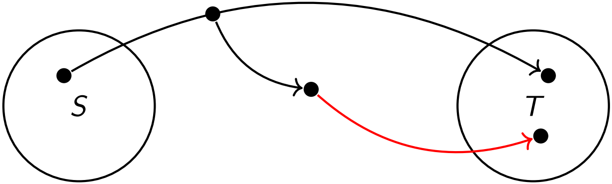
Definition of $\Rrightarrow$

$S \Rrightarrow T$ iff

1. $\forall s \in S \quad \exists t \in T : \quad s \overset{*}{\Longrightarrow} t$, and



2. $\forall s \in S :$ every execution from s goes through T



Proof System for $\Rrightarrow$

$$\frac{}{S \Rrightarrow S}\ \textsc{Refl} \qquad \frac{S \Rrightarrow T \qquad T \Rrightarrow U}{S \Rrightarrow U}\ \textsc{Trans} \qquad \frac{S \equiv T}{S \Rrightarrow T}\ \textsc{Iff}$$

$$\frac{S \Rrightarrow T \qquad S' \Rrightarrow T'}{S \cup S' \Rrightarrow T \cup T'}\ \textsc{Union}$$

$$\frac{P \to C \Longrightarrow D \qquad P \to C \lhd \{\, D \mid P \,\}}{\{\, C \mid P \,\} \qquad \Rrightarrow \{\, D \mid P \,\}}\ \textsc{Step}$$

Proof System for $\Rrightarrow$

$$\frac{}{S \Rrightarrow S}\ \textsc{Refl} \qquad \frac{S \Rrightarrow T \qquad T \Rrightarrow U}{S \Rrightarrow U}\ \textsc{Trans} \qquad \frac{S \equiv T}{S \Rrightarrow T}\ \textsc{Iff}$$

$$\frac{S \Rrightarrow T \qquad S' \Rrightarrow T'}{S \cup S' \Rrightarrow T \cup T'}\ \textsc{Union}$$

$$\frac{P \to C \Longrightarrow D \qquad P \to C \lhd \{\, D \mid P \,\} \cup T}{\{\, C \mid P \,\} \cup T \Rrightarrow \{\, D \mid P \,\} \cup T}\ \textsc{Step}$$

Toy Example: Arithmetic Expressions + Random

Syntax

$$\textbf{data}\ \mathit{Expr} = \mathit{Val}\ \mathit{Int} \mid \mathit{Add}\ \mathit{Expr}\ \mathit{Expr} \mid \mathit{Rnd}\ \mathit{Expr}$$

Toy Example: Arithmetic Expressions + Random

Syntax

$$\textbf{data}\ \mathit{Expr} = \mathit{Val}\ \mathit{Int} \mid \mathit{Add}\ \mathit{Expr}\ \mathit{Expr} \mid \mathit{Rnd}\ \mathit{Expr}$$

Toy Example: Arithmetic Expressions + Random

Syntax

$$\textbf{data}\ Expr = Val\ Int \mid Add\ Expr\ Expr \mid Rnd\ Expr$$

Semantics

$$\frac{}{Val\ n \Downarrow n}\ \textsc{Val} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{Add\ e_1\ e_2 \Downarrow (n_1 + n_2)}\ \textsc{Add}$$

$$\frac{e \Downarrow n \qquad 0 \leqslant m \leqslant |n|}{Rnd\ e \Downarrow m}\ \textsc{Rnd}$$

Toy Example: Arithmetic Expressions + Random

Syntax

$$\textbf{data}\ \mathit{Expr} = \mathit{Val}\ \mathit{Int} \mid \mathit{Add}\ \mathit{Expr}\ \mathit{Expr} \mid \mathit{Rnd}\ \mathit{Expr}$$

Semantics

$$\frac{}{\mathit{Val}\ n \Downarrow n}\ \textsc{Val} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{\mathit{Add}\ e_1\ e_2 \Downarrow (n_1 + n_2)}\ \textsc{Add}$$

$$\frac{e \Downarrow n \qquad 0 \leqslant m \leqslant |n|}{\mathit{Rnd}\ e \Downarrow m}\ \textsc{Rnd}$$

Compiler Correctness Property

$$\forall\ e\ c\ s\ :\quad \{(\mathit{comp}'\ e\ c, s)\quad\} \Rightarrow \{(c, n : s) \mid e \Downarrow n\quad\}$$

Toy Example: Arithmetic Expressions + Random

Syntax

$$\textbf{data}\ \mathit{Expr} = \mathit{Val}\ \mathit{Int} \mid \mathit{Add}\ \mathit{Expr}\ \mathit{Expr} \mid \mathit{Rnd}\ \mathit{Expr}$$

Semantics

$$\frac{}{\mathit{Val}\ n \Downarrow n}\ \textsc{Val} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2}{\mathit{Add}\ e_1\ e_2 \Downarrow (n_1 + n_2)}\ \textsc{Add}$$

$$\frac{e \Downarrow n \qquad 0 \leqslant m \leqslant |n|}{\mathit{Rnd}\ e \Downarrow m}\ \textsc{Rnd}$$

Compiler Correctness Property

$$\forall\ e\ c\ P: \quad \{(\mathit{comp}'\ e\ c, s) \mid P\ s\} \Rrightarrow \{(c, n : s) \mid e \Downarrow n \land P\ s\}$$

Calculate!

$$\{(comp' \quad e \quad c, s) \mid P\ s\} \Rightarrow \{(c, n : s) \mid \quad e \quad \Downarrow n \wedge P\ s\}$$

Calculate!

$$\{\,(\mathit{comp}'\,(\mathit{Val}\ m)\,c, s) \mid P\ s\,\} \Rightarrow \{\,(c, n : s) \mid\ (\mathit{Val}\ m)\ \Downarrow n\ \wedge\ P\ s\,\}$$

Calculate!

$$\{(comp'\,(Val\ m)\,c, s) \mid P\ s\} \Rrightarrow \{(c, n : s) \mid (Val\ m) \Downarrow n \wedge P\ s\}$$

$$\begin{aligned}
& \{(c, n : s) \mid Val\ m \Downarrow n \wedge P\ s\} \\
\equiv \quad & \{\ \text{by } \textsc{Val}\ \} \\
& \{(c, m : s) \mid P\ s\} \\
\Longleftarrow \quad & \{\ \text{define } \text{VM-}\textsc{Push}\ \} \\
& \{(PUSH\ m : c, s) \mid P\ s\} \\
\equiv \quad & \{\ \text{define: } comp'\ (Val\ m)\ c = PUSH\ m : c\ \} \\
& \{(comp'\ (Val\ m)\ c, s) \mid P\ s\}
\end{aligned}$$

Calculate!

$$\{(comp' \; (Val \; m) \; c, s) \mid P \; s\} \Rrightarrow \{(c, n : s) \mid (Val \; m) \Downarrow n \land P \; s\}$$

$$\frac{}{Val \; m \; \Downarrow \; m} \; \textsc{Val}$$

$$
\begin{array}{ll}
 & \{(c, n : s) \mid Val \; m \; \Downarrow \; n \land P \; s\} \\
\equiv & \{ \text{ by } \textsc{Val} \; \} \\
 & \{(c, m : s) \mid P \; s\} \\
\Longleftarrow & \{ \text{ define } \textsc{VM-Push} \; \} \\
 & \{(PUSH \; m : c, s) \mid P \; s\} \\
\equiv & \{ \text{ define: } comp' \; (Val \; m) \; c = PUSH \; m : c \; \} \\
 & \{(comp' \; (Val \; m) \; c, s) \mid P \; s\}
\end{array}
$$

Calculate!

$$\{(comp'\,(Val\ m)\,c, s) \mid P\ s\} \Rrightarrow \{(c, n : s) \mid (Val\ m) \Downarrow n \wedge P\ s\}$$

$$
\begin{array}{ll}
& \{(c, n : s) \mid Val\ m \Downarrow n \wedge P\ s\} \\
\equiv & \quad \{\ \text{by Val} \\
& \{(c, m : s) \mid P\ s\} \\
\Longleftarrow & \quad \{\ \text{define VM-Push}\ \} \\
& \{(PUSH\ m : c, s) \mid P\ s\} \\
\equiv & \quad \{\ \text{define: } comp'\ (Val\ m)\ c = PUSH\ m : c\ \} \\
& \{(comp'\ (Val\ m)\ c, s) \mid P\ s\}
\end{array}
$$

$$(PUSH\ m : c, s) \Longrightarrow (c, m : s)$$

Calculate!

$$
\begin{array}{ll}
& \{(c, n : s) \mid Add\ e_1\ e_2 \Downarrow n \wedge P\ s\} \\
\equiv & \{\ \text{by } \textsc{Add}\ \} \\
& \{(c, (n_1 + n_2) : s) \mid e_1 \Downarrow n_1 \wedge e_2 \Downarrow n_2 \wedge P\ s\} \\
\Longleftarrow & \{\ \text{define } \textsc{VM-Add}\ \} \\
& \{(ADD : c, n_2 : n_1 : s) \mid e_1 \Downarrow n_1 \wedge e_2 \Downarrow n_2 \wedge P\ s\} \\
\equiv & \{\ \text{move existential quantifier}\ \} \\
& \{(ADD : c, n_2 : s') \mid e_2 \Downarrow n_2 \wedge (\exists\ s\ n_1, e_1 \Downarrow n_1 \wedge s' = n_1 : s \wedge P\ s)\} \\
\Lleftarrow & \{\ \text{induction hypothesis for } e_2\ \} \\
& \{(comp'\ e_2\ (ADD : c), s) \mid \exists\ s'\ n_1, e_1 \Downarrow n_1 \wedge s = n_1 : s' \wedge P\ s'\} \\
\equiv & \{\ \text{move existential quantifier}\ \} \\
& \{(comp'\ e_2\ (ADD : c), n_1 : s) \mid e_1 \Downarrow n_1 \wedge P\ s\} \\
\Lleftarrow & \{\ \text{induction hypothesis for } e_1\ \} \\
& \{(comp'\ e_1\ (comp'\ e_2\ (ADD : c)), s) \mid P\ s\} \\
\equiv & \{\ \text{define } comp'\ (Add\ e_1\ e_2)\ c = \ldots\ \} \\
& \{(comp'\ (Add\ e_1\ e_2)\ c, s) \mid P\ s\}
\end{array}
$$

Calculate!

$$\{(comp' \quad e \qquad c, s) \mid P\ s\} \Rightarrow \{(c, m : s) \mid \quad e \qquad \Downarrow m \ \wedge\ P\ s\}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rightarrow \{(c', m : s) \mid e\ \Downarrow\ m \wedge P'\ s\}$$

Calculate!

$$\{(comp' \,(Rnd\ e)\, c, s) \mid P\ s\} \Rrightarrow \{(c, m : s) \mid (Rnd\ e) \Downarrow m \wedge P\ s\}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rrightarrow \{(c', m : s) \mid e \Downarrow m \wedge P'\ s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\,c, s) \mid P\ s\} \Rrightarrow \{(c, m : s) \mid\ (Rnd\ e) \Downarrow m \wedge\ P\ s\}$$

$$\{(c, m : s) \mid Rnd\ e\ \Downarrow\ m \wedge P\ s\}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rrightarrow \{(c', m : s) \mid e\ \Downarrow\ m \wedge P'\ s\}$$

Calculate!

$$\{(comp' \, (Rnd \; e) \, c, s) \mid P \; s\} \Rightarrow \{(c, m : s) \mid \; (Rnd \; e) \Downarrow m \; \wedge \; P \; s\}$$

$$\begin{aligned} & \{(c, m : s) \mid Rnd \; e \; \Downarrow \; m \wedge P \; s\} \\ \equiv & \quad \{ \text{ by } \textsc{Rnd} \; \} \\ & \{(c, m : s) \mid e \; \Downarrow \; n \wedge 0 \leqslant m \leqslant |n| \wedge P \; s\} \end{aligned}$$

Induction Hypothesis

$$\{(comp' \; e \; c', s) \mid P' \; s\} \Rightarrow \{(c', m : s) \mid e \; \Downarrow \; m \wedge P' \; s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\ \ \ \ \frac{e \Downarrow n \qquad 0 \leqslant m \leqslant |n|}{Rnd\ e \Downarrow m}\ \textsc{Rnd} \ \ \ \ \Downarrow m \land P\ s\}$$

$$\{(c, m : s) \mid Rnd\ e \Downarrow m \land P\ s\}$$

$$\equiv \quad \{\text{ by } \textsc{Rnd} \ \}$$

$$\{(c, m : s) \mid e \Downarrow n \land 0 \leqslant m \leqslant |n| \land P\ s\}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rrightarrow \{(c', m : s) \mid e \Downarrow m \land P'\ s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\,c, s) \mid P\ s\} \Rrightarrow \{(c, m : s) \mid (Rnd\ e) \Downarrow m \wedge P\ s\}$$

$$\begin{array}{ll}
& \{(c, m : s) \mid Rnd\ e \Downarrow m \wedge P\ s\} \\
\equiv & \quad \{\ \text{by } \textsc{Rnd}\ \} \\
& \{(c, m : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\
\Longleftarrow & \quad \{\ \text{define } \textsc{VM-Rnd}\ \} \\
& \{(RND : c, n : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\}
\end{array}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rrightarrow \{(c', m : s) \mid e \Downarrow m \wedge P'\ s\}$$

Calculate!

$$\{(comp' \; (Rnd \; e) \; c, s) \mid P \; s\} \Rightarrow \{(c, m : s) \mid (Rnd \; e) \Downarrow m \wedge P \; s\}$$

$$\{(c, m : s) \mid Rnd \; e \Downarrow m \wedge P \; s\}$$

$\equiv$ { by Rn… }

$$\{(c, m : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P \; s\}$$

$\Longleftarrow$ { define VM-Rnd }

$$\{(RND : c, n : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P \; s\}$$

$$(RND : c, n : s) \Longrightarrow (c, m : s)$$

Induction Hypothesis

$$\{(comp' \; e \; c', s) \mid P' \; s\} \Rightarrow \{(c', m : s) \mid e \Downarrow m \wedge P' \; s\}$$

Calculate!

$$\{(comp' \, (Rnd \; e) \, c, s) \mid P \; s\} \Rrightarrow \{(c, m : s) \mid (Rnd \; e) \Downarrow m \wedge P \; s\}$$

$$\{(c, m : s) \mid Rnd \; e \Downarrow m \wedge P \; s\}$$

$$\equiv \quad \{ \text{ by } \textsc{Rnd} \}$$

$$\{(c, m : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P \; s\}$$

$$\Longleftarrow \quad \{ \text{ define } \textsc{VM-Rnd} \}$$

$$\{(RND : c, n : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P \; s\}$$

$$(RND : c, n : s) \Longrightarrow (c, m : s) \quad \textit{if } 0 \leqslant m \leqslant |n|$$

Induction Hypothesis

$$\{(comp' \; e \; c', s) \mid P' \; s\} \Rrightarrow \{(c', m : s) \mid e \Downarrow m \wedge P' \; s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\,c, s) \mid P\ s\} \Rightarrow \{(c, m : s) \mid (Rnd\ e) \Downarrow m \wedge P\ s\}$$

$$\begin{array}{ll} & \{(c, m : s) \mid Rnd\ e \Downarrow m \wedge P\ s\} \\ \equiv & \{\ \text{by } \textsc{Rnd}\ \} \\ & \{(c, m : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\ \Longleftarrow & \{\ \text{define } \text{VM-}\textsc{Rnd}\ \} \\ & \{(RND : c, n : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\ \equiv & \{\ \text{eliminate tautology } \exists\, m, 0 \leqslant m \leqslant |n|\ \} \\ & \{(RND : c, n : s) \mid e \Downarrow n \wedge P\ s\} \end{array}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rightarrow \{(c', m : s) \mid e \Downarrow m \wedge P'\ s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\,c, s) \mid P\ s\} \Rrightarrow \{(c, m : s) \mid (Rnd\ e) \Downarrow m \wedge P\ s\}$$

$$\begin{array}{ll}
 & \{(c, m : s) \mid Rnd\ e \Downarrow m \wedge P\ s\} \\
\equiv & \{\ \text{by } \textsc{Rnd}\ \} \\
 & \{(c, m : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\
\Longleftarrow & \{\ \text{define } \textsc{VM-Rnd}\ \} \\
 & \{(RND : c, n : s) \mid e \Downarrow n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\
\equiv & \{\ \text{eliminate tautology } \exists\, m, 0 \leqslant m \leqslant |n|\ \} \\
 & \{(RND : c, n : s) \mid e \Downarrow n \wedge P\ s\} \\
\Lleftarrow & \{\ \text{induction hypothesis for } e\ \} \\
 & \{(comp'\ e\ (RND : c), s) \mid P\ s\}
\end{array}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rrightarrow \{(c', m : s) \mid e \Downarrow m \wedge P'\ s\}$$

Calculate!

$$\{(comp'\,(Rnd\ e)\,c, s) \mid P\ s\} \Rightarrow \{(c, m : s) \mid\ (Rnd\ e) \Downarrow m\ \wedge\ P\ s\}$$

$$\begin{array}{ll}
& \{(c, m : s) \mid Rnd\ e\ \Downarrow\ m \wedge P\ s\} \\
\equiv & \quad \{\ \text{by } \textsc{Rnd}\ \ \} \\
& \{(c, m : s) \mid e\ \Downarrow\ n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\
\Longleftarrow & \quad \{\ \text{define } \textsc{VM-Rnd}\ \ \} \\
& \{(RND : c, n : s) \mid e\ \Downarrow\ n \wedge 0 \leqslant m \leqslant |n| \wedge P\ s\} \\
\equiv & \quad \{\ \text{eliminate tautology } \exists\ m, 0 \leqslant m \leqslant |n|\ \} \\
& \{(RND : c, n : s) \mid e\ \Downarrow\ n \wedge P\ s\} \\
\Leftarrow & \quad \{\ \text{induction hypothesis for } e\ \} \\
& \{(comp'\ e\ (RND : c), s) \mid P\ s\} \\
\equiv & \quad \{\ \text{define } comp'\ (Rnd\ e)\ c = RND : c\ \} \\
& \{(comp'\ (Rnd\ e)\ c, s) \mid P\ s\}
\end{array}$$

Induction Hypothesis

$$\{(comp'\ e\ c', s) \mid P'\ s\} \Rightarrow \{(c', m : s) \mid e\ \Downarrow\ m \wedge P'\ s\}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD} \mid \mathit{RND}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD} \mid \mathit{RND}$$

$$\begin{array}{ll}
\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code} & \\
\mathit{comp}'\ (\mathit{Val}\ m) & c = \mathit{PUSH}\ m : c \\
\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2) & c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c)) \\
\mathit{comp}'\ (\mathit{Rnd}\ x) & c = \mathit{comp}'\ x\ (\mathit{RND} : c)
\end{array}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH}\ \mathit{Int} \mid \mathit{ADD} \mid \mathit{RND}$$

$$
\begin{aligned}
&\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code}\\
&\mathit{comp}'\ (\mathit{Val}\ m)\qquad c = \mathit{PUSH}\ m : c\\
&\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c))\\
&\mathit{comp}'\ (\mathit{Rnd}\ x)\qquad c = \mathit{comp}'\ x\ (\mathit{RND} : c)
\end{aligned}
$$

$$(\mathit{PUSH}\ m : c, s) \Longrightarrow (c, m : s) \qquad \text{(VM-Push)}$$

$$(\mathit{ADD} : c, n_2 : n_1 : s) \Longrightarrow (c, (n_1 + n_2) : s) \qquad \text{(VM-Add)}$$

$$(\mathit{RND} :: c, n :: s) \Longrightarrow (c, m :: s) \quad \text{if } 0 \leqslant m \leqslant |n| \qquad \text{(VM-Rnd)}$$

The Result

$$\textbf{data}\ \mathit{Instr} = \mathit{PUSH\ Int} \mid \mathit{ADD} \mid \mathit{RND}$$

$$\begin{aligned}
&\mathit{comp}' :: \mathit{Expr} \to \mathit{Code} \to \mathit{Code}\\
&\mathit{comp}'\ (\mathit{Val}\ m) \qquad\quad c = \mathit{PUSH}\ m : c\\
&\mathit{comp}'\ (\mathit{Add}\ e_1\ e_2)\ c = \mathit{comp}'\ e_1\ (\mathit{comp}'\ e_2\ (\mathit{ADD} : c))\\
&\mathit{comp}'\ (\mathit{Rnd}\ x) \qquad\quad c = \mathit{comp}'\ x\ (\mathit{RND} : c)
\end{aligned}$$

$$(\mathit{PUSH}\ m : c, s) \Longrightarrow (c, m : s) \qquad \text{(VM-Push)}$$

$$(\mathit{ADD} : c, n_2 : n_1 : s) \Longrightarrow (c, (n_1 + n_2) : s) \qquad \text{(VM-Add)}$$

$$(\mathit{RND} :: c, n :: s) \Longrightarrow (c, m :: s) \quad \text{if } 0 \leqslant m \leqslant |n| \qquad \text{(VM-Rnd)}$$

$$\forall\ e\ c\ s : \quad \{(\mathit{comp}'\ e\ c, s)\} \Rrightarrow \{(c, n : s) \mid e \Downarrow n\}$$

Concluding Remarks

Implementation in Coq

- Verified proof rules
- Proof search for STEP rule
- Syntax close to informal notation
- available at: `http://j.mp/CompCalc`

Co

```
Theorem correctness : forall e P c,
  {s, ⟨comp' e c, s ⟩ | P s} =|> {s n, ⟨c , n :: s ⟩ | e ⇓ n /\ P s}.
Proof.
induction e;intros.

begin
  ({s n', ⟨c, n' :: s ⟩ | Val n ⇓ n' /\ P s}).
=   { by_eval }
  ({s, ⟨c, n :: s ⟩ | P s}) .
<== { apply vm_push }
  ({s, ⟨PUSH n :: c, s ⟩ | P s}) .
 [].

begin
  ({s n, ⟨c, n :: s ⟩ | Add e1 e2 ⇓ n /\ P s }) .
=   { by_eval }
  ({s n m, ⟨c, (n + m) :: s ⟩ | e1 ⇓ n /\ e2 ⇓ m /\ P s}) .
<== { apply vm_add }
  ({s n m, ⟨ADD :: c, m :: n :: s ⟩ | e1 ⇓ n  /\ e2 ⇓ m /\ P s}).
=   { eauto }
  ({s' m, ⟨ADD :: c, m :: s' ⟩ | e2 ⇓ m
            /\ (exists s n, e1 ⇓ n /\ s' = n :: s /\ P s)}).
<|= { apply IHe2 }
```

Concluding Remarks

Implementation in Coq

- Verified proof rules
- Proof search for STEP rule
- Syntax close to informal notation
- available at: `http://j.mp/CompCalc`

Concluding Remarks

Implementation in Coq

- Verified proof rules
- Proof search for STEP rule
- Syntax close to informal notation
- available at: `http://j.mp/CompCalc`

Future Work

- Concurrency
- Abstraction vs. full details
- Register machines

Calculating Certified Compilers for Non-Deterministic Languages

Patrick Bahr

University of Copenhagen
`paba@diku.dk`

Coq source code available at: `http://j.mp/CompCalc`