Termination
Automated Reasoning and its Tools

Christian Sternagel

Ogawa Laboratory
School of Information Science
Japan Advanced Institute of Science and Technology

March 5th, 2012

JAIST Spring School 2012

Certified Termination

Automated Reasoning and its Tools

Christian Sternagel

Ogawa Laboratory
School of Information Science
Japan Advanced Institute of Science and Technology

March 5th, 2012

JAIST Spring School 2012

Certified Termination

Automated Reasoning and its Tools

Christian Sternagel

Ogawa Laboratory
School of Information Science
Japan Advanced Institute of Science and Technology

March 5th, 2012

JAIST Spring School 2012

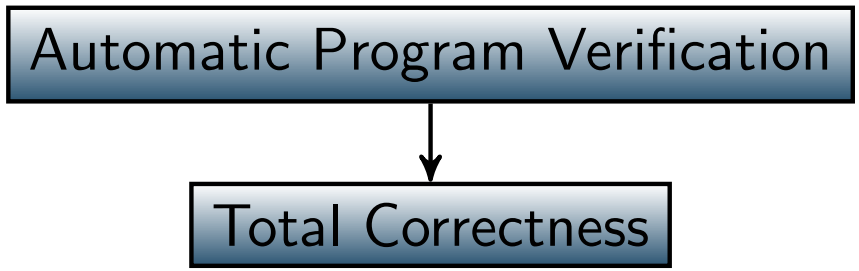
Overview

- Motivation
- Isabelle/HOL
- Certification
- Termination of HOL Functions

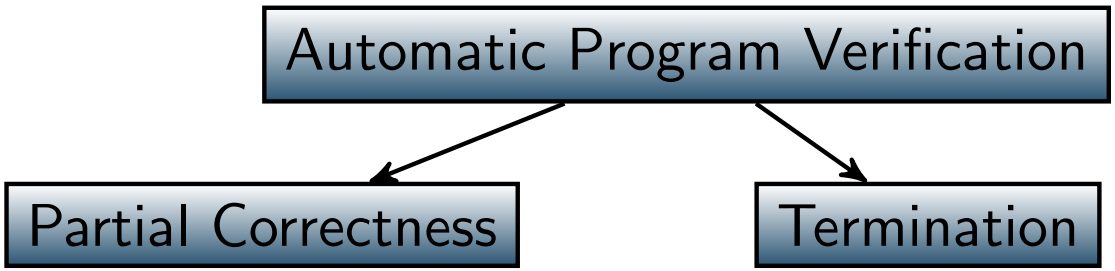
Motivation

Automatic Program Verification

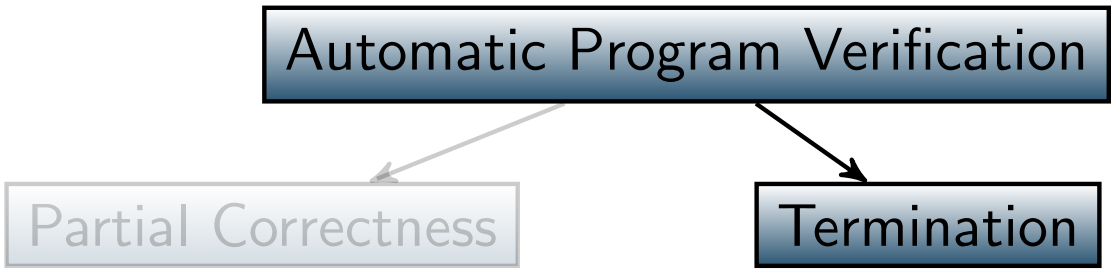
Motivation



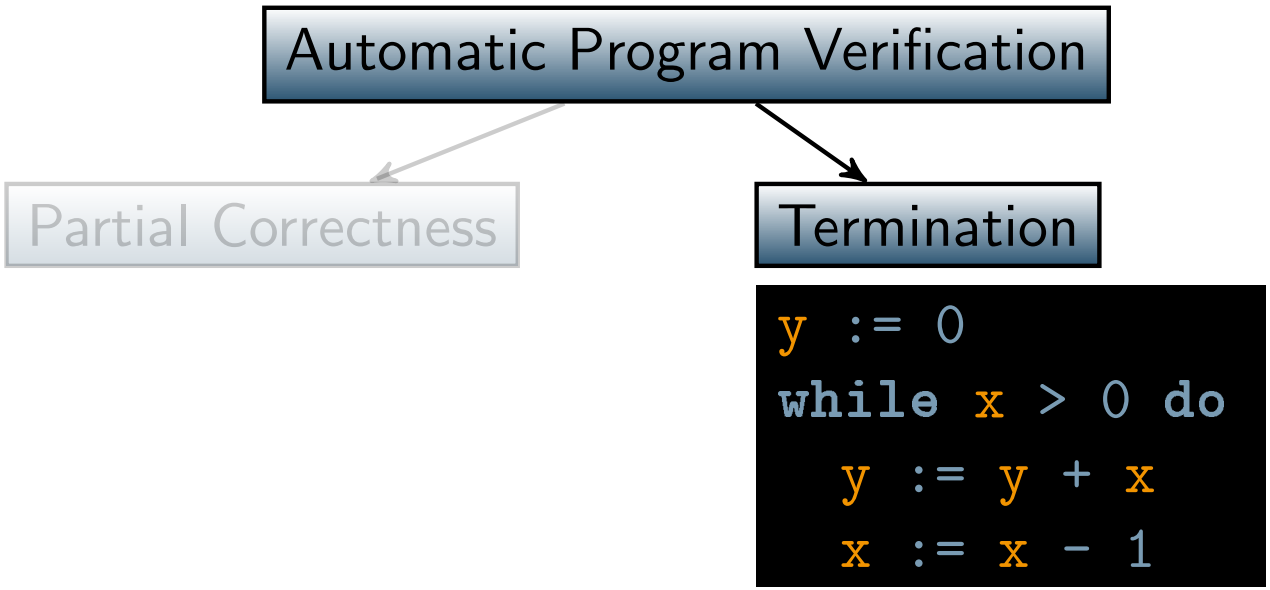
Motivation


Automatic Program Verification
Partial Correctness
Termination

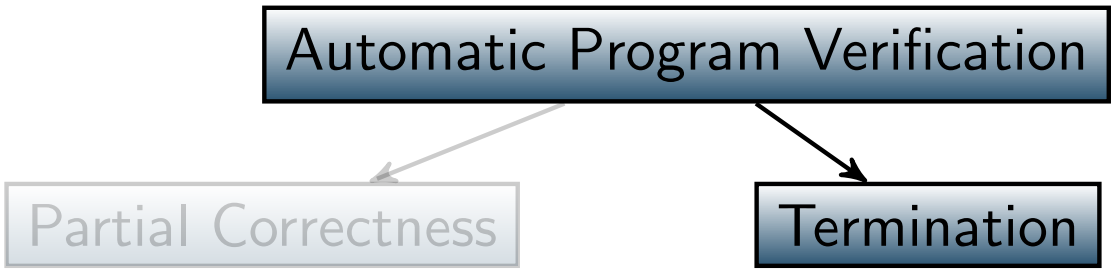
Motivation


Automatic Program Verification
Partial Correctness
Termination

Motivation

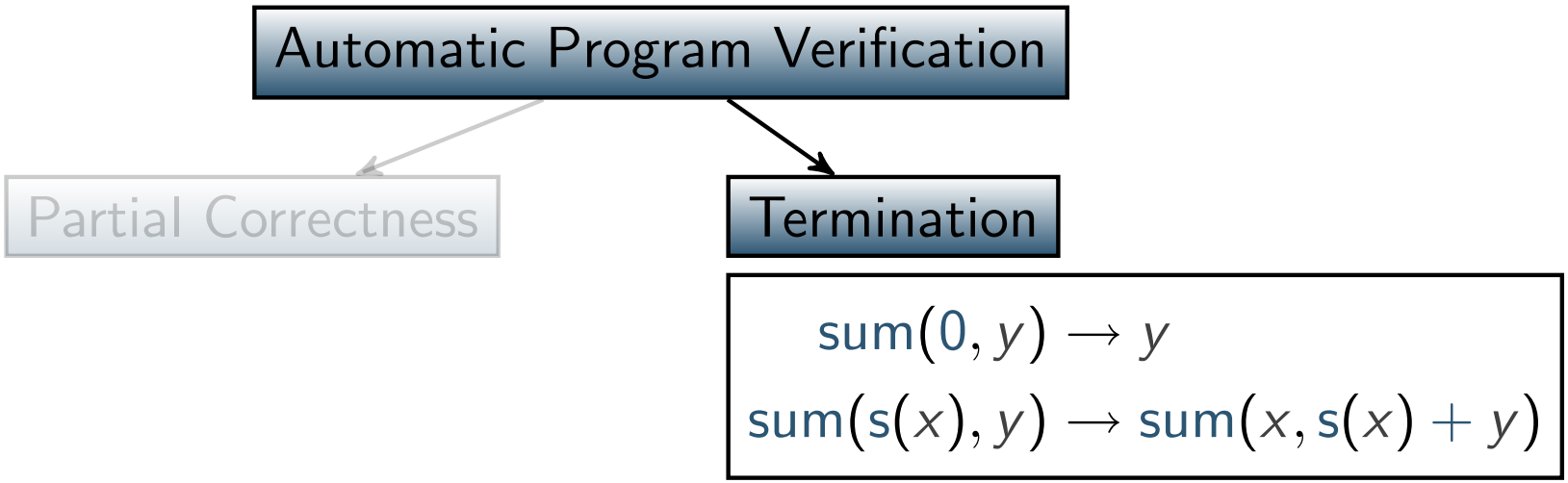

Automatic Program Verification
Partial Correctness
Termination
y := 0
while x > 0 do
y := y + x
x := x - 1

Motivation

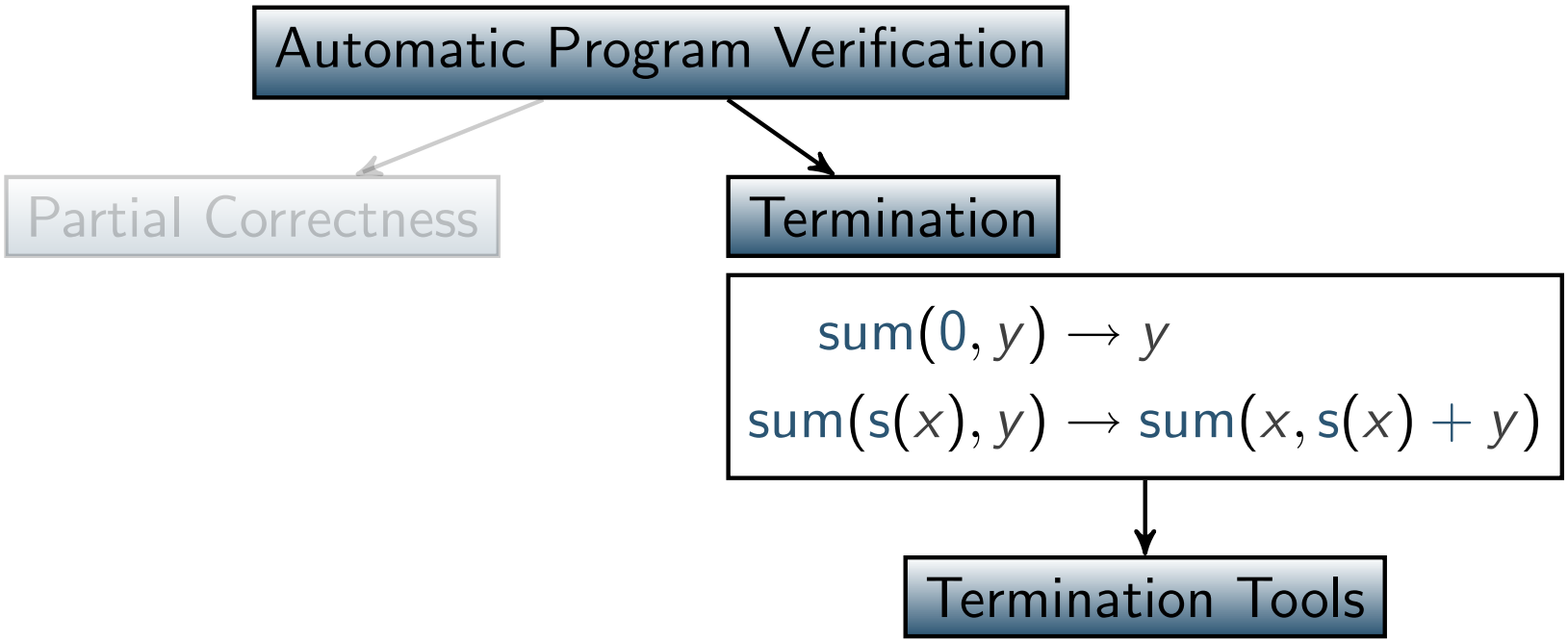



```
int sum(int x, int y) {
  if (x > 0)
    return sum(x-1, y+x);
  else
    return y;
}
```

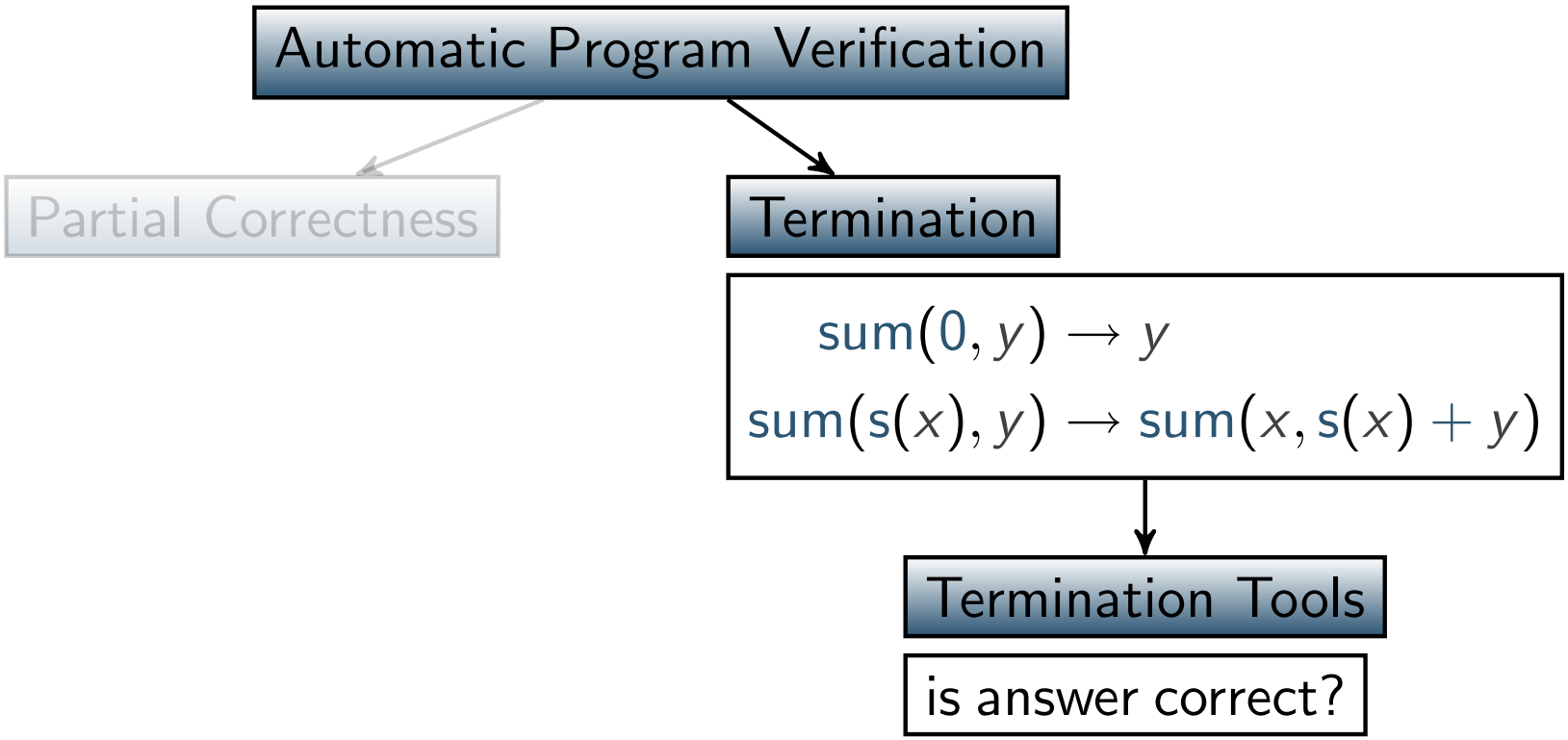
Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$

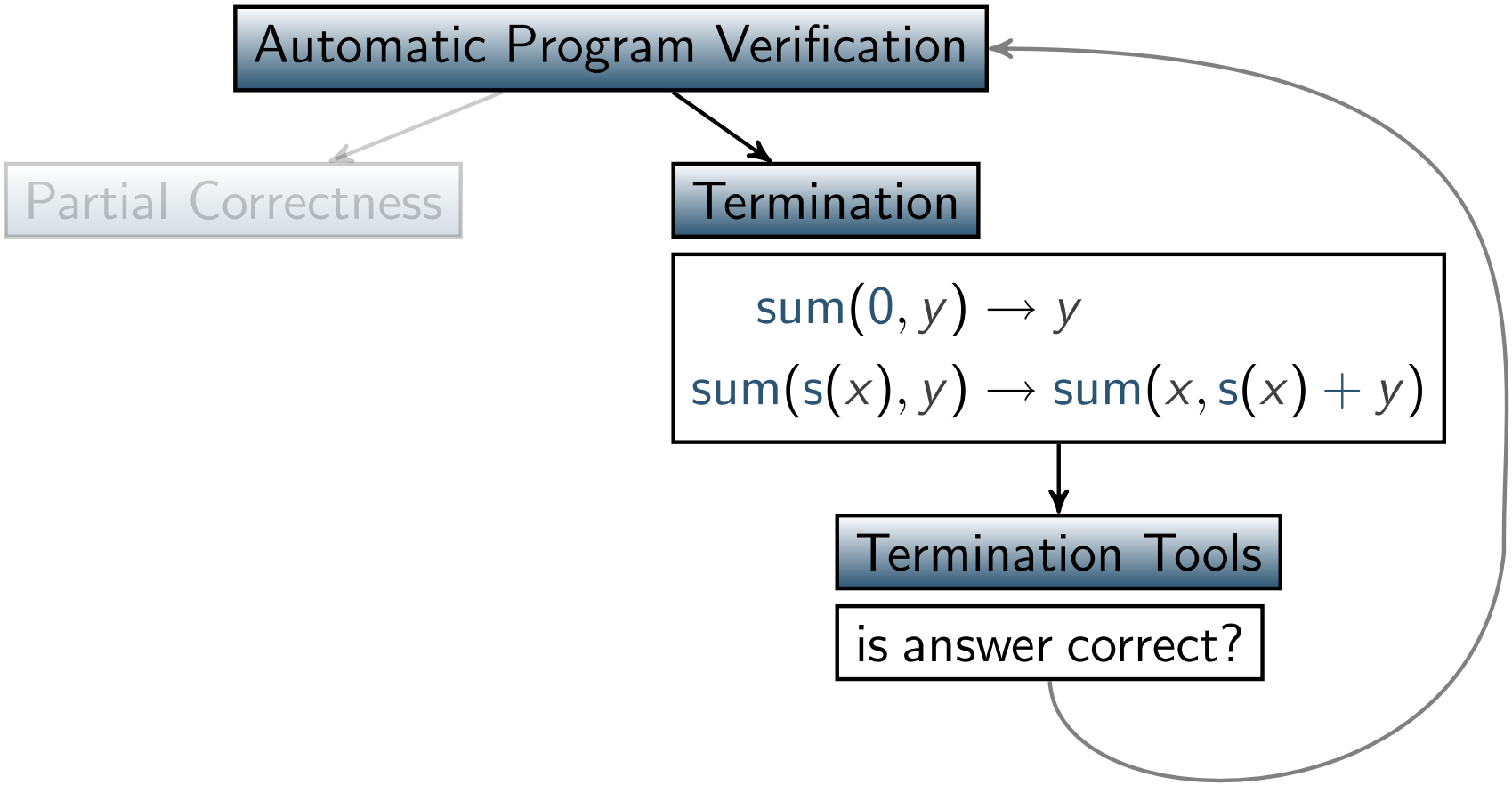
Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools

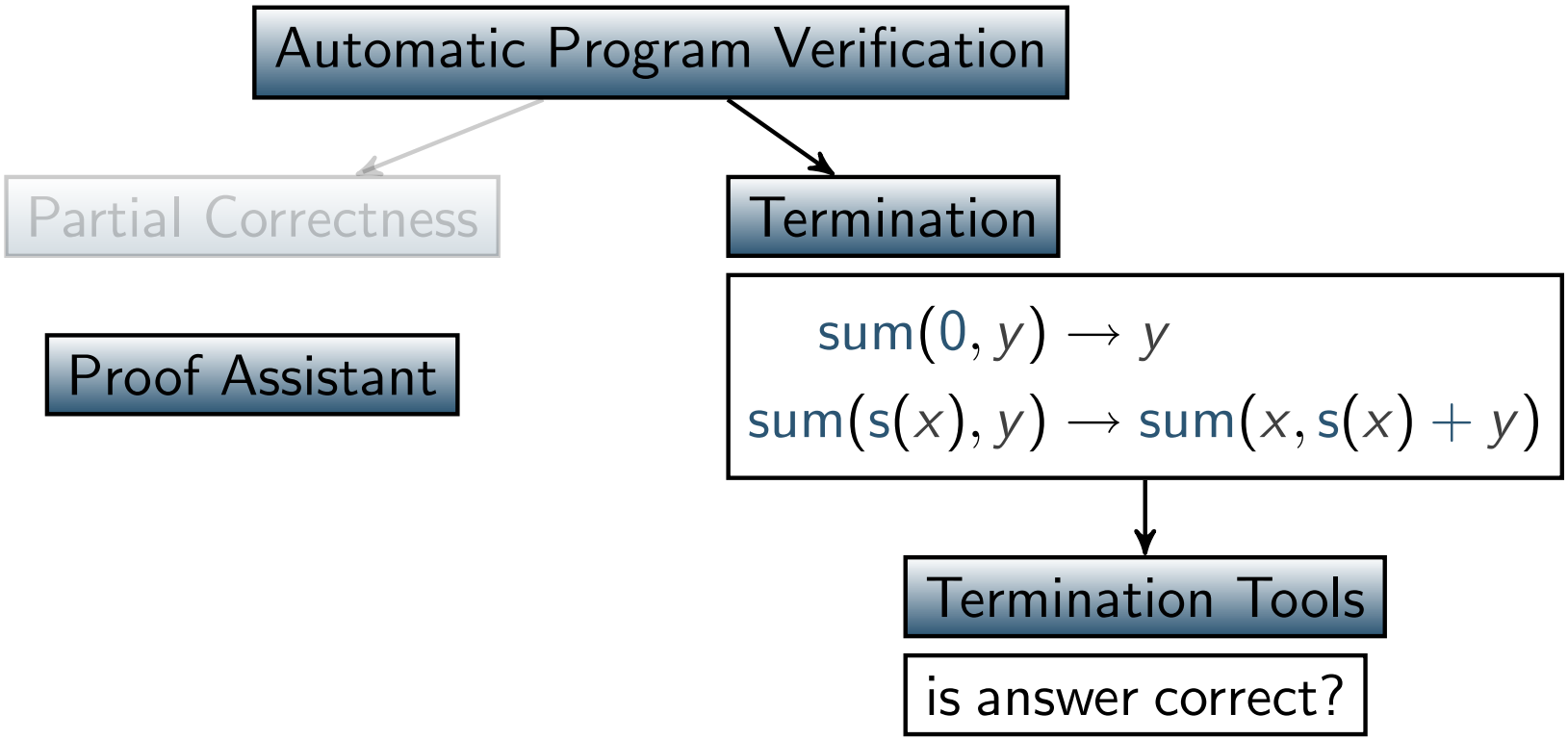
Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
is answer correct?

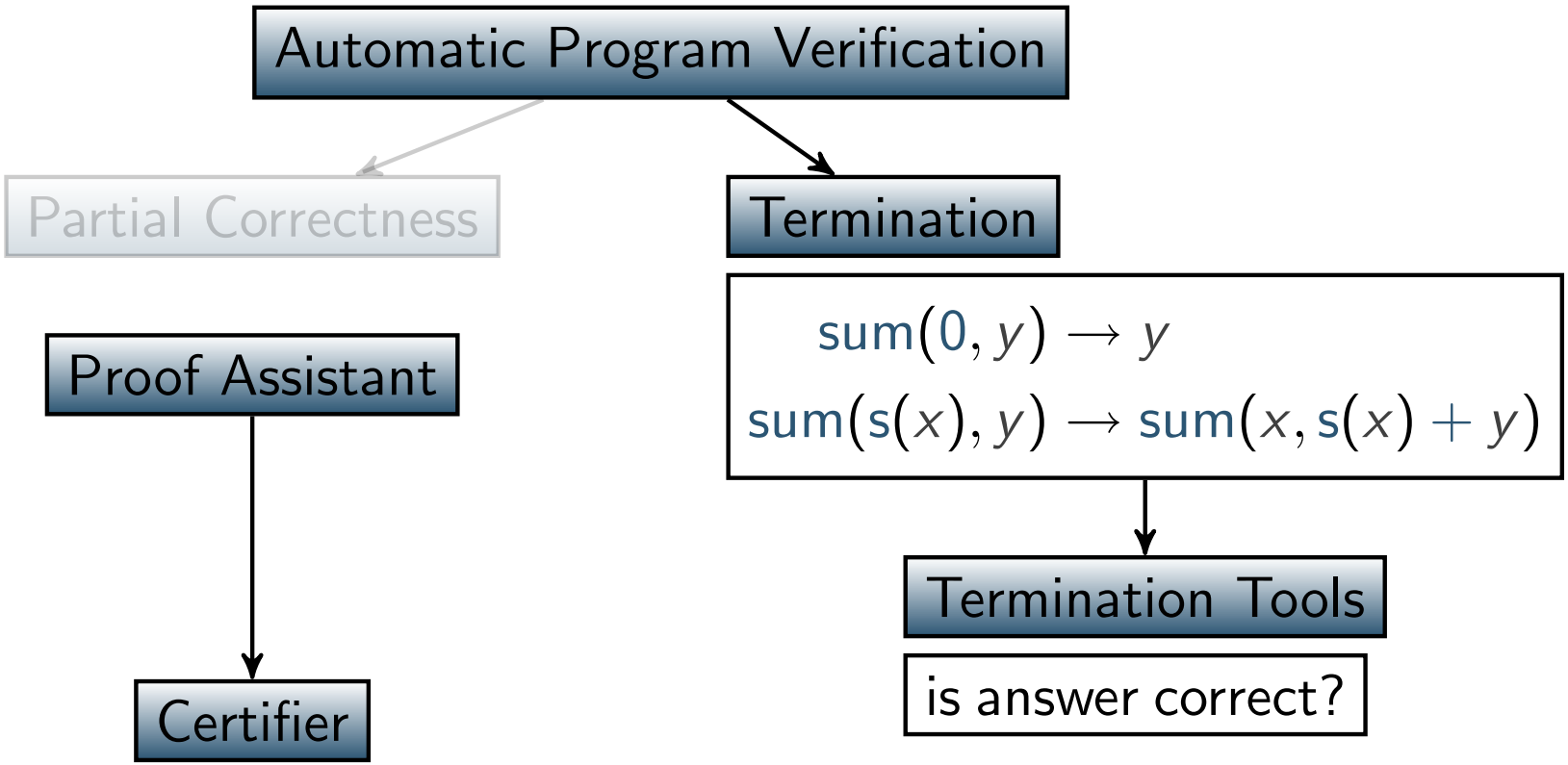
Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
is answer correct?

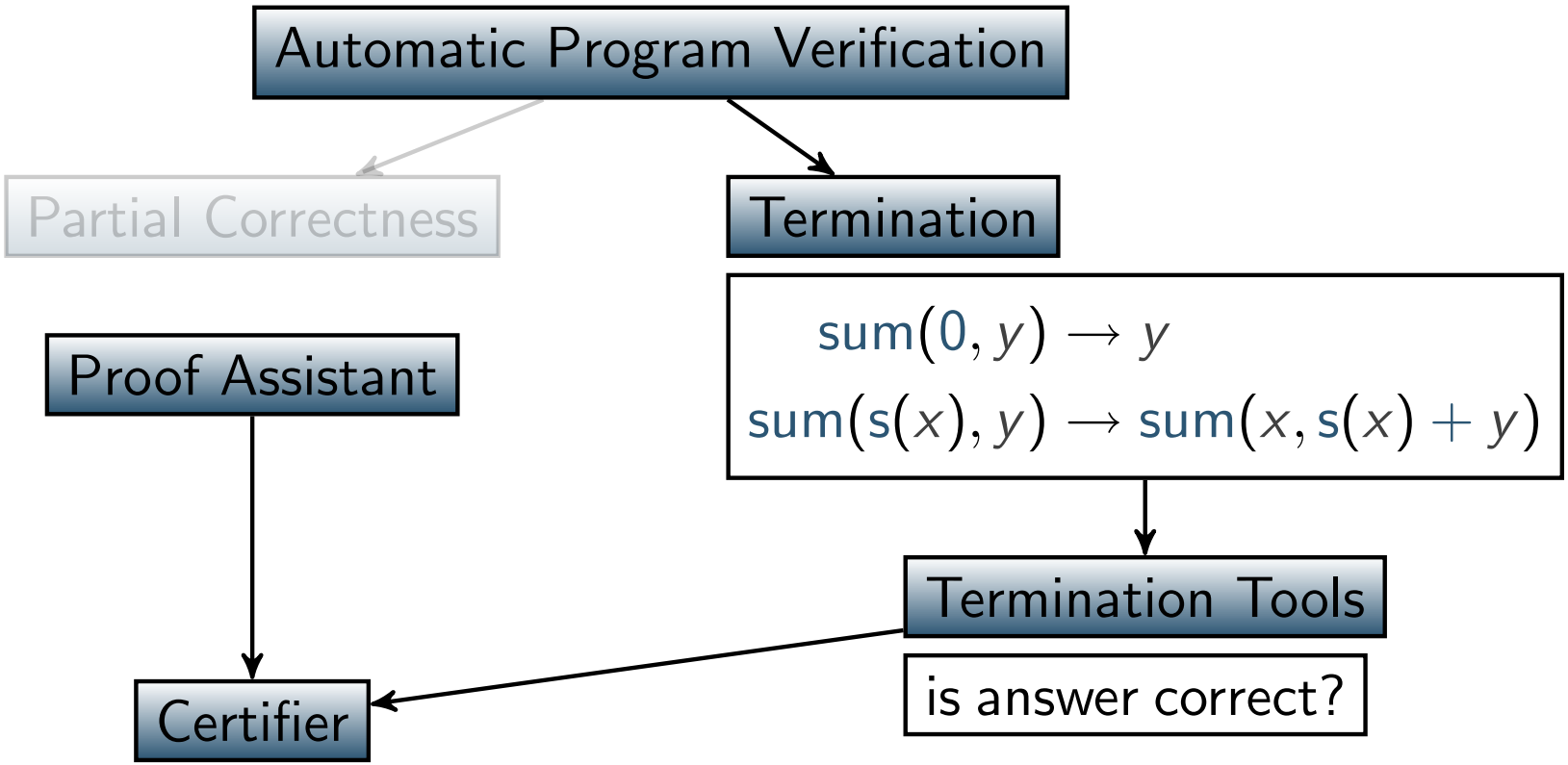
Motivation


Automatic Program Verification
Partial Correctness
Termination
Proof Assistant
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
is answer correct?

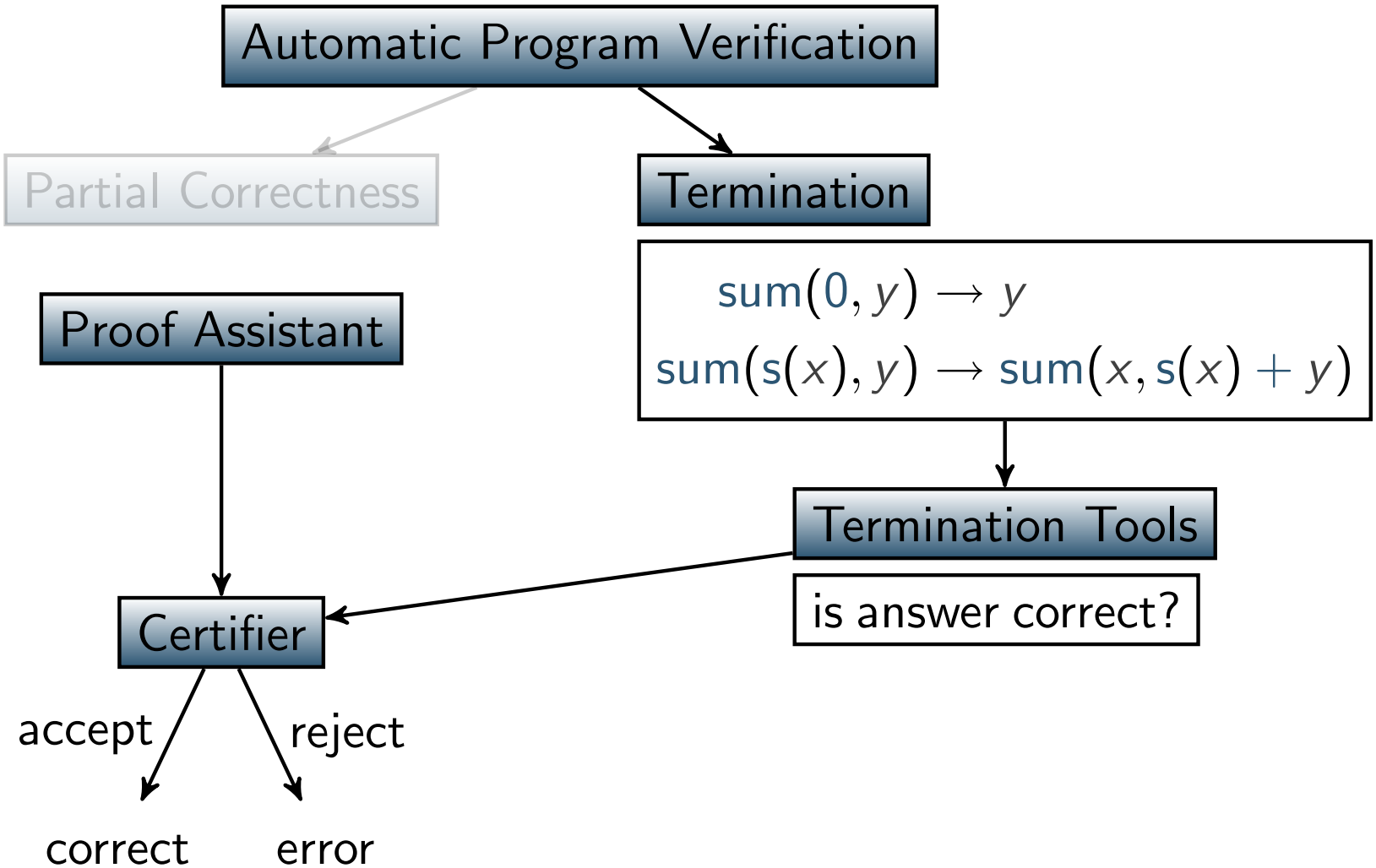
Motivation


Automatic Program Verification
Partial Correctness
Termination
Proof Assistant
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
Certifier
is answer correct?

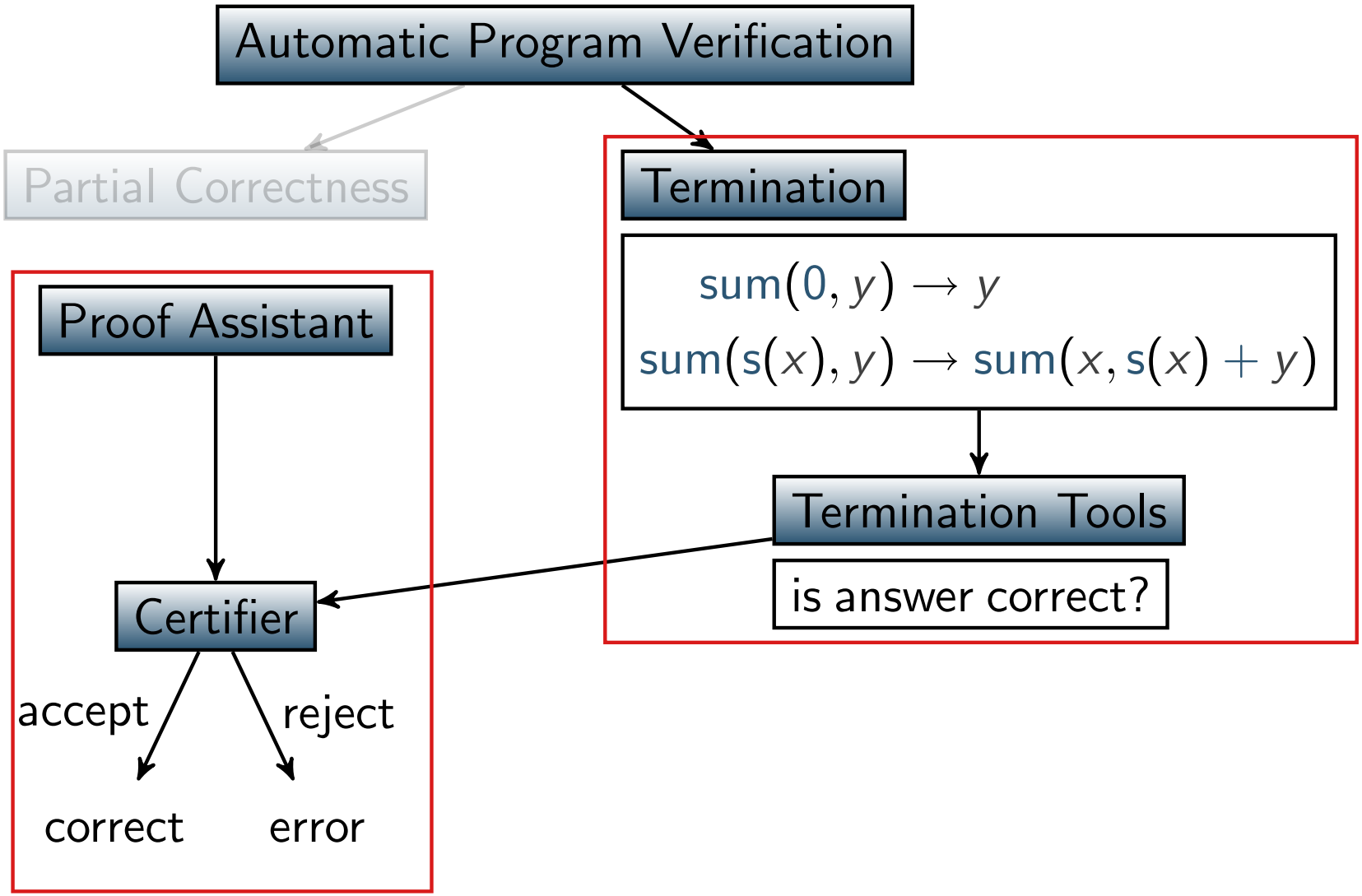
Motivation


Automatic Program Verification
Partial Correctness
Termination
Proof Assistant
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
is answer correct?
Certifier

Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Proof Assistant
Termination Tools
is answer correct?
Certifier
accept
reject
correct
error

Motivation


Automatic Program Verification
Partial Correctness
Termination
$\mathsf{sum}(0, y) \to y$
$\mathsf{sum}(\mathsf{s}(x), y) \to \mathsf{sum}(x, \mathsf{s}(x) + y)$
Termination Tools
is answer correct?
Proof Assistant
Certifier
accept
reject
correct
error

Introduction to Isabelle

A Bit of History

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*

A Bi

Scott's Logic for Computable Functions

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*
1977	Edinburgh LCF	*inference rules as ML functions, theorems as abstract data types (small trusted kernel, correct by construction)*

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theo-*
		Meta Language (e.g., Standard ML, OCaml)
1977	Edinburgh LCF	*inference rules as ML functions, theorems as abstract data types (small trusted kernel, correct by construction)*

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*
1977	Edinburgh LCF	*inference rules as ML functions, theorems as abstract data types (small trusted kernel, correct by construction)*
1987	Cambridge LCF	*Isabelle precursor [Larry Paulson]*

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*
1977	Edinburgh LCF	*inference rules as ML functions, theorems as abstract data types (small trusted kernel, correct by construction)*
1987	Cambridge LCF	*Isabelle precursor [Larry Paulson]*
1986	Isabelle	*generic proof assistant based on intuitionistic HOL and higher-order unification*

A Bit of History

1972	Stanford LCF	*compromise between fully automatic theorem proving (difficult) and single step proof checking (tedious) [Robin Milner]*
1977	Edinburgh LCF	*inference rules as ML functions, theorems as abstract data types (small trusted kernel, correct by construction)*
1987	Cam	*rsor [Larry Paulson]*
1986	Isabelle	*generic proof assistant based on intuitionistic HOL and higher-order unification*

Higher-Order Logic

System Architecture

Standard ML implementation language

System Architecture

Isabelle/Pure generic proof assistant

Standard ML implementation language

System Architecture

Isabelle/HOL Higher-Order Logic

Isabelle/Pure generic proof assistant

Standard ML implementation language

System Architecture

Proof General Emacs interface

Isabelle/HOL Higher-Order Logic

Isabelle/Pure generic proof assistant

Standard ML implementation language

System Architecture

Isabelle/jEdit jEdit based interface

Isabelle/Scala connects ML to JVM

Proof General Emacs interface

Isabelle/HOL Higher-Order Logic

Isabelle/Pure generic proof assistant

Standard ML implementation language

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*

terms depending on terms

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv$ :: $\alpha \Rightarrow \alpha \Rightarrow$ **prop**	$P\ x \equiv Q\ x$	*equality*

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv$ :: $\alpha \Rightarrow \alpha \Rightarrow$ **prop**	$P\ x \equiv Q\ x$	*equality*
$\bigwedge$ :: $(\alpha \Rightarrow$ **prop**$) \Rightarrow$ **prop**	$\bigwedge x.\ P\ x$	*all-quantification*

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$		
$\equiv :: \alpha \Rightarrow \alpha \Rightarrow \mathtt{prop}$	$P\ x \equiv Q\ x$	*equa*
$\bigwedge :: (\alpha \Rightarrow \mathtt{prop}) \Rightarrow \mathtt{prop}$	$\bigwedge x.\ P\ x$	*all-quantification*

proofs depending on terms

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv$ :: $\alpha \Rightarrow \alpha \Rightarrow$ **prop**	$P\ x \equiv Q\ x$	*equality*
$\bigwedge$ :: $(\alpha \Rightarrow$ **prop**$) \Rightarrow$ **prop**	$\bigwedge x.\ P\ x$	*all-quantification*
$\Longrightarrow$ :: **prop** $\Rightarrow$ **prop** $\Rightarrow$ **prop**	$A \Longrightarrow B$	*implication*

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv :: \alpha \Rightarrow \alpha \Rightarrow \mathtt{prop}$		
$\bigwedge :: (\alpha \Rightarrow \mathtt{prop}) \Rightarrow \mathtt{prop}$	$\bigwedge x.\, P\ x$	*universal quantification*
$\Longrightarrow :: \mathtt{prop} \Rightarrow \mathtt{prop} \Rightarrow \mathtt{prop}$	$A \Longrightarrow B$	*implication*

proofs depending on proofs

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv$ :: $\alpha \Rightarrow \alpha \Rightarrow$ **prop**	$P\ x \equiv Q\ x$	*equality*
$\bigwedge$ :: $(\alpha \Rightarrow$ **prop**$) \Rightarrow$ **prop**	$\bigwedge x.\ P\ x$	*all-quantification*
$\Longrightarrow$ :: **prop** $\Rightarrow$ **prop** $\Rightarrow$ **prop**	$A \Longrightarrow B$	*implication*

Basic Inference Rules

$$\frac{\text{if } \phi \text{ implies } \psi}{\phi \Longrightarrow \psi} \qquad \frac{\phi \Longrightarrow \psi \quad \phi}{\psi} \qquad \frac{\phi[x]}{\bigwedge x.\ \phi[x]}\star \qquad \frac{\bigwedge x.\ \phi[x]}{\phi[t]}$$

$$\frac{[\![\psi_1; \ldots; \psi_m]\!] \Longrightarrow \psi \qquad [\![\phi_1; \ldots; \phi_n]\!] \Longrightarrow \phi}{[\![\phi_1\sigma; \ldots; \phi_{i-1}\sigma; \psi_1\sigma; \ldots; \psi_m\sigma; \phi_{i+1}\sigma; \ldots; \phi_n\sigma]\!] \Longrightarrow \phi\sigma}(\psi\sigma = \phi_i\sigma)$$

Isabelle/Pure

Basic Objects

simply-typed λ-terms with equality modulo α (renaming of bound variables), β (computation), and η (extensionality)

Basic Constructs

Syntax	Example	Description
$\Rightarrow$	$\alpha \Rightarrow \beta$	*function space*
$\equiv :: \alpha \Rightarrow \alpha \Rightarrow \mathbf{prop}$	$P\ x \equiv Q\ x$	*equality*
$\bigwedge :: (\alpha \Rightarrow \mathbf{prop}) \Rightarrow \mathbf{prop}$	$\bigwedge x.\ P\ x$	*all-quantification*
$\Longrightarrow :: \mathbf{prop} \Rightarrow \mathbf{prop} \Rightarrow \mathbf{prop}$	$A \Longrightarrow B$	*implication*

Basic Inference Rules

x not free in assumptions

$$\frac{\text{if } \phi \text{ implies } \psi}{\phi \Longrightarrow \psi} \qquad \frac{\phi \Longrightarrow \psi \quad \phi}{\psi} \qquad \frac{\phi[x]}{\bigwedge x.\ \phi[x]}\star \qquad \frac{\bigwedge x.\ \phi[x]}{\phi[t]}$$

$$\frac{[\![\psi_1; \ldots; \psi_m]\!] \Longrightarrow \psi \qquad [\![\phi_1; \ldots; \phi_n]\!] \Longrightarrow \phi}{[\![\phi_1\sigma; \ldots; \phi_{i-1}\sigma; \psi_1\sigma; \ldots; \psi_m\sigma; \phi_{i+1}\sigma; \ldots; \phi_n\sigma]\!] \Longrightarrow \phi\sigma}(\psi\sigma = \phi_i\sigma)$$

Isabelle/HOL

Axioms

refl:	$t = t$
subst:	$\llbracket s = t; P\ s \rrbracket \Longrightarrow P\ t$
ext:	$(\bigwedge x.\ f\ x = g\ x) \Longrightarrow f = g$
the-eq-trivial:	$(\varepsilon x.\ x = a) = a$
impl:	$(P \Longrightarrow Q) \Longrightarrow P \longrightarrow Q$
mp:	$\llbracket P \longrightarrow Q; P \rrbracket \Longrightarrow Q$
iff:	$\llbracket P \longrightarrow Q; Q \longrightarrow P \rrbracket \Longrightarrow P = Q$
True-or-False:	$P = \texttt{True} \vee P = \texttt{False}$
eq-reflection:	$x = y \longrightarrow x \equiv y$

Isabelle/HOL

Axioms

refl:	$t = t$
subst:	$[\![s = t; P\ s]\!] \Longrightarrow P\ t$
ext:	$(\bigwedge x.\ f\ x = g\ x) \Longrightarrow f = g$
the-eq-trivial:	$(\varepsilon x.\ x = a) = a$
impl:	$(P \Longrightarrow Q) \Longrightarrow P \longrightarrow Q$
mp:	$[\![P \longrightarrow Q; P]\!] \Longrightarrow Q$
iff:	$[\![P \longrightarrow Q; Q \longrightarrow P]\!] \Longrightarrow P = Q$
True-or-False:	$P = \mathtt{True} \vee P = \mathtt{False}$
eq-reflection:	$x = y \longrightarrow x \equiv y$

Definitions

$$
\begin{aligned}
\mathtt{True} &\equiv (\lambda x.\ x) = (\lambda x.\ x) \\
\forall x.\ P\ x &\equiv P = (\lambda x.\ \mathtt{True}) \\
\exists x.\ P\ x &\equiv \forall Q.\ (\forall x.\ P\ x \longrightarrow Q) \longrightarrow Q \\
\mathtt{False} &\equiv \forall P.\ P \\
\neg P &\equiv P \longrightarrow \mathtt{False} \\
P \wedge Q &\equiv \forall R.\ (P \longrightarrow Q \longrightarrow R) \longrightarrow R \\
P \vee Q &\equiv \forall R.\ (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R
\end{aligned}
$$

Higher-Order Logic

- HOL = Functional Programming + Logic
- data types (**`datatype`**)
- recursive functions (**`fun`**)
- logical operators ($\wedge$, $\vee$, $\longrightarrow$, $\forall$, $\exists$, ...)

Example - Append in Isabelle/HOL

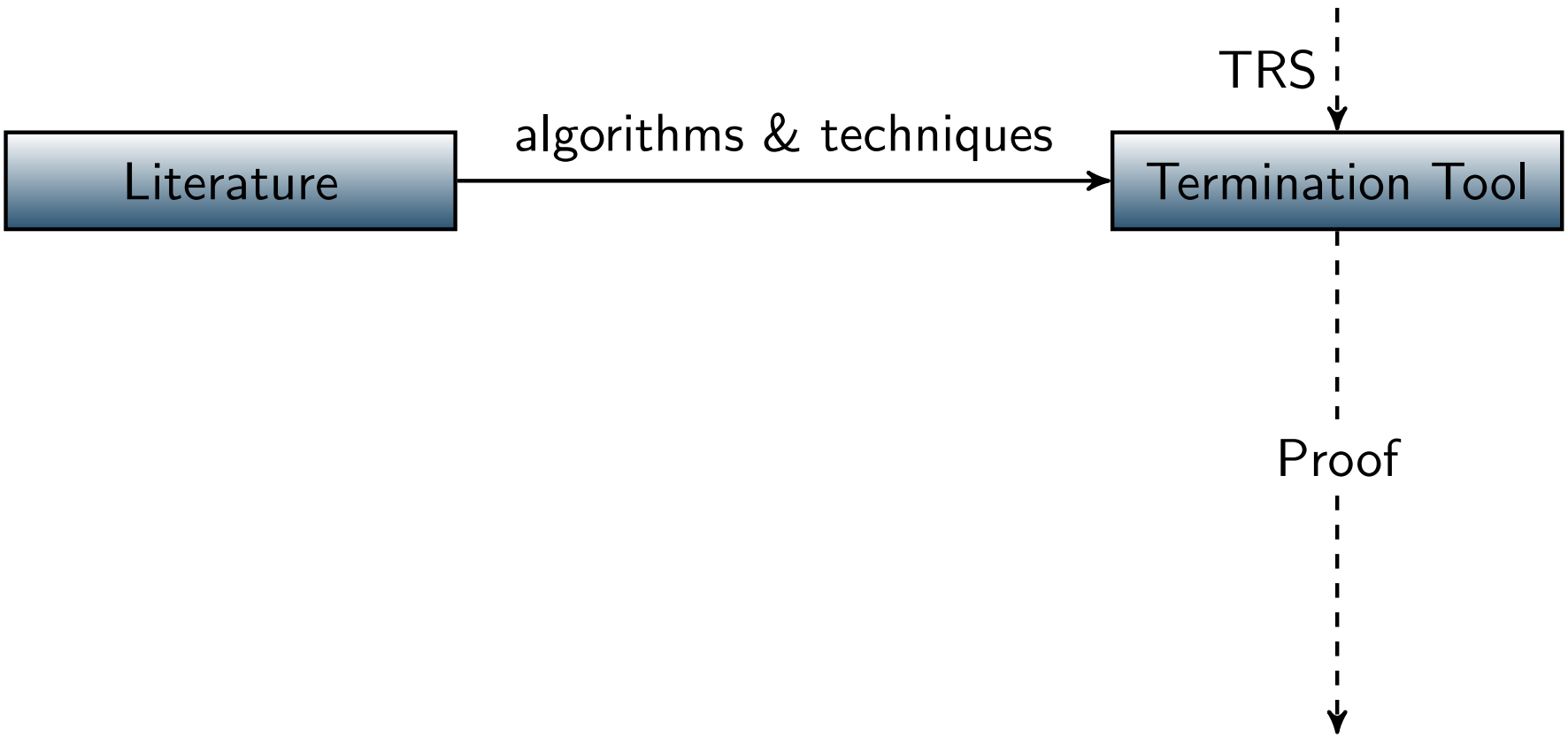
```
fun
  append :: "'a list => 'a list => 'a list"
    (infixr "@" 65)
where
  "[] @ ys = ys"
| "(x#xs) @ ys = x # (xs @ ys)"
```

Demo - Functional Programming in HOL

Certification of Termination Proofs

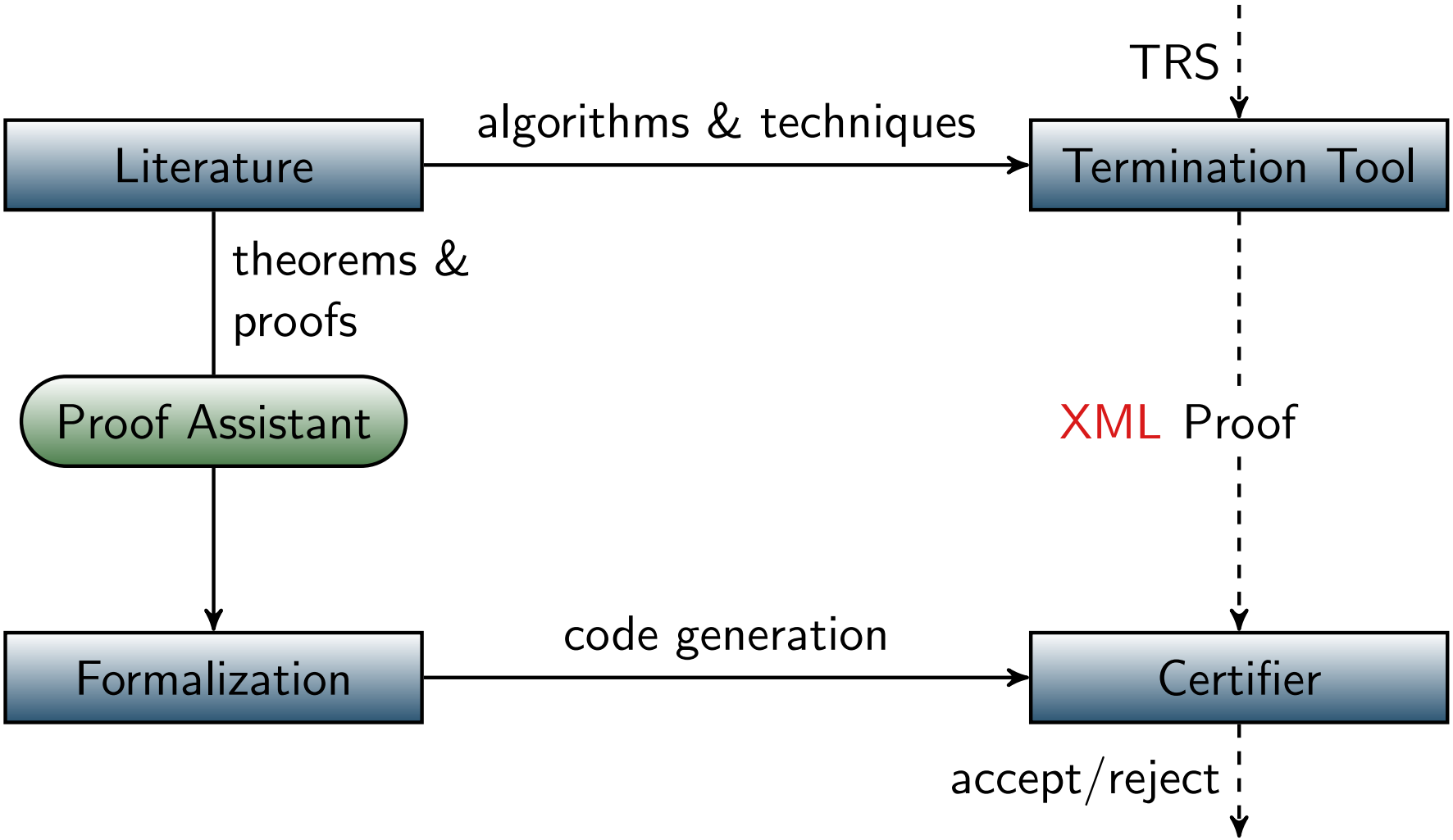
Automatic Termination Analysis

- (automatically) provide evidence of (non)termination



Automatic Termination Analysis

- (automatically) provide evidence of (non)termination
- (automatically) certify correctness of evidence



Example - Append as TRS

$$\mathsf{append}(\mathsf{nil}, ys) \longrightarrow ys$$
$$\mathsf{append}(\mathsf{cons}(x, xs), ys) \longrightarrow \mathsf{cons}(x, \mathsf{append}(xs, ys))$$

Demo - Termination Tools and Certification

An Isabelle Formalization of Rewriting

IsaFoR

- `http://cl-informatik.uibk.ac.at/software/ceta`
- abstract rewriting, relative rewriting, term rewriting, termination, the dependency pair framework, ...

An Isabelle Formalization of Rewriting

- `http://cl-informatik.uibk.ac.at/software/ceta`
- abstract rewriting, relative rewriting, term rewriting, termination, the dependency pair framework, ...

An Isabelle Formalization of Rewriting

IsaFoR

- `http://cl-informatik.uibk.ac.at/software/ceta`
- abstract rewriting, relative rewriting, term rewriting, termination, the dependency pair framework, ...

CPF

- *Certification Problem Format*
- common XML format
- output of termination tools
- input for certifiers

An Isabelle Formalization of Rewriting

IsaFoR

- `http://cl-informatik.uibk.ac.at/software/ceta`
- abstract rewriting, relative rewriting, term rewriting, termination, the dependency pair framework, ...

CPF

- *Certification Problem Format*
- common XML format
- output of termination tools
- input for certifiers

CeTA

- *Certify Termination Analysis*
- code generated (and thus verified) Haskell program

An Isabelle Formalization of Rewriting

IsaFoR

- `http://cl-informatik.uibk.ac.at/software/ceta`
- abstract rewriting, relative rewriting, term rewriting, termination, the dependency pair framework, ...

CPF

- *Certification Problem Format*
- common XML format
- output of termination tools
- input for certifiers

CeTA

- *Certify Termination Analysis*
- code generated (and thus verified) Haskell program

Demo - Code Generation

Termination of Isabelle/HOL Functions

Example - Why Termination Matters

```
fun f :: "nat => nat" where
  "f x = Suc (f x)"
```

Proving Termination of Isabelle/HOL Functions

Built-In Automation

- primitive recursion (syntactic)
- lexicographic orders
- size-change principle

Proving Termination of Isabelle/HOL Functions

Built-In Automation

- primitive recursion (syntactic)
- lexicographic orders
- size-change principle

Alternatives . . .

- provide appropriate measure manually
- do full termination proof inside HOL

Proving Termination of Isabelle/HOL Functions

Built-In Automation

- primitive recursion (syntactic)
- lexicographic orders
- size-change principle

Alternatives . . .

- provide appropriate measure manually
- do full termination proof inside HOL

Or

- use external (fully automatic) termination tool

Proving Termination of Isabelle/HOL Functions

Built-In Automation

- primitive recursion (syntactic)
- lexicographic orders
- size-change principle

Alternatives . . .

- provide appropriate measure manually
- do full termination proof inside HOL

Or

- use external (fully automatic) termination tool

Termination Techniques

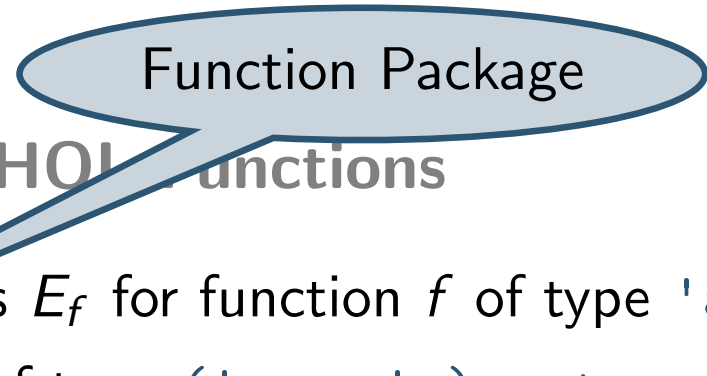
transformations (semantic labeling, root-labeling, uncurrying, . . .), interpretations (polynomial, matrix, arctic, . . .), orders (Knuth-Bendix, lexicographic, multiset, RPO, . . .), advanced methods (dependency pairs, dependency graph, usable rules, . . .),

Two Worlds

Termination of Isabelle/HOL Functions

- input: defining equations E_f for function f of type `'a => 'b`
- output: call-relation $\mathcal{C}_f$ of type `('a × 'a) set`
- goal: show well-foundedness of $\mathcal{C}_f$

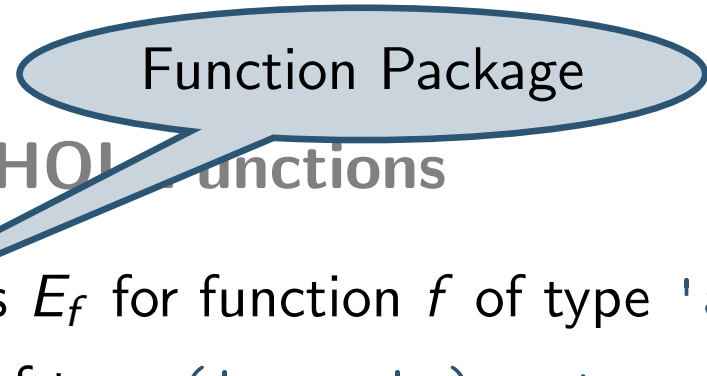
Two Worlds



Termination of Isabelle/HOL Functions

- input: defining equations E_f for function f of type `'a => 'b`
- output: call-relation $\mathcal{C}_f$ of type `('a × 'a) set`
- goal: show well-foundedness of $\mathcal{C}_f$

Two Worlds



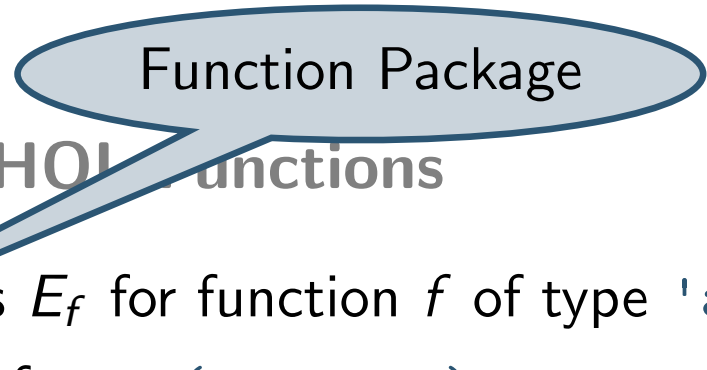
Termination of Isabelle/HOL Functions

- input: defining equations E_f for function f of type `'a => 'b`
- output: call-relation $\mathcal{C}_f$ of type `('a × 'a) set`
- goal: show well-foundedness of $\mathcal{C}_f$

Termination of TRSs

- input: TRS $\mathcal{R}$
- goal: show well-foundedness of rewrite relation $\rightarrow_{\mathcal{R}}$

Two Worlds



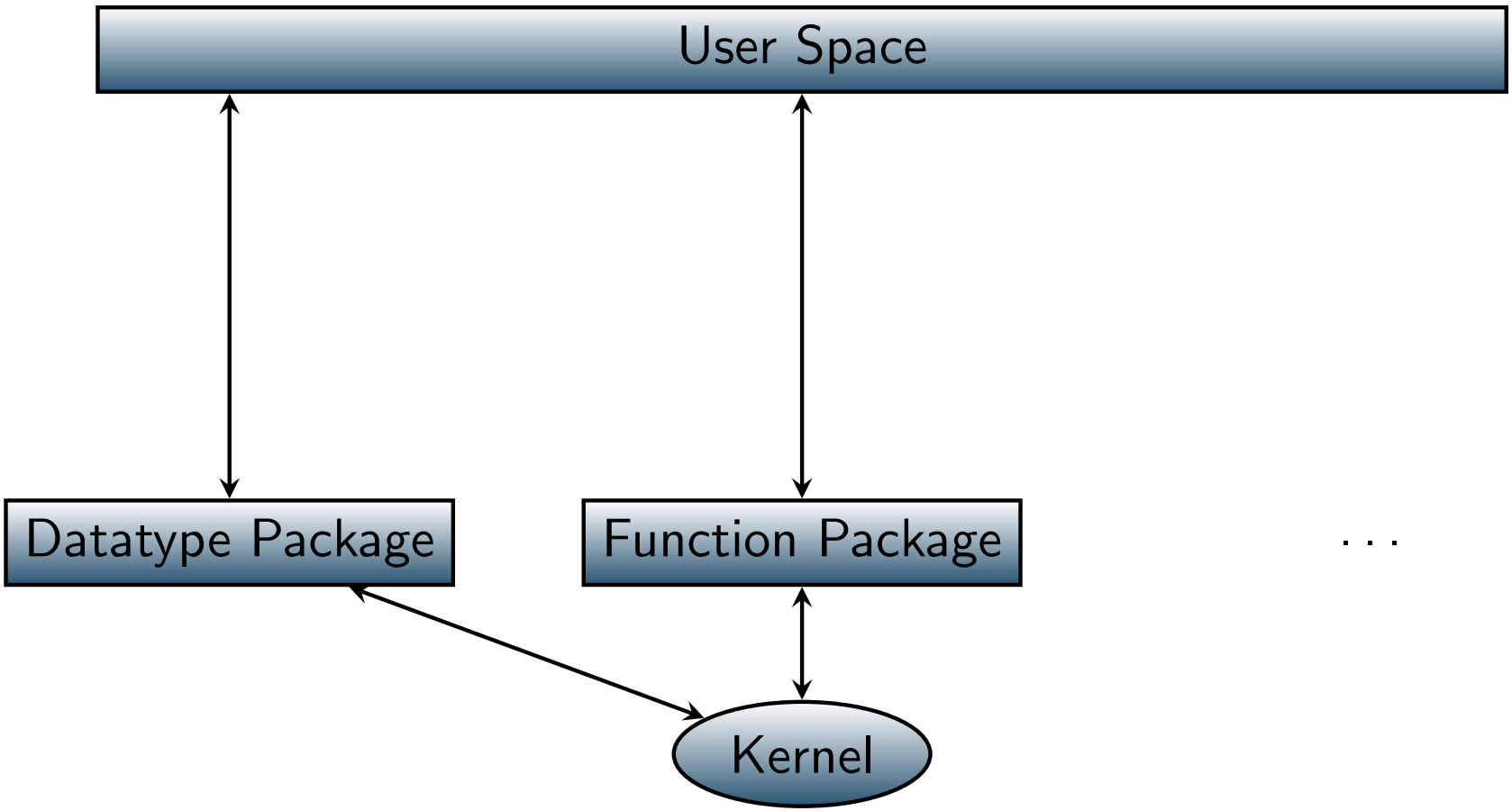
Termination of Isabelle/HOL Functions

- input: defining equations E_f for function f of type `'a => 'b`
- output: call-relation $\mathcal{C}_f$ of type `('a × 'a) set`
- goal: show well-foundedness of $\mathcal{C}_f$

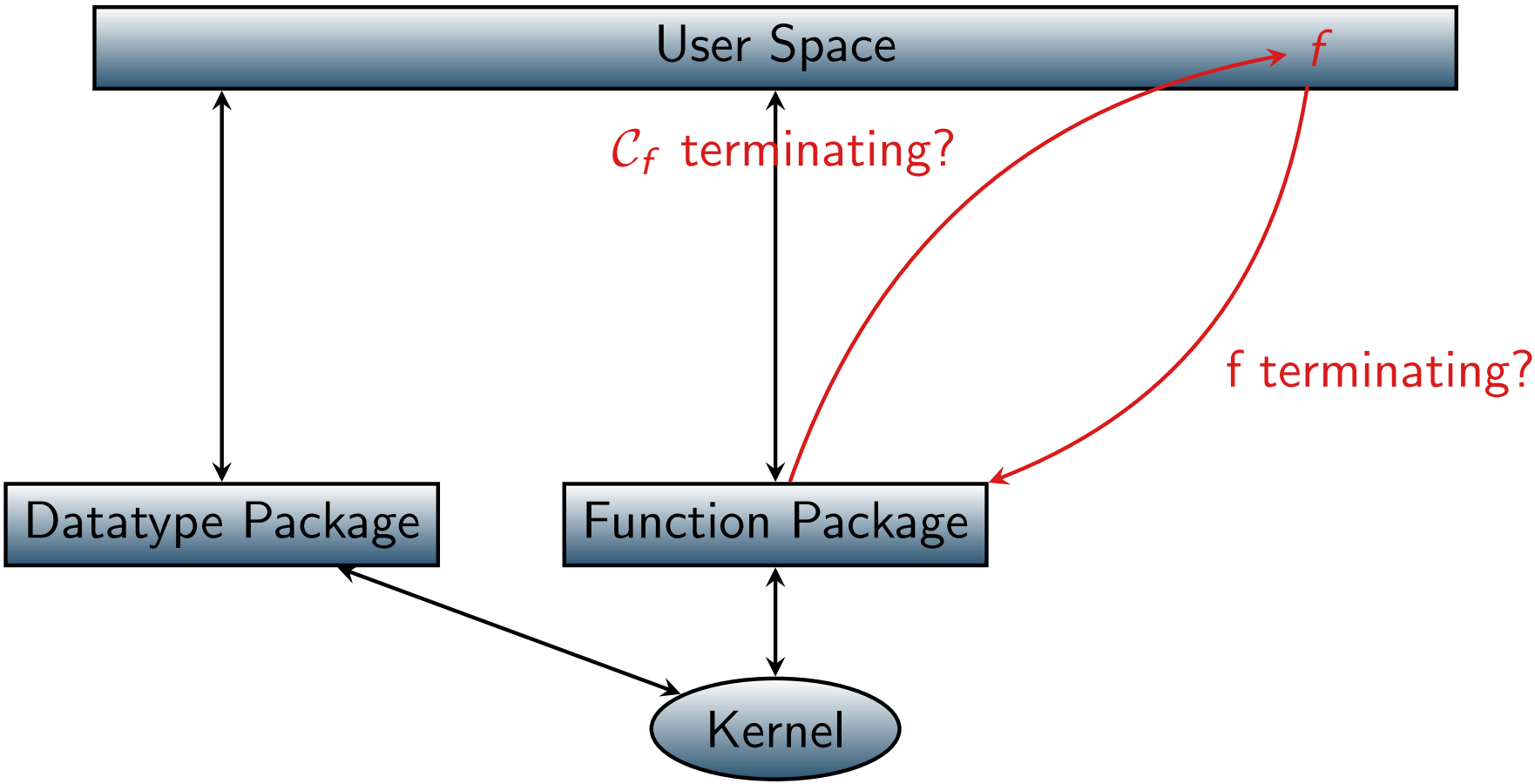
Termination of TRSs

- input: TRS $\mathcal{R}$
- goal: show well-foundedness of rewrite relation $\rightarrow_{\mathcal{R}}$

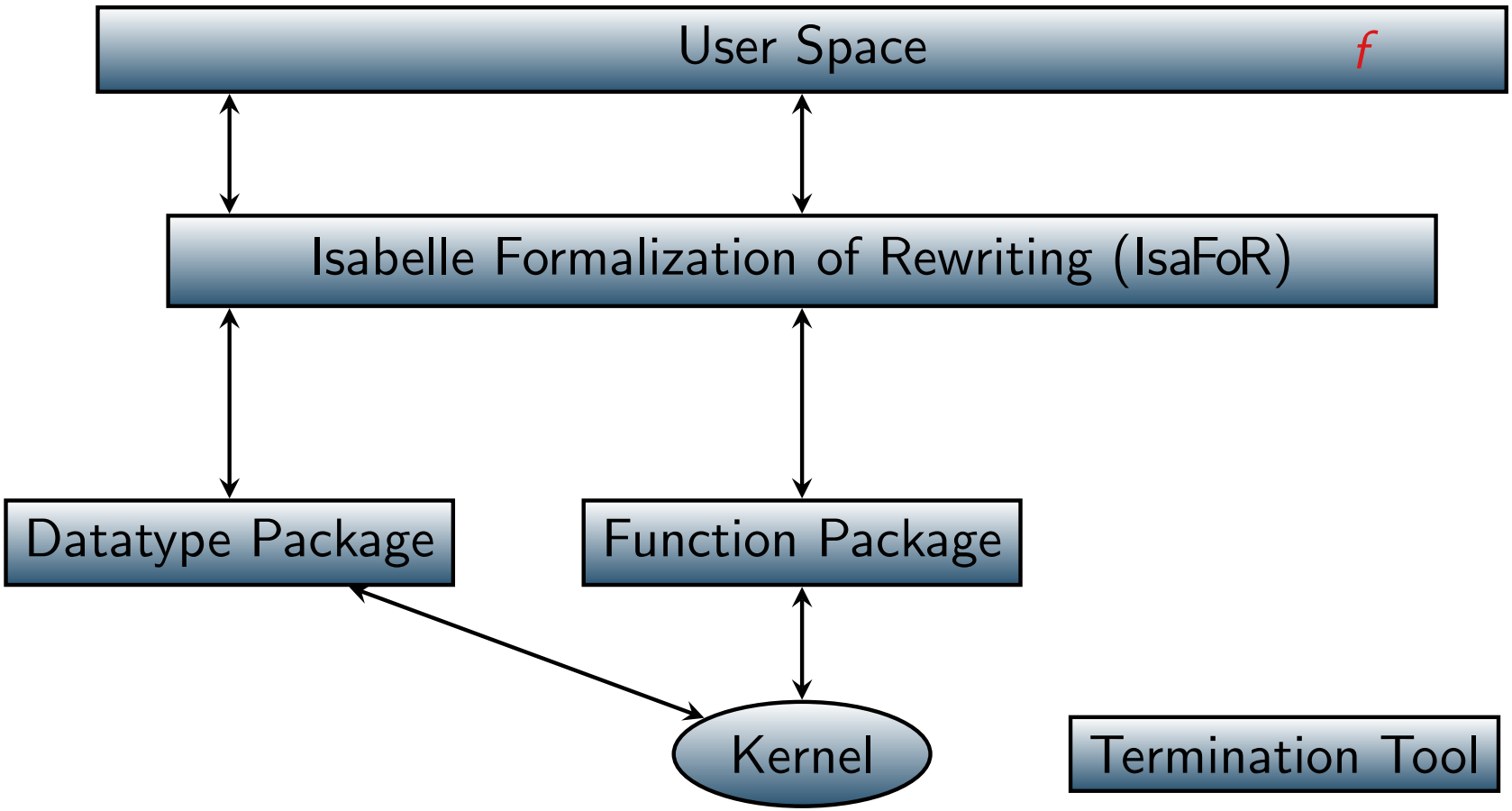
The Big Picture



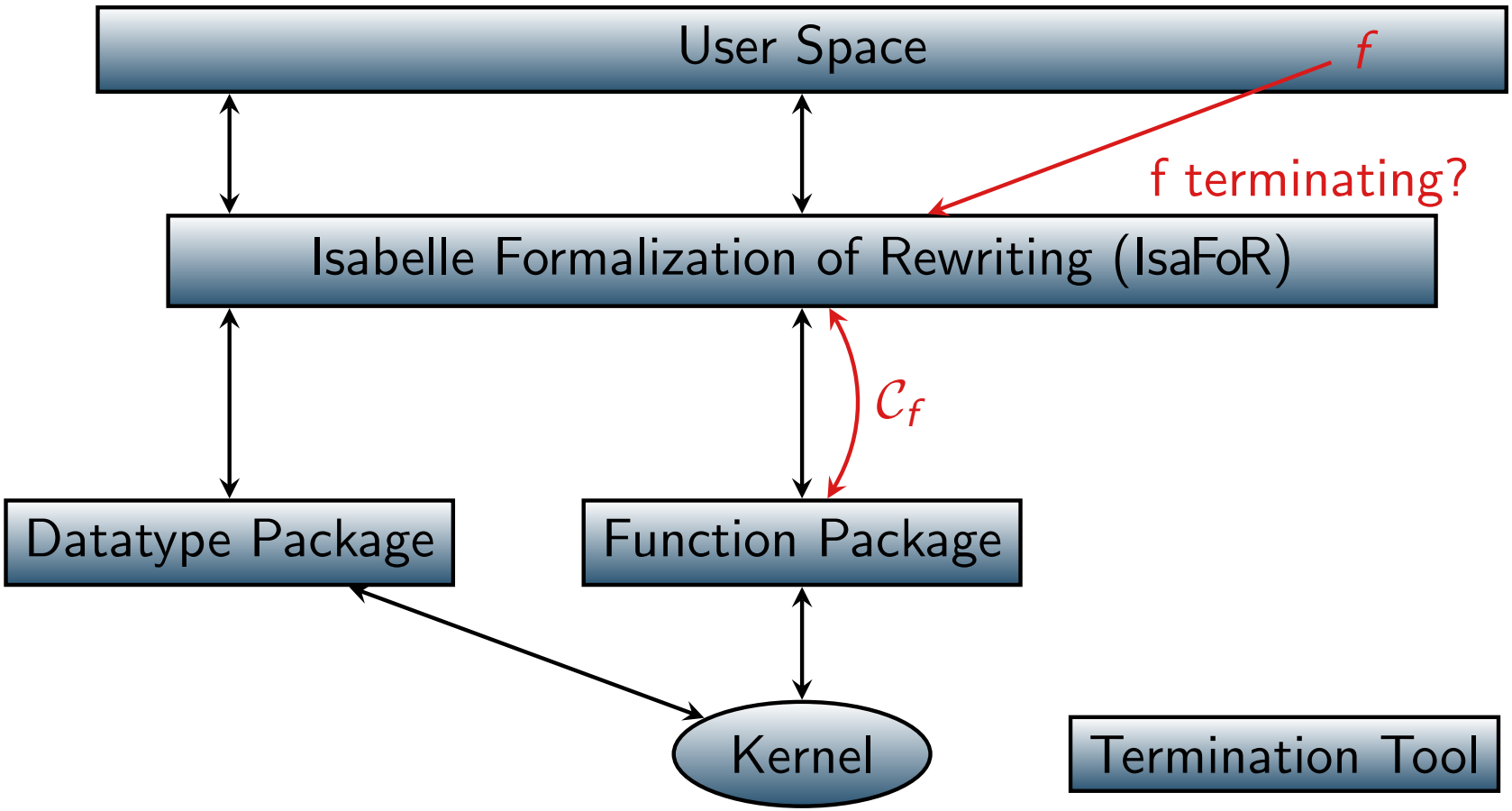
The Big Picture



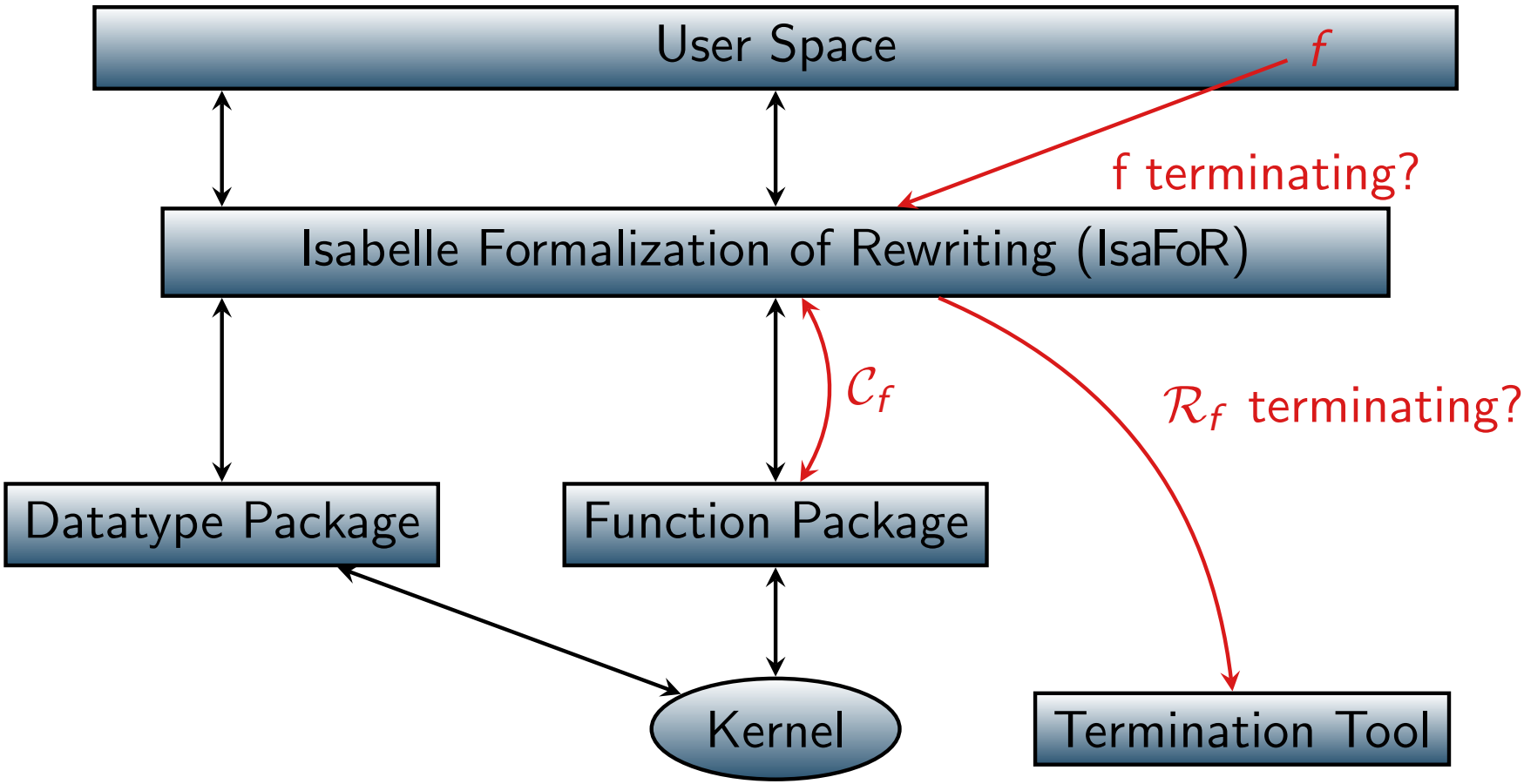
The Big Picture



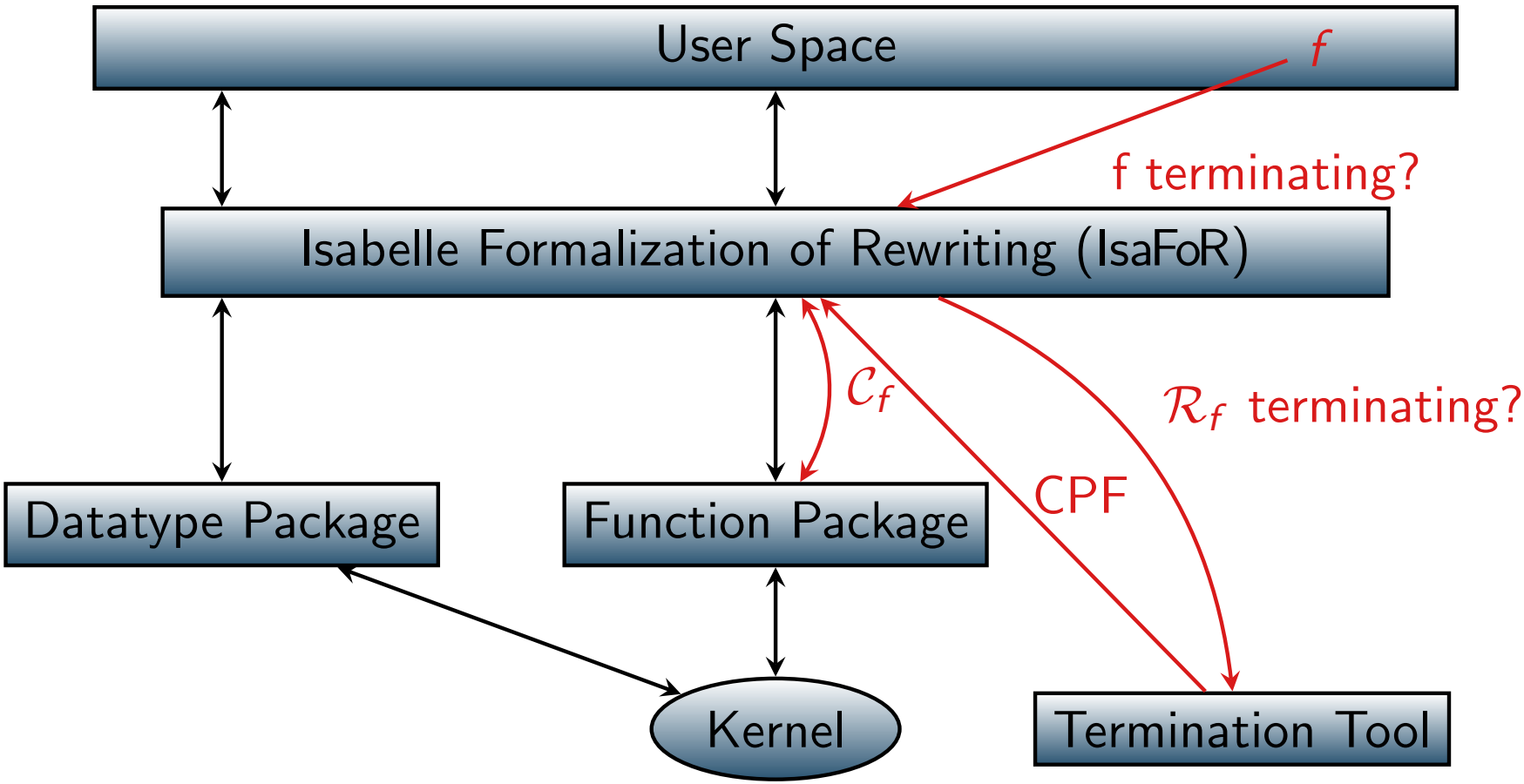
The Big Picture



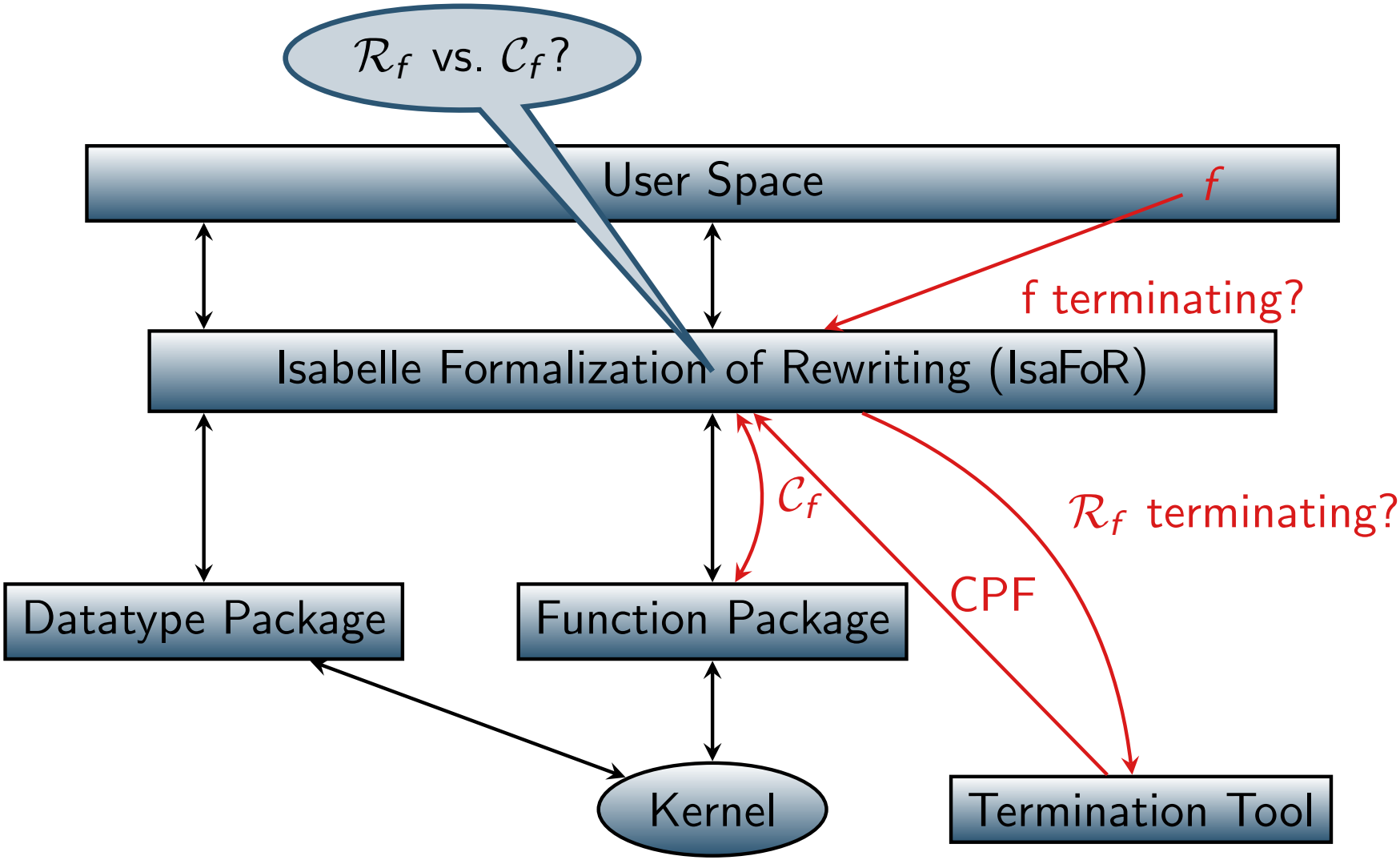
The Big Picture



The Big Picture



The Big Picture



Necessary Glue

- import CPF certificate into Isabelle (using IsaFoR)
- generate TRS $\mathcal{R}_f$ corresponding to definition of function f
- relate termination of $\rightarrow_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Necessary Glue

- import CPF certificate into Isabelle (using IsaFoR)
- generate TRS $\mathcal{R}_f$ corresponding to definition of function f
- relate termination of $\rightarrow_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Encoding Function Specifications

Necessary Glue

- import CPF certificate into Isabelle (using IsaFoR)
- generate TRS $\mathcal{R}_f$ corresponding to definition of function f
- relate termination of $\rightarrow_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Encoding Function Specifications

- internal type

```
term = Var string | Fun string (term list)
```

Necessary Glue

- import CPF certificate into Isabelle (using IsaFoR)
- generate TRS $\mathcal{R}_f$ corresponding to definition of function f
- relate termination of $\rightarrow_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Encoding Function Specifications

- internal type

```
term = Var string | Fun string (term list)
```

- encoding Isabelle/HOL expressions

$$\text{ENC}(f\ \vec{e}_n) = \texttt{Fun f}\ [\text{ENC}(e_1), \ldots, \text{ENC}(e_n)]$$
$$\text{ENC}(x) = \texttt{Var}\ x$$

Necessary Glue

- import CPF certificate into Isabelle (using IsaFoR)
- generate TRS $\mathcal{R}_f$ corresponding to definition of function f
- relate termination of $\to_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Encoding Function Specifications

- internal type

```
term = Var string | Fun string (term list)
```

- encoding Isabelle/HOL expressions

$$\textsc{enc}(f\ \vec{e}_n) = \texttt{Fun f}\ [\textsc{enc}(e_1), \ldots, \textsc{enc}(e_n)]$$
$$\textsc{enc}(x) = \texttt{Var}\ x$$

- rewrite rules for equations $l_1 = r_1, \ldots, l_k = r_k$

$$\begin{aligned}\textsc{rules}(f) = \{\ & \textsc{enc}(l_1) \to \textsc{enc}(r_1),\\ & \vdots\\ & \textsc{enc}(l_k) \to \textsc{enc}(r_k)\}\end{aligned}$$

Example - An Odd Even/Odd Function

```
function evenodd :: "bool => nat => bool" where
  "evenodd True 0 = False"
| "evenodd False x = (¬ (evenodd True x))"
| "evenodd True (Suc x) = evenodd False x"
```

Example - An Odd Even/Odd Function

```
function evenodd :: "bool => nat => bool" where
  "evenodd True 0 = False"
| "evenodd False x = (¬ (evenodd True x))"
| "evenodd True (Suc x) = evenodd False x"
```

Goal

```
1. wf ?R
2. ⋀x. ((True,x),(False,x)) ∈ ?R
3. ⋀x. ((False,x),(True,Suc x)) ∈ ?R
```

Demo - Termination by External Tool

Main Goal

- embedding *emb* of type `'a => term`
- prove that $\mathcal{C}_f$ is contained in

$$\{(x, y) \mid \texttt{Fun f [}\mathit{emb}\ x\texttt{]}\ (\rightarrow_{\mathcal{R}_f} \cup \rhd)^{+}\ \texttt{Fun f [}\mathit{emb}\ y\texttt{]}\}$$

Main Goal

- embedding *emb* of type `'a => term`
- prove that $\mathcal{C}_f$ is contained in

$$\{(x, y) \mid \texttt{Fun f [}emb\ x\texttt{]}\ (\rightarrow_{\mathcal{R}_f} \cup \rhd)^+\ \texttt{Fun f [}emb\ y\texttt{]}\}$$

Simulation Lemmas

- n-ary function f
- lemma:

$$\texttt{Fun f [}emb\ x_1, \ldots, emb\ x_n\texttt{]} \rightarrow^*_{\mathcal{R}_f} emb\ (f\ \vec{x}_n)$$

Restrictions

Supported

- variables, function applications
- case-expressions (let, if)

Restrictions

Supported

- variables, function applications
- case-expressions (let, if)

Not Supported

- no data type constructors with functional arguments
- no "lambdas"
- no function variables
- no overlapping patterns
- no incomplete patterns
- no mutual recursion

Summary

- underlying principles of LCF-style proof assistants
- basics of HOL
- prove termination of HOL functions by external tools
- free user from tedious termination proofs (open problem since certification of termination proofs started)

Bibliography


A. Krauss et al.
Termination of Isabelle Functions via Termination of Rewriting.
ITP'11. doi:10.1007/978-3-642-22863-6_13.


C. Sternagel.
Automatic Certification of Termination Proofs. PhD'10.
www.jaist.ac.jp/~c-sterna/publications/PhD-thesis.pdf.


R. Thiemann and C. Sternagel.
Certification of Termination Proofs using CeTA. TPHOLs'09.
doi:10.1007/978-3-642-03359-9_31.


M. Gordon.
From LCF to HOL: A Short History. 2000.


L. C. Paulson.
Isabelle: The Next 700 Theorem Provers. 1990.