Termination of Isabelle Functions via Termination of Rewriting

ITP 2011

Alexander Krauss **Christian Sternagel** René Thiemann
Carsten Fuhs Jürgen Giesl

Technische Universität München
University of Innsbruck
RWTH Aachen University

August 22, 2011

What?

Why?

How?

What?

Why?

How?

What?

Why?

How?

talk

What? talk

Why?

How?

paper

Functional Programming in Isabelle/HOL

Datatypes

```
datatype tree =
  E (*the empty tree*)
| N tree nat tree (*node having 2 subtrees*)
```

Functional Programming in Isabelle/HOL

Datatypes

```
datatype tree =
  E (*the empty tree*)
| N tree nat tree (*node having 2 subtrees*)
```

Recursive Functions

```
(*assuming search tree property*)
fun getmax :: "tree => nat" where
  "getmax E            =  0"
| "getmax (N _ x E)   =  x"
| "getmax (N _ _ r)   =  getmax r"
```

Functional Programming in Isabelle/HOL

Datatypes

```
datatype tree =
  E (*the empty tree*)
| N tree nat tree (*node having 2 subtrees*)
```

Recursive Functions

need to be total

```
(*assuming search tree property*)
fun getmax :: "tree => nat" where
  "getmax E           =  0"
| "getmax (N _ x E)   =  x"
| "getmax (N _ _ r)   =  getmax r"
```

Functional Programming in Isabelle/HOL

Datatypes

```
datatype tree =
  E (*the empty tree*)
| N tree nat tree (*node having 2 subtrees*)
```

Recursive Functions

need to be total

```
(*assuming search tree property*)
fun getmax :: "tree => nat" where
  "getmax E           =  0"
| "getmax (N _ x E)   =  x"
| "getmax (N _ _ r)   =  getmax r"
```

Consider

```
fun f where "f x = f x + 1"
```

Proving Totality of Isabelle/HOL Functions

Built-In Automation

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

Proving Totality of Isabelle/HOL Functions

Built-In Automation

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

Alternatives . . .

- provide appropriate measure manually
- do the full totality/termination proof inside HOL

Proving Totality of Isabelle/HOL Functions

Built-In Automation

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

Alternatives ...

- provide appropriate measure manually
- do the full totality/termination proof inside HOL

Or

- use external (fully automatic) termination tool

Proving Totality of Isabelle/HOL Functions

Built-In Automation

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

Alternatives . . .

- provide appropriate measure manually
- do the full totality/termination proof inside HOL

Or

What

- use external (fully automatic) termination tool

First-Order Term Rewriting - "Replacing Equals by Equals"

Definition by Example

$$\begin{aligned} \mathsf{getmax}(\mathsf{E}) &\to \mathsf{0} \\ \mathsf{getmax}(\mathsf{N}(\mathsf{x}, \mathsf{y}, \mathsf{E})) &\to \mathsf{y} \\ \mathsf{getmax}(\mathsf{N}(\mathsf{x}, \mathsf{y}, \mathsf{N}(\mathsf{z}, \mathsf{u}, \mathsf{v}))) &\to \mathsf{getmax}(\mathsf{N}(\mathsf{z}, \mathsf{u}, \mathsf{v})) \end{aligned}$$

First-Order Term Rewriting - “Replacing Equals by Equals”

Definition by Example

term rewrite system (TRS)

$$\begin{aligned} \mathsf{getmax}(\mathsf{E}) &\to \mathsf{0} \\ \mathsf{getmax}(\mathsf{N}(x, y, \mathsf{E})) &\to y \\ \mathsf{getmax}(\mathsf{N}(x, y, \mathsf{N}(z, u, v))) &\to \mathsf{getmax}(\mathsf{N}(z, u, v)) \end{aligned}$$

First-Order Term Rewriting - "Replacing Equals by Equals"

Definition by Example

term rewrite system (TRS)

$$\mathsf{getmax}(\mathsf{E}) \to 0$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{E})) \to y$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{N}(z, u, v))) \to \mathsf{getmax}(\mathsf{N}(z, u, v))$$

$$\mathsf{getmax}(\mathsf{N}(\mathsf{E}, 1, \mathsf{N}(\mathsf{E}, 2, \mathsf{E})))$$

First-Order Term Rewriting - "Replacing Equals by Equals"

Definition by Example

term rewrite system (TRS)

$$
\begin{aligned}
\mathsf{getmax}(\mathsf{E}) &\to \mathsf{0} \\
\mathsf{getmax}(\mathsf{N}(\mathsf{x},\mathsf{y},\mathsf{E})) &\to \mathsf{y} \\
\mathsf{getmax}(\mathsf{N}(\mathsf{x},\mathsf{y},\mathsf{N}(\mathsf{z},\mathsf{u},\mathsf{v}))) &\to \mathsf{getmax}(\mathsf{N}(\mathsf{z},\mathsf{u},\mathsf{v}))
\end{aligned}
$$

$$\mathsf{getmax}(\mathsf{N}(\mathsf{E},1,\mathsf{N}(\mathsf{E},2,\mathsf{E}))) \to \mathsf{getmax}(\mathsf{N}(\mathsf{E},2,\mathsf{E}))$$

First-Order Term Rewriting - "Replacing Equals by Equals"

Definition by Example

term rewrite system (TRS)

$$\begin{aligned} \mathsf{getmax}(\mathsf{E}) &\to 0 \\ \mathsf{getmax}(\mathsf{N}(x, y, \mathsf{E})) &\to y \\ \mathsf{getmax}(\mathsf{N}(x, y, \mathsf{N}(z, u, v))) &\to \mathsf{getmax}(\mathsf{N}(z, u, v)) \end{aligned}$$

$$\mathsf{getmax}(\mathsf{N}(\mathsf{E}, 1, \mathsf{N}(\mathsf{E}, 2, \mathsf{E}))) \to \mathsf{getmax}(\mathsf{N}(\mathsf{E}, 2, \mathsf{E})) \to 2$$

First-Order Term Rewriting - "Replacing Equals by Equals"

Definition by Example

term rewrite system (TRS)

$$\mathsf{getmax}(\mathsf{E}) \rightarrow 0$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{E})) \rightarrow y$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{N}(z, u, v))) \rightarrow \mathsf{getmax}(\mathsf{N}(z, u, v))$$

rewrite sequence

$$\mathsf{getmax}(\mathsf{N}(\mathsf{E}, 1, \mathsf{N}(\mathsf{E}, 2, \mathsf{E}))) \rightarrow \mathsf{getmax}(\mathsf{N}(\mathsf{E}, 2, \mathsf{E})) \rightarrow 2$$

First-Order Term Rewriting - “Replacing Equals by Equals”

Definition by Example

term rewrite system (TRS)

$$\mathsf{getmax}(\mathsf{E}) \to 0$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{E})) \to y$$
$$\mathsf{getmax}(\mathsf{N}(x, y, \mathsf{N}(z, u, v))) \to \mathsf{getmax}(\mathsf{N}(z, u, v))$$

rewrite sequence

$$\mathsf{getmax}(\mathsf{N}(\mathsf{E}, 1, \mathsf{N}(\mathsf{E}, 2, \mathsf{E}))) \to \mathsf{getmax}(\mathsf{N}(\mathsf{E}, 2, \mathsf{E})) \to 2$$

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, . . .), . . .

First-Order Term Rewriting

Termination Tools

AProVE, C*i*ME, Jambox, Matchbox, NTI, VMTL, Torpa, TPA, $\mathsf{T_TT_2}$, ...

First-Order Term Rewriting

Termination Tools

AProVE, C*i*ME, Jambox, Matchbox, NTI, VMTL, Torpa, TPA, $\mathsf{T_TT_2}$, ...

Problems

- no uniform output
- not stable (introduction of new techniques)
- huge proofs (several megabytes)
- complex and tuned for efficiency (thus sometimes buggy)

First-Order Term Rewriting

Termination Tools

AProVE, C*i*ME, Jambox, Matchbox, NTI, VMTL, Torpa, TPA, $\mathsf{T_TT_2}$, ...

Problems

- no uniform output
- not stable (introduction of new techniques)
- huge proofs (several megabytes)
- complex and tuned for efficiency (thus sometimes buggy)

Solutions

- XML format for proofs (Certification Problem Format - CPF)
- automatic certification of CPF files (using a proof assistant)

Two Worlds

Totality 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

Totality of Isabelle/HOL Functions

Function Package

- 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

Totality of Isabelle/HOL Functions

Function Package

- 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 $\to_{\mathcal{R}}$

Two Worlds

Totality of Isabelle/HOL Functions

Function Package

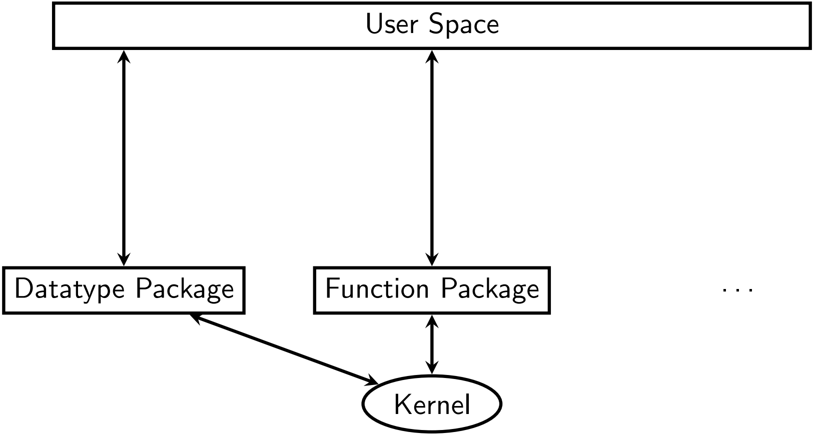
- input: defining equations E_f for function f of type `'a => 'b`
- output: call-relation $\mathcal{C}_f$ of type `('a × 'a) set`
- goal: show well-foundedness of $\mathcal{C}_f$

Termination of TRSs

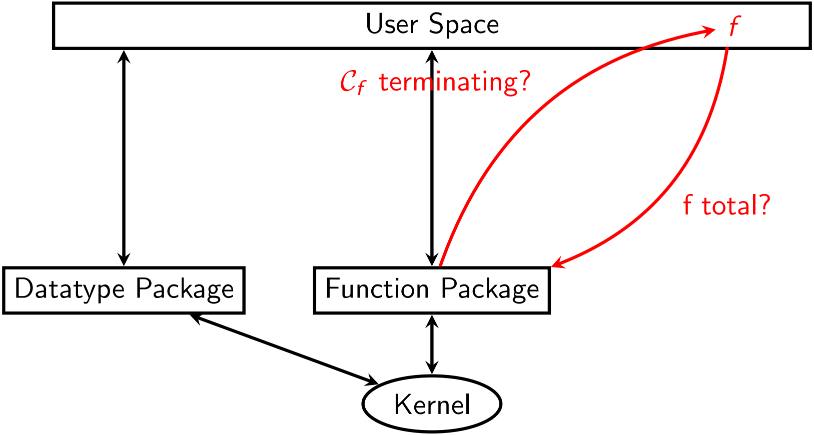
Termination Tool

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

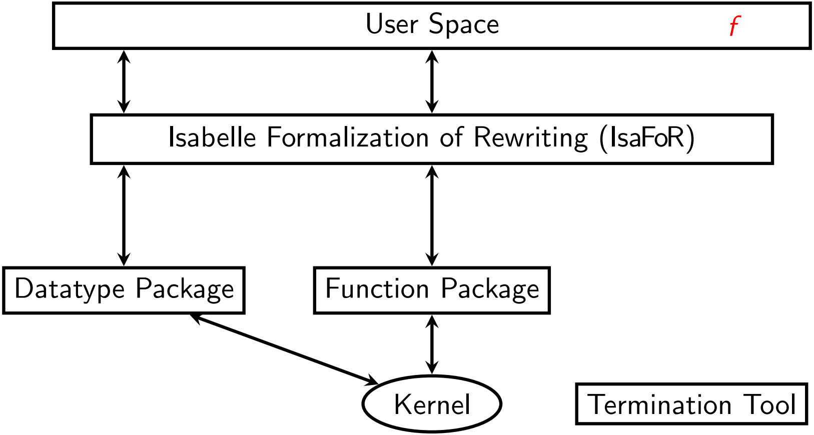
The Big Picture



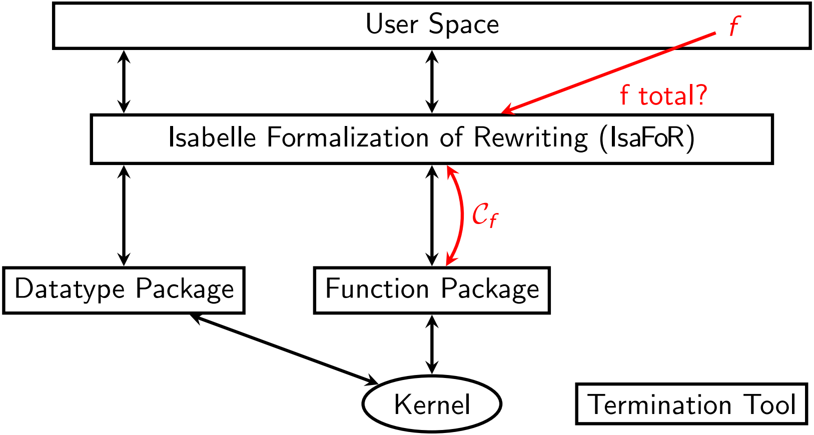
The Big Picture



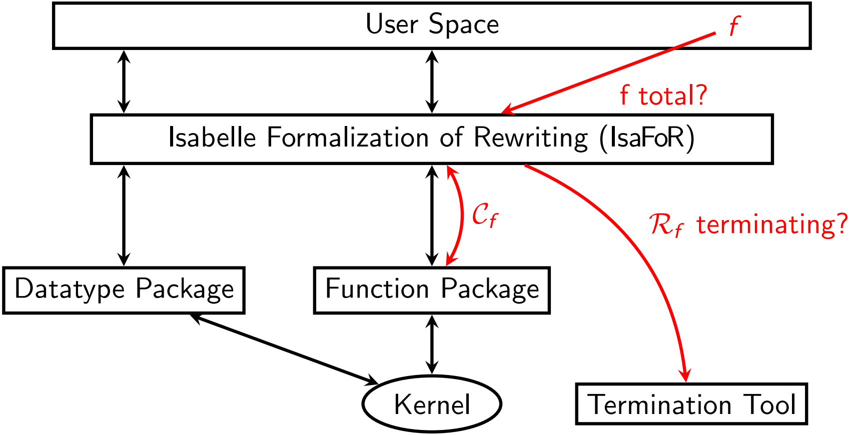
The Big Picture



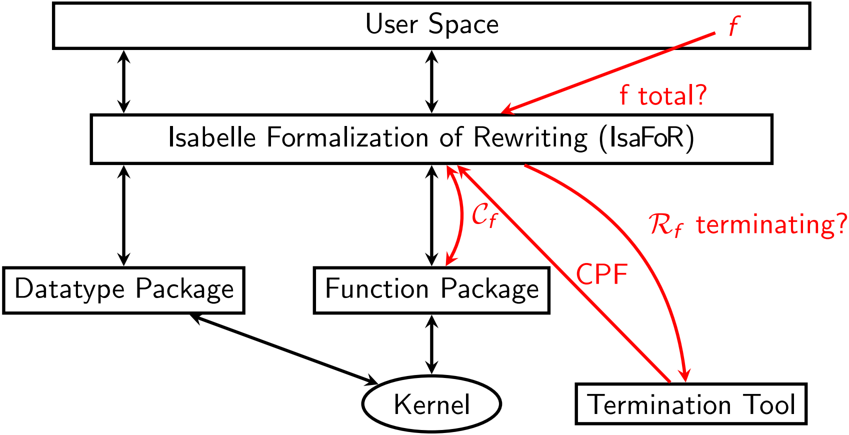
The Big Picture



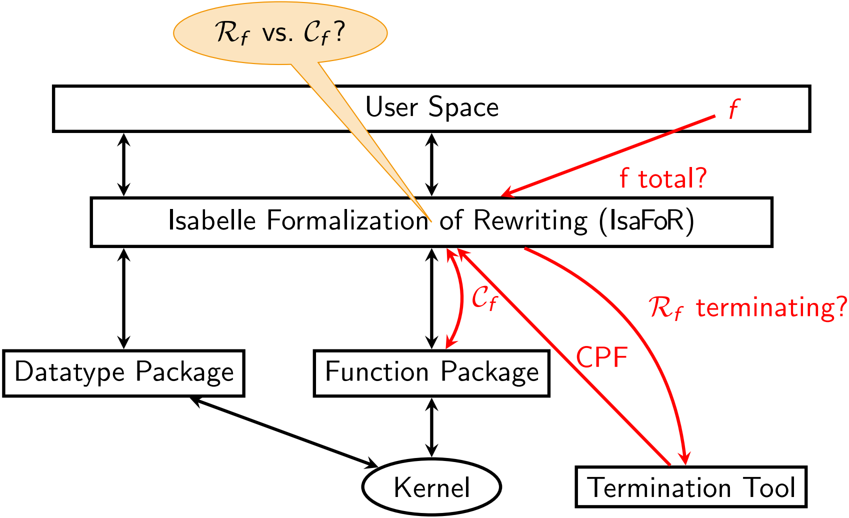
The Big Picture



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 $\to_{\mathcal{R}_f}$ 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 $\to_{\mathcal{R}_f}$ to well-foundedness of $\mathcal{C}_f$?

Encoding Function Specifications

- internal type
  `term = Var string | Fun string (term list)`
- encoding Isabelle/HOL expressions

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

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

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

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

$$\textsc{rules}(f) = \{ \begin{array}{l} \textsc{enc}(l_1) \rightarrow \textsc{enc}(r_1), \\ \vdots \\ \textsc{enc}(l_k) \rightarrow \textsc{enc}(r_k)\} \end{array}$$

Main Goal

- encoding *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

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

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

Simulation Lemmas

- n-ary function f
- lemma:

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

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

What

prove termination of Isabelle/HOL functions by external termination tool

Summary

What

prove termination of Isabelle/HOL functions by external termination tool

Why

free user from tedious termination proofs; open problem since certification of termination proofs started

Summary

What

prove termination of Isabelle/HOL functions by external termination tool

Why

free user from tedious termination proofs; open problem since certification of termination proofs started

How

refer to paper and Isabelle/HOL formalization
`http://cl-informatik.uibk.ac.at/software/ceta`