Certification of Constrained Rewriting

René Thiemann
joint work with Christian Sternagel

Computational Logic, University of Innsbruck

05 July 2012
Outline

- Task 5: Certification
- IsaFoR + CeTA
- Progress in Task 5
Task 5: Certification

IsaFoR + CeTA

Progress in Task 5
Overview of Task 5

- abstract goal: certification of constrained rewriting
- 3 concrete goals:

  1. formalization and certification of loops w.r.t. strategy (does a loop $t \rightarrow +C[t\mu]$ respect strategy w.r.t. forbidden patterns?)
  2. formalization and certification of soundness and completeness results of unraveling transformation from conditional to unconditional rewriting
  3. formalization and certification of constrained rewriting steps, taking output of general rewrite tool of Task 1

- working plan: late start

  1st year
  2nd year
  3rd year

  Thiemann
  1M
  2M

  Zankl
  1M
  1M

  PD2
  6M
  12M

  Sternagel
  2M


Overview of Task 5

- abstract goal: certification of constrained rewriting
- 3 concrete goals:
  1. formalization and certification of loops w.r.t. strategy
     (does a loop $t \rightarrow^{+} C[t\mu]$ respect strategy w.r.t. forbidden patterns?)
Overview of Task 5

- abstract goal: certification of constrained rewriting
- 3 concrete goals:
  1. formalization and certification of loops w.r.t. strategy
     (does a loop $t \rightarrow^+ C[t\mu]$ respect strategy w.r.t. forbidden patterns?)
  2. formalization and certification of soundness and completeness results of unraveling transformation from conditional to unconditional rewriting
Overview of Task 5

- abstract goal: certification of constrained rewriting
- 3 concrete goals:
  1. formalization and certification of loops w.r.t. strategy
     (does a loop $t \rightarrow^+ C[t\mu]$ respect strategy w.r.t. forbidden patterns?)
  2. formalization and certification of soundness and completeness results of unraveling transformation from conditional to unconditional rewriting
  3. formalization and certification of constrained rewriting steps, taking output of general rewrite tool of Task 1
Overview of Task 5

- abstract goal: certification of constrained rewriting
- 3 concrete goals:
  1. formalization and certification of loops w.r.t. strategy (does a loop $t \rightarrow^+ C[t\mu]$ respect strategy w.r.t. forbidden patterns?)
  2. formalization and certification of soundness and completeness results of unraveling transformation from conditional to unconditional rewriting
  3. formalization and certification of constrained rewriting steps, taking output of general rewrite tool of Task 1
- working plan: late start

<table>
<thead>
<tr>
<th></th>
<th>1st year</th>
<th>2nd year</th>
<th>3rd year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thiemann</td>
<td>1M</td>
<td>2M</td>
<td>1M</td>
</tr>
<tr>
<td>Zankl</td>
<td>1M</td>
<td>1M</td>
<td>1M</td>
</tr>
<tr>
<td>PD2</td>
<td>6M</td>
<td>12M</td>
<td></td>
</tr>
<tr>
<td>Sternagel</td>
<td></td>
<td>2M</td>
<td></td>
</tr>
</tbody>
</table>
Goal 1: Loops w.r.t. strategy

Consider leftmost-outermost rewriting: $\xrightarrow{lo}$

\[
x == y \rightarrow eq(chk(x), chk(y))
\]
\[
eq(x, x) \rightarrow true
\]
\[
chk(x) \rightarrow false
\]
\[
eq(false, y) \rightarrow false
\]

Above TRS can be used to check equality of arbitrary terms in constant number of reductions

- $t == t \xrightarrow{lo} eq(chk(t), chk(t)) \xrightarrow{lo} true$
Goal 1: Loops w.r.t. strategy

Consider leftmost-outermost rewriting: $\xrightarrow{lo}$

\[
\begin{align*}
x == y & \rightarrow eq(chk(x), chk(y)) \\
eq(x, x) & \rightarrow true \\
chk(x) & \rightarrow false \\
eq(false, y) & \rightarrow false
\end{align*}
\]

Above TRS can be used to check equality of arbitrary terms in constant number of reductions

- $t == t \xrightarrow{lo} eq(chk(t), chk(t)) \xrightarrow{lo} true$
- $s == t \xrightarrow{lo} eq(chk(s), chk(t)) \xrightarrow{lo} eq(false, chk(t)) \xrightarrow{lo} false$ if $s \neq t$
 Goal 1: Loops w.r.t. strategy

Consider leftmost-outermost rewriting: \( \xrightarrow{lo} \)

\[
\begin{align*}
x &= y &\rightarrow& \text{eq(chk}(x), \text{chk}(y)) \\
\text{eq}(x, x) &\rightarrow& \text{true} \\
\text{chk}(x) &\rightarrow& \text{false} \\
\text{eq}(\text{false}, y) &\rightarrow& \text{false}
\end{align*}
\]

Above TRS can be used to check equality of arbitrary terms in constant number of reductions

- \( t = t \xrightarrow{lo} \text{eq(chk}(t), \text{chk}(t)) \xrightarrow{lo} \text{true} \)
- \( s = t \xrightarrow{lo} \text{eq(chk}(s), \text{chk}(t)) \xrightarrow{lo} \text{eq}(\text{false}, \text{chk}(t)) \xrightarrow{lo} \text{false if } s \neq t \)

If strategy is ignored, unwanted behaviour can occur

- \( t = t \rightarrow \text{eq(chk}(t), \text{chk}(t)) \rightarrow \text{eq}(\text{false}, \text{chk}(t)) \rightarrow \text{false} \)
- \( s = t \rightarrow \text{eq(chk}(s), \text{chk}(t)) \rightarrow \text{eq(chk}(t), \text{false}) \rightarrow \text{eq}(\text{false}, \text{false}) \rightarrow \text{true} \)
Goal 1: Loops w.r.t. strategy

Consider leftmost-outermost rewriting: $\xrightarrow{\text{lo}}$

\[
x == y \rightarrow \text{eq}(\text{chk}(x), \text{chk}(y)) \\
\text{eq}(x, x) \rightarrow \text{true} \\
\text{chk}(x) \rightarrow \text{false} \\
\text{eq}(\text{false}, y) \rightarrow \text{false}
\]

Above TRS can be used to check equality of arbitrary terms in constant number of reductions

- $t == t \xrightarrow{\text{lo}} \text{eq}(\text{chk}(t), \text{chk}(t)) \xrightarrow{\text{lo}} \text{true}$
- $s == t \xrightarrow{\text{lo}} \text{eq}(\text{chk}(s), \text{chk}(t)) \xrightarrow{\text{lo}} \text{eq}(\text{false}, \text{chk}(t)) \xrightarrow{\text{lo}} \text{false}$ if $s \neq t$

If strategy is ignored, unwanted behaviour can occur

- $t == t \rightarrow \text{eq}(\text{chk}(t), \text{chk}(t)) \rightarrow \text{eq}(\text{false}, \text{chk}(t)) \rightarrow \text{false}$
- $s == t \rightarrow \text{eq}(\text{chk}(s), \text{chk}(t)) \rightarrow \text{eq}(\text{chk}(t), \text{false}) \rightarrow \text{eq}(\text{false}, \text{false}) \rightarrow \text{true}$

$\Rightarrow$ (Non-)Termination analysis has to be strategy aware
Goal 1: Loops w.r.t. strategy

Considered strategies for rewrite step $s \to_p t$

- innermost (no redex strictly below position $p$)
- outermost (no redex strictly above position $p$)
- forbidden patterns generalizes innermost and outermost strategies
  (includes flexible combination of no redex below/at/above position $p$)
Goal 1: Loops w.r.t. strategy

Considered strategies for rewrite step $s \rightarrow_p t$

- innermost (no redex strictly below position $p$)
- outermost (no redex strictly above position $p$)
- forbidden patterns generalizes innermost and outermost strategies
  (includes flexible combination of no redex below/at/above position $p$)

Loop under strategy

- loop $t \rightarrow \ldots u \rightarrow_p v \rightarrow \ldots C[t\mu]_q$ respects strategy iff reduction

$$C[C[C[\ldots C[u\mu]_n \ldots \mu]_{n\mu}]_{n\mu}] \rightarrow_{q \ldots q_p} C[C[C[\ldots C[v\mu]_n \ldots \mu]_{n\mu}]_{n\mu}]$$

respects strategy for all $n$ and all steps $u \rightarrow_p v$ in the loop
Goal 1: Loops w.r.t. strategy

Considered strategies for rewrite step $s \rightarrow_p t$

- innermost (no redex strictly below position $p$)
- outermost (no redex strictly above position $p$)
- forbidden patterns generalizes innermost and outermost strategies
  (includes flexible combination of no redex below/at/above position $p$)

Loop under strategy

- loop $t \rightarrow \ldots u \rightarrow_p v \rightarrow \ldots C[t\mu]_q$ respects strategy iff reduction

$$C[C[C[\ldots C[u\mu]_\mu]_\mu]_\mu]_\mu] \rightarrow_{q\ldots qp} C[C[C[\ldots C[v\mu]_\mu]_\mu]_\mu]_\mu]$$

respects strategy for all $n$ and all steps $u \rightarrow_p v$ in the loop

- decidable for
  - context-sensitive rewriting (trivial)
  - innermost strategy (difficult)
  - outermost strategy (even more difficult)
  - forbidden patterns (most difficult)
Goal 2: Unraveling

- Unraveling is transformation \( U \) from conditional to standard rewriting
- Evaluation of conditions is encoded using \( U \)-symbols
- Variations in how many auxiliary variables are used as arguments of \( U \)

\[
\text{Example: } \quad \text{div}(x, s(y)) \rightarrow 0 | x \leq y \rightarrow \ast \text{true} \\
\text{becomes} \quad \text{div}(x, s(y)) \rightarrow U_1(x \leq y, x, y) U_1(\text{true}, x, y) \rightarrow 0 \\
\text{div}(x, s(y)) \rightarrow U_2(x \leq y, x, y) U_2(\text{false}, x, y) \rightarrow s(\text{div}(x - s(y), s(y)))
\]

red variables are subject to optimizations
Goal 2: Unraveling

- Unraveling is transformation $U$ from conditional to standard rewriting
- Evaluation of conditions is encoded using $U$-symbols
- Variations in how many auxiliary variables are used as arguments of $U$
- Example:

  \[
  \text{div}(x, s(y)) \rightarrow 0 \quad | \quad x \leq y \rightarrow^* \text{true} \\
  \text{div}(x, s(y)) \rightarrow s(\text{div}(x - s(y), s(y))) \quad | \quad x \leq y \rightarrow^* \text{false}
  \]

  becomes

  \[
  \text{div}(x, s(y)) \rightarrow U_1(x \leq y, x, y) \quad U_1(\text{true}, x, y) \rightarrow 0 \\
  \text{div}(x, s(y)) \rightarrow U_2(x \leq y, x, y) \quad U_2(\text{false}, x, y) \rightarrow s(\text{div}(x - s(y), s(y)))
  \]

  red variables are subject to optimizations
Goal 2: Unraveling

- Completeness of unraveling is easy: \( s \rightarrow^R t \) implies \( s \rightarrow^+_U(R) t \)
- Soundness is much harder and does not always hold:
  if \( s \rightarrow^*_U(R) t \) then \( s \rightarrow^*_R t \) for \( s \) and \( t \) not containing \( U \)-symbols
  sufficient conditions like
    - ultra left linearity (for optimized and non-optimized unraveling)
    - ultra weak left linearity
Goal 3: Certifying constrained rewriting

- Completely depends on results of Task 1:
  - what kinds of constraints?
  - definition of rewrite relation?
  - ...
Outline

- Task 5: Certification
- IsaFoR + CeTA
- Progress in Task 5
Let $\succ$ be some order. If $\ell \succ r \ldots$
(Let \( \succ \) be some order. If \( \ell \succ r \ldots \))
(Let $\succ$ be some order. If $\ell \succ r \ldots$)
(Let \( \succ \) be some order. If \( \ell \succ r \ldots \))

proof tree (> 500 MB)

certifier

accept / error
(Let $\succ$ be some order. If $\ell \succ r$ ... )
(Let $\succ$ be some order. If $\ell \succ r \ldots$)
(Let $\succ$ be some order. If $\ell \succ r$ ...)

search algorithms

termination tool

proof tree ($>500$ MB)

code generator

certifier

accept / error
Isabelle/HOL

- Isabelle/HOL: Interactive theorem proving for higher order logic
- HOL: functional programming + theorem proving

datatype 'a list = Nil | Cons 'a ('a list)

fun insert where
  insert x Nil = Cons x Nil
  insert x (Cons y ys) = if x ≤ y then Cons x (Cons y ys) else ...

(funspecifying+)

fun sorted where
  sorted Nil = True
  sorted (Cons x Nil) = True
  sorted (Cons x (Cons y ys)) = x ≤ y ∧ sorted (Cons y ys)

(proving+)

lemma: sorted xs ⇒ sorted (insert x xs)
proof (induction xs ...

12 / 22
Isabelle/HOL

- Isabelle/HOL: Interactive theorem proving for higher order logic
- HOL: functional programming + theorem proving

(* programming *)

datatype 'a list = Nil | Cons 'a ('a list)

fun insert where
  insert x Nil = Cons x Nil
  insert x (Cons y ys) = if x ≤ y then Cons x (Cons y ys) else ...
Isabelle/HOL

- Isabelle/HOL: Interactive theorem proving for higher order logic
- HOL: functional programming + theorem proving

(*) *programming* *

datatype 'a list = Nil | Cons 'a ('a list)

fun insert where
insert x Nil = Cons x Nil
insert x (Cons y ys) = if x ≤ y then Cons x (Cons y ys) else ...

(*) *specifying* *

fun sorted where
sorted Nil = True
sorted (Cons x Nil) = True
sorted (Cons x (Cons y ys)) = x ≤ y ∧ sorted (Cons y ys)
Isabelle/HOL

- Isabelle/HOL: Interactive theorem proving for higher order logic
- HOL: functional programming + theorem proving

(* programming *)

datatype 'a list = Nil | Cons 'a ('a list)
fun insert where
  insert x Nil = Cons x Nil
  insert x (Cons y ys) = if x ≤ y then Cons x (Cons y ys) else ...

(* specifying *)
fun sorted where
  sorted Nil = True
  sorted (Cons x Nil) = True
  sorted (Cons x (Cons y ys)) = x ≤ y ∧ sorted (Cons y ys)

(* proving *)
lemma: sorted xs → sorted (insert x xs)
proof(induction xs ... )
library on rewriting, deep embedding (TRS is data object)

datatype ('f,'v)term = Var 'v | Fun 'f (('f,'v)term list)
type_synonym ('f,'v)trs = (('f,'v)term × ('f,'v)term) set
IsaFoR: Isabelle/HOL Formalization of Rewriting

- library on rewriting, deep embedding (TRS is data object)

\[
\text{datatype } ('f,'v)\text{term} = \text{Var }'v | \text{Fun }'f (('f,'v)\text{term list})
\]

\[
\text{type synonym } ('f,'v)\text{trs} = (('f,'v)\text{term }\times\text{ ('f,'v)term}) \text{ set}
\]

- supported constraints:

  - innermost strategy (many results)
  - substitution constraint
    
    \[
    C[\ell\sigma] \text{nfs} \rightarrow QR \quad C[r\sigma] \quad \ell \rightarrow r \in R
    \]
    
    - all arguments of \( \ell\sigma \) are in normal form w.r.t. \( \rightarrow Q \) (innermost)
    - if \( \text{nfs} \) then \( x\sigma \) is in normal form w.r.t. \( \rightarrow Q \) for all \( x \in V(\ell) \cup V(r) \)

  - outermost strategy (few results)
    - focus on termination criteria
    - recently added results on confluence, complexity, completion, and equational reasoning
IsaFoR: Isabelle/HOL Formalization of Rewriting

- library on rewriting, deep embedding (TRS is data object)

\[
\text{datatype } (\text{'f}, \text{'v})\text{term} = \text{Var 'v} | \text{Fun 'f (('f,'v)term list)}
\]

\[
\text{type_synonym } (\text{'f}, \text{'v})\text{trs} = ((\text{'f}, \text{'v})\text{term} \times (\text{'f}, \text{'v})\text{term}) \text{ set}
\]

- supported constraints:
  - innermost strategy (many results)
IsaFoR: Isabelle/HOL Formalization of Rewriting

• library on rewriting, deep embedding (TRS is data object)

\[
\text{datatype} \ (\text{'}f, \text{'}v)\text{term} = \text{Var } \text{'}v \mid \text{Fun } f ((\text{'}f, \text{'}v)\text{term list})
\]

\[
\text{datatype} \ (\text{'}f, \text{'}v)\text{trs} = ((\text{'}f, \text{'}v)\text{term} \times (\text{'}f, \text{'}v)\text{term}) \text{ set}
\]

• supported constraints:
  • innermost strategy (many results)
  • substitution constraint \(nfs\) to allow free variables in right-hand sides:

\[
C[\ell_\sigma] \overset{nfs_{Q}}{\underset{R}{\rightarrow}} C[r_\sigma] \text{ for } \ell \rightarrow r \in R \text{ if}
\]

• all arguments of \(\ell_\sigma\) are in normal form w.r.t. \(\rightarrow_Q\) (innermost)
• if \(nfs\) then \(x_\sigma\) is in normal form w.r.t. \(\rightarrow_Q\) for all \(x \in \mathcal{V}(\ell) \cup \mathcal{V}(r)\)
library on rewriting, deep embedding (TRS is data object)

\[
\begin{align*}
\text{datatype } (f, v)\text{term} &= \text{Var } v \mid \text{Fun } f \ ((f, v)\text{term list}) \\
\text{type synonym } (f, v)\text{trs} &= ((f, v)\text{term} \times (f, v)\text{term}) \text{ set}
\end{align*}
\]

supported constraints:

- innermost strategy (many results)
- substitution constraint \(nfs\) to allow free variables in right-hand sides:

\[
C[\ell\sigma] \xrightarrow[nfs]{}^{Q} C[r\sigma] \text{ for } \ell \rightarrow r \in \mathcal{R} \text{ if}
\]

- all arguments of \(\ell\sigma\) are in normal form w.r.t. \(\rightarrow_{Q}\) (innermost)
- if \(nfs\) then \(x\sigma\) is in normal form w.r.t. \(\rightarrow_{Q}\) for all \(x \in \mathcal{V}(\ell) \cup \mathcal{V}(r)\)
- outermost strategy (few results)
IsaFoR: Isabelle/HOL Formalization of Rewriting

• library on rewriting, deep embedding (TRS is data object)

\[
\text{datatype } (f,v)\text{term} = \text{Var } v | \text{Fun } f ((f,v)\text{term list})
\]

\[
\text{type synonym } (f,v)\text{trs} = ((f,v)\text{term} \times (f,v)\text{term}) \text{ set}
\]

• supported constraints:
  • innermost strategy (many results)
  • substitution constraint \(nfs\) to allow free variables in right-hand sides:

\[
C[l\sigma] \stackrel{nfs}{\xrightarrow{Q}} C[r\sigma] \text{ for } l \rightarrow r \in R \text{ if}
\]

  • all arguments of \(l\sigma\) are in normal form w.r.t. \(\rightarrow_Q\) (innermost)
  • if \(nfs\) then \(x\sigma\) is in normal form w.r.t. \(\rightarrow_Q\) for all \(x \in V(l) \cup V(r)\)

• outermost strategy (few results)

• focus on \textit{termination} criteria
IsaFoR: Isabelle/HOL Formalization of Rewriting

- library on rewriting, deep embedding (TRS is data object)

  \[
  \text{datatype} \ (\, f, \nu)\text{term} = \text{Var} \ \nu \ | \ \text{Fun} \ f ((\, f, \nu)\text{term list})
  \]

  \[
  \text{type synonym} \ (\, f, \nu)\text{trs} = ((\, f, \nu)\text{term} \times (\, f, \nu)\text{term}) \text{ set}
  \]

- supported constraints:
  - innermost strategy (many results)
  - substitution constraint \( nfs \) to allow free variables in right-hand sides:

    \[
    C[l\sigma] \xrightarrow{nfs}^Q C[r\sigma] \text{ for } l \rightarrow r \in \mathcal{R} \text{ if }
    \]

    \[
    \begin{align*}
    & \text{all arguments of } l\sigma \text{ are in normal form w.r.t. } \rightarrow_Q \text{ (innermost)} \\
    & \text{if } nfs \text{ then } x\sigma \text{ is in normal form w.r.t. } \rightarrow_Q \text{ for all } x \in \mathcal{V}(l) \cup \mathcal{V}(r)
    \end{align*}
    \]

  - outermost strategy (few results)

- focus on termination criteria

- recently added results on confluence, complexity, completion, and equational reasoning
Supported termination techniques

- Dependency Pairs
- Dependency Pair Framework
- (Innermost) Dependency Graph Processor
- (Innermost) DP Transformations Narrowing, Rewriting, and (Forward)-Instantiation
- Flat Context Closure
- Innermost Switch
- Innermost- and Outermost-Loops
- Match- and roof-bounds
- Reduction Pair Processors (poly, matrix, LPO, RPO, KBO, …)
- Root-Labeling
- Semantic Labeling
- Size-Change Termination
- String Reversal
- Subterm Criterion
- Uncurrying
- (Innermost) Usable Rules
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
  - load IsaFoR in Isabelle
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
  - load IsaFoR in Isabelle
  - export-code
    - (check-proof :: string -> certification-result)
    - in Haskell
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
  - load IsaFoR in Isabelle
  - export-code
    (check-proof :: string -> certification-result)
    in Haskell
  - write Haskell file Main.hs to load file and call check-proof
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
  - load IsaFoR in Isabelle
  - export-code
    - (check-proof :: string -> certification-result)
      in Haskell
  - write Haskell file Main.hs to load file and call check-proof
  - invoke Haskell-compiler and obtain CeTA as binary (outside Isabelle)
CeTA: Certified Termination Analysis

- **CeTA** is executable proof checker within IsaFoR
  - load IsaFoR in Isabelle
  - export-code
    (check-proof :: string -> certification-result)
    in Haskell
  - write Haskell file Main.hs to load file and call check-proof
  - invoke Haskell-compiler and obtain CeTA as binary (outside Isabelle)
- both IsaFoR and CeTA are easily available (repository or distribution)
  http://cl-informatik.uibk.ac.at/software/ceta/
Task 5: Certification

IsaFoR + CeTA

Progress in Task 5
Goal 1: Loops under strategies for $t \rightarrow^+ \mathcal{C}[t\mu]$

- Innermost Loops: original proof works in three phases

1. Setting up matching problems: $s\mu n$ never matches lhs for any $n$
2. Simplified to identity problems using matching algorithm: $s\mu n \neq t\mu n$ for all $n$
3. Complex algorithm to decide identity problems

- Example: lhs $\text{eq}(z, z)$, redex $f(\text{eq}(x, y))$, $\mu = \{x/f(y), y/f(x)\}$

- Example: $\text{lhs } \text{eq}(z, z)$, redex $f(\text{eq}(x, y))$, $\mu = \{x/f(y), y/f(x)\}$
Goal 1: Loops under strategies for $t \to^+ C[t \mu]$

- Innermost Loops: original proof works in three phases
  1. setting up matching problems:

    $s \mu^n$ never matches lhs for any $n$

- Example: lhs $\text{eq}(z, z)$, redex $f(\text{eq}(x, y))$, $\mu = \{x/f(y), y/f(x)\}$
  1. for $s = \text{eq}(x, y)$ or $s = f(x)$ or $s = f(y)$, $s \mu^n$ does not match $\text{eq}(z, z)$
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Innermost Loops: original proof works in three phases
  1. setting up matching problems:

    $$s\mu^n \text{ never matches lhs for any } n$$

  2. simplified to identity problems using matching algorithm:

    $$s\mu^n \neq t\mu^n \quad \text{for all } n$$

- Example: lhs eq($z, z$), redex f(eq($x, y$)), $\mu = \{x/f(y), y/f(x)\}$
  1. for $s = \text{eq}(x, y)$ or $s = f(x)$ or $s = f(y)$, $s\mu^n$ does not match eq($z, z$)
  2. $x\mu^n \neq y\mu^n$ for all $n$
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$ 

- Innermost Loops: original proof works in three phases
  1. setting up matching problems:
     
     $s\mu^n$ never matches lhs for any $n$
  
  2. simplified to identity problems using matching algorithm:
     
     $s\mu^n \neq t\mu^n$ for all $n$
  
  3. complex algorithm to decide identity problems

- Example: lhs $\text{eq}(z, z)$, redex $f(\text{eq}(x, y))$, $\mu = \{x/f(y), y/f(x)\}$
  1. for $s = \text{eq}(x, y)$ or $s = f(x)$ or $s = f(y)$, $s\mu^n$ does not match $\text{eq}(z, z)$
  2. $x\mu^n \neq y\mu^n$ for all $n$
  3. True (the variable in both terms is toggling)
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Innermost Loops: original proof works in three phases
  1. setting up matching problems:
    
    \[ s\mu^n \text{ never matches lhs for any } n \]

  2. simplified to identity problems using matching algorithm:
    
    \[ s\mu^n \neq t\mu^n \text{ for all } n \]

  3. complex algorithm to decide identity problems

- Example: lhs $\text{eq}(z, z)$, redex $f(\text{eq}(x, y))$, $\mu = \{ x/f(y), y/f(x) \}$
  1. for $s = \text{eq}(x, y)$ or $s = f(x)$ or $s = f(y)$, $s\mu^n$ does not match $\text{eq}(z, z)$
  2. $x\mu^n \neq y\mu^n$ for all $n$
  3. True (the variable in both terms is toggling)

- everything formalized, deviation from original proof in 3. step: improved complex algorithm, Kruskal no longer required
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Outermost Loops: original proof works in three phases
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Outermost Loops: original proof works in three phases
  1. setting up matching and extended matching problems:

\[
D[s(C, \mu)^k]_{\mu^n} \text{ never matches lhs for any } n, k
\]

\[
(C, \mu)^k: k\text{-times application of } u \mapsto C[u\mu]
\]

18 / 22
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Outermost Loops: original proof works in three phases
  1. setting up matching and extended matching problems:

$$D[s(C, \mu)^k] \mu^n \text{ never matches lhs for any } n, k$$

$$(C, \mu)^k: k\text{-times application of } u \mapsto C[u\mu]$$

2. simplified to identity and extended identity problems:

$$D[s(C, \mu)^k] \mu^n \neq t\mu^n \quad \text{for all } n, k$$

(*)
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- Outermost Loops: original proof works in three phases
  1. setting up matching and extended matching problems:

  \[ D[s(C, \mu)^k] \mu^n \text{ never matches lhs for any } n, k \]

  $(C, \mu)^k$: $k$-times application of $u \mapsto C[u\mu]$

  2. simplified to identity and extended identity problems:

  \[ D[s(C, \mu)^k] \mu^n \neq t\mu^n \text{ for all } n, k \quad (\star) \]

  3. complex algorithm to decide extended identity problems
Goal 1: Loops under strategies for \( t \rightarrow^+ C[t\mu] \)

- Outermost Loops: original proof works in three phases
  1. setting up matching and extended matching problems:

\[
D[s(C, \mu)^k]\mu^n \text{ never matches lhs for any } n, k
\]

\((C, \mu)^k: k\text{-times application of } u \mapsto C[u\mu]\)

2. simplified to identity and extended identity problems:

\[
D[s(C, \mu)^k]\mu^n \neq t\mu^n \quad \text{for all } n, k \quad (\star)
\]

3. complex algorithm to decide extended identity problems

- main result formalized without any formalization of 3. step:
  for extended identity problems from outermost loops
  \( k \) in \((\star)\) can be fixed to 0
  \(\implies\) get non-extended identity problem
Goal 1: Loops under strategies for \( t \rightarrow^+ C[t\mu] \)

- forbidden patterns: original proof works in two phases
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- forbidden patterns: original proof works in two phases
  1. setting up matching and extended matching problems (much more involved than in innermost and outermost case)
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- forbidden patterns: original proof works in two phases
  1. setting up matching and extended matching problems (much more involved than in innermost and outermost case)
  2. solve (extended) matching problems
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- forbidden patterns: original proof works in two phases
  1. setting up matching and extended matching problems (much more involved than in innermost and outermost case)
  2. solve (extended) matching problems
- several open tasks
  - formalization of rewrite relation with forbidden patterns
Goal 1: Loops under strategies for $t \rightarrow^+ C[t\mu]$

- forbidden patterns: original proof works in two phases
  1. setting up matching and extended matching problems (much more involved than in innermost and outermost case)
  2. solve (extended) matching problems
- several open tasks
  - formalization of rewrite relation with forbidden patterns
  - complete formalization of 1. step
Goal 1: Loops under strategies for $t \xrightarrow{+} C[t\mu]$

- forbidden patterns: original proof works in two phases
  1. setting up matching and extended matching problems (much more involved than in innermost and outermost case)
  2. solve (extended) matching problems

- several open tasks
  - formalization of rewrite relation with forbidden patterns
  - complete formalization of 1. step
  - figure out whether trick for outermost loops (fix $k = 0$) is also possible for extended identity problems from forbidden patterns; if not, formalize complex algorithm for extended identity problems
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
  - generic unraveling-transformation:
    
    \[
    \text{for } \ell \rightarrow r \iff s_1 = t_1, \ldots s_n = t_n \text{ unraveled rules are}
    \]

    \[
    \ell \rightarrow C_1[s_1] \\
    C_1[t_1] \rightarrow C_2[s_2] \\
    \ldots \rightarrow \ldots \\
    C_n[t_n] \rightarrow r
    \]

    \(C_i\) are arbitrary, standard unraveling by choosing \(C_1 = U_1(\cdot, \mathcal{V}(\ell)), \ldots\)
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
  - generic unraveling-transformation:
    for $\ell \rightarrow r \iff s_1 = t_1, \ldots, s_n = t_n$ unraveled rules are
    \[
    \ell \rightarrow C_1[s_1] \\
    C_1[t_1] \rightarrow C_2[s_2] \\
    \ldots \rightarrow \ldots \\
    C_n[t_n] \rightarrow r
    \]
    $C_i$ are arbitrary, standard unraveling by choosing $C_1 = U_1(\cdot, V(\ell)), \ldots$
- completeness result:
  \[
  SN(U_{C_i}(R)) \implies quasi-decreasing(R) \implies SN(\rightarrow R)
  \]
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
  - generic unraveling-transformation:
    for $\ell \rightarrow r \iff s_1 = t_1, \ldots s_n = t_n$ unraveled rules are
    $$\ell \rightarrow C_1[s_1]$$
    $$C_1[t_1] \rightarrow C_2[s_2]$$
    $$\ldots \rightarrow \ldots$$
    $$C_n[t_n] \rightarrow r$$
  - $C_i$ are arbitrary, standard unraveling by choosing $C_1 = U_1(\cdot, V(\ell)), \ldots$
- completeness result:
  $$SN(U_{C_i}(R)) \implies \text{quasi-decreasing}(R) \implies SN(\rightarrow_R)$$
- several open tasks
  - certification algorithm for unraveling and suitable format
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
  - generic unraveling-transformation:
    for $\ell \rightarrow r \iff s_1 = t_1, \ldots s_n = t_n$ unraveled rules are
    $$\ell \rightarrow C_1[s_1]$$
    $$C_1[t_1] \rightarrow C_2[s_2]$$
    $$\ldots \rightarrow \ldots$$
    $$C_n[t_n] \rightarrow r$$

  $C_i$ are arbitrary, standard unraveling by choosing $C_1 = U_1(\cdot, V(\ell)), \ldots$

- completeness result:
  $$SN(U_{C_i}(\mathcal{R})) \implies \text{quasi-decreasing(}\mathcal{R}\text{)} \implies SN(\rightarrow_{\mathcal{R}})$$

- several open tasks
  - certification algorithm for unraveling and suitable format
  - algorithm to compute $\rightarrow_{\mathcal{R}}$ (for quasi-decreasing $\mathcal{R}$)
Goal 2: Unraveling

- last week: nothing formalized in IsaFoR
- this week: formalization of easy results
  - conditional rewrite relation, quasi-decreasingness
  - generic unraveling-transformation:
    for $\ell \rightarrow r \Leftarrow s_1 = t_1, \ldots s_n = t_n$ unraveled rules are
    \[
    \ell \rightarrow C_1[s_1] \\
    C_1[t_1] \rightarrow C_2[s_2] \\
    \ldots \rightarrow \ldots \\
    C_n[t_n] \rightarrow r
    \]
    $C_i$ are arbitrary, standard unraveling by choosing $C_1 = U_1(\cdot, V(\ell)), \ldots$
- completeness result:
  \[
  SN(U_{C_i}(R)) \implies \text{quasi-decreasing}(R) \implies SN(\rightarrow_R)
  \]
- several open tasks
  - certification algorithm for unraveling and suitable format
  - algorithm to compute $\rightarrow_R$ (for quasi-decreasing $R$)
  - soundness results for unravelings
Goal 3: Constraint rewriting

- wait for outcome of Task 1
- then there will be several open tasks
Task 5: Certification, Summary

1. goal: certifying loops
   - large parts already done (innermost, outermost)
   - generalization to forbidden patterns currently open

2. goal: unraveling
   - only completeness results in IsaFoR at the moment

3. goal: constrained rewriting
   - not clear what to do at the moment

open questions
   - which tools provide proofs to certify (TTT2?, VMTL?, Nagoya?)
   - which soundness result for unraveling (Nagoya?, Vienna?, combined?)