universität
innsbruck

History and Future of the CeTA-Certifier for CoCo

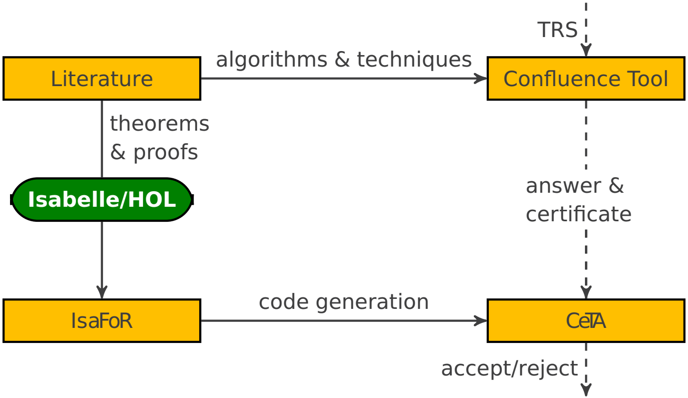
(Including a New Decision Procedure for Pattern Completeness)

René Thiemann

Supported by the Austrian Science Fund (FWF) project I 5943

International Workshop on Confluence, August 24, 2023

The Certification Approach for CoCo



- machine checked theorems
- verified algorithms that check correct application of confluence techniques

History of CeTA 1/3

- 2009: first release of CeTA: Certified Termination Analysis
  - main development by Sternagel and T.
  - focus on termination techniques for TRSs
- 2009 – : CeTA checks proofs in termination competition
  (2022: CeTA can check more than 90 % of classified TRSs in "TRS standard")
- 2011: new release of CeTA: Certified Tool Assertions
  - support of $SN(\to_{\mathcal{R}}) = SN(\xrightarrow{i}_{\mathcal{R}})$ for locally confluent overlay systems
  - support of confluence via Newman's lemma
- 2012: non-confluence support, weak orthogonality
- 2012: CeTA is used in demo certification track in CoCo
- 2013 – : CeTA is participating in certification track in CoCo

History of CeTA 2/3

- many confluence contributions from UIBK \ T.
  - Felgenhauer
  - Middeldorp
  - Nagele
  - Sternagel
  - Winkler
  - Zankl
- covering
  - modularity
  - layer framework
  - rule labeling
  - almost parallel closed
  - unraveling
  - infeasibility, including ordered completion

History of CeTA 3/3

- latest contributions (2022, 2023)
  - Kohl, Middeldorp (CPP and ITP 2023): almost development closed
  - Kim, T. (unpublished): criteria based on parallel critical pairs
  - Kim, Kohl, Middeldorp, T.: support for commutation
- CoCo 2022
  - winner TRS (CSI): 68 / 100 — winner COM (CoLL): 58 / 85
  - certified: 51 / 100 — certified: 0 / 85
- CoCo 2023
  - winner TRS: 71 / 100 — winner commutation: 63 / 85
  - certified: 60 / 100 — certified: 26 / 85
- many thanks to all confluence tool authors that teamed with CeTA
  - ACP, ConCon, CSI, Hakusan, nonreach

Design of CeTA

- try to make certificate generation easy $\Longrightarrow$ small certificates
- try to make certificate checking robust $\Longrightarrow$ flexible check-functions
- example: check that certain critical pairs are joinable in some way
  - CeTA: verified algorithm to compute critical pairs
  - confluence tools might slightly deviate
  - CeTA: check that variant of every non-trivial critical pair exists in certificate
- TRS example:

$$\mathsf{f}(x, \mathsf{a}) \to \mathsf{b}$$
$$\mathsf{f}(x, y) \to \mathsf{g}(y, x)$$

  - CeTA: $CP = \{(\mathsf{b}, \mathsf{g}(\mathsf{a}, x1)), (\mathsf{g}(\mathsf{a}, x1), \mathsf{b}), (\mathsf{b}, \mathsf{b}), (\mathsf{g}(x1, x2), \mathsf{g}(x1, x2))\}$
  - certificate 1: $CP = \{(\mathsf{g}(\mathsf{a}, x47), \mathsf{b}),\ (\mathsf{b}, \mathsf{g}(\mathsf{a}, x23))\}$
  - certificate 2: $CP = \{(\mathsf{b}, \mathsf{g}(\mathsf{a}, u)),\ (\mathsf{b}, \mathsf{f}(\mathsf{g}(y, x), \mathsf{a}))\}$
  - certificate 3: $CP = \{(\mathsf{g}(\mathsf{a}, v), \mathsf{b}),\ (\mathsf{iwc2023}, \mathsf{cocoweb})\}$

Design of CeTA

- try to make certificate generation easy $\Longrightarrow$ small certificates
- try to make certificate checking robust $\Longrightarrow$ flexible check-functions
- example: check that certain critical pairs are joinable in some way
  - consider rule labeling with parallel critical pairs in commutation version
  - required decrease for parallel critical pair $t \;{}_{\mathcal{R}_k}\!\!\mathrel{\leftarrow\!\!\!\Vert\!\!-} \cdot \xrightarrow{\varepsilon}_{\mathcal{S}_m} u$:

$$t \left({}_{\mathcal{R}_{<k}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<k}}\right)^* \cdot \mathrel{-\!\!\Vert\!\!\!\to}_{\mathcal{S}_{\leqslant m}} \cdot \left({}_{\mathcal{R}_{<n}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<n}}\right)^* \cdot {}_{\mathcal{R}_{\leqslant k}}\!\!\mathrel{\leftarrow\!\!\!\Vert\!\!-} \cdot \left({}_{\mathcal{R}_{<m}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<m}}\right)^* u$$

    for $n = \max(k, m)$ and Toyama-restriction on ${}_{\mathcal{R}_{\leqslant k}}\!\!\mathrel{\leftarrow\!\!\!\Vert\!\!-}$-step
  - checker variant 1: breadth-first search with limit on number of steps

$$\underbrace{t \longrightarrow^{c_1}_{\mathcal{R}^{-1}_{<k} \cup \mathcal{S}_{<k}} \cdot \mathrel{-\!\!\Vert\!\!\!\to}_{\mathcal{S}_{\leqslant m}} \cdot \longrightarrow^{c_2}_{\mathcal{S}_{<n}}}_{\dashrightarrow} \cdot \underbrace{{}_{\mathcal{R}_{<n}}\!\!\overset{c_3}{\longleftarrow} \cdot {}_{\mathcal{R}_{\leqslant k}}\!\!\mathrel{\leftarrow\!\!\!\Vert\!\!-} \cdot {}_{\mathcal{R}_{<m} \cup \mathcal{S}^{-1}_{<m}}\!\!\overset{c_4}{\longleftarrow} u}_{\dashleftarrow}$$

    - trivial certificate
    - breadth-first search might become time-critical: $c_1 + c_2 + c_3 + c_4 \leqslant \mathit{limit}$
    - cannot deal with conversions very well (extra variables will not be instantiated)

Design of CeTA

- try to make certificate generation easy $\Longrightarrow$ small certificates
- try to make certificate checking robust $\Longrightarrow$ flexible check-functions
- example: check that certain critical pairs are joinable in some way
  - required decrease for parallel critical peak $t\ {}_{\mathcal{R}_k}\!\!\leftarrow\!\!\Vdash \cdot \xrightarrow{\varepsilon}_{\mathcal{S}_m} u$:

    $$t \left({}_{\mathcal{R}_{<k}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<k}}\right)^* \cdot \twoheadrightarrow\!\!\!\Vdash_{\mathcal{S}_{\leqslant m}} \cdot \left({}_{\mathcal{R}_{<n}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<n}}\right)^* \cdot {}_{\mathcal{R}_{\leqslant k}}\!\!\twoheadleftarrow\!\!\Vdash \cdot \left({}_{\mathcal{R}_{<m}}\!\!\longleftarrow \cup \longrightarrow_{\mathcal{S}_{<m}}\right)^* u$$

    for $n = \max(k, m)$ and Toyama-restriction on ${}_{\mathcal{R}_{\leqslant k}}\!\!\twoheadleftarrow\!\!\Vdash$-step
  - checker variant 2: require intermediate terms $t, s_1, s_2, s_3, \ldots, s_{42}, u$
    - non-verbose certificate: no specification of applied rules
    - automatic partitioning of sequence
    - flexible: step from s_i to s_{i+1} might be single step or parallel step

Automatic Partitioning

- required join

$$s_0 \ ({}_{\mathcal{R}_1}\!\longleftarrow \cup \longrightarrow_{\mathcal{R}_2})^* \cdot \mathrel{\twoheadrightarrow\!\!\!\!\!\!\shortmid\!\shortmid}_{\mathcal{R}_3} \cdot ({}_{\mathcal{R}_4}\!\longleftarrow \cup \longrightarrow_{\mathcal{R}_5})^* \cdot {}_{\mathcal{R}_6}\!\twoheadleftarrow\!\!\!\shortmid\!\shortmid \cdot ({}_{\mathcal{R}_7}\!\longleftarrow \cup \longrightarrow_{\mathcal{R}_8})^* \ s_n$$

- provided: sequence of terms $s_0, s_1, s_2, s_3, \ldots, s_n$
- greedy approach
  - try to find maximum i such that $s_0 \ ({}_{\mathcal{R}_1}\!\longleftarrow \cup \longrightarrow_{\mathcal{R}_2})^* \ s_i$
    - $i := 0$
    - if $s_i \mathrel{\twoheadrightarrow\!\!\!\!\!\!\shortmid\!\shortmid}_{\mathcal{R}_1^{-1} \cup \mathcal{R}_2} s_{i+1}$ then $i := i + 1$ and iterate (decision procedure for $\mathrel{\twoheadrightarrow\!\!\!\!\!\!\shortmid\!\shortmid}$)
    - otherwise return i
  - continue with checking $s_i \mathrel{\twoheadrightarrow\!\!\!\!\!\!\shortmid\!\shortmid}_{\mathcal{R}_3} \cdot \ldots$
    - if $s_i \mathrel{\twoheadrightarrow\!\!\!\!\!\!\shortmid\!\shortmid}_{\mathcal{R}_3} s_{i+1}$ then $i := i + 1$
  - continue with checking $s_i \ ({}_{\mathcal{R}_4}\!\longleftarrow \cup \longrightarrow_{\mathcal{R}_5})^* \cdot \ldots$
  - finally check $s_i = s_n$
- greedy approach is complete since all relations in join are reflexive

Future Work: Certification of Ground Confluence

- ground confluence: given many-sorted TRS $\mathcal{R}$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$, determine whether $\mathcal{R}$ is confluent on ground terms
- internally it is based on
  - rewriting induction (certification is ongoing work with Aoto, Kim, and Yamada), and
  - quasi-reducibility as a criterion to ensure sufficient completeness

Quasi-Reducibility and Pattern Completeness

- general idea: is program sufficiently defined, i.e., does not get stuck
- setup: fix TRS $\mathcal{R}$, split signature into $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$
- basic ground terms: $\mathcal{B}(\mathcal{C},\mathcal{D}) = \{f(t_1,\dots,t_n) \mid f \in \mathcal{D}, t_1,\dots,t_n \in \mathcal{T}(\mathcal{C})\}$
- quasi-reducibility: $\forall t \in \mathcal{B}(\mathcal{C},\mathcal{D}).\ \exists s.\ t \to_{\mathcal{R}} s$
- strong quasi-reducibility: $\forall t \in \mathcal{B}(\mathcal{C},\mathcal{D}).\ \exists s.\ t \xrightarrow{\leqslant 1}_{\mathcal{R}} s$
- pattern completeness: $\forall t \in \mathcal{B}(\mathcal{C},\mathcal{D}).\ \exists s.\ t \xrightarrow{\varepsilon}_{\mathcal{R}} s$
- pattern completeness $\Longrightarrow$ strong quasi-reducibility $\Longrightarrow$ quasi-reducibility
- free constructors ($\mathcal{T}(\mathcal{C}) \subseteq \mathit{NF}$): pattern completeness = quasi-reducibility

Quiz

- consider TRS with signature $\mathcal{D} = \{\mathsf{even}\}$ and $\mathcal{C} = \{\mathsf{true}, \mathsf{false}, \mathsf{0}, \mathsf{s}\}$

$$\mathsf{even}(\mathsf{0}) \to \mathsf{true}$$
$$\mathsf{even}(\mathsf{s}(\mathsf{0})) \to \mathsf{false}$$
$$\mathsf{even}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{even}(x)$$

- is it quasi-reducible? strongly-reducible? pattern complete?
- if the sorts are $\mathsf{true} : \mathsf{Bool}$, $\mathsf{false} : \mathsf{Bool}$, $\mathsf{0} : \mathsf{Num}$, and $\mathsf{s} : \mathsf{Num} \to \mathsf{Num}$
- if we add $\mathsf{p} : \mathsf{Num} \to \mathsf{Num}$ to $\mathcal{C}$ and add further rules?

$$\mathsf{even}(\mathsf{p}(\mathsf{0})) \to \mathsf{false}$$
$$\mathsf{even}(\mathsf{p}(\mathsf{p}(x))) \to \mathsf{even}(x)$$
$$\mathsf{s}(\mathsf{p}(x)) \to x$$
$$\mathsf{p}(\mathsf{s}(x)) \to x$$

Quasi-Reducibility: Basic Ground Terms are Reducible

- decidable, co-NP complete [Kapur, Narendran]
- ensuring quasi-reducibility via Kapur, Narendran requires enumeration of exponentially many terms
- for left-linear TRSs, decidable via tree-automata:

$$\mathcal{B}(\mathcal{C},\mathcal{D}) \subseteq \text{“terms that contain redex”}$$

  - contains expensive subset-check
  - certification requires verified tree-automata library
  - restriction to left-linear TRSs

Pattern Completeness: Basic Ground Terms have Root Redex

- for left-linear TRSs, decidable via tree-automata
- for left-linear TRSs, decidable via complement algorithm [Lazrek, Lescanne and Thiel]
- this talk: decision procedure for pattern completeness and for strong quasi-reducibility
  - does not require tree-automata techniques
  - inspired by matching algorithm
  - linear lower bound, exponential upper bound
  - no explicit computation of complements
  - no restriction to left-linear TRS
  - verified implementation in Isabelle/HOL (soundness, not complexity)
  - outperforms complement algorithm

The Algorithm

- modified matching algorithm
  - matching problem is finite set $mp = \{(t_1, \ell_1), \dots, (t_n, \ell_n)\}$ where t_i, ℓ_i are terms
  - ℓ_i is subterm of left-hand side of TRS: is t_i matched by ℓ_i
  - modification: vars in t_i represent constr. ground terms: is $t_i\sigma$ matched by ℓ_i
  - notion: mp is solvable w.r.t. σ if there is γ such that $t_i\sigma = \ell_i\gamma$ for all i
  - $\bot_{mp}$ is special matching problem that is never solvable
- pattern problem = disjunctive combination of matching problems
  - pattern problem is finite set $pp = \{mp_1, \dots, mp_k\}$ of matching problems
  - pp is solvable if for each constr. ground substitution σ, at least one of the matching problems mp_i is solved w.r.t. σ
  - $\top_{pp}$ is special pattern problem that is always solvable
- examples for three-rule even-TRS: basic-terms represented by $\mathsf{even}(y)$
  - pattern completeness: $pp :=$ $\{\{(\mathsf{even}(y), \mathsf{even}(0))\}, \{(\mathsf{even}(y), \mathsf{even}(\mathsf{s}(0)))\}, \{(\mathsf{even}(y), \mathsf{even}(\mathsf{s}(\mathsf{s}(x))))\}\}$
  - strong quasi-reducibility: $pp \cup \{\{(y, \mathsf{even}(0))\}, \{(y, \mathsf{even}(\mathsf{s}(0)))\}, \{(y, \mathsf{even}(\mathsf{s}(\mathsf{s}(x))))\}\}$

Simplification Rules for Matching and Pattern Problems

- we transform matching problems ($\to$) and pattern problems ($\Rightarrow$)
- whenever $mp \to mp'$ then mp is solvable w.r.t. σ iff so is mp'
- whenever $pp \Rightarrow pp'$ then pp is solvable iff so is pp'

$$\{(f(t_1,\dots,t_n),f(\ell_1,\dots,\ell_n))\} \uplus mp \to \{(t_1,\ell_1),\dots,(t_n,\ell_n)\} \cup mp \qquad \text{(decompose)}$$

$$\{(f(\dots),g(\dots))\} \uplus mp \to \bot_{mp} \quad \text{if } f \neq g \qquad \text{(clash)}$$

$$\{(t,x)\} \uplus mp \to mp \quad \text{if ``}x \notin mp\text{''} \qquad \text{(match)}$$

$$\{mp\} \uplus pp \Rightarrow \{mp'\} \cup pp \quad \text{if } mp \to mp' \qquad \text{(simp-mp)}$$

$$\{\bot_{mp}\} \uplus pp \Rightarrow pp \qquad \text{(remove-mp)}$$

$$\{\emptyset\} \uplus pp \Rightarrow \top_{pp} \qquad \text{(success)}$$

- example (pattern completeness of three-rule even-TRS): pp

$$= \{\{(\mathsf{even}(y),\mathsf{even}(0))\},\{(\mathsf{even}(y),\mathsf{even}(\mathsf{s}(0)))\},\{(\mathsf{even}(y),\mathsf{even}(\mathsf{s}(\mathsf{s}(x))))\}\}$$

$$\Rightarrow \{\{(y,0)\},\{(\mathsf{even}(y),\mathsf{even}(\mathsf{s}(0)))\},\{(\mathsf{even}(y),\mathsf{even}(\mathsf{s}(\mathsf{s}(x))))\}\}$$

$$\Rightarrow^2 \{\{(y,0)\},\{(y,\mathsf{s}(0))\},\{(y,\mathsf{s}(\mathsf{s}(x)))\}\}$$

Towards Instantiation

- current rules get stuck on $\{\{(y, 0)\}, \{(y, \mathsf{s}(0))\}, \{(y, \mathsf{s}(\mathsf{s}(x)))\}\}$
- usual matching algorithm would indicate failure for all three matching problems
- here, y is variable, but represents an arbitrary constructor ground term
- $\Longrightarrow$ perform case-analysis by replacing y by all possible ground terms
  - $y : \mathsf{Num}$
  - all constructor ground terms of sort Num can be represented as 0 or as $\mathsf{s}(z)$ for some fresh variable z
  - logically, we have to build a conjunction over all these cases: for each σ ...
  - $\Longrightarrow$ work on sets P of pattern problems; P is solvable iff each $pp \in P$ is solvable
  - define transformation rules $\Rrightarrow$ on sets of pattern problems
  - $\bot_P$ denotes unsolvable set of pattern problems

Instantiation

- new rules for sets of pattern problems

$$\{pp\} \uplus P \Rrightarrow \{pp'\} \cup P \qquad \text{if } pp \Rightarrow pp' \qquad \text{(simp-pp)}$$

$$\{\emptyset\} \uplus P \Rrightarrow \bot_P \qquad \text{(failure)}$$

$$\{\top_{pp}\} \uplus P \Rrightarrow P \qquad \text{(remove-pp)}$$

$$\{pp\} \uplus P \Rrightarrow \{pp[x/c(\vec{xs})] \mid c \in \mathcal{C}_\tau\} \cup P \quad \text{if } mp \in pp, (x, f(\dots)) \in mp, \text{ and } x \in \mathcal{V}_\tau \qquad \text{(instantiate)}$$

- example (continued): $\{pp\} \Rrightarrow^* \{\{\{(y,0)\},\{(y,\mathsf{s}(0))\},\{(y,\mathsf{s}(\mathsf{s}(x)))\}\}\}$

$$\begin{aligned}
&\Rrightarrow \{\{\{(0,0)\},\{(0,\mathsf{s}(0))\},\{(0,\mathsf{s}(\mathsf{s}(x)))\}\},\\
&\qquad \{\{(\mathsf{s}(z),0)\},\{(\mathsf{s}(z),\mathsf{s}(0))\},\{(\mathsf{s}(z),\mathsf{s}(\mathsf{s}(x)))\}\}\}\\
&\Rrightarrow \{\{\emptyset,\{(0,\mathsf{s}(0))\},\{(0,\mathsf{s}(\mathsf{s}(x)))\}\},\\
&\qquad \{\{(\mathsf{s}(z),0)\},\{(\mathsf{s}(z),\mathsf{s}(0))\},\{(\mathsf{s}(z),\mathsf{s}(\mathsf{s}(x)))\}\}\}\\
&\Rrightarrow \{\top_{pp},\{\{(\mathsf{s}(z),0)\},\{(\mathsf{s}(z),\mathsf{s}(0))\},\{(\mathsf{s}(z),\mathsf{s}(\mathsf{s}(x)))\}\}\}\\
&\Rrightarrow \{\{\{(\mathsf{s}(z),0)\},\{(\mathsf{s}(z),\mathsf{s}(0))\},\{(\mathsf{s}(z),\mathsf{s}(\mathsf{s}(x)))\}\}\}
\end{aligned}$$

Example – Finalized

$$
\begin{aligned}
\{pp\} &\Rrightarrow^* \{\{\{(\mathsf{s}(z), 0)\}, \{(\mathsf{s}(z), \mathsf{s}(0))\}, \{(\mathsf{s}(z), \mathsf{s}(\mathsf{s}(x)))\}\}\} \\
&\Rrightarrow \{\{\bot_{mp}, \{(\mathsf{s}(z), \mathsf{s}(0))\}, \{(\mathsf{s}(z), \mathsf{s}(\mathsf{s}(x)))\}\}\} \\
&\Rrightarrow \{\{\{(\mathsf{s}(z), \mathsf{s}(0))\}, \{(\mathsf{s}(z), \mathsf{s}(\mathsf{s}(x)))\}\}\} \\
&\Rrightarrow^2 \{\{\{(z, 0)\}, \{(z, \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\{(0, 0)\}, \{(0, \mathsf{s}(x))\}\}, \{\{(\mathsf{s}(y), 0)\}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\emptyset, \{(0, \mathsf{s}(x))\}\}, \{\{(\mathsf{s}(y), 0)\}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\top_{pp}, \{\{(\mathsf{s}(y), 0)\}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\{(\mathsf{s}(y), 0)\}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\bot_{mp}, \{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\{(\mathsf{s}(y), \mathsf{s}(x))\}\}\} \\
&\Rrightarrow \{\{\{(y, x)\}\}\} \\
&\Rrightarrow \{\{\emptyset\}\} \\
&\Rrightarrow \{\top_{pp}\} \\
&\Rrightarrow \emptyset
\end{aligned}
$$

Properties

Theorem

- *whenever* $\{pp\} \Rrightarrow^* \emptyset$ *then* pp *is solvable*
- *whenever* $\{pp\} \Rrightarrow^* \bot_P$ *then* pp *is not solvable*
- $\Rrightarrow$ *is terminating*
- *whenever* pp *is linear then normal forms of* $\{pp\}$ *w.r.t.* $\Rrightarrow$ *are either* $\emptyset$ *or* $\bot_P$

The General Case

- $\Rrightarrow$ might get stuck on non-linear inputs
- example: consider TRS with lhss $\{\mathsf{f}(x,x,y), \mathsf{f}(x,y,x), \mathsf{f}(y,x,x)\}$
- $\{\{\{(\mathsf{f}(x_1,x_2,x_3), \mathsf{f}(x,x,y))\}, \{(\mathsf{f}(x_1,x_2,x_3), \mathsf{f}(x,y,x))\}, \{(\mathsf{f}(x_1,x_2,x_3), \mathsf{f}(y,x,x))\}\}\}$
  $\Rrightarrow^* \{\{\{(x_1,x),(x_2,x)\}, \{(x_1,x),(x_3,x)\}, \{(x_2,x),(x_3,x)\}\}\}$
- general case requires three more rules

$$\{(t,x),(t',x)\} \uplus mp \to \bot_{mp} \qquad \text{if } t|_p = f(\dots) \neq g(\dots) = t'|_p \qquad \text{(clash')}$$

$$\{pp\} \uplus P \Rrightarrow \{pp[x/c(\vec{xs})] \mid c \in \mathcal{C}_\tau\} \cup P \qquad \text{(instantiate')}$$

$$\text{if } mp \in pp,\ \{(t,y),(t',y)\} \subseteq mp,\ t|_p = x \neq t'|_p,\ x \in \mathcal{V}_\tau, \text{ and } \tau \text{ is finite}$$

$$\{pp\} \uplus P \Rrightarrow \bot_P \qquad \text{if for each } mp \in pp \text{ there are } \{(t,y),(t',y)\} \subseteq mp \qquad \text{(failure')}$$

$$\text{such that } t|_p = x \neq t'|_p,\ x \in \mathcal{V}_\tau \text{ and } \tau \text{ is infinite}$$

- $\Rrightarrow$ is still terminating; moreover pp is solvable iff $\{pp\} \Rrightarrow^* \emptyset$

Example

- assume x_1, x_2, x_3 : Num, i.e., infinite sort

$$\{\{\{(x_1,x),(x_2,x)\},\{(x_1,x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \Rrightarrow \bot_P$$

- assume x_1, x_2, x_3 : Bool, i.e., finite sort

$$\begin{aligned}
& \{\{\{(x_1,x),(x_2,x)\},\{(x_1,x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \\
\Rrightarrow\ & \{\{\{(\mathsf{true},x),(x_2,x)\},\{(\mathsf{true},x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}, \\
& \ \{\{(\mathsf{false},x),(x_2,x)\},\{(\mathsf{false},x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \\
\Rrightarrow\ & \{\{\{(\mathsf{true},x)\},\{(\mathsf{true},x),(x_3,x)\},\{(\mathsf{true},x),(x_3,x)\}\}, \\
& \ \{\{(\mathsf{true},x),(\mathsf{false},x)\},\{(\mathsf{true},x),(x_3,x)\},\{(\mathsf{false},x),(x_3,x)\}\}, \\
& \ \{\{(\mathsf{false},x),(x_2,x)\},\{(\mathsf{false},x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \\
\Rrightarrow^*\ & \{\{\{(\mathsf{true},x),(x_3,x)\},\{(\mathsf{false},x),(x_3,x)\}\}, \\
& \ \{\{(\mathsf{false},x),(x_2,x)\},\{(\mathsf{false},x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \\
\Rrightarrow^*\ & \{\{\{(\mathsf{false},x),(x_2,x)\},\{(\mathsf{false},x),(x_3,x)\},\{(x_2,x),(x_3,x)\}\}\} \Rrightarrow^* \emptyset
\end{aligned}$$

Proof of Termination.

- part 1: (instantiate) rule: instantiate $(x, f(\dots))$ by all $[x/c(\vec{xs})]$ for all $c \in \mathcal{C}$
  - define difference measure $|t - \ell|$ for term pairs (t, ℓ) of matching problems
    - $|x - \ell|$ is the number of function symbols in ℓ,
    - $|f(t_1, \dots, t_n) - f(\ell_1, \dots, \ell_n)| = \sum_{i=1}^{n} |t_i - \ell_i|$, and
    - $|t - \ell| = 0$ in all other cases.
  - define $|pp| = \sum_{mp \in pp, (t,\ell) \in mp} |t - \ell|$
  - define $P \succ P'$ as $\{|pp| \mid pp \in P\} >^{mul} \{|pp| \mid pp \in P'\}$
  - whenever $P \Rrightarrow P'$ then $P \succeq P'$, and $P \succ P'$ for (instantiate)
- part 2: (instantiate') rule: instantiate $\{(t, y), (t', y)\}$ by all $[x/c(\vec{xs})]$ if $t|_p = x \neq t'|_p$ if sort of x is finite
  - define $|x|$ as maximal size of term of (finite) sort of x
  - define $|pp| = \sum_{x \in mp, \text{sort of } x \text{ is finite}} |x|$
  - define $P \succ P'$ as $\{|pp| \mid pp \in P\} >^{mul} \{|pp| \mid pp \in P'\}$
  - whenever $P \Rrightarrow P'$ (except instantiate) then $P \succeq P'$, and $P \succ P'$ for (instantiate')
- part 3: remaining rules are trivially terminating $\square$

Auxiliary Algorithms

- technical precondition of decision procedure: each sort must be inhabited by constructor ground term
- figure out whether sorts are inhabited by constructor ground terms
  $\Longrightarrow$ easy marking algorithm to detect inhabited sorts
  - whenever $c : \sigma_1 \times \ldots \times \sigma_n \to \sigma \in \mathcal{C}$ and all $\sigma_1, \ldots, \sigma_n$ are marked, then mark σ
  - iterate until no new sorts are marked
  - finally, sort σ is marked iff $\mathcal{T}(\mathcal{C})_\sigma \neq \emptyset$
- tried adjustment to detect infinite sorts, by marking recursive constructors
  $\Longrightarrow$ not successful, e.g., consider mutually recursive sorts
  $\Longrightarrow$ perhaps encode as lasso problem in graph
- however, simple marking algorithm is successful by marking finite sorts
  - whenever $c : \sigma_1 \times \ldots \times \sigma_n \to \sigma \in \mathcal{C}$ and all $\sigma_1, \ldots, \sigma_n$ are marked for all $c \in \mathcal{C}_\sigma$, then mark σ
  - iterate until no new sorts are marked
  - finally, sort σ is marked iff $|\mathcal{T}(\mathcal{C})_\sigma| < \infty$

Experimental Results

- see demo

Summary

- history of CeTA, including new developments in 2022 and 2023
- small certificates for joins via greedy partitioning algorithm
- novel decision procedure for pattern completeness
  - verified in Isabelle
  - based on IsaFoR and Yamada's library on sorted terms
  - 2,800 lines: main algorithm
  - 385 lines: checking that sorts are inhabited + decide whether sort is infinite
- future work in Isabelle/HOL (with Dohan and Akihisa):
  verify rewriting induction rules in sorted setting for certifying ground confluence proofs of AGCP

Questions?

ISR 2024 – 14th International School on Rewriting

- August 25 – September 1, 2024, Obergurgl, Austria
- three tracks
  (A) introductory course to first-order term rewriting
  (B) introductory course to lambda-calculus and type theory
  (C) advanced courses on recent developments and applications
- lecturers
  - Aart Middeldorp (A)
  - Herman Geuvers (B)
  - Niels van der Weide (B)
  - Frédéric Blanqui (C)
  - Ugo Dal Lago (C)
  - Nao Hirokawa (C)
  - Cynthia Kop (C)
  - Sarah Winkler (C)

`http://cl-informatik.uibk.ac.at/isr24/`