

# Towards a Verified Decision Procedure for Confluence of Ground Term Rewrite Systems in Isabelle/HOL\*

Bertram Felgenhauer<sup>1</sup> Aart Middeldorp<sup>1</sup>  
T. V. H. Prathamesh<sup>1</sup> Franziska Rapp<sup>1</sup>

Department of Computer Science, University of Innsbruck, Austria  
{bertam.felgenhauer,aart.middeldorp,venkata.turaga,franziska.rapp}@uibk.ac.at

## Abstract


Confluence is a decidable property of ground rewrite systems. We present a formalization effort in Isabelle/HOL of the decision procedure based on ground tree transducers.

## 1 Introduction

Confluence is an undecidable property of term rewrite systems. Oyamaguchi [7] was the first to prove the decidability of confluence for *ground* rewrite systems. Dauchet, Heuillard, Lescanne, and Tison [3] presented a simpler decidability proof based on *ground tree transducers*. Comon, Godoy, and Nieuwenhuis [2] were the first to prove that confluence of ground rewrite systems is decidable in polynomial time and Felgenhauer [6] presented a cubic time algorithm.

In [4] the decision procedure of [3] was extended to *left-linear right-ground* rewrite systems. Dauchet and Tison [5] showed that the *first-order theory* of rewriting is decidable for ground rewrite systems. In this theory properties definable by a first-order formula over rewrite predicates like  $\rightarrow$  and  $\rightarrow^*$  are expressible. This includes confluence. The decision procedure (extended to left-linear right-ground rewrite systems) is implemented in FORT [8]. Ground tree transducers and their closure properties play a key role in the decision procedure.

Our long-term aim is to formalize the decision procedure in the proof assistant Isabelle/HOL such that the output of FORT can be certified. In this paper we present a formalization of ground tree transducers their closure properties. Furthermore, a number of results on the interplay between rewriting and ground tree transducers are formalized, bringing us close to the first formalized proof of the decidability of confluence of ground rewrite systems.

Our formalization is based on IsaFoR [9]<sup>1</sup>. Our own development can be found at <http://cl-informatik.uibk.ac.at/software/fortissimo/iwc2018/>. Furthermore most definitions, theorems, and lemmas directly correspond to the formalization. These are indicated by the  symbol, which links to a HTML presentation in the PDF version of the paper.

## 2 Preliminaries

We assume familiarity with term rewriting and (bottom-up) tree automata. Let  $\mathcal{R}$  be a ground term rewrite system (TRS for short) over a signature  $\mathcal{F}$ , where  $\mathcal{F}$  contains at least one constant (which is assured if  $\mathcal{R} \neq \emptyset$ .) A tree automaton  $\mathcal{A} = (Q, Q_f, \Delta)$  consists of a set of states  $Q$ , a set of final states  $Q_f$ , and a set of transitions  $\Delta$ . Ordinary transitions have the form  $f(q_1, \dots, q_n) \rightarrow q$  where  $q_1, \dots, q_n$  and  $q$  are states, and  $f \in \mathcal{F}$  has arity  $n$ , while  $\epsilon$ -transitions

---

\*This work is supported by the Austrian Science Fund (FWF): project P30301.

<sup>1</sup><http://cl-informatik.uibk.ac.at/isafor>

**inductive** *gtt\_accept'* :: ('q, 'f) *gtt*  $\Rightarrow$  ('f, 'q) *term*  $\Rightarrow$  ('f, 'q) *term*  $\Rightarrow$  *bool*  
**for**  $\mathcal{G}$  **where**  
*mctxt* [*intro*]: *length ss = length ts*  $\Longrightarrow$  *num\_holes C = length ss*  $\Longrightarrow$   
 $\forall i < \text{length } ts. \exists q. q \in \text{ta\_res } (\text{fst } \mathcal{G}) (ss ! i) \wedge q \in \text{ta\_res } (\text{snd } \mathcal{G}) (ts ! i) \Longrightarrow$   
*gtt\_accept' G (fill\_holes C ss) (fill\_holes C ts)*

Listing 1: Definition of GTT acceptance.

$p \rightarrow q$  are between states. Noting that the transitions are ground rewrite rules, we write  $\rightarrow_{\mathcal{A}}$  for  $\rightarrow_{\Delta}$ . To decide confluence of  $\mathcal{R}$ , first a ground tree transducer (GTT for short)  $\mathcal{G} = (\mathcal{A}, \mathcal{B})$  is constructed that recognizes a relation in between<sup>2</sup>  $\rightarrow_{\mathcal{R}}$  and  $\rightarrow_{\mathcal{R}}^*$ . A GTT  $\mathcal{G}$  consists of two tree automata  $\mathcal{A}$  and  $\mathcal{B}$  that operate on the same signature. A pair of ground terms  $s$  and  $t$  is accepted by  $\mathcal{G}$  if  $s \rightarrow_{\mathcal{A}}^* \cdot \mathcal{B} \leftarrow t$ ; we denote the relation consisting of all such pairs  $(s, t)$  by  $\mathcal{L}(\mathcal{G})$ .

Next the transitive closure  $\mathcal{G}^*$  of  $\mathcal{G}$  is computed by an iterative procedure in which certain  $\epsilon$ -transitions are added to the involved tree automata. Since  $(\rightarrow_{\mathcal{R}}^*)^* = \rightarrow_{\mathcal{R}}^*$ , the GTT  $\mathcal{G}^*$  recognizes reachability. The relation  $\mathcal{R} \leftarrow^*$  is recognized by the inverse  $\mathcal{G}^{*-}$  of  $\mathcal{G}^*$  (which is simply obtained by interchanging the two tree automata that make up  $\mathcal{G}^*$ ).


The GTTs  $\mathcal{G}^*$  and  $\mathcal{G}^{*-}$  are composed to obtain GTTs  $\mathcal{G}_1$  and  $\mathcal{G}_2$  that recognize the relations  $\uparrow_{\mathcal{R}} = \mathcal{R} \leftarrow^* \cdot \rightarrow_{\mathcal{R}}^*$  and  $\downarrow_{\mathcal{R}} = \rightarrow_{\mathcal{R}}^* \cdot \mathcal{R} \leftarrow^*$ . The final step of the decision procedure is the inclusion check  $\mathcal{L}(\mathcal{G}_1) \subseteq \mathcal{L}(\mathcal{G}_2)$ . In [4] this is done by applying an ad-hoc recognizability preserving transformation from GTTs to tree automata over an extended signature, and subsequently use a decision procedure for tree language inclusion. In our formalization we instead associate  $\text{RR}_2$  automata to  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , followed by an inclusion check for  $\text{RR}_n$  automata. The reason for this approach is that  $\text{RR}_n$  automata play a key role in the decision procedure for the first-order theory of rewriting. Therefore we can reuse our results when formalizing further aspects of the theory implemented in FORT.

The most complicated part of the above procedure is the closure of GTT relations under composition and transitive closure. Proofs of these results are presented in detail in [1, Section 3.2]. Below we present (simpler) paper proofs that correspond to our formalization.

### 3 Formalizing the Confluence Check

We rely on `IsaFoR`'s formalization of tree automata, where a tree automaton is a triple consisting of the set of final states (which is irrelevant for GTTs), the set of ordinary transitions, and the set of epsilon transitions. The set of states of the automaton is left implicit. For example,

$$\text{ta.make } \{0\} \{a [] \rightarrow 1, f [1] \rightarrow 0\} \{(0, 1)\}$$

would be an automaton that accepts  $f^k(a)$  for  $k \geq 1$  (the transitions are  $a \rightarrow 1$ ,  $f(1) \rightarrow 0$ , and  $0 \rightarrow 1$ ). We can check whether an automaton  $\mathcal{A}$  accepts a term  $t$  in state  $q$  using  $q \in \text{ta\_res } \mathcal{A} t$ . The language accepted by  $\mathcal{A}$  is provided as *ta\_lang*  $\mathcal{A}$ . GTTs are formalized as pairs of tree automata with the same state and function symbol types. The relation accepted by a GTT is formalized by the predicate *gtt\_accept*, which is equivalent  to *gtt\_accept'* given in Listing 1.

The first step of the construction is to obtain a GTT from the given ground TRS  $\mathcal{R}$ . To this end, we follow the construction by Dauchet et al. [4]. Let  $\langle s \rangle$  be a state for each subterm  $s \leq \mathcal{R}$

<sup>2</sup>While it is true that  $\nrightarrow_{\mathcal{R}}$  is recognizable by a GTT, we have not yet formalized this fact.

**definition**  $cmn\_ta\_rules::('f, 'v) \text{ term set} \Rightarrow (('f, 'v) \text{ term option}, 'f) \text{ ta\_rule set}$   
**where**

$$cmn\_ta\_rules \ T = \{(f \ (map \ Some \ ts) \rightarrow \ Some \ (Fun \ f \ ts)) \mid f \ ts \ t. \ Fun \ f \ ts \sqsubseteq \ t \wedge \ t \in \ T\}$$

**definition**  $trs\_to\_ta\_A::('f, 'v) \text{ trs} \Rightarrow (('f, 'v) \text{ term option}, 'f) \text{ ta}$  **where**  
 $trs\_to\_ta\_A \ R = ta.make \ \{\} \ (cmn\_ta\_rules \ (TRS\_terms \ R))$   
 $\{(Some \ l, \ Some \ r) \mid l \ r. \ (l, r) \in \ R\}$

Listing 2: Associating a GTT to a TRS.

(meaning there is a rule  $l \rightarrow r$  in  $\mathcal{R}$  such that  $s \sqsubseteq l$  or  $s \sqsubseteq r$ ). Let  $\mathcal{G} = (\mathcal{A}, \mathcal{B})$  where

$$\Delta_{\mathcal{A}} = \{f(\langle t_1 \rangle, \dots, \langle t_n \rangle) \rightarrow \langle f(t_1, \dots, t_n) \rangle \mid f(t_1, \dots, t_n) \sqsubseteq \mathcal{R}\} \cup \{\langle l \rangle \rightarrow \langle r \rangle \mid l \rightarrow r \in \mathcal{R}\}$$

and  $\Delta_{\mathcal{B}}$  is defined symmetrically (replacing  $\langle l \rangle \rightarrow \langle r \rangle$  by  $\langle r \rangle \rightarrow \langle l \rangle$  in the second subset). In the formalization,  $\langle s \rangle$  is represented by *Some s*.<sup>3</sup> This gives rise to the definitions in Listing 2. The resulting GTT is suitable for simulating sequences of  $\mathcal{R}$  steps, by the following theorem.

**Theorem 1.**  $\rightarrow_{\mathcal{R}} \subseteq \mathcal{L}(\mathcal{A}, \mathcal{B}) \subseteq \rightarrow_{\mathcal{R}}^*$

*Example 2.* The construction is illustrated on the ground TRS  $\mathcal{R}$  consisting of the rules  $a \rightarrow f(a)$ ,  $a \rightarrow b$ , and  $f(b) \rightarrow c$ . We construct the GTT  $\mathcal{G} = (\mathcal{A}, \mathcal{B})$  with  $\Delta$  consisting of the rules

$$a \rightarrow \langle a \rangle \quad b \rightarrow \langle b \rangle \quad c \rightarrow \langle c \rangle \quad f(\langle a \rangle) \rightarrow \langle f(a) \rangle \quad f(\langle b \rangle) \rightarrow \langle f(b) \rangle$$

to recognize all subterms in the rules of  $\mathcal{R}$ ,  $\Delta_{\mathcal{A}} = \Delta \cup \{\langle a \rangle \rightarrow \langle f(a) \rangle, \langle a \rangle \rightarrow \langle b \rangle, \langle f(b) \rangle \rightarrow \langle c \rangle\}$ , and  $\Delta_{\mathcal{B}} = \Delta \cup \{\langle f(a) \rangle \rightarrow \langle a \rangle, \langle b \rangle \rightarrow \langle a \rangle, \langle c \rangle \rightarrow \langle f(b) \rangle\}$ . Note that  $L(\mathcal{G})$  accepts more than  $\mapsto_{\mathcal{R}}$ . For instance,  $(a, f(b)) \in L(\mathcal{G})$  as  $a \rightarrow_{\mathcal{A}}^* \langle f(a) \rangle \xrightarrow{\mathcal{B}}^* f(b)$  but  $a \not\mapsto_{\mathcal{R}} f(b)$  does not hold.

To illustrate one of the minor (but tedious) issues that come up in the formalization, note that the state type of the GTT seeps into terms accepted by the GTT: they are objects of type  $('f, ('f, 'q) \text{ term option}) \text{ term}$ . On the other hand,  $\mathcal{R}^*$  is a relation between terms of type  $('f, 'v) \text{ term}$ , with a completely different variable type. But actually, since we deal with ground terms, the variable type does not matter. In order to express this property, we use the existent *adapt\_vars* function that changes the variable type arbitrarily.

The next step in the decision procedure is the computation of the transitive closure. However, that computation builds on top of the composition of GTT relations, so we present that first. The composition combines the transitions of the constituent GTTs, and adds carefully chosen epsilon transitions.

**Definition 3.**  Let  $\mathcal{G}_1 = (\mathcal{A}_1, \mathcal{B}_1)$  and  $\mathcal{G}_2 = (\mathcal{A}_2, \mathcal{B}_2)$  be GTTs. We let

$$GTT\_comp(\mathcal{G}_1, \mathcal{G}_2) = (\Delta_{\mathcal{A}_1} \cup \Delta_{\mathcal{A}_2} \cup \Delta_{\epsilon}(\mathcal{B}_1, \mathcal{A}_2), \Delta_{\mathcal{A}_1} \cup \Delta_{\mathcal{A}_2} \cup \Delta_{\epsilon}(\mathcal{A}_2, \mathcal{B}_1))$$

Here  $\Delta_{\epsilon}(\mathcal{A}, \mathcal{B}) = \{(p, q) \mid t \rightarrow_{\mathcal{A}}^* p \text{ and } t \rightarrow_{\mathcal{B}}^* q \text{ for some } t \in \mathcal{T}(\mathcal{F})\}$ .

This construction is simplified compared to [1, 3]. Compared to [3], only  $\epsilon$ -transitions are added, while [1] actually adds fewer  $\epsilon$ -transitions than our definition, but at the cost of a less symmetric definition.

<sup>3</sup>This use of the option type is not really necessary, but it was helpful to distinguish states and terms while developing the proofs.

*Example 4.* Continuing Example 2, we let  $Q$  be the set of states in  $\Delta$  and compute  $\Delta_\epsilon(\mathcal{A}, \mathcal{B}) = \text{Id}_Q \cup \{\langle \mathbf{b} \rangle \rightarrow \langle \mathbf{a} \rangle, \langle \mathbf{c} \rangle \rightarrow \langle \mathbf{f}(\mathbf{b}) \rangle\} \cup \{\langle \mathbf{c} \rangle, \langle \mathbf{f}(\mathbf{a}) \rangle, \langle \mathbf{f}(\mathbf{b}) \rangle\} \times \{\langle \mathbf{a} \rangle, \langle \mathbf{f}(\mathbf{a}) \rangle\}$  and  $\Delta_\epsilon(\mathcal{B}, \mathcal{A}) = \Delta_\epsilon(\mathcal{A}, \mathcal{B})^-$ . For instance, the transition rule  $\langle \mathbf{c} \rangle \rightarrow \langle \mathbf{a} \rangle \in \Delta_\epsilon(\mathcal{A}, \mathcal{B})$  is witnessed by the term  $\mathbf{f}(\mathbf{b})$ .

**Theorem 5.**  $\checkmark$  *If  $R_1$  and  $R_2$  are recognizable relations then  $R_1 \circ R_2$  is a recognizable relation. More precisely, if  $R_1$  and  $R_2$  are recognized by  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , where the states of  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are disjoint, then  $R_1 \circ R_2$  is recognized by  $\text{GTT\_comp}(\mathcal{G}_1, \mathcal{G}_2)$ .*

The transitive closure of a GTT  $\mathcal{G}$  is computed by taking  $\mathcal{G}_0 = \mathcal{G}$  and then iterating  $\mathcal{G}_{n+1} = \text{GTT\_comp}(\mathcal{G}_n, \mathcal{G}_n)$  until a fixed point is reached. If  $\mathcal{G}$  is finite, this process terminates.

$\checkmark$  We have proved that the GTT produced that way accepts the transitive closure of the original GTT.  $\checkmark$  One interesting aspect is that transitivity of the resulting GTT relation follows immediately from the first part of the proof of Theorem 5 (where the assumption that the states of  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are disjoint is not used).  $\checkmark$

*Example 6.* Returning to our example, let  $\mathcal{A}_1 = \mathcal{A} \cup \Delta_\epsilon(\mathcal{B}, \mathcal{A})$  and  $\mathcal{B}_1 = \mathcal{B} \cup \Delta_\epsilon(\mathcal{A}, \mathcal{B})$ . The GTT  $\mathcal{G}_1 = (\mathcal{A}_1, \mathcal{B}_1)$  recognizes  $\rightarrow_{\mathcal{R}}^*$  while its inverse  $\mathcal{R}^* \leftarrow$  is recognized by  $\mathcal{G}_1^- = (\mathcal{B}_1, \mathcal{A}_1)$ . Next we compose  $\mathcal{G}_1^-$  and  $\mathcal{G}_1$  to obtain a GTT  $\mathcal{G}_\uparrow$  that recognizes  $\mathcal{R}^* \leftarrow \cdot \rightarrow_{\mathcal{R}}^*$ . This requires a renaming of states in one of the GTTs. Similarly, composing  $\mathcal{G}_1$  and  $\mathcal{G}_1^-$  produces a GTT  $\mathcal{G}_\downarrow$  recognizing the joinability relation  $\rightarrow_{\mathcal{R}}^* \cdot \mathcal{R}^* \leftarrow$ .

Finally, we need to check whether one GTT language is a subset of another one. To this end, we formalized the result  $\checkmark$  that any GTT relation is an  $\text{RR}_2$  relation, where  $\text{RR}_n$  relations are a way of capturing  $n$ -ary relations on terms as regular tree languages [1]. (The detour via GTTs is necessary because  $\text{RR}_2$  relations are not closed under transitive closure.)

**Theorem 7.**  $\checkmark$  *Let  $\mathcal{R}$  be a ground TRS and let  $\mathcal{G} = (\mathcal{A}, \mathcal{B})$  be the GTT simulating  $\mathcal{R}$ -steps as in Theorem 1. Then  $\mathcal{R}$  is confluent on ground terms if and only if*

$$\text{ta\_lang}(\text{GTT\_to\_RR2}(\text{GTT\_comp}(\mathcal{G}^{*-}, \mathcal{G}^{*-}))) \subseteq \text{ta\_lang}(\text{GTT\_to\_RR2}(\text{GTT\_comp}(\mathcal{G}^*, \mathcal{G}^{*-})))$$

*Example 8.* To finish our running example, we transform  $\mathcal{G}_\uparrow$  and  $\mathcal{G}_\downarrow$  into  $\text{RR}_2$  automata. A subsequent language inclusion check returns a negative answer from which we infer that the TRS  $\mathcal{R}$  lacks confluence.

Note that the results presented so far are purely theoretical, and cannot be executed directly. Here we sketch how to derive executable code for  $\Delta_\epsilon$ , cf. Definition 3. Note that a direct implementation of the definition would require iterating over all ground terms  $t$ , of which there are infinitely many. The first step is to define an inductive set  $\Delta'_\epsilon$   $\checkmark$  that is equal to  $\Delta_\epsilon$ :  $\checkmark$

$$\frac{f(\vec{p}) \rightarrow p \in \mathcal{A} \quad f(\vec{q}) \rightarrow q \in \mathcal{B} \quad \text{len } \vec{p} = \text{len } \vec{q} = n \quad (p_i, q_i) \in \Delta'_\epsilon \quad (1 \leq i \leq n)}{(p, q) \in \Delta'_\epsilon} \text{cong}$$

$$\frac{(p, q) \in \Delta'_\epsilon \quad p \rightarrow p' \in \mathcal{A}}{(p', q) \in \Delta'_\epsilon} \epsilon_1 \quad \frac{(p, q) \in \Delta'_\epsilon \quad q \rightarrow q' \in \mathcal{B}}{(p, q') \in \Delta'_\epsilon} \epsilon_2$$

We then plug this into a generic algorithm for Horn inference (which we regard as the foundation of saturation algorithms), which works on inference rules of the shape  $a_1 \cdots a_n \rightarrow a$ , where  $a_i, a$  are all of the same type. The idea here is that proving correctness and termination can be done once and for all on this generic level, and then be reused for any saturation procedure.

In the case of  $\Delta'_\epsilon$ , the inference rules work on pairs of states  $(p, q)$ , i.e., the potential elements of  $\Delta'_\epsilon$ . We turn the inferences of  $\Delta'_\epsilon$  into Horn clauses by keeping only the premises of the form  $(p, q) \in \Delta'_\epsilon$ , evaluating the other premises statically based on  $\mathcal{A}$  and  $\mathcal{B}$ .  $\checkmark$  Then we show that

the resulting Horn inferences characterize  $\Delta'_\epsilon$ .  In order to use the generic procedure, we have to provide a function that computes the inferences with no premises ( $\Delta'_\epsilon\_infer0$ ),  and a function that computes inferences that use a particular premise  $(p, q)$  and other premises from a given set ( $\Delta'_\epsilon\_infer1$ ).  With those functions we can instantiate the generic procedure.

$$\Delta'_\epsilon\_impl \mathcal{A} \mathcal{B} = saturate\_impl (\Delta'_\epsilon\_infer0 \mathcal{A} \mathcal{B}) (\Delta'_\epsilon\_infer1 \mathcal{A} \mathcal{B})$$

Partial correctness follows from partial correctness of the generic procedure.

## 4 Conclusion

We have outlined an ongoing effort to formalize decidability of (ground) confluence of ground TRSs, which is a useful test case for the decidability of the full first-order theory of rewriting for ground TRSs. The main remaining challenge is to provide executable algorithms for all these results and prove their termination. We have already made significant progress to this end; in fact there are executable versions of all constructions needed for the confluence check, except for the final tree language subset check.

Our immediate goal is to provide a verified confluence checker for ground TRSs. Many tasks remain as future work. We want to adapt the basic TRS to GTT construction to cover the larger class of linear, variable separated (extended) TRSs. For the full first-order theory of rewriting, while we already have constructions for intersection, union, complement, cylindrification and projection (the latter are used for dealing with quantifiers), these are not yet executable.

## References

- [1] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available from [www.grappa.univ-lille3.fr/tata](http://www.grappa.univ-lille3.fr/tata), 2008.
- [2] H. Comon, G. Godoy, and R. Nieuwenhuis. The confluence of ground term rewrite systems is decidable in polynomial time. In *Proc. 42nd FOCS*, pages 298–307, 2001. doi: [10.1109/SFCS.2001.959904](https://doi.org/10.1109/SFCS.2001.959904).
- [3] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of ground term rewriting systems. In *Proc. 2nd LICS*, pages 353–359, 1987.
- [4] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewriting systems and of other related term rewriting systems. *I&C*, 88(2):187–201, 1990. doi: [10.1016/0890-5401\(90\)90015-A](https://doi.org/10.1016/0890-5401(90)90015-A).
- [5] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. 5th LICS*, pages 242–248, 1990. doi: [10.1109/LICS.1990.113750](https://doi.org/10.1109/LICS.1990.113750).
- [6] B. Felgenhauer. Deciding confluence of ground term rewrite systems in cubic time. In *Proc. 23rd RTA*, volume 15 of *LIPICs*, pages 165–175, 2012. doi: [10.4230/LIPICs.RTA.2012.165](https://doi.org/10.4230/LIPICs.RTA.2012.165).
- [7] M. Oyamaguchi. The Church-Rosser property for ground term-rewriting systems is decidable. *Theoretical Computer Science*, 49:43–79, 1987. doi: [10.1016/0304-3975\(87\)90100-9](https://doi.org/10.1016/0304-3975(87)90100-9).
- [8] F. Rapp and A. Middeldorp. Automating the first-order theory of left-linear right-ground term rewrite systems. In *Proc. 1st FSCD*, volume 52 of *LIPICs*, pages 36:1–36:12, 2016. doi: [10.4230/LIPICs.FSCD.2016.36](https://doi.org/10.4230/LIPICs.FSCD.2016.36).
- [9] R. Thiemann and C. Sternagel. Certification of termination proofs using CeTA. In *Proc. 22nd TPHOLs*, volume 5674 of *LNCS*, pages 452–468, 2009. doi: [10.1007/978-3-642-03359-9\\_31](https://doi.org/10.1007/978-3-642-03359-9_31).