Cetera: Certified Termination with Agda

Dieter Hofbauer
ASW Saarland

Johannes Waldmann
HTWK Leipzig

Abstract

We report on Cetera, a program and library for checking termination certificates for string rewriting. The library is certified in the dependently typed programming language Agda. For now, Cetera contains well-founded monotone monoids (instantiated to weights and standard matrices) and allows modular removal of rules (by relative termination). We are currently working on the RFC (right-hand sides of forward closures) theorem. Proofs in Agda are constructive, so they sometimes differ from standard presentation.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Theory of computation → Rewrite systems

Keywords and phrases termination, string rewriting

1 Introduction

The termination method of sparse tiling [8] for string rewriting systems, implemented in Matchbox and MultumNonMulta, turned out to be quite powerful (for present benchmarks, at least). It uses the RFC (right-hand sides of forward closures) method, which was absent from libraries for verified termination [6, 4, 11]. The goal of our library Cetera (`https://git.imn.htwk-leipzig.de/waldmann/cetera/`) is to verify sparse tiling.

Matchbox is implemented in the pure functional programming language Haskell. Haskell's type system is moving towards dependent types (for example, it has generalized algebraic data types), and this suggests verification with Agda [5], a dependently typed programming language based on Martin-Löf type theory.

2 Termination and Non-Termination in Agda

Termination is formalized constructively by an accessibility predicate [10]. The following Agda type `SN R x` specifies that for relation R on a set a, the element $x \in a$ is terminating (strongly normalizing), if each R-successor y of x is terminating.

```
data SN {a : Set} (R : Rel a) (x : a) : Set  where
  sn : (forall (y : a) -> R x y -> SN R y) -> SN R x
```

A termination proof of R is an Agda expression of type `forall (x : a) -> SN R x`, that is, an Agda-computable function that builds (more precisely, bounds the depth of) the R-derivation tree of input x. Ultimately, this method reduces termination of R to termination of Agda programs in our library, checked by Agda's type checker [1, 2]. There are terminating R that have no such proof, since Agda is, by design, not Turing-complete.

The element x is non-terminating if there is a function $f : \mathbb{N} \to a$ that represents an infinite R-derivation $x = f(0) \to_R f(1) \to_R \dots$, formally,

```
data INF {a : Set} (R : Rel a) (x : a) : Set  where
  inf : (f : Nat -> a) -> (f zero == x)
      -> (forall (y : Nat) -> R (f y) (f (succ y))) -> INF R x
```

Again, since the function f is Agda-computable, we miss some forms of non-termination.

We can easily prove in Agda that `SN R x` and `INF R x` cannot hold at the same time.

3 The Present: Well-founded Monotone Monoids

We consider pairs $(>, \geq)$ where $>$ is terminating, $\geq$ is reflexive and transitive, and $> \circ \geq \subseteq >$ as well as $\geq \circ > \subseteq >$. Then $R \subseteq >$ and $S \subseteq \geq$ implies termination of R relative to S, allowing to remove rules (of R) in an incremental proof of termination (of $R \cup S$). We realize such pairs by orders on matrices [9] of shape $E_{1,d} = \{m \mid m \in \mathbb{N}^{d\times d}, m_{1,1} \geq 1, m_{d,d} \geq 1\}$ where $L > R$ if $(L - R) \in P_{1,n} = \{m \mid m \in \mathbb{N}^{d\times d}, m_{1,d} \geq 1\}$ and $L \geq R$ if $(L - R) \in \mathbb{N}^{d\times d}$.

In the Agda program, we represent matrices as functions `Fin d -> Fin d -> Nat`, where `Fin d` denotes $\{0, \dots, d-1\}$. This allows for easier proofs than the concrete representation `Vec d (Vec d Nat)`, where `Vec d` denotes lists of length d. If we want to use Agda's built-in equality for functional matrices, then we need the extensionality axiom.

The extracted code for checking certificates does indeed multiply matrices. The functional matrix representation does not share computations for intermediate products, so a chain of matrix multiplications has exponential cost. This can be reduced to linear cost in the length of the chain by conversion from function to data, which allows sharing.

4 Implementation and Experiments

For now, Cetera uses Haskell syntax for systems and certificates. Identifiers are renamed to numbers. Printers and parsers are obtained from `deriving (Show, Read)`. A certificate for $\{abaabb \to aababba\}$ (Zantema_04/z049) is

```
Certificate
  { system = [Rule  { lhs = [0, 1, 0, 0, 1, 1], rhs = [0, 0, 1, 0, 1, 1, 0]}]
  , reason = MatrixInterpretation
    { minterpretation = MatrixInterpretation
      { dim = 4
      , int = [ (1, [[1, 0, 0, 0], [0, 0, 1, 0], [0, 1, 1, 1], [0, 0, 0, 1]])
              , (0, [[1, 1, 0, 0], [0, 1, 0, 0], [0, 0, 0, 0], [0, 0, 0, 1]]) ]}
    , remove = [Rule  { lhs = [0, 1, 0, 0, 1, 1], rhs = [0, 0, 1, 0, 1, 1, 0]}]
    , sub = Certificate  { system = [], reason = Empty}}}
```

Using only weights and matrix interpretations, with a time-out of 30 seconds, Matchbox proves termination for 488 (of 1658) benchmarks in SRS_Standard. The largest certificate occurs for ICFP_2010/26871. It has 60k non-whitespace characters, and takes 0.07 seconds to verify. Total verification time over all certificates is < 5 seconds.

Larger certificates will appear with semantic labelling via full or sparse tiling [8].

5 The Future: Right-Hand Sides of Forward Closures

Termination of R follows from termination of R on the set of right-sides of forward closures $\mathsf{RFC}(R)$ [7]. We formalize this claim in Agda as

```
sn-RFC : {C : Set l} (R : Rel (List C) l)
  -> (forall (x : List C) -> RFC R x -> SN-from (Rewrites R) x) -> SN (Rewrites R)
```

The classical proof is non-constructive, as it derives a contradiction from the existence of an infinite derivation. For a constructive proof, we need to embed an arbitrary R-derivation into a well-founded order. The idea is to simulate the original derivation by a derivation on sequences of *blocks* where a block is a single letter, or the right-hand side of a forward closure. The key idea is that a step on a sequence of blocks is length-lexicographically (from the right) decreasing w.r.t. the order $(\to_R \cup \sqsupset_s)^+$ on blocks. Here

$\sqsubset_s$ is the strict suffix relation, and it comes into play since an extension (narrowing) step of a closure eats a prefix of the neighbouring block. We use a commutation property [3] $(T \circ S) \subseteq (S \circ T) \wedge \mathsf{SN}(T) \wedge \mathsf{SN}(S) \Rightarrow \mathsf{SN}(S \cup T)$ for $T = \sqsupset_s$, $S = \to_R$. Note that $\to_R^+$ is only used when starting from $\mathsf{RFC}(R)$, as in the premise of Theorem `sn-RFC`.

We are currently working on proving the theorem. To use it for certificate checking, Cetera is still missing a finite description of (an over-approximation of) $\mathsf{RFC}(R)$. This could be a matchbound automaton, or a shift automaton (of tiles) [8].

References

1 Andreas Abel. foetus - termination checker for simple functional programs. 1998. doi:10.48550/ARXIV.2407.06924.

2 Andreas Abel. Termination checking with types. *RAIRO Theor. Informatics Appl.*, 38(4):277–319, 2004. doi:10.1051/ITA:2004015.

3 Leo Bachmair and Nachum Dershowitz. Commutation, transformation, and termination. In *Proc. Conference on Automated Deduction, CADE*, volume 230 of *LNCS*, pages 5–20. Springer, 1986. doi:10.1007/3-540-16780-3_76.

4 Frédéric Blanqui and Adam Koprowski. CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Math. Struct. Comput. Sci.*, 21(4):827–859, 2011. doi:10.1017/S0960129511000120.

5 Ana Bove, Peter Dybjer, and Ulf Norell. A brief overview of Agda - A functional language with dependent types. In *Proc. Theorem Proving in Higher Order Logics, TPHOLs*, volume 5674 of *LNCS*, pages 73–78. Springer, 2009. doi:10.1007/978-3-642-03359-9_6.

6 Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. Certification of automated termination proofs. In *Proc. Frontiers of Combining Systems, FroCoS*, volume 4720 of *LNCS*, pages 148–162. Springer, 2007. doi:10.1007/978-3-540-74621-8_10.

7 Nachum Dershowitz. Termination of linear rewriting systems. In *Proc. International Colloquium on Automata, Languages and Programming, ICALP*, pages 448–458, 1981. doi:10.1007/3-540-10843-2_36.

8 Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Sparse tiling through overlap closures for termination of string rewriting. In *Proc. Formal Structures for Computation and Deduction, FSCD*, volume 131 of *LIPIcs*, pages 21:1–21:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSCD.2019.21.

9 Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In *Proc. Term Rewriting and Applications, RTA*, volume 4098 of *LNCS*, pages 328–342. Springer, 2006. doi:10.1007/11805618_25.

10 Lawrence C. Paulson. Constructing recursion operators in intuitionistic type theory. *J. Symb. Comput.*, 2(4):325–355, 1986. doi:10.1016/S0747-7171(86)80002-5.

11 René Thiemann and Christian Sternagel. Certification of termination proofs using CeTA. In *Proc. Theorem Proving in Higher Order Logics, TPHOLs*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.