Some results on confluence: decision and what to do without
Florent Jacquemard

To cite this version:
Florent Jacquemard. Some results on confluence: decision and what to do without. 5th International Workshop on Confluence, Beniamino Accattoli, Ashish Tiwari Sep 2016, Obergurgl, Austria. hal-01367535

HAL Id: hal-01367535
https://inria.hal.science/hal-01367535
Submitted on 16 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Some results on confluence:
decision and what to do without.

Florent Jacquemard\textsuperscript{1*}

INRIA – Sorbonne Universités
STMS (IRCAM-CNRS-UPMC), Paris
florent.jacquemard@inria.fr

Abstract
We recall first some decidability results on the confluence of TRS, and related properties about unicity of normal forms. In particular we put it in perspective old proofs of undecidability of confluence for the class of flat systems with more recent results, in order to discuss the importance of linearity wrt these decision problems.

Second, we describe a case study on musical rhythm notation involving modeling rewrite systems which are not confluent. In this case, instead of applying rewrite rules directly, we enumerate the equivalence class of a given term using automata-based representations and dynamic programming.

1 Confluence (un)decidability

When term rewriting systems (TRS) are used as models in fields such as functional programming languages semantics, automated deduction or system or program verification, the application of rewrite rules can be highly non-deterministic. Confluence permits to relax from this problem by guaranteeing that divergent reduction will eventually converge to a canonical form, in case of termination. It is therefore an crucial property to decide for TRS.

Decidability of confluence for linear TRS. Confluence of TRS is undecidable in general, even for linear systems (every variable can occur at most once in every left- or right-hand-side of rules) \cite{28}. It has been shown decidable for ground TRS (rewrite rules without variables) \cite{18, 3} and for left-linear right-ground TRS \cite{2}. Polynomial time decision procedures have been proposed years later for ground TRS \cite{1, 22}, for left-shallow-linear and right-ground TRS (every variable can occur at most once and at depth at most one in every left-hand-side of rule) \cite{22}, for linear-shallow TRS (every variable occurs at most once in each rule and at depth at most one) \cite{22, 10}, and for linear and shallow TRS (every variable occurs at most once and at depth at most one in each side rule but can occur twice in a rule) \cite{7}.

Uniqueness of Normal Forms. The decidability of several alternatives to confluence has been studied. A first alternative, uniqueness of normal forms (UN\textsuperscript{=}), implied by confluence, expresses that no two distinct normal forms (irreducible terms) can be equivalent modulo the rewrite system considered. UN\textsuperscript{=} has been shown decidable for ground TRS \cite{28}, and for shallow TRS (without the restriction of linearity) \cite{19}. It is also polynomial time decidable for shallow and linear TRS \cite{24}. It is undecidable for right ground TRS \cite{26}, for linear, non-collapsing (the right-hand-side of rules cannot be a variable), variable-preserving, and depth-two TRS \cite{25}, for

*co-authors for the results mentioned in §1: Ichiro Mitsuhashi, Michio Oyamaguchi, Guillem Godoy, and in S2: Jean Bresson, Masahiko Sakai, Adrien Ycart, Adrien Maire, Pierre Donat Bouillud, Slawek Staworko.
left-linear and left-flat TRS with with depth-two right-hand sides of rules [19] as well as for right-ground, right-flat TRS [25].

A second alternative, unique normalization (UN) expresses that every term can reach at most one normal form using the TRS considered. UN implies UN but the converse is not true. UN is decidable in polynomial time for ground TRS [27], and also for for shallow and linear TRS [9]. On the negative side, UN is undecidable for right-ground TRS [23], for flat TRS (left- and right-hand side of rules have depth at most one) [8], for linear and right-flat TRS [11] and for flat and right-linear TRS [9].

Decidability of confluence for non-linear TRS. The linearity is often considered as a yardstick when considering decision of properties of TRS such as confluence, reachability or joinability. For instance, tree automata based methods sometimes used in this context [18, 3, 2, 9] need, in case of non-linear TRS, generalized models with difficult decision problems. Confluence is shown undecidable for flat (non-linear) TRS [14, 17] by reduction of reachability, also shown undecidable in this case (note that this is in contrast with UN [19]). The latter proofs have been simplified drastically in [8]. However, confluence has been shown decidable for some classes of TRS allowing non-linear rules, like right-ground TRS (without restriction on the left-hand-sides of rules) [16], and shallow and right-linear TRS [12].

The latter proof uses decidability of reachability and joinability, both implied by regularity preservation result. To our knowledge, it is an open question whether confluence is decidable for other classes of TRS preserving regularity such as right-linear and finite-path-overlapping TRS [21] (shallow right-linear TRS are a particular case) or Layer Transducing TRS [20]. It is also interesting to consider the decision of confluence for particular rewriting strategies e.g. bottom-up [5, 6]. Finally, it can be observed that collapsing (right-variable) rules are essential in shifted pairing like constructions for undecidability proofs [14, 17, 8]. It is also unknown whether confluence is decidable for shallow and non-collapsing TRS.

2 What to do when there is no confluence

Traditional music notation is since centuries the standard format for the communication, exchange, and preservation of musical works in Western musical practice. We have been working recently on modeling the notation of rhythm (durations), following an approach based on formal languages and term rewriting.

In common western music notation, durations values are expressed proportionally, by recursive subdivisions of a unit (beat). This hierarchical definition induces naturally tree-structured representations called rhythm trees (RT). Every position in a RT is associated to a duration value. In a simple variant (see Figure 1), the root position is associated a fixed duration value and every non-root position is associated the duration of its parent divided by the number of edges outgoing from it. Moreover, if a leaf position labeled by \( \circ \), the the duration of \( p \) is added to the duration of the next leaf \( p' \) in depth-first-traversal (if it exists). The other leafs may be labeled by symbols giving information on notes, rests etc, and the labels of inner positions are not significant (here we use named after their arity 2, 3, 4…). To a RT, we associate the sequence of durations of the non-\( \circ \) leaves (in dfs). To capture more complex rhythm notations, we use a dag representations not described here.

The RT representations are used in a new tool for the transcription of timestamped event sequences into a music notation [29]. It is implemented as a library of the algorithmic composition framework OpenMusic (Figure 2). We are also developing Music Information Retrieval
Figure 1: Rhythm Trees with associated duration sequences (symbol \( n \) represents a note).

Figure 2: Transcription library for OpenMusic (Ircam). [http://repmus.ircam.fr/cao/rq](http://repmus.ircam.fr/cao/rq)

Tasks based on RT representations, in particular for querying bases of digital music scores (e.g. by query by tapping) and for musicologist research, using similarity measures and tree edit distances.

**Structural theory of RT.** For reasoning about rhythm notations in the above tasks, we define an equivalence between RT with term rewriting rules [15, 4]. For instance, the rules \( 2(o, n) \rightarrow n, 3(o, o, n) \rightarrow n \), ... and \( 2(o, o) \rightarrow o \), ... comply with the semantics of \( o \) presented above, and rules of the form \( 3(2(x_1, x_2), 2(x_3, x_4), 2(x_5, x_6)) \rightarrow 2(3(x_1, x_2, x_3), 3(x_4, x_5, x_6)) \) can be used in order to simplify RT. The TRS containing these simple rules is not confluent. For instance, starting from \( t = 3(2(o, o), 2(n, o), 2(o, o)) \), we have the following non-joinable critical peak:

\[
3(o, 2(n, o), n) \leftrightarrow t \rightarrow 2(3(o, o, n), 3(o, o, n)) \rightarrow 2(n, n).
\]

**Exploring sets of equivalent terms.** Therefore, in order to reason about sets of equivalent terms (in particular the set \([t]\) of terms equivalent to a given RT \( t \)), instead of applying rewriting to reach a canonical normal form that does not exist, we use automata-based characterizations. Some techniques like tree automata completion, can be used to compute a tree automaton recognizing the rewrite closure of a given regular tree set (in particular recognizing \([t]\) given \( \{t\} \)), by superposition of rewrite rules into tree automata transition rules. Such techniques have been used for verify safety properties of programs or systems modeled as TRS (possibly not confluent) by reduction to the problem of emptiness of tree automata intersection (regular tree model checking).
With rewrite rules like the above ones, it is not easy to establish the termination of standard tree automata completion procedures. Even though in our case in practice we only need to consider terms of a bounded depth, hence finite set of terms, it is neither easy to reasonably bound the size of the automaton obtained this way. As an alternative, we have developed an ad hoc construction using the duration sequence associated to a given RT, and a tree automaton representing the family of RT that we want to consider. Once an automaton recognizing $J_tK$ is constructed, we use dynamic programming for the lazy enumeration of this set, according to a measure of tree complexity, following techniques of $k$-best parsing [13]. This way, we can enumerate efficiently the rhythms equivalent to a given rhythm, by increasing complexity.

References


