An Automata Theoretic Approach to Rational Tree Relations

Frank G. Radmacher

Lehrstuhl für Informatik 7, RWTH Aachen, Germany
radmacher@automata.rwth-aachen.de

Abstract. We investigate rational relations over trees. Our starting point is the definition of rational tree relations via rational expressions by Raoult (Bull. Belg. Math. Soc. 1997). We develop a new class of automata, called asynchronous tree automata, which recognize exactly these relations. The automata theoretic approach is convenient for the solution of algorithmic problems (like the emptiness problem). The second contribution of this paper is a new subclass of the rational tree relations, called separate-rational tree relations, defined via a natural restriction on asynchronous tree automata. These relations are closed under composition, preserve regular tree languages, and generate precisely the regular sets in the unary case (all these properties fail for the general model), and they are still more powerful than, for instance, the automatic tree relations.

1 Introduction

Automata definable relations over words are widely investigated. Recognizable, automatic, deterministic rational, and (non-deterministic) rational relations result in a well-known hierarchy [3]. Proper generalizations of these theories to trees have been established over the past years in the case of recognizable relations and automatic relations [2,4]. However, it is still debatable how to obtain a reasonable generalization of rational word relations to trees.

Rational relations over words can be introduced in several equivalent ways: First, they are definable via rational expressions (a generalization of regular expressions), which means that rational relations are generated from the finite relations by closure under union, componentwise concatenation, and Kleene star. On the other hand rational relations are recognized by a generalized model of finite automata, so-called *asynchronous automata* (sometimes also called *multi-tape automata*). The theory was developed in [14,7,8,6,1,3].

Generalizing rational relations to trees (resp. terms) is not straightforward. A survey focussing on binary relations (transductions) was given by Raoult in [17]. Attractive results on rational word relations which one would also like for rational tree relations are the following:

- Applied to unary trees, the rational word relations should be generated (also in the case of n-ary relations).
- A characterization via rational expressions should exist (this implies closure under union, some kind of componentwise concatenation, and Kleene star).

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 424–435, 2008.

- A natural automata theoretic characterization should exist.
- Restricted to unary relations the class of regular tree languages should be generated.
- Binary rational tree relations should be closed under composition.
- Binary rational tree relations (transductions) should preserve regular tree languages.

Our automata theoretic approach is a step towards the definition of *deterministic* rational tree relations (cf. [14,13,11] for the word case) and rational relations over *unranked* trees. These theories were started in [16].

Towards a generalization of rational relations to trees, Raoult suggests in [18] defining relations over trees by tree grammars in which non-terminals are represented by tuples of letters (called *multivariables*), so that a synchronization between the productions is possible. Raoult calls these relations *rational tree relations* and gives also a characterization in terms of rational expressions.

Complementary Raoult's grammars, the first contribution of this paper are so-called *asynchronous tree automata* which recognize exactly the rational tree relations. With our automata theoretic approach it is possible to address certain properties and (un-) decidability results of rational tree relations.

Rational tree relations in the mentioned format have a few drawbacks. They do not coincide with regular tree languages in the unary case, they are not closed under composition, and if considered as transductions they do not preserve regular tree languages. In [18] Raoult proposes a restriction of his tree grammars to so-called *transduction grammars* which resolve these problems. But these have the disadvantage that, when applied to unary trees, they can only be considered as a generalization of binary rational word relations, but not of the n-ary case. Furthermore, Raoult's restriction is difficult to adapt to tree automata, i. e. it misses a natural automata theoretic characterization. To take account of these problems the second contribution of this paper is such a natural restriction of rational tree relations (which semantically differs from Raoult's one). These so-called *separate-rational tree relations* meet all the properties demanded above and are still more powerful than automatic tree relations [2].

The remainder of this paper is structured as follows. First we fix a few notations in Sect. 2. In Sect. 3 we define rational tree relation introduced by Raoult, develop asynchronous tree automata, and show the equivalence. In Sect. 4 we introduce separate-rational relations and corresponding separate-asynchronous automata. Section 5 contains a conclusion and an outlook on further research.

2 Preliminaries

We assume the reader is familiar with the basics of tree automata [9,4] and with rational relations over words [1,6]. Here, we fix just a few notations and conventions used throughout this paper.

We consider trees and tuple of trees over *ranked alphabets* $\Sigma = \Sigma_0 \uplus \ldots \uplus \Sigma_m$ (where Σ_i contains exactly the symbols of rank i). Often we will state the rank of a symbol in parentheses as superscript. So, $f^{(2)}$ means that the symbol f has

rank 2. A tree t is represented as a pair $(\mathrm{dom}_t, \mathrm{val})$ where dom_t is the set of tree nodes and $\mathrm{val} : \mathrm{dom}_t \to \Sigma$ maps each node of rank k to a symbol in Σ_k. Similarly, a tuple $\bar{t} = (t_1, \ldots, t_n)$ of trees is represented as $(\mathrm{dom}_{\bar{t}}, \mathrm{val})$ where $\mathrm{dom}_{\bar{t}}$ is the disjoint union of the dom_{t_i}. We write trees as terms in the standard way. The *height* of a tree resp. a tuple of trees is defined as the number of nodes of a longest path from a root to a leaf. For example a tree which only consists of the root has a height of 1. With T_Σ we denote the set of all trees over Σ. A *tree language* resp. *tree relation* is a subset of T_Σ resp. $(T_\Sigma)^n$. In Sect. 4 we will also distinguish alphabets for each (projection to one) component of a relation.

3 Rational Tree Relations

In this section we present the theory of rational tree relations starting from Raoult's definition via rational expressions [18]. Then we define asynchronous tree automata, show the equivalence to Raoult's definition, and deal with some closure properties and (un-) decidability results of rational tree relations.

3.1 Definition of Rational Tree Relations Via Rational Expressions

Example 1. Consider the rational expression

$$(cx_1y_1, cbx_2y_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (a, a) \cdot_{x_1x_2} (bz_1, bz_2)^{*_{z_1z_2}} \cdot_{z_1z_2} (a, a)$$

over the ranked alphabet $\Sigma = \{a^{(0)}, b^{(1)}, c^{(2)}\}$. In this example we use "multivariables" x_1x_2, y_1y_2, z_1z_2 (written also as X, Y, Z) which are subject to simultaneous substitution. The form of tuples of the rational tree relation defined by above expression is depicted in Fig. 1. We see that the multivariable $X = x_1x_2$ occurs in distinct instances $x_1x_2, x_1'x_2', \ldots$ which have to be distinguished. Here, the number of possible instances of X cannot be bounded by a natural number. Each instance of the multivariable X becomes substituted with two unary trees of same height (see Fig. 3(a) on page 429 for a full example pair of trees).

Towards the formal definition, let $\mathcal{V}$ be a set of variables. A *multivariable* is a sequence in $\mathcal{V}^+$ containing at most one occurrence of any variable. For

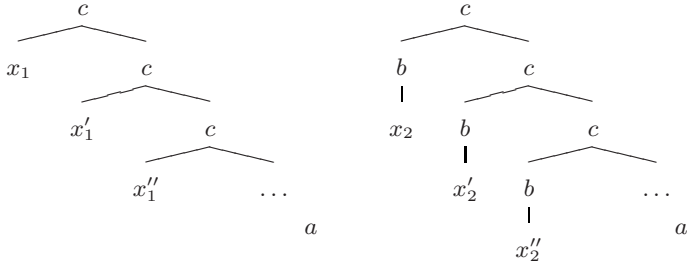

Fig. 1. Generating an unbounded number of instances of a multivariable

$X = x_1 \cdots x_n$ $(n > 0)$ we say that the multivariable X has length $|X| := n$. The set of all *instances* of variables resp. multivariables is the cartesian product $\mathcal{V} \times \mathbb{N}$ resp. $\mathcal{V}^+ \times \mathbb{N}$. We say (x, j) is the j-th instance of variable x, written x^j, and (X, j) is the j-th instance of multivariable X, written X^j. In order to avoid too many indices in the notation, we write instances $x_i^0, x_i^1, x_i^2, \ldots$ of a variable x_i also in the form $x_i, x_i', x_i'', \ldots$.

Instances of variables are nullary symbols which can only occur as leaves. Furthermore, each instance of a multivariable can occur in a tuple of trees at most once, and if an instance of a variable occurs, so all other variables of the same multivariable and same instance: Formally, let $\bar{t} \in T_\Sigma^m$, let $X = x_1 \cdots x_n$ be a multivariable where x_i^j occurs in $\bar{t}$; then each $x_{i'}^j$ occurs in $\bar{t}$ exactly once (and as leaf) for $1 \leq i, i' \leq n$.

Let $X = x_1 \cdots x_n$ be a multivariable of length n, R a relation over n-tuples of trees, S a relation over m-tuples of trees, and $\bar{t}$ a m-tuple of trees containing k instances of X. Then the concatenation of a tuple with a tree relation is defined as $\bar{t} \cdot_X R := \{\bar{t}' \mid \bar{t}' \text{ results from } \bar{t} \text{ by substituting each of the } k \text{ instances of } X \text{ with a tuple from } R\}$. The concatenation of two tree relations is defined as $S \cdot_X R := \{\bar{t} \cdot_X R \mid \bar{t} \in S\}$, and the iterated concatenation and the Kleene star for tree relations are defined as $R^{0X} := \{(x_1^j, \ldots, x_n^j)\}$, $R^{nX} := \{(x_1^j, \ldots, x_n^j)\} \cup R \cdot_X R^{(n-1)X}$, and $R^{*X} := \bigcup_{n \geq 0} R^{nX}$. In the case of the iterated concatenation the instance $j \in \mathbb{N}$ is chosen as a new instance, so that it occurs in the resulting relation only once.

Definition 1 ([18]). *The classes Rat_n of* rational tree relations *are defined inductively as follows:*

- *Each finite n-ary tree relation is in Rat_n.*
- $R \in Rat_n \wedge S \in Rat_n \Rightarrow R \cup S \in Rat_n$.
- $R \in Rat_n \wedge |X| = m \wedge S \in Rat_m \Rightarrow R \cdot_X S \in Rat_n$.
- $R \in Rat_n \wedge |X| = n \Rightarrow R^{*X} \in Rat_n$.

We denote the unary relations in the class Rat_1 as *rational tree languages*. Note that the class Rat_1 does not coincide with the class of regular tree languages:

Example 2. The rational expression $(fx_1x_2) \cdot_{x_1x_2} (gy_1, gy_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (aa)$ over the ranked alphabet $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ describes the tree language $T_{\mathrm{sim}} = \{f(g^n a, g^n a) \mid n \in \mathbb{N}\} \in \mathrm{Rat}_1$, but T_{sim} is not regular.

3.2 Asynchronous Tree Automata

Now we introduce a class of automata recognizing exactly the class Rat_n of rational tree relations. The above considered Examples 1 and 2 show that these automata basically have to provide the following three mechanisms:

- Certain transitions are supposed to be used simultaneously. We will achieve this by combining states to tuples of states which we will call *macro states*. In the runs of our automata all states of a macro state have to be reached and left simultaneously.

We write a finite set of *macro states* as $\mathfrak{Q} = \{\mathfrak{q}_1, \ldots, \mathfrak{q}_k\}$ where $\mathfrak{q}_1, \ldots, \mathfrak{q}_k$ are tuples of states taken from a finite set Q of states (Q contains all states that occur in some $\mathfrak{q} \in \mathfrak{Q}$). A macro state has the form $\mathfrak{q} = (q_1, \ldots, q_l)$ with $l \geq 1$. All macro states in $\mathfrak{Q}$ are "pairwise disjoint", i. e. $\{q_1, \ldots, q_l\} \cap \{p_1, \ldots, p_m\} = \emptyset$ for all macro states $\mathfrak{q} = (q_1, \ldots, q_l)$ and $\mathfrak{p} = (p_1, \ldots, p_m)$ in $\mathfrak{Q}$.

- In addition we require some mechanism to allow asynchronous moves. We will achieve this by the addition of ε-transitions. This enables the automaton to do a bottom-up step in one component and to stay in place in another component (possibly just changing the state).
- An unbounded number of instances of macro states has to be distinguished. In a run we have to distinguish whether states belong to the same or to different instances. We will achieve this by combining each state in a transition with a variable. States with same variables must belong to the same instance when these transitions are used. In a run of our automaton, variables will be instantiated with natural numbers to denote the different instances.

Example 3. Consider macro states $\mathfrak{p} = (p_1, p_2)$ and $\mathfrak{q} = (q_1, q_2)$. Then two transitions $((p_1, x), (p_1, y), (p_2, y), f, (q_1, z))$, $((p_2, x), \varepsilon, (q_2, z))$ enable a bottom-up computation step as depicted in Fig. 2.

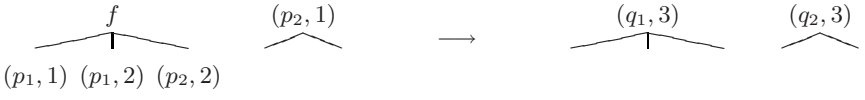

Fig. 2. A computation step of an asynchronous tree automaton

Before we give the formal definition of asynchronous tree automata, we start with a comprehensive example.

Example 4. Consider the rational relation from Example 1. We define an asynchronous tree automaton recognizing this relation. Formally, we will denote our automaton with $\mathcal{A}^{(2)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$ (the superscript indicates that the automaton runs on pairs of trees). $\Sigma = \{a^{(0)}, b^{(1)}, c^{(2)}\}$ is a ranked alphabet. The used macro state set $\mathfrak{Q} = \{(q_{a_1}, q_{a_2}), (q_{b_1}, q_{b_2}), (q_{c_1}, q_{c_2})\}$ consists of pairwise disjoint tuples of states in Q. We declare the macro states of the set $\mathfrak{F} \subseteq \mathfrak{Q}$ as final. In this example we declare only the macro state (q_{c_1}, q_{c_2}) as final. $\mathcal{A}^{(2)}$ has the following transitions in its transition relation Δ which employ variables of the set $\mathrm{Var} = \{x, y, z\}$:

$$
\begin{array}{ll}
(a, (q_{a_1}, x)), & (a, (q_{c_1}, x)), \\
(a, (q_{a_2}, x)), & (a, (q_{c_2}, x)), \\
((q_{a_1}, x), b, (q_{a_1}, x)), & ((q_{a_1}, x), \varepsilon, (q_{b_1}, x)), \\
((q_{a_2}, x), b, (q_{a_2}, x)), & ((q_{a_2}, x), b, (q_{b_2}, x)), \\
((q_{b_1}, x), (q_{c_1}, y), c, (q_{c_1}, z)), & ((q_{b_2}, x), (q_{c_2}, y), c, (q_{c_2}, z)) \ .
\end{array}
$$

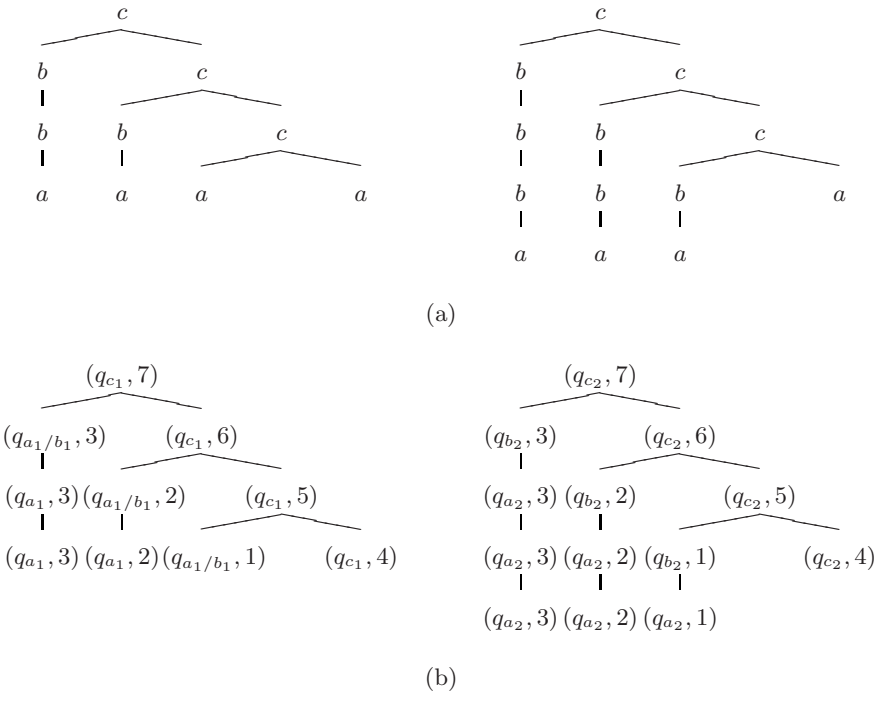

Fig. 3. (a) A pair of trees; (b) an accepting run of $\mathcal{A}^{(2)}$ on this pair of trees

Figure 3 shows a pair of trees and an accepting bottom-up run of $\mathcal{A}^{(2)}$ on this pair. For instance, in a first step of this accepting run the first instantiation of the macro state (q_{a_1}, q_{a_2}) is assigned to a pair of leaves resulting in the labelings $(q_{a_1}, 1)$ and $(q_{a_1}, 2)$. In a second step this macro state changes to (q_{b_1}, q_{b_2}) by application of the transitions $((q_{a_1}, x), \varepsilon, (q_{b_1}, x))$ and $((q_{a_2}, x), b, (q_{b_2}, x))$. Note that the numbering of instances is rather arbitrary as long as different instances of variables can be distinguished. Due to lack of space we illustrate all intermediate configurations of the run in one tree. If a node is part of two different cuts in the run (due to the use of ε-transitions), we label this node with both configurations in an abbreviated form, e. g. for a node v and two configurations $c_1(v) = (q_{a_1}, 3)$ and $c_2(v) = (q_{b_1}, 3)$ we label v with $(q_{a_1/b_1}, 3)$.

Now we give a formal definition of asynchronous tree automata.

Definition 2. *An asynchronous tree automaton over a ranked alphabet* $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$ *is a tuple* $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, \mathit{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$ *with*

- *a finite set* Q *of states,*
- *a set* $\mathfrak{Q}$ *of macro states over* Q *(i. e. pairwise disjoint tuples of states in* Q*),*
- *a finite set* Var *of variables,*

– *a transition relation*

$$\Delta \subseteq \bigcup_{i=0}^{m} \big((Q \times Var)^i \times \Sigma_i \times Q \times Var\big) \ \cup \ (Q \times Var \times \{\varepsilon\} \times Q \times Var) \ ,$$

– *and a set* $\mathfrak{F} \subseteq \mathfrak{Q}$ *of final macro states.*

An instantiation *of a set* $\mathcal{V} \subseteq Var$ *of variables is an injective function* $I_{\mathcal{V}} : \mathcal{V} \to \mathbb{N}, x \mapsto \alpha$. *We also refer to* $\alpha \in \mathbb{N}$ *as the* instance α. *A* cut C *of an* n*-tuple* $(t_1, \ldots, t_n)$ *of trees is an antichain in* $dom_{(t_1,\ldots,t_n)}$ *(consisting of pairwise incomparable nodes w. r. t. the prefix ordering). The computation shifts the cut stepwise upwards until it reaches the antichain of the root nodes of* $t_1, \ldots, t_n$ *(if possible). A* configuration *is a mapping* $c : C \to Q \times \mathbb{N}$ *which associates an instantiated state to each node of* C *through the tuple* $(t_1, \ldots, t_n)$. *We require that the instances of states are the same within each macro state of a configuration; also different occurrences of a state in a configuration appear with different instances (formally* $c(v_1) \neq c(v_2)$ *for all* $v_1 \neq v_2$*).*

A makes a computation step $c_1 \to c_2$ *between two configurations* $c_1 : C_1 \to Q \times \mathbb{N}$ *and* $c_2 : C_2 \to Q \times \mathbb{N}$ *where* C_2 *contains the parents of* C_1*-nodes reached via a proper transition, those* C_1*-nodes which are only subject to state changes by* ε*-transitions, and those* C_1*-nodes which are not affected by any transitions in this step and hence stay unchanged. More precisely, we require that there exist nodes* $v_1, \ldots v_k$ *with children* $v_{1,1}, \ldots, v_{1,l}$ *of* v_1*, children* $v_{2,1}, \ldots, v_{2,l}$ *of* v_2*, ... and children* $v_{k,1}, \ldots, v_{k,l}$ *of* v_k *as well as nodes* $v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}$*, so that the following conditions are fulfilled:*

1. $\{v_{1,1}, \ldots, v_{k,l}, v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}\} \subseteq C_1$.
2. $C_2 = (C_1 \setminus \{v_{1,1}, \ldots, v_{k,l}\}) \cup \{v_1, \ldots, v_k\}$.
3. *There exist proper transitions*

   $$((q_{1,1}, x_{1,1}), \ldots, (q_{1,l}, x_{1,l}), val(v_1), (q_1, x)), \ \ldots \ , ((q_{k,1}, x_{k,1}), \ldots, (q_{k,l}, x_{k,l}), val(v_k), (q_k, x))$$

   and ε*-transitions*

   $$((q_{\varepsilon_1}, x_{\varepsilon_1}), \varepsilon, (q_{\varepsilon'_1}, x)), \ \ldots \ , ((q_{\varepsilon_j}, x_{\varepsilon_j}), \varepsilon, (q_{\varepsilon'_j}, x))$$

   in Δ*, so that*
   - $q_1, \ldots, q_l, q_{\varepsilon'_1}, \ldots, q_{\varepsilon'_j}$ *form exactly one macro state,*
   - $q_{1,1}, \ldots, q_{k,l}, q_{\varepsilon_1}, \ldots, q_{\varepsilon_j}$ *form a union of certain macro states, and all states belonging to the same macro state occur with the same variable,*
   - *there exist an instantiation* $I_{\mathcal{V}}$ *of a variable set* $\mathcal{V} \subseteq Var$*, so that these transitions with each variable* $x \in \mathcal{V}$ *replaced by* $I_{\mathcal{V}}(x)$ *match exactly the computation step* $c_1 \to c_2$.
4. c_2 *is identical to* c_1 *on* $(C_1 \cap C_2) \setminus \{v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}\}$.

The configuration $c : C \to Q \times \mathbb{N}$ *with* $C = \emptyset$ *is called* start configuration. *A configuration* $c : C \to Q \times \mathbb{N}$ *is* accepting *iff* $C = \{root_1, \ldots, root_n\}$ *with roots* $root_i$ *of* t_i *(*$1 \leq i \leq n$*), and there exist a final macro state* $(q_1 \ldots q_n) \in \mathfrak{F}$ *and an* $\alpha \in \mathbb{N}$*, so that* $c(root_1) = (q_1, \alpha), \ldots, c(root_n) = (q_n, \alpha)$. *A sequence of*

configurations is a run *iff* $c_1 \to \ldots \to c_m$ *and* c_1 *is the start configuration. Such a run is called* accepting *iff* c_m *is accepting.* $\mathcal{A}^{(n)}$ *recognizes the* n*-ary relation*

$$R(\mathcal{A}^{(n)}) = \{(t_1, \ldots, t_n) \mid \textit{there exists an accepting run of } \mathcal{A}^{(n)} \textit{ on } (t_1, \ldots, t_n)\} \ .$$

3.3 The Equivalence Theorem

The equivalence theorem is an adaption of the Kleene-Theorem for tree languages (see [9,4]). (For the detailed proof we refer to [15].)

Theorem 1. *A relation* R *of* n*-tuples of trees is rational if and only if there exists an asynchronous tree automaton* $\mathcal{A}^{(n)}$ *with* $R(\mathcal{A}^{(n)}) = R$.

Proof (Sketch). The $\Rightarrow$-direction of the proof goes by induction over rational expressions. For the induction start the construction of an asynchronous tree automaton for a singleton of a tuple of trees suffices. Here it is important to prepare the induction step by reading each instance of a multivariables at the leaves simultaneously. For the induction step asynchronous tree automata for the operations $\cup$, $\cdot_X$ and $*_X$ according to Definition 1 are easy to construct.

For the $\Leftarrow$-direction it can be shown for each asynchronous tree automaton $\mathcal{A}^{(n)}$ that its recognized relation is rational. The result can be shown by an induction over the set of "intermediate macro states" $\mathfrak{S}$ of the runs of $\mathcal{A}^{(n)}$. As intermediate macro states we count macro states which occur in other configurations than start configurations at the leaves or an end configuration at the root. For the induction start ($\mathfrak{S} = \emptyset$) we have to consider trees accepted by $\mathcal{A}^{(n)}$ without intermediate macro states. These are n-tuples of trees of height 1 or 2 only. Since these are only finitely many, they form a rational relation. For the induction step ($|\mathfrak{S}| > 0$) it suffices to give a rational expression which composes relations with $|\mathfrak{S}| - 1$ intermediate macro states to a relations with $|\mathfrak{S}|$ intermediate macro states and which is accepted by $\mathcal{A}^{(n)}$. □

3.4 Properties of Rational Tree Relations

Now we present some closure properties and (un-) decidability results, also recalling some "defects" of the rational tree relations which were noted in [18].

For a word relation R we define a tree relation $\mathrm{TRel}(R)$ by interpreting each word $u = a_1 a_2 \cdots a_n$ of a tuple of R as an unary tree $u\$ = a_1(a_2(\ldots(a_n(\$))\ldots))$. For an n-ary word relation $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$ the tree relation $\mathrm{TRel}(R)$ over $\Sigma_1 \cup \{\$^{(0)}\}, \ldots, \Sigma_n \cup \{\$^{(0)}\}$ is defined as $\mathrm{TRel}(R) := \{(u_1\$, \ldots, u_n\$) \mid (u_1, \ldots, u_n) \in R\}$. The following results are easy to prove by construction of corresponding automata for each direction:

Lemma 1. *Let* R *be a word relation. Then* R *is rational iff* $TRel(R)$ *is rational.*

Due to Lemma 1 some elementary closure properties and all undecidability results of rational word relations can be extended to trees easily:

Proposition 1. *(a) The class* Rat_n *of* n*-ary rational tree relations is closed under union, not closed under intersection, and not closed under complementation.*

(b) For rational tree relations $R_1, R_2 \in Rat_n$ it is undecidable to determine whether $R_1 \cap R_2 = \emptyset$, $R_1 \subseteq R_2$, and $R_1 = R_2$.

The *membership problem* for asynchronous tree automata is decidable, i. e. it is decidable whether $(t_1, \ldots, t_n) \in R(\mathcal{A})$. Also the *emptiness problem*, i. e. the question whether $R(\mathcal{A}) = \emptyset$, and the *infinity problem*, i. e. the question whether $|R(\mathcal{A})|$ is infinite, are decidable. (The proofs can be found in [15].)

Theorem 2. *Given an asynchronous tree automata with macro state set $\mathfrak{Q}$ and transition relation Δ, and a tuple of trees with m nodes. The membership problem is decidable in $O(|\Delta|^m)$ time, and the emptiness and the infinity problem are decidable in $O(|\mathfrak{Q}|^2 \cdot |\Delta|)$ time.*

Unlike binary rational relations over words, the class Rat_2 of binary rational tree relations is not closed under composition:

Example 5. The binary tree relations $R_1 = \{(b^m a^n \$, f(a^n \$, b^m \$)) \mid m, n \in \mathbb{N}\}$ and $R_2 = \{(f(a^n \$, b^m \$), a^n b^m \$) \mid m, n \in \mathbb{N}\}$ are rational, but the composition $\{(b^m a^n \$, a^n b^m \$) \mid m, n \in \mathbb{N}\}$ is *not* rational.

Binary rational relations over words are also called *(rational) transductions*. They preserve regular and context-free languages, i. e. the image and the inverse image of a regular (resp. a context-free) language under a transduction is again a regular (resp. context-free) language [1]. Here we note that binary rational tree relations do not even preserve regularity:

Example 6 ([18]). Consider the rational tree relation $T_{\mathrm{sim}} = \{f(g^n a, g^n a) \mid n \in \mathbb{N}\}$ from Example 2. Clearly, $R := \Sigma^* \times T_{\mathrm{sim}}$ is rational. The image of a regular language under R is T_{sim} which is not regular. An analogous result for the inverse image can be proved with a relation $R' := T_{\mathrm{sim}} \times \Sigma^*$.

4 Separate-Rational Tree Relations

We have seen a few drawbacks of rational tree relations. They do not coincide with regular tree languages in the unary case, are not closed under composition, and do not preserve regular tree languages. In [18] Raoult proposes a restriction of rational tree relations, generated by so-called *transduction grammars*. These reestablish the demanded properties, but as mentioned in the introduction they have other drawbacks: They are not a proper generalization of rational word relations in the n-ary case, and the restriction is difficult to adapt to asynchronous tree automata. So, we define yet another restriction, both for rational expressions and asynchronous tree automata, resolving these issues.

The idea is to define a class of relations which can be computed by asynchronous tree automata which have all their macro states separated between the components, i. e. each state of a macro state can only occur in one component.

Definition 3. *The classes $SepRat_n$ of* separate-rational tree relations *are defined inductively as follows:*

- $\emptyset \in SepRat_n$.
- $\{(t_1, \ldots, t_n)\} \in SepRat_n$, *where* $t_1, \ldots, t_n$ *are only trees of height* 1 *or* 2 *and each component contains at most one variable of each multivariable.*
- $R \in SepRat_n \wedge S \in SepRat_n \Rightarrow R \cup S \in SepRat_n$.
- $R \in SepRat_n \wedge |X| = m \wedge S \in SepRat_m \Rightarrow R \cdot_X S \in SepRat_n$, $m \leq n$, *where each component of a tuple in* R *contains at most one variable of* X.
- $R \in SepRat_n \wedge |X| = n \Rightarrow R^{*_X} \in SepRat_n$, *where each component of a tuple in* R *contains exactly one variable of* X.

Example 7. (a) The relation from Example 1 is separate-rational. The rational expression can be rewritten as $(cx_1y_1, cx_2y_2)^{*_{y_1y_2}} \cdot_{y_1y_2} (a,a) \cdot_{x_1x_2} (x_1, bx_2) \cdot_{x_1x_2} (bz_1, bz_2)^{*_{z_1z_2}} \cdot_{z_1z_2} (a,a)$. It is generated by trees of height 2 at most, and all multivariables are separated between the components of the tuples.

(b) The rational relations R_1 and R_2 from Example 5 are *not* separate-rational, because multivariables of length 3 are are easily seen to be necessary in order to define these relations. So, at least two variables of one multivariable have to occur in the same component of a tuple.

We will restrict asynchronous tree automata, so that these recognize exactly the class of separate-rational relations. For the separate-asynchronous case we allow the automata to utilize a specific ranked alphabet for each component.

Definition 4. *A separate-asynchronous tree automaton* $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, Var, \Sigma_1, \ldots, \Sigma_n, \Delta, \mathfrak{F} \rangle$ *is an asynchronous tree automaton over* $\Sigma_1 \cup \ldots \cup \Sigma_n$ *(each* $\Sigma_j = \Sigma_{0_j} \cup \ldots \cup \Sigma_{m_j}$ *is a ranked alphabets) with the following restrictions:*

- *the set* Q *of states is partitioned in* $Q = Q_1 \uplus \ldots \uplus Q_n$,
- *for each macro state* $(q_1, \ldots, q_m) \in \mathfrak{Q}$ *and all* $q_k \neq q_l$, $1 \leq k, l \leq m$, $1 \leq j \leq n$ *holds:* $q_k \in Q_j \Rightarrow q_l \notin Q_j$,
- *the transition relation is partitioned in* $\Delta = \Delta_1 \uplus \ldots \uplus \Delta_n$ *with*

$$\Delta_j \subseteq \cup_{i=0}^{m} \big((Q_j \times Var)^i \times \Sigma_{i_j} \times Q_j \times Var \big) \cup (Q_j \times Var \times \{\varepsilon\} \times Q_j \times Var) \ ,$$

- *each final macro state* $\mathfrak{q} \in \mathfrak{F}$ *has the form* $\mathfrak{q} = (q_1, \ldots, q_n)$ *with* $q_i \in Q_i$ *for all* $1 \leq i \leq n$.

The Equivalence Theorem (Theorem 1) can be reformulated for separate-rational relations. Only slight modifications are necessary. It should be mentioned that the restriction to elementary trees of height 1 or 2 in Definition 4 is important for the "$\Rightarrow$"-direction of the proof in order to handle the induction start. Also, this condition is not a restriction for the "$\Leftarrow$"-direction, because in the original proof the induction start only results in trees of height 1 or 2.

Theorem 3. *A relation* R *of* n*-tuples of trees is separate-rational if and only if there exists a separate-asynchronous tree automaton* $\mathcal{A}^{(n)}$ *with* $R(\mathcal{A}^{(n)}) = R$.

Lemma 1 can be reformulated for separate-rational relations. So, we obtain the same undecidability results and closure properties which we derived for rational tree relations from Lemma 1. Beyond this, separate-rational relations resolve the issues raised in Sect. 3.

Theorem 4. *(a) The class $SepRat_1$ of separate-rational tree languages is the class of regular tree languages.*
(b) The class $SepRat_2$ of binary separate-rational tree relations is closed under composition.
(c) The image and the inverse image of a regular tree language under a binary separate-rational tree relation R are again regular tree languages.

Proof. (a) For $n = 1$ all multivariables have length 1 resp. all macro states have size 1, yielding regular tree languages.

(b) Construct a separate-asynchronous automaton recognizing $R \circledcirc S := \{(t, t', t'') \mid (t, t') \in R, (t', t'') \in S\}$ for separate-rational tree relations R and S by synchronization of the common component. The projection on the first and third component yields a separate-asynchronous automaton for $R \circ S$. (We refer to [15] for the detailed proof.)

(c) Due to symmetry of Definition 4, it suffices to show that the image of a regular tree language under a binary separate-rational relation is regular. Clearly, the identity $\mathrm{id}_T = \{(t, t) \mid t \in T\}$ of a regular tree language T is separate-rational. Thus, the image of T under a separate-rational relation R is the projection on the second component of $\mathrm{id}_T \circ R$. Due to Theorem 4(b) $\mathrm{id}_T \circ R$ is also a separate-rational. The projection on the second component yields a regular tree language (due to the closure of SepRat under projections [15] and Theorem 4(a)). □

If we consider rational relations over words, they also preserve context-free languages [1]. It is an open question whether separate-rational tree relations also preserve context-free tree languages as defined in [10].

5 Conclusion

We presented an automata theoretic approach to rational tree relations which now can be described by three equivalent formalisms: Rational expressions, tree grammars [18], and asynchronous tree automata. Separate-rational tree relations overcome some drawbacks of the rational tree relations. This restriction is natural, since it is easy to apply to all three formalisms (tree grammars were not discussed here, but can be restricted like rational expressions). Separate-rational tree relations are a proper generalization of rational word relations and are still more powerful than, for instance, automatic tree relations.

Outlook: Rational tree relations are more powerful than *linear tree transducers* (as defined in [4]) and some cases of *term rewriting systems* [12]. These results do not hold for the separate-rational restriction. More expressive extensions of separate-rational relations with such features need to be investigated.

Asynchronous tree automata allow the definition of rational relations over unranked trees and the definition of deterministic rational tree relations (both over ranked and unranked trees). For the deterministic top-down model see [5,16]. A deterministic bottom-up model seems to be more challenging (due to the non-deterministic grouping of nodes in a run for the instantiation with macro

states). A further restriction of separate-rational automata may yield a model which generalizes deterministic rational word relations on the one hand and includes recognizable and automatic tree relations on the other hand.

Acknowledgements. This work contains some results of my diploma thesis [16]. Special thanks go to Wolfgang Thomas for supervising this work and for his numerous helpful suggestions.

References

1. Berstel, J.: Transductions and Context-Free Languages. Leitfäden der angewandten Mathematik und Mechanik 38. Teubner, Stuttgart (1979)
2. Blumensath, A., Grädel, E.: Finite presentations of infinite structures: Automata and interpretations. Theory of Computing Systems 37, 641–674 (2004)
3. Carton, O., Choffrut, C., Grigorieff, S.: Decision problems among the main subfamilies of rational relations. Theor. Informat. Appl. 40(2), 255–275 (2006)
4. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. Unpublished electronic book (1997), `http://www.grappa.univ-lille3.fr/tata`
5. Cristau, J., Löding, C., Thomas, W.: Deterministic automata on unranked trees. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 68–79. Springer, Heidelberg (2005)
6. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic Press, New York (1974)
7. Elgot, C.C., Mezei, J.E.: On relations defined by generalized finite automata. IBM Journal of Research and Development 9(1), 47–68 (1965)
8. Fischer, P.C., Rosenberg, A.L.: Multitape one-way nonwriting automata. Journal of Computer and System Sciences 2(1), 88–101 (1968)
9. Gécseg, F., Steinby, M.: Tree Automata, Akadémiai Kiadó, Budapest (1984)
10. Gécseg, F., Steinby, M.: Tree Languages. In: Handbook of Formal Languages, Beyond Words, vol. 3, pp. 1–68. Springer, Heidelberg (1997)
11. Grigorieff, S.: Modelization of deterministic rational relations. Theoretical Computer Science 281(1-2), 423–453 (2002)
12. Meyer, A.: On term rewriting systems having a rational derivation. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 378–392. Springer, Heidelberg (2004)
13. Pelletier, M., Sakarovitch, J.: On the representation of finite deterministic 2-tape automata. Theoretical Computer Science 225(1-2), 1–63 (1999)
14. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development 3(2), 115–125 (1959)
15. Radmacher, F.G.: An automata theoretic approach to the theory of rational tree relations. Tech. Rep. (2007), `http://www.automata.rwth-aachen.de/~radmacher/`
16. Radmacher, F.G.: Automatendefinierbare Relationen über Bäumen (Automata Definable Relations over Trees). Diploma thesis (revised version), RWTH Aachen (2007), `http://www.automata.rwth-aachen.de/~radmacher/`
17. Raoult, J.-C.: A survey of tree transductions. In: Nivat, M., Podelski, A. (eds.) Tree Automata and Languages, pp. 311–326. Elsevier, Amsterdam (1992) (also published as report 1410 INRIA-Rennes, 1991)
18. Raoult, J.-C.: Rational tree relations. Bulletin of the Belgian Mathematical Society 4(1), 149–176 (1997)