Department of Computer Science
Technical Report

An Automata Theoretic Approach to the Theory of Rational Tree Relations

Frank G. Radmacher

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2008-05

RWTH Aachen · Department of Computer Science · March 2008

An Automata Theoretic Approach to the Theory of Rational Tree Relations*

Frank G. Radmacher

Lehrstuhl für Informatik 7, RWTH Aachen University, 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 [CCG06]. Proper generalizations of these theories to trees have been established over the past years in the case of recognizable relations and automatic relations [CDG$^+$07, Blu99, BG04]. However, it is still debatable how to obtain a reasonable generalization of rational word relations to trees.

The theory of rational relations over words is a well-developed theory [Ber79, Eil74]. 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*). They were introduced in their weaker deterministic variant by Rabin and Scott in the late fifties [RS59]. Nowadays the non-deterministic model introduced by Elgot and Mezei in [EM65] which exactly recognize the rational relations is considered as the right generalization. Elgot and Mezei showed that rational relations includes many other known binary relations. They also studied the closure properties of rational relations. Fischer and Rosenberg discovered most of the undecidability results [FR68].

Generalizing rational relations to trees (resp. terms) is not straightforward. A survey focussing on binary relations (transductions) was given by Raoult in [Rao92]. Attractive results on rational word relations which one would also like for rational tree relations are the following:

* This article is an extended version of [Rad08].

– 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).
– 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.

An additional motivation for our automata theoretic approach is the possibility of defining other classes of relations over trees which have their natural definition in automata theory. Examples for such languages and relations are regular languages over unranked trees which are defined via unranked tree automata [BMW01, CDG$^+$07], and deterministic rational relations which are defined via deterministic asynchronous automata [RS59, PS99, Gri02]. The theories of rational relations over *unranked* trees and *deterministic* rational tree relations were started in [Rad07].

Towards a generalization of rational relations to trees, Raoult suggests in [Rao97] 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 [Rao97] 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 in our framework. 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 [CDG$^+$07, BG04].

1.1 Related Works

As mentioned above, deterministic rational tree relations and rational tree relations over unranked trees were addressed in [Rad07]. A deterministic top-down approach was applied to recognizable, automatic, and rational tree relations which results in a whole hierarchy of automata definable tree relations.

An application of rational relations is the classification of word-rewriting systems. Caucal investigated left,- right, suffix- and, prefix-systems and showed these have a rational derivation [Cau00]. Meyer adapted this classification to term-rewriting systems using Raoult's rational tree relations. He examined bottom-up-, top-down-, suffix- and prefix-systems and showed the former three have a rational derivation, but some prefix-systems have not [Mey04, Mey05]. With respect to Raoult's restriction of transduction grammars Meyer notes that even some top-down-systems do not satisfy this restriction.

Morvan investigated the class of *rational graphs* over words whose nodes are represented by words and whose edge relation is recognizable by a rational word relation [Mor00]. The model-checking problem on those graphs is undecidable for first order logic. However, decidability of the model-checking problem for modal logic is easy to obtain. Since rational word relations preserve regular languages, a set of nodes which satisfies a given formula (with regular sets of nodes as propositions) have a finite representation (cf. [Rad07]).

1.2 Structure of the paper

This 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 use asynchronous tree automata to investigate the properties of rational tree relations. We will see to what extent they are a good generalizations of the word case. In Sect. 5 we introduce separate-rational relations and corresponding separate-asynchronous automata. A more detailed discussion on the restrictions of separate-rational relations and Raoult's transduction grammars is given at the end of Sect. 5. Section 6 contains a conclusion and an outlook on further research. There, we will give some ideas for defining rational relations over unranked trees and deterministic rational tree relations.

Acknowledgements. This work contains some results of my diploma thesis [Rad07]. Special thanks go to Wolfgang Thomas for supervising this work and for his numerous helpful suggestions.

2 Preliminaries

We assume the reader is familiar with the basics of tree automata [GS84, CDG$^+$07] and with rational relations over words [Ber79, Eil74]. 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. 5 we will also use a distinguished alphabet for each projection to one component of a relation. An n-ary *recognizable relation* is a finite union of products $T_1 \times \ldots \times T_n$ of regular tree languages $T_1, \ldots, T_n$.

3 Rational Tree Relations

In this section we present the theory of rational tree relations starting from Raoult's definition via rational expressions [Rao97]. Then we define asynchronous tree automata and show the equivalence to Raoult's definition.

3.1 Definition of Rational Tree Relations via Rational Expressions

Rational relations over words are defined inductively as the sets which are closed under union, componentwise concatenation, and Kleene star. This yields a description of rational relations by rational expressions. In order to define the concatenation for tuple of trees, Raoult has the idea of using multivariables. These enable a simultaneous substitution on distinguished leaves. This approach is a natural generalization of the concatenation of tree languages in which single leaves are distinguished as variables for substitution.

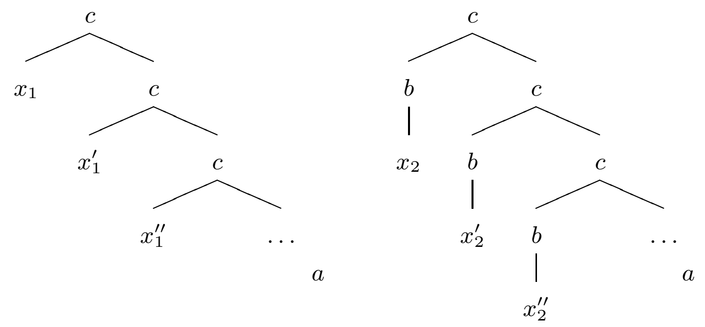

Fig. 1. Generating an unbounded number of instances of a multivariable.

Example 3.1. 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 $x_1x_2, x'_1x'_2, \ldots$ 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 7 for a full example pair of trees).

Alternatively, rational tree relations can also be defined via tree grammars [Rao97]. The relation defined by this rational expression can also be generated by the tree grammar with multivariables $S = S_1S_2$ and $A = A_1A_2$ as non-terminals (S is the start symbol), and following production:

$$\begin{aligned} S_1S_2 &\rightarrow c(A_1, S_1)\, c(b(A_2), S_2) \;\mid\; a\,a \\ A_1A_2 &\rightarrow b(A_1)\, b(A_2) \;\mid\; a\,a \;. \end{aligned}$$

Towards the formal definition of rational tree relations, 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 $X = x_1 \cdots x_n$ $(n > 0)$ 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 $\bar{t} \in T_\Sigma^m$ be a tuple of trees, 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

$$\begin{aligned} \bar{t} \cdot_X R := \{\bar{t}' \mid\ & \bar{t}' \text{ results from } \bar{t} \text{ by substituting each of the } k \text{ instances of } X \\ & \text{with a tuple from } R\} \;, \end{aligned}$$

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 $R^{0_X} := \{(x_1^j, \ldots, x_n^j)\}$, $R^{n_X} := \{(x_1^j, \ldots, x_n^j)\} \cup R \cdot_X R^{(n-1)_X}$, and

$$R^{*_X} := \bigcup_{n \geq 0} R^{n_X} \;.$$

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 3.2 ([Rao97]). The classes Rat_n of *rational tree relations* are defined inductively as follows:

- Each finite n-ary tree relation is in Rat_n.
- $R \in \mathrm{Rat}_n \wedge S \in \mathrm{Rat}_n \;\Rightarrow\; R \cup S \in \mathrm{Rat}_n$.
- $R \in \mathrm{Rat}_n \wedge |X| = m \wedge S \in \mathrm{Rat}_m \;\Rightarrow\; R \cdot_X S \in \mathrm{Rat}_n$.
- $R \in \mathrm{Rat}_n \wedge |X| = n \;\Rightarrow\; R^{*_X} \in \mathrm{Rat}_n$.

3.2 Rational Tree Languages

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 3.3. 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.

So, we obtain a difference to rational relations over words which coincide to regular languages when restricted to unary sets. This fact already motivates restrictions of rational tree relations that will be discussed in Sect. 5.

3.3 Asynchronous Tree Automata

Now we introduce a class of automata recognizing exactly the class Rat_n of rational tree relations. The above considered Examples 3.1 and 3.3 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 3.4. 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.

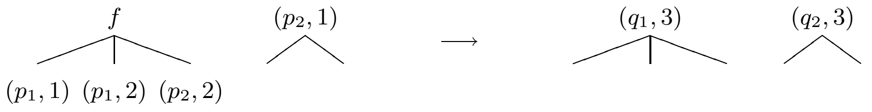

Fig. 2. A computation step of an asynchronous tree automaton.

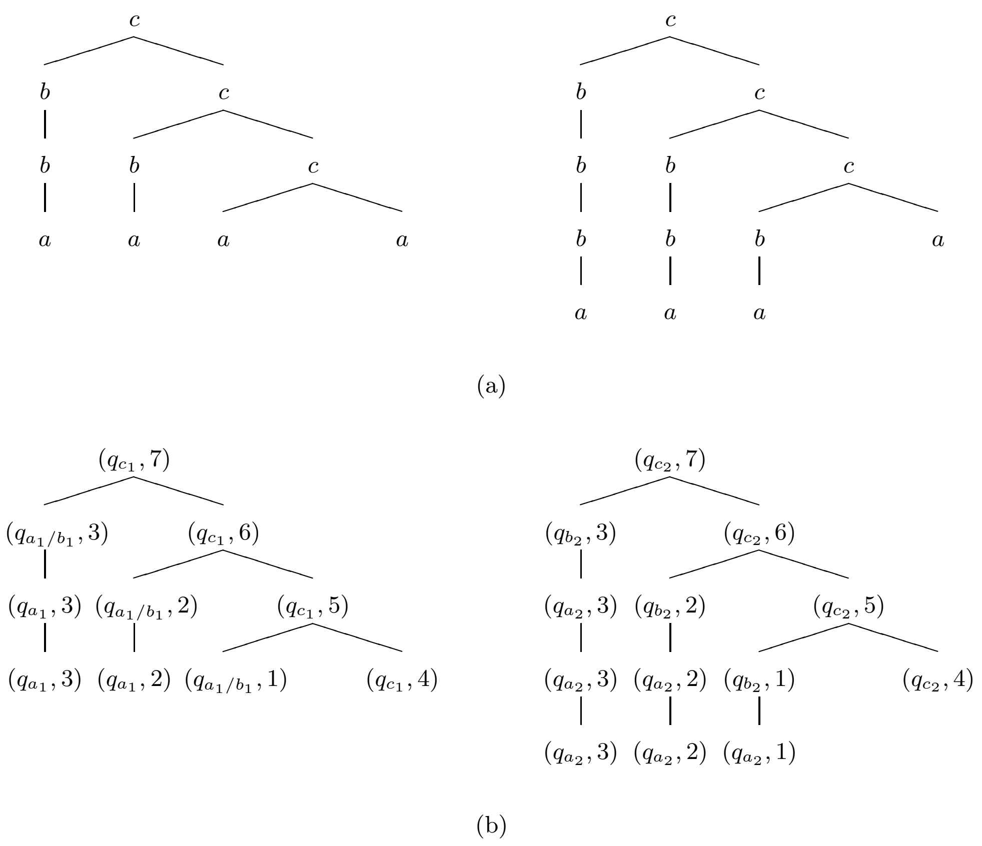

Fig. 3. (a) A pair of trees; (b) an accepting run of $\mathcal{A}^{(2)}$ on this pair of trees.

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

Example 3.5. Consider the rational relation from Example 3.1. 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, i. e. the automaton accepts a pair of trees iff there exists a run of $\mathcal{A}^{(2)}$ which reaches q_{c_1} at the first and q_{c_2} at the second root node. $\mathcal{A}^{(2)}$ has the following transitions in its transition relation Δ which employ variables of the set $\mathrm{Var} = \{x, y, z\}$:

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

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 ε-transition $((q_{a_1}, x), \varepsilon, (q_{b_1}, x))$ in the first component and the proper transition $((q_{a_2}, x), b, (q_{b_2}, x))$ in the second component. 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 $(q_{a_1/b_1}, 3)$.

The definition of asynchronous tree automata follows. Informally, the description of the computation steps of the bottom-up run formalize the of following:

– Same states in the same configuration have to be distinguishable, i. e. they have to be instantiated with different natural numbers.
– Only entire instances of macro states occur.
– In each computation step exactly one instance of a macro state is reached.

Definition 3.6. An asynchronous tree automaton over a ranked alphabet $\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_m$ is a tuple $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$ 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

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

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

An *instantiation* of a set $\mathcal{V} \subseteq \mathrm{Var}$ 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 $\mathrm{dom}_{(t_1,\ldots,t_n)}$ (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 $v_1 \neq v_2$).

$\mathcal{A}$ 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 $v_{11}, \ldots, v_{1l}$ of v_1, children $v_{21}, \ldots, v_{2l}$ of v_2, … and children $v_{k1}, \ldots, v_{kl}$ of v_k as well as nodes $v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}$, so that the following conditions are fulfilled:

1. $\{v_{11}, \ldots, v_{kl}, v_{\varepsilon_1}, \ldots, v_{\varepsilon_j}\} \subseteq C_1$.
2. $C_2 = (C_1 \setminus \{v_{11}, \ldots, v_{kl}\}) \cup \{v_1, \ldots, v_k\}$.
3. There exist proper transitions

$$((q_{11}, x_{11}), \ldots, (q_{1l}, x_{1l}), \mathrm{val}(v_1), (q_1, x)), \ldots, ((q_{k1}, x_{k1}), \ldots, (q_{kl}, x_{kl}), \mathrm{val}(v_k), (q_k, x))$$

   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,
   – $q_{11}, \ldots, q_{kl}, q_{\varepsilon_1}, \ldots, q_{\varepsilon_j}$ 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 $\mathcal{V} \subseteq \mathrm{Var}$, so that these transitions with each variable $x \in \mathcal{V}$ replaced by $I_{\mathcal{V}}(x)$ match both configurations c_1 and c_2 of 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 $C = \{\mathrm{root}_1, \ldots, \mathrm{root}_n\}$ with roots root_i of t_i $(1 \leq i \leq n)$, and there exist a final macro state $(q_1, \ldots, q_n) \in \mathfrak{F}$, so that $c(\mathrm{root}_1) = (q_1, \alpha), \ldots, c(\mathrm{root}_n) = (q_n, \alpha)$ for an $\alpha \in \mathbb{N}$. 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

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

3.4 The Equivalence Theorem

The equivalence theorem is an adaption of the Kleene-Theorem for tree languages (see [GS84, CDG+07]). (For the detailed proof we refer to Appendix A.)

Theorem 3.7. *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 3.2 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. $\square$

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 [Rao97]. So we will see to what extent they are a good generalization of rational relations over words.

4.1 The Interrelationship to Rational Word Relations

We start with the interrelationship between rational relations over words and rational relations over trees. For a word relation R we define a tree relation $\mathrm{TreeRel}(R)$ and a tree language $\mathrm{TreeLang}(R)$. These are two possible views to see a word relation as tree relation. In $\mathrm{TreeRel}(R)$ each word $u = a_1 a_2 \cdots a_n$ of a tuple of R is seen as a unary tree $u\$ = a_1(a_2(\ldots(a_n(\$))\ldots))$. In $\mathrm{TreeLang}(R)$ an n-tuple $(u_1, \ldots, u_n)$ of words is seen as a single tree $f(u_1\$, \ldots, u_n\$)$ with a new n-ary symbol f. This not so common view allows a closer characterization of the class Rat_1 of relation tree languages.

Definition 4.1. For an n-ary word relation $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$ the tree language $\mathrm{TreeLang}(R)$ over $\Sigma_1 \cup \ldots \cup \Sigma_n \uplus \{f^{(n)}, \$^{(0)}\}$ is defined as

$$\mathrm{TreeLang}(R) = \{f(u_1\$, \ldots, u_n\$) \mid (u_1, \ldots, u_n) \in R\}$$

and the tree relation $\mathrm{TreeRel}(R)$ over $\Sigma_1 \uplus \{\$^{(0)}\}, \ldots, \Sigma_n \uplus \{\$^{(0)}\}$ is defined as

$$\mathrm{TreeRel}(R) = \{(u_1\$, \ldots, u_n\$) \mid (u_1, \ldots, u_n) \in R\} \ .$$

The following results are easy to prove by construction of corresponding automata for each direction. (The construction for part (d) can be found in Appendix C; part (b) can be found in [Rad07].)

Lemma 4.2. *Let R be a word relation. The following equivalences hold:*

(a) R is recognizable iff TreeRel(R) recognizable.
(b) R is recognizable iff TreeLang(R) recognizable.
(c) R is rational iff TreeRel(R) is rational.
(d) R is rational iff TreeLang(R) is rational.

4.2 Closure Properties

Some elementary closure properties of rational word relations can be extended to trees easily.

Proposition 4.3. *The class Rat_n of n-ary rational tree relations is closed under union, not closed under intersection, and not closed under complementation.*

Proof. Closure under union holds per Definition 3.2. The latter follows from Lemma 4.2(c), because the class of rational relations over words are closed neither under intersection nor under complementation. □

Rational tree relations are closed under intersection with recognizable tree relations. For the proof construct the product automaton of the automata recognizing R and S which is an asynchronous tree automaton for $R \cap S$.

Proposition 4.4. *Given an n-ary rational tree relation R and an n-ary recognizable tree relation S. Then $R \cap S$ is rational.*

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

Example 4.5. 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.

4.3 Decision Problems

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:

Theorem 4.6. *Given an asynchronous tree automaton 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.*

Proof (Sketch). For the membership problem consider a tuple of trees $(t_1, \ldots, t_n)$ and an asynchronous tree automaton $\mathcal{A}$. We make use of the fact that there are only finitely many possibilities of constructing a run for a given tuple of trees (except for different naming of instances). By constructing a run in a top-down manner, we get a time complexity of $O(|\Delta|^m)$ for a tuple of trees with m nodes overall. The length of a run can be bounded by m and in each step we have the choice of Δ transitions at most.

The emptiness problem can be solved by a reachability test. The idea is to compute successively for a given $\mathcal{A}$ the set $E_{\mathcal{A}}$ of all reachable macro states. Then $R(\mathcal{A}) = \emptyset$ iff $E_{\mathcal{A}} \cap \mathfrak{F} = \emptyset$. Note that the reachability of a macro state is only subject to the existence of an instance, so that instantiations have not to be distinguished for the emptiness test. A corresponding algorithm has to perform $|E_{\mathcal{A}}| \leq |\mathfrak{Q}|$ iterations at most, and in each step the transitions of Δ have to be applied on the so far calculated set $E_k \subseteq E_{\mathcal{A}}$. So we get a time complexity of $O(|\mathfrak{Q}|^2 \cdot |\Delta|)$.

The infinity problem is a variant of the emptiness problem which is decidable in an analogous way. It can be checked, whether there exists a final macro state which can be visited twice. If so, the transition relation permits a loop and the final macro state can be visited infinitely often, hence $\mathcal{A}$ recognizes an infinite language.

A formalization of the reachability tests for the emptiness problem and the infinity problem can be found in [Rad07]. □

Of course, due to Lemma 4.2(c) rational tree relations inherit all undecidable properties of rational word relations:

Proposition 4.7. *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 $R_1 = R_2$.*

Lemma 4.2 has another application in a question which arises from the rational tree languages (cf. Sect. 3.2). We prove that it is undecidable whether a rational tree language is regular.

Theorem 4.8. *It is undecidable*

(a) whether for a given rational word relation R the tree language $TreeLang(R)$ is regular,
(b) whether a rational tree language is regular.

Proof. (a) Given a rational word relation $R \subseteq \Sigma^* \times \Gamma^*$ defined by an asynchronous automaton $\mathcal{A}$. Assume, it is decidable whether the tree language $\mathrm{TreeLang}(R)$ is regular. Then we can decide whether R is recognizable: Due to Lemma 4.2(d), we know $\mathrm{TreeLang}(R)$ is rational. We decide whether $\mathrm{TreeLang}(R)$ is a regular tree language (due to our assumption). If so, R is recognizable, otherwise not (due to Lemma 4.2(b)). This is a contradiction, because it is undecidable whether a rational word language R is recognizable (cf. [CCG06]).

(b) follows from (a). □

4.4 Transductions Ought to Preserve

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 [Ber79]. Here we note that binary rational tree relations do *not* even preserve regularity:

Example 4.9 ([Rao97]). Consider the rational tree relation $T_{\mathrm{sim}} = \{f(g^n a, g^n a) \mid n \in \mathbb{N}\}$ from Example 3.3. 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 $R' := T_{\mathrm{sim}} \times \Sigma^*$.

5 Separate-Rational Tree Relations

We have seen a few drawbacks of rational tree relations. They do not coincide with regular tree language when restricted to unary sets, they are not closed under composition, and they are unsuitable as transductions in the sense that they do not preserve regular tree languages.

In [Rao97] Raoult proposes a restriction of rational tree relations, generated by so-called *transduction grammars*. These satisfy 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. We will discuss the assets and drawbacks of Raoult's transduction grammars and our so-called *separate-rational tree relations* at the end of this section.

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.

5.1 Definition of Separate-Rational Tree Relations via Separate-Rational Expressions

For the definition via *separate-rational expressions* we demand that variables of the same multivariable may only occur in different components of the generated tuples and that relations are generated by tuples of trees of height 2 at most (this will assure the recognizability of the tuples by separate-asynchronous tree automata as defined in the next section).

Definition 5.1. The classes SepRat_n of *separate-rational tree relations* are defined inductively as follows:

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

Example 5.2. (a) The relation from Example 3.1 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 relation T_{sim} from Example 3.3 is *not* separate-rational, because in order to define T_{sim} we need two different variables of one multivariable to occur in the same component of a tuple.

(c) The rational relations R_1 and R_2 from Example 4.5 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 different variables of one multivariable have to occur in the same component of a tuple.

5.2 Separate-Asynchronous Tree Automata

Now we will restrict asynchronous tree automata, so that these recognize exactly the class of separate-rational tree relations. In contrast to asynchronous tree automata, which use the same ranked alphabet for all components, we allow separate-asynchronous automata to utilize a specific ranked alphabet for each component.

Definition 5.3. A separate-asynchronous tree automaton $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma_1, \dots, \Sigma_n, \Delta, \mathfrak{F}\rangle$ 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 alphabet) with the following restrictions:

– the set Q of states is partitioned in $Q = Q_1 \uplus \ldots \uplus Q_n$,
– for each macro state $(q_1, \dots, q_m) \in \mathfrak{Q}$ and all $q_k \neq q_l$ with $1 \leq k, l \leq m$ and $1 \leq j \leq n$ 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

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

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

The Equivalence Theorem (Theorem 3.7) 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 5.3 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 results in trees of height 1 or 2 only.

Theorem 5.4. *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 $R(\mathcal{A}^{(n)}) = R$.*

5.3 The Interrelationship to Rational Word Relations

We have already discussed the interrelationship between rational tree relations and rational word relations in Sect. 4. Now we discuss these thoughts for separate-rational tree relations.

We note that a unary separate-rational tree language is regular, because by restricting all multivariables to length 1 (resp. all macro states to size 1) we yield a regular tree language. So, a rational word relation cannot be represented by a separate-rational tree language (in the spirit of Lemma 4.2(d)), since the separate-rational tree languages are precisely the regular sets. For a rational word relation R the tree language TreeLang(R) is not separate-rational in general.

The reason Lemma 4.2(d) does not inherit to separate-rational relations is only up to the fact that synchronization is not allowed within the same projection to one component. If we understand each component (of a tuple in a word relation) as a unary tree, this restriction is insignificant. Therefore, we can reformulate Lemma 4.2(c) for separate-rational tree relations:

Lemma 5.5. *A word relation R is rational iff TreeRel(R) is separate-rational.*

5.4 Properties of Separate-Rational Tree Relations

Due to Lemma 5.5, we obtain the same undecidability results and closure properties as for rational tree relations. Beyond this, separate-rational relations resolve the issues raised in Sect. 4.

Theorem 5.6. *(a) The class $SepRat_1$ of separate-rational tree languages is precisely 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) For separate-rational tree relations R and S construct a separate-asynchronous automaton recognizing $R \circledcirc S := \{(t, t', t'') \mid (t, t') \in R, (t', t'') \in S\}$ 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 Appendix B for the detailed proof.)

(c) Due to symmetry of Definition 5.3, 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 5.6(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 (see Lemma B.1 in Appendix B) and Theorem 5.6(a)). □

If we consider rational relations over words, they also preserve context-free languages [Ber79]. It is an open question whether separate-rational tree relations also preserve context-free tree languages as defined in [GS97].

Of course, per definition every separate-rational tree relation is also rational and every separate-asynchronous tree automaton is an ordinary asynchronous tree automaton. The decidability of the converse questions is more interesting:

Theorem 5.7. *(a) It is decidable whether a given asynchronous tree automaton over Σ is also separate-asynchronous over $\Sigma \times \ldots \times \Sigma$.*
(b) It is undecidable whether a rational tree relation is separate-rational.

Proof. (a) The additional conditions of Definition 5.3 are easy to verify. It suffices to check whether there exists an adequate partition of the state set and the transition relation.

(b) Assume, it is decidable whether a rational tree relation is separate-rational. Given a rational tree language T, we know T is separate-rational iff T is regular (see Theorem 5.6(a)). So, we can decide whether T is a regular tree language. But this is undecidable due to Theorem 4.8. □

5.5 Discussion on Restrictions of Rational Tree Relations

In [Rao97] Raoult proposes a restriction of rational tree relations generated by so-called *transduction grammars*. Raoult's restriction requires a $(p + q)$-ary relation to be decomposed into two parts, so that each multivariable have all their variables in two trees at most, one tree in the first p and the other tree in the last q components (of a tuple in the relation) (see [Rao97] for details). This restriction also solves the discussed issues of rational tree relations (i. e. Theorem 5.6 hold for Raoult's restriction), but from our point of view two other drawbacks arises.

The main drawback of Raoult's transduction grammars is the lack of a natural automata theoretic characterization in our framework of asynchronous tree automata. The other drawback is that transduction grammars can only be considered as a generalization of binary rational relations over words, but not as a generalization of n-ary rational relations:

Example 5.8. The trinary relation $R_3 = \{(g^n a, g^n a, g^n a) \mid n \in \mathbb{N}\}$ is separate-rational, but does not satisfy the restriction which arises from transduction grammars in [Rao97].

Considering binary relations only, Raoult's transduction grammar are more general than binary separate-rational relations:

Example 5.9. Consider the relations R, S, and $R \circ S$ depicted in Fig. 4. These relations can be generated by transduction grammars, but are not separate-rational.

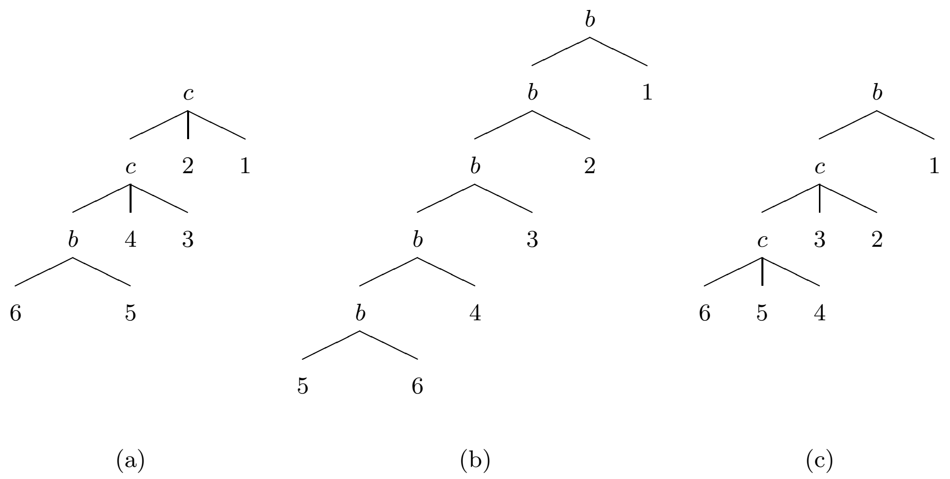

Fig. 4. Pairs of numbers represent the identity, i. e. same numbers denote same subtrees. The relation between trees of the form (a) and (b) is denoted by R, the relation between trees of the form (b) and (c) is denoted by S. So, the composition $R \circ S$ is the relation between trees of the form (a) and (c).

Remark 5.10. Example 5.9 was first proposed by Arnold and Dauchet [AD82]. They investigated an interesting type of tree relations which are definable via *morphisms (of "magmoids") strict and separated with delay* (see [AD82, Rao92] for details). These relations are also closed under composition and include the relations R, S, and $R \circ S$ from the example. Hence, separate-rational tree relations miss the definability of some relations definable via these morphisms. Raoult mentionend in [Rao97] that his transduction grammars contain the relations of Arnold and Dauchet.

The relations from the example also illustrate the reason for restricting trees to a height of at most 2 in Definition 5.1. Otherwise, on the one hand problems regarding the closure under composition occur with our approach, and on the other hand a corresponding restriction for asynchronous automata would be hard to define.

Separate-rational tree relations have a drawback regarding their expressiveness. Since those relations are generated by tuple of trees of height 2 at most, they miss the inclusion of relations relaying on this feature. For example the left rotation and the right rotation (cf. [Rao97], Fig. 2) are not separate-rational, and even some relations definable by *linear* (top-down or bottom-up) tree transducers (cf. [CDG$^+$07]) are not separate-rational, but are definable by transduction grammars. However, separate-rational relations comprise automatic tree relations (cf. [Blu99, BG04]), whereas transduction grammars misses this inclusion in the n-ary case.

Furthermore, a proper generalization of Raoult's restriction of transduction grammars to the n-ary case seems possible by demanding that each multivariable has all their variables in n trees at most, one tree in each component (of a tuple in the relation). Generalizing Raoult's restriction to the n-ary case in this way results in a less restrictive class which comprises more relations than separate-rational tree relations. So, with respect to expressiveness Raoult's transduction grammars are perhaps a more promising approach to restrict rational tree relations.

Nevertheless, the main drawback of Raoult's restriction remains that it does not seem to be natural in the framework of asynchronous tree automata. An automata theoretic description for transduction grammars should assure that all states of a macro state in the same component belong to one "elemental tree". So, such automata should rather have transitions over trees instead of transitions over single symbols.

To sum it up, separate-rational tree relations and relations generated by transduction grammars are incomparable in the n-ary case, and in the binary case Raoult's tree transductions are strictly stronger than separate-rational relations. Separate-rational tree relations are a proper generalization of rational relations over words with a natural automata theoretic description.

6 Conclusion

We presented a reasonable automata theoretic approach to rational tree relations which now can be described by three equivalent formalisms: Rational expressions, tree grammars [Rao97], and asynchronous tree automata. We argued to what extent the properties of rational relations over words apply to trees. Separate-rational tree relations overcome some drawbacks of the rational tree relations. They generate exactly the regular sets when restricted to the unary case, are closed under composition, and preserve regular tree languages. 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). We discussed the differences between the restrictions of separate-rational relations and Raoult's transduction grammars. While Raoult's transduction grammars have some advantages regarding the expressiveness, separate-rational tree relations are a proper generalization of n-ary 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 [CDG$^+$07]) and some classes of *term rewriting systems* [Mey04]. These results do not hold for the separate-rational restriction. More expressive extensions of separate-rational relations with such features need to be investigated. As discussed in Sect. 5.5, automata which allow transitions over entire trees (instead of transitions over single symbols) could be a promising approach to such a class.

As mentioned in the introduction asynchronous automata enable the definition of rational relations over unranked trees. A straightforward approach is the usage of regular expressions in transitions; but the problem arises that we have used a finite set Var of variables in Definition 3.6 which is problematic for a node of unbounded rank. In [Rad07] we propose a solution by numbering the occurrences of variables in a given sequence of states. So, we still use a finite set Var of variables in the transitions, but interpret it in an unbounded manner. For example if we use a transition $(L, a, (q', i'))$ with L given by a regular expression $((q_1, i)(q_2, i)(q, i'))^+$ and $(q), (q_1, q_2) \in \mathfrak{Q}$, $i, i' \in$ Var for a node of rank 6, we will interpret it as $((q_1, i_1), (q_2, i_1), (q, i'_1), (q_1, i_2), (q_2, i_2), (q, i'_2), a, (q', i'))$.

In the word case asynchronous automata enable the definition of a deterministic subclass of rational relations (cf. [RS59, PS99, Gri02]). Analogously, asynchronous tree automata enable the definition of deterministic rational tree relations (both over ranked and unranked trees). A deterministic *top-down* model of rational tree relations is easy to define (see [CLT05, Rad07]). The automaton starts in a unique macro state $\mathfrak{q}_0$ at the root nodes. Then the transition relation is made deterministic by partitioning of the macro state set. In a macro state $(q_1, \ldots, q_k)$ of the i-th partition the automaton only reads the symbol at the node marked with q_i and makes a top-down step at this node with a proper transition. In the other nodes marked with the states $q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_k$ only ε-transition are applied. The state q_i and the labeling (in Σ) of this node deterministically determine the computation step. Unfortunately a top-down approach have the same weakness as ordinary deterministic top-down tree automata, so such automata do not recognize all regular tree languages (resp. recognizable tree relations). Hence, this approach yields a hierarchy of automata definable relations over trees which differ from the case of words. Due to the non-deterministic mechanism of merging states to instances of macro states in a run, a deterministic bottom-up model seems to be challenging. 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.

References

[AD82] André Arnold and Max Dauchet. Morphismes et bimorphismes d'arbres. *Theoretical Computer Science*, 20:33–93, 1982.

[Ber79] Jean Berstel. *Transductions and Context-Free Languages.* Number 38 in Leitfäden der angewandten Mathematik und Mechanik. Teubner, Stuttgart, 1979.

[BG04] Achim Blumensath and Erich Grädel. Finite presentations of infinite structures: Automata and interpretations. *Theory of Computing Systems*, 37:641–674, 2004.

[Blu99] Achim Blumensath. Automatic structures. Diploma thesis, RWTH Aachen, 1999.

[BMW01] Anne Brüggemann-Klein, Makoto Murata, and Derick Wood. Regular tree and regular hedge languages over unranked alphabets: Version 1. Technical Report HKUST-TCSC-2001-05, The Hongkong University of Science and Technology, 2001.

[Cau00] Didier Caucal. On word rewriting systems having a rational derivation. In *Proceedings of FoSSaCS*, volume 1784 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2000.

[CCG06] Olivier Carton, Christian Choffrut, and Serge Grigorieff. Decision problems among the main subfamilies of rational relations. *Theoretical Informatics and Applications*, 40(2):255–275, 2006.

[CDG$^+$07] Hubert Comon, Max Dauchet, Rémi Gilleron, Christof Löding, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. Unpublished electronic book, release October, 12th, 2007. `http://tata.gforge.inria.fr`.

[CLT05] Julien Cristau, Christof Löding, and Wolfgang Thomas. Deterministic automata on unranked trees. In *Proceedings of FCT*, volume 3623 of *Lecture Notes in Computer Science*, pages 68–79. Springer, 2005.

[Eil74] Samuel Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, New York, 1974.

[EM65] Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68, 1965.

[FR68] Patrick C. Fischer and Arnold L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2(1):88–101, 1968.

[Gri02] Serge Grigorieff. Modelization of deterministic rational relations. *Theoretical Computer Science*, 281(1-2):423–453, 2002.

[GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata.* Akadémiai Kiadó, Budapest, 1984.

[GS97] Ferenc Gécseg and Magnus Steinby. *Handbook of Formal Languages, Vol. 3: Beyond Words*, chapter Tree Languages, pages 1–68. Springer, 1997.

[Mey04] Antoine Meyer. On term rewriting systems having a rational derivation. In *Proceedings of FoSSaCS*, volume 2987 of *Lecture Notes in Computer Science*, pages 378–392. Springer, 2004.

[Mey05] Antoine Meyer. *Finitely presented infinite graphs.* PhD thesis, Université de Rennes 1, 2005.

[Mor00] Christophe Morvan. On rational graphs. In *Proceedings of FoSSaCS*, volume 1784 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2000.

[PS99] Maryse Pelletier and Jacques Sakarovitch. On the representation of finite deterministic 2-tape automata. *Theoretical Computer Science*, 225(1-2):1–63, 1999.

[Rad07] Frank G. Radmacher. Automatendefinierbare Relationen über Bäumen (Automata Definable Relations over Trees). Diploma thesis (revised version), RWTH Aachen, 2007. `http://www.automata.rwth-aachen.de/~radmacher/`.

[Rad08] Frank G. Radmacher. An automata theoretic approach to rational tree relations. In *Proceedings of SOFSEM*, volume 4910 of *Lecture Notes in Computer Science*, pages 424–435. Springer, 2008.

[Rao92] Jean-Claude Raoult. A survey of tree transductions. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*, pages 311–326. Elsevier, 1992. (also published as report 1410 INRIA-Rennes, 1991).

[Rao97] Jean-Claude Raoult. Rational tree relations. *Bulletin of the Belgian Mathematical Society*, 4(1):149–176, 1997.

[RS59] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):115–125, 1959.

A Proof of the Equivalence Theorem (Theorem 3.7)

Proof (⇒). Given a rational relation of n-tuples of trees. According to Definition 3.2 we can assume that this relation is specified by iterated application of operators $\cup$, $\cdot_X$ and $*_X$ on finite relations of tuples of trees. Hence, we shall prove our claim inductively.

Induction start: Every finite relation can be understood as union of relations which consists of one n-tuple of trees at most. The special case of the empty relation is recognized by an asynchronous tree automaton with an empty set of final macro states. For the relations consisting of exactly one n-tuple of trees we have to prepare the automaton in such a way that the concatenation $\cdot_X$ and the star $*_X$ with respect to a multivariable $X = x_1 \cdots x_n$ can be applied. Thus, when reading an instance of $x_1 \cdots x_n$ at the leaves the automaton have to assume a macro state $(q_{x_1}, \ldots, q_{x_n})$.

Example A.1. Consider the tuple (t_1, t_2, t_3) of trees in Fig. 5. Thereby $X = x_1x_2$ is a multivariable with two occurrences of instances in (t_1, t_2, t_3). We construct an asynchronous automaton which only accepts (t_1, t_2, t_3) and assumes a macro state (q_{x_1}, q_{x_2}) when reading an instance of $X = x_1x_2$ at the leaves.

We construct $\mathcal{A}^{(3)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F}\rangle$ over $\Sigma = \{a^{(0)}, x_1^{(0)}, x_2^{(0)}, b^{(1)}, c^{(2)}, \}$ with

- $Q = \{\, q^1_{x_1}, q^1_{x_2}, q^2_{x_1}, q^2_{x_2}, r, s, q_{f_1}, q_{f_2}, q_{f_3} \,\}$,
- $\mathfrak{Q} = \{\, (q^1_{x_1}, q^1_{x_2}),\ (q^2_{x_1}, q^2_{x_2}),\ (r),\ (s),\ (q_{f_1}, q_{f_2}, q_{f_3}) \,\}$,
- $\mathfrak{F} = \{\, (q_{f_1}, q_{f_2}, q_{f_3}) \,\}$,
- $\mathrm{Var} = \{\, i, j, k, l \,\}$, and
- $\Delta = \{\ (x_1, (q^1_{x_1}, i)),\ (x_2, (q^1_{x_2}, i)),\ (a, (r, i)),$
  $((q^1_{x_1}, i), \varepsilon, (q^2_{x_1}, i)),\ ((q^1_{x_2}, i), b, (q^2_{x_2}, i)),\ ((r, i), b, (s, i)),$
  $((q^2_{x_1}, i), (q^2_{x_2}, i), c, (q_{f_1}, l)),\ ((q^1_{x_1}, j), b, (q_{f_2}, l)),$
  $((q^1_{x_2}, j), (s, k), c, (q_{f_3}, l))\ \}$.

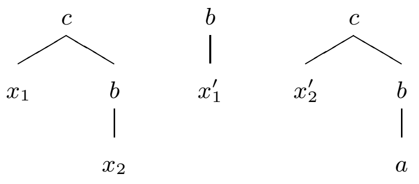

Fig. 5. A triple of trees (t_1, t_2, t_3) with two instances of a multivariable $X = x_1x_2$.

It should be quite clear that every relation which only consists of one tuple of trees is recognizable in this way (nevertheless, formally writing down the construction seems to be cumbersome).

induction step: We construct asynchronous tree automata for the operations $\cup$, $\cdot_X$ and $*_X$:

- Given asynchronous tree automata $\mathcal{A}_1^{(n)} = \langle Q_1, \mathfrak{Q}_1, \mathrm{Var}_1, \Sigma, \Delta_1, \mathfrak{F}_1\rangle$ and $\mathcal{A}_2^{(n)} = \langle Q_2, \mathfrak{Q}_2, \mathrm{Var}_2, \Sigma, \Delta_2, \mathfrak{F}_2\rangle$ with $Q_1 \cap Q_2 = \emptyset$. The relation $R(\mathcal{A}_1^{(n)}) \cup R(\mathcal{A}_2^{(n)})$ can be recognized by the asynchronous tree automaton $\mathcal{A}^{(n)} = \langle Q_1 \cup Q_2, \mathfrak{Q}_1 \cup \mathfrak{Q}_2, \mathrm{Var}_1 \cup \mathrm{Var}_2, \Sigma, \Delta_1 \cup \Delta_2, \mathfrak{F}_1 \cup \mathfrak{F}_2\rangle$.

– Given asynchronous tree automata $\mathcal{A}_1^{(n)} = \langle Q_1, \mathfrak{Q}_1, \mathrm{Var}_1, \Sigma, \Delta_1, \mathfrak{F}_1 \rangle$ and $\mathcal{A}_2^{(m)} = \langle Q_2, \mathfrak{Q}_2, \mathrm{Var}_2, \Sigma, \Delta_2, \mathfrak{F}_2 \rangle$ with $Q_1 \cap Q_2 = \emptyset$. Furthermore let $X = x_1 \cdots x_m$ be a multivariable, and $\mathcal{A}_1^{(n)}$ assumes the macro state $\mathfrak{q}^1 = (q_1^1, \ldots, q_m^1) \in \mathfrak{Q}_1$ when reading $x_1, \ldots, x_m$ at the leaves. The relation $R(\mathcal{A}_1^{(n)}) \cdot_X R(\mathcal{A}_2^{(m)})$ can be recognized by the asynchronous tree automaton $\mathcal{A}^{(n)} = \langle Q_1 \cup Q_2, \mathfrak{Q}_1 \cup \mathfrak{Q}_2, \mathrm{Var}_1 \cup \mathrm{Var}_2 \cup \{k, k'\}, \Sigma, \Delta, \mathfrak{F}_1 \rangle$ with

$$\begin{aligned}\Delta = (&\Delta_1 \setminus \{(x_1, (q_1^1, i)), \ldots, (x_m, (q_m^1, i)) \mid i \in \mathrm{Var}_1\}) \cup \Delta_2 \\ &\cup \{((q_1^2, k'), \varepsilon, (q_1^1, k)), \ldots, ((q_m^2, k'), \varepsilon, (q_m^1, k)) \mid \\ &\qquad \text{there exist } \mathfrak{q}^2 = (q_1^2, \ldots, q_m^2) \in \mathfrak{F}_2 \\ &\qquad \text{and } (x_1, (q_1^1, i)), \ldots, (x_m, (q_m^1, i)) \in \Delta_1 \text{ with } i \in \mathrm{Var}_1\} \ .\end{aligned}$$

– Given an asynchronous tree automaton $\mathcal{A}_1^{(n)} = \langle Q_1, \mathfrak{Q}_1, \mathrm{Var}_1, \Sigma, \Delta_1, \mathfrak{F}_1 \rangle$. Furthermore let $X = x_1 \cdots x_n$ be a multivariable, and $\mathcal{A}_1^{(n)}$ assumes the macro state $\mathfrak{q} = (q_1, \ldots, q_n) \in \mathfrak{Q}_1$ when reading $x_1, \ldots, x_n$ at the leaves. The relation $R(\mathcal{A}_1^{(n)})^{*X}$ can be recognized by the asynchronous tree automaton $\mathcal{A}^{(n)} = \langle Q_1, \mathfrak{Q}_1, \mathrm{Var}_1 \cup \{k, k'\}, \Sigma, \Delta, \mathfrak{F}_1 \rangle$ with

$$\begin{aligned}\Delta = \Delta_1 \cup \{&((q_1', k'), \varepsilon, (q_1, k)), \ldots, ((q_n', k'), \varepsilon, (q_n, k)) \mid \\ &\text{there exist } \mathfrak{q}' = (q_1', \ldots, q_n') \in \mathfrak{F}_1 \\ &\text{and } (x_1, (q_1, i)), \ldots, (x_n, (q_n, i)) \in \Delta_1 \text{ with } i \in \mathrm{Var}_1\} \ .\end{aligned}$$

□

Proof (⇐). Given an asynchronous tree automaton $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$, we show that $R(\mathcal{A}^{(n)})$ is a n-ary rational tree relation according to Definition 3.2.

First of all we extent the alphabet Σ with new nullary symbols, in order to apply the operations $\cdot_X$ and $*_X$. For this purpose we missuse the states of the given automaton and extent Σ to $\Sigma \uplus Q$ where all symbols in Q have arity 0.

Example A.2. Let $\Sigma = \{a^{(0)}, b^{(1)}, c^{(2)}\}$ and $\mathfrak{Q} = \{(p_1, p_2), (q_1, q_2), (r)\}$. An example tree over the extended alphabet $\Sigma \cup Q$ is shown in Fig. 6 (a).

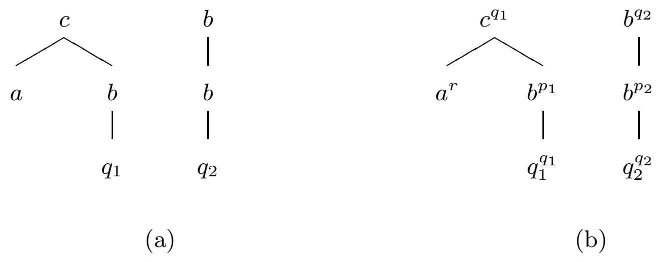

Fig. 6. (a) A pair of trees over $\Sigma \cup Q$; (b) a run on this pair in $R_2(\mathfrak{R}, \mathfrak{S}, (q_1, q_2))$.

After extending the alphabet we construct for $\mathcal{A}^{(n)}$ a new automaton $\mathcal{A}'^{(n)}$ by adding transitions $(p_1, (p_1, i)), \ldots, (p_k, (p_k, i))$ for all $\mathfrak{p} = (p_1, \ldots, p_k) \in \mathfrak{Q}$. This allows the automaton to read the new nullary symbols at the leaves and in doing so assume the corresponding states (having the same name). Notice, this assures that symbols in Q which form an entire macro state are supposed to appear together, because only in this case the automaton $\mathcal{A}'^{(n)}$ can assume the macro state $\mathfrak{p} = (p_1, \ldots, p_k)$.

For $\mathfrak{R}, \mathfrak{S} \subseteq \mathfrak{Q}, \mathfrak{q} \in \mathfrak{Q}$ we define a set $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$. It consists of all n-tuple of trees over $\Sigma \uplus Q$ for which an accepting run of $\mathcal{A}'^{(n)}$ exists, so that

- on all leaves labeled with symbols in Q, the run begins with macro states in $\mathfrak{R}$,
- the run ends in an accepting configuration at the root nodes assuming the macro state $\mathfrak{q} = (q_1, \ldots, q_n)$,
- in the remaining configurations of the run only macro states in $\mathfrak{S}$ occur.

We call the elements of $\mathfrak{R}$ "source states", $\mathfrak{q}$ the "target state" and the elements of $\mathfrak{S}$ "intermediate states".

Example A.3. Let $\mathfrak{R} = \{(q_1, q_2)\}$ and $\mathfrak{S} = \{(p_1, p_2), (r)\}$. An example run on a pair in $R_2(\mathfrak{R}, \mathfrak{S}, (q_1, q_2))$ is shown in Fig. 6 (b).

By definition of $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ holds:

$$R(\mathcal{A}^{(n)}) = \bigcup_{\mathfrak{q} \in \mathfrak{F}} R_n(\emptyset, \mathfrak{Q}, \mathfrak{q}) \ .$$

Hence, it suffices to show that all sets $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ are rational. We show this by induction over $\mathfrak{S}$.

Induction start: $\mathfrak{S} = \emptyset$. In this case $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ only consists of n-tuple of trees of height 1 or 2. These are finitely many, so $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ is rational.

Induction step: $|\mathfrak{S}| > 0$. Let $\mathfrak{S} = \mathfrak{S}_0 \cup \{\mathfrak{s}\}$ (with $\mathfrak{s} \notin \mathfrak{S}_0$), where for $\mathfrak{S}_0$ the induction hypothesis holds. It holds $(t_1, \ldots, t_n) \in R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ iff there exists a run on $(t_1, \ldots, t_n)$ with a source state in $\mathfrak{R}$, target state $\mathfrak{q}$ and an intermediate state in $\mathfrak{S}_0 \cup \{\mathfrak{s}\}$. We can unambiguously decompose $(t_1, \ldots, t_n)$ at the nodes where $\mathfrak{s} = (s_1, \ldots, s_k)$ occurs in the run. An example of a decomposition with respect to a macro state is depicted in Fig. 7. The decomposed parts have all their intermediate states in $\mathfrak{S}_0$, hence they are rational by induction hypothesis. So, $R_n(\mathfrak{R}, \mathfrak{S}, \mathfrak{q})$ is rational, because it can be composed by the rational expression

$$\begin{aligned}
&R_n(\mathfrak{R}, \mathfrak{S}_0 \cup \{\mathfrak{s}\}, \mathfrak{q}) = \\
&R_n(\mathfrak{R}, \mathfrak{S}_0, \mathfrak{q}) \ \cup \ R_n(\mathfrak{R} \cup \{\mathfrak{s}\}, \mathfrak{S}_0, \mathfrak{q}) \cdot_S (R_n(\mathfrak{R} \cup \{\mathfrak{s}\}, \mathfrak{S}_0, \mathfrak{s}))^{*_S} \cdot_S R_n(\mathfrak{R}, \mathfrak{S}_0, \mathfrak{s})
\end{aligned}$$

with $S = s_1 \cdots s_k$.

The concatenation of the decomposed parts is possible, because the states $s_1, \ldots, s_k$ in a run of $\mathcal{A}'^{(n)}$ can only be reached and left simultaneously, and in each bottom-up computation step only one instance of $\mathfrak{s} = (s_1, \ldots, s_k)$ can be reached. For any run this is assured by the requirements on states and variables in Definition 3.6. Examples for *impossible* decompositions are depicted in Fig. 8. □

B Separate-Rational Tree Relations are Closed Under Composition (Proof of Theorem 5.6(b))

One major advantage of separate-rational relations is their closure under composition. Towards the proof, we need to show that separate-rational relations

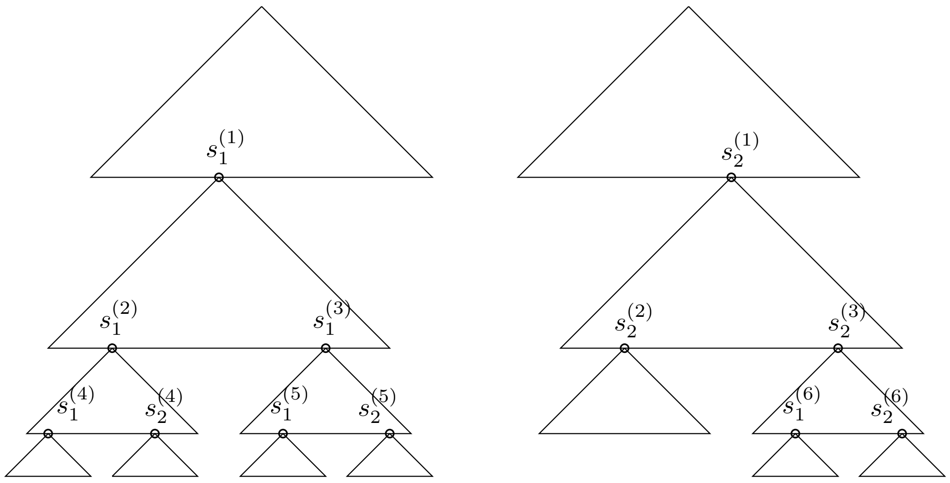

Fig. 7. Decomposition of a run of an asynchronous tree automaton with respect to a macro state (s_1, s_2). Superscripts number distinct instances.

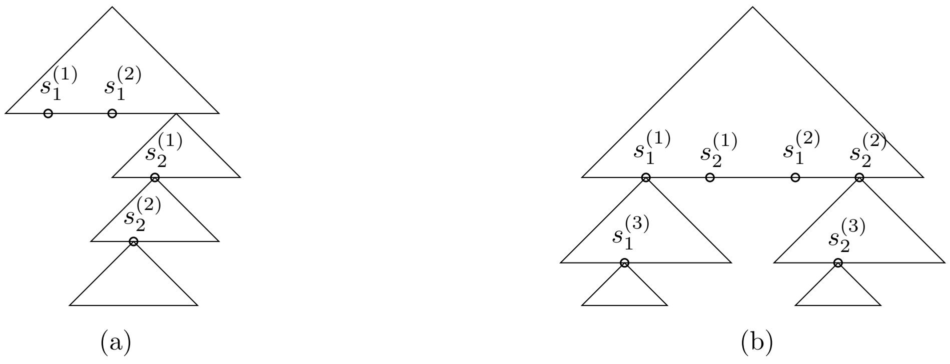

Fig. 8. Decompositions with respect to a macro state (s_1, s_2) which *cannot* occur in a run of an asynchronous tree automaton.

are closed under projection (a result Raoult has shown already for rational tree relations [Rao97]).

Lemma B.1 (Projection Lemma). *The class $SepRat_n$ of separate-rational tree relation is closed under projection, i. e. for every n-ary separate-rational relation R each $(n-1)$-ary relation R' with*

$$(t_1, \ldots, t_{i-1}, t_{i+1} \ldots, t_n) \in R' \iff \exists t_i : (t_1, \ldots, t_{i-1}, t_i, t_{i+1} \ldots, t_n) \in R$$

is also separate-rational (for an arbitrary $1 \le i \le n$).

Proof. Given an n-ary R by a separate-asynchronous tree automaton $\mathcal{A}_R^{(n)}$ with states in $Q_1 \uplus \ldots \uplus Q_n$ according to Definition 5.3. Assume that all transitions are reachable and productive, i. e. for each transition τ there exists an accepting run using τ. This can be easily checked (e. g. by modifying $\mathcal{A}_R^{(n)}$ and using the emptiness test, or by directly using a suitable reachability test). Then we can construct the separate-asynchronous tree automaton $\mathcal{A}_{R'}^{(n-1)}$ for R' by deleting the i-th component, i. e. delete all transitions in Δ_i, and change each macro state $(q_1, \ldots, q_{k-1}, q_k, q_{k+1} \ldots, q_l)$ with $q_k \in Q_i$ to $(q_1, \ldots, q_{k-1}, q_{k+1} \ldots, q_l)$. □

Towards the closer under composition, we construct for binary separate-rational tree relations R and S a separate-asynchronous tree automaton for

$$R \circledcirc S := \{(t, t', t'') \mid (t, t') \in R, (t', t'') \in S\}$$

by synchronization of the common component. Then the Projection Lemma yields a separate-asynchronous automaton for $R \circ S$.

We introduce the abbreviated form "sync(Δ')" to express that a set Δ' of transitions can be applied simultaneously in a run of an asynchronous automata:

Definition B.2. Let $\mathcal{A}^{(n)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma, \Delta, \mathfrak{F} \rangle$ be an asynchronous tree automaton. It holds sync(Δ') for a subset $\Delta' \subseteq \Delta$ iff a computation step is possible exactly with the transitions in Δ', i. e. there exist a tuple of trees and configurations c_1, c_2 (on this tuple), so that exactly the transitions in Δ' are used for the computation step $c_1 \to c_2$.

Now we can give the formal construction for $R \circledcirc S$ that proves Theorem 5.6(b), i. e. that the class SepRat_2 of binary separate-rational tree relations is closed under composition.

Proof (of Theorem 5.6(b)). Given $R, S \in \mathrm{SepRat}_2$ by separate-asynchronous automata $\mathcal{A}_R = \langle Q, \mathfrak{Q}_1, \mathrm{Var}_1, \Sigma_1, \Gamma, \Delta_1, \mathfrak{F}_1 \rangle$ with a partitioned state set $Q = Q_1 \cup Q_2$ and $\mathcal{A}_S = \langle P, \mathfrak{Q}_2, \mathrm{Var}_2, \Gamma, \Sigma_2, \Delta_2, \mathfrak{F}_2 \rangle$ with a partitioned state set $P = P_1 \cup P_2$. We assume w. l. o. g. $(q_1, q_2) \in \mathfrak{Q}_1 \Rightarrow q_1 \in Q_1 \wedge q_2 \in Q_2$ (analogously for $\mathcal{A}_S$). We further assume w. l. o. g. that for every macro state $(q_1, q_2) \in \mathfrak{Q}_1$ also ε-transitions $((q_1, i), \varepsilon, (q_1, i))$, $((q_2, i), \varepsilon, (q_2, i))$ are in Δ_1 (analogously for $\mathcal{A}_S$). Generally these ε-transitions are needed for a synchronization of the common component.

For $R \circledcirc S := \{(t,t',t'') \mid (t,t') \in R, (t',t'') \in S\}$ we construct the separate-asynchronous automaton $\mathcal{A}^{(3)}_{RS} = \langle Q', \mathfrak{Q}', \mathrm{Var}', \Sigma_1, \Gamma, \Sigma_2, \Delta', \mathfrak{F}' \rangle$ with

$$
\begin{aligned}
Q' &:= Q_1 \uplus Q_2 \times P_1 \uplus P_2\,,\\
\mathfrak{Q}' &:= \{(q) \mid (q) \in \mathfrak{Q}_1,\, q \in Q_1\}\\
&\quad \cup \{(p) \mid (p) \in \mathfrak{Q}_2,\, p \in P_2\}\\
&\quad \cup \{(qp) \mid (q) \in \mathfrak{Q}_1,\, q \in Q_2,\ (p) \in \mathfrak{Q}_2,\, q \in P_1\}\\
&\quad \cup \{(q_1, q_2p) \mid (p) \in \mathfrak{Q}_2,\, p \in P_1,\ (q_1,q_2) \in \mathfrak{Q}_1\}\\
&\quad \cup \{(qp_1, p_2) \mid (q) \in \mathfrak{Q}_1,\, q \in Q_2,\ (p_1,p_2) \in \mathfrak{Q}_2\}\\
&\quad \cup \{(q_1, q_2p_1, p_2) \mid (q_1,q_2) \in \mathfrak{Q}_1,\ (p_1,p_2) \in \mathfrak{Q}_2\}\,,\\
\mathrm{Var}' &:= \mathrm{Var}_1 \cup \mathrm{Var}_2 \cup (\mathrm{Var}_1 \times \mathrm{Var}_2)\,,\\
\Delta' &:= \{(q_1,\dots,q_k,x,(q,i)) \mid (q) \in \mathfrak{Q}_1,\ q \in Q_1,\ (q_1,\dots,q_k,x,(q,i)) \in \Delta_1\}\\
&\quad \cup \{(p_1,\dots,p_k,y,(p,i)) \mid (p) \in \mathfrak{Q}_2,\ p \in P_2,\ (p_1,\dots,p_k,y,(p,i)) \in \Delta_2\}\\
&\quad \cup \{(q_1p_1,\dots,q_kp_k,z,(qp,(i_1,i_2))) \mid (q) \in \mathfrak{Q}_1,\, q \in Q_2,\ (p) \in \mathfrak{Q}_2,\, p \in P_1,\\
&\qquad\qquad (q_1,\dots,q_k,z,(q,i_1)) \in \Delta_1,\ (p_1,\dots,p_k,z,(p,i_2)) \in \Delta_2\}\\
&\quad \cup \{(q'_1,\dots,q'_k,x,(q_1,(i_1,i_2))),\ (r_1s_1,\dots,r_ls_l,z,(q_2p,(i_1,i_2))) \mid\\
&\qquad\qquad (q_1,q_2) \in \mathfrak{Q}_1,\ (p) \in \mathfrak{Q}_2,\, p \in P_1,\ (s_1,\dots,s_l,z,(p,i_2)) \in \Delta_2,\\
&\qquad\qquad \tau_1 : (q'_1,\dots,q'_k,x,(q_1,i_1)) \in \Delta_1,\ \tau_2 : (r_1,\dots,r_l,z,(q_2,i_1)) \in \Delta_1,\\
&\qquad\qquad \text{and } \mathrm{sync}(\{\tau_1,\tau_2\}) \text{ holds}\}\\
&\quad \cup \{(\dots,z,(qp_1,(i_1,i_2))),\ (\dots,y,(p_2,(i_1,i_2))) \mid\\
&\qquad\qquad (q) \in \mathfrak{Q}_1,\, q \in Q_2,\ (p_1,p_2) \in \mathfrak{Q}_2,\ (\dots,z,(q,i_1)) \in \Delta_1,\\
&\qquad\qquad \tau_1 : (\dots,z,(p_1,i_2)) \in \Delta_2,\ \tau_2 : (\dots,y,(p_2,i_2)) \in \Delta_2,\\
&\qquad\qquad \text{and } \mathrm{sync}(\{\tau_1,\tau_2\}) \text{ holds}\}\\
&\quad \cup \{(\dots,x,(q_1,(i_1,i_2))),\ (\dots,z,(q_2p_1,(i_1,i_2))),\ (\dots,y,(p_2,(i_1,i_2))) \mid\\
&\qquad\qquad (q_1,q_2) \in \mathfrak{Q}_1,\ (p_1,p_2) \in \mathfrak{Q}_2,\\
&\qquad\qquad \tau_1 : (\dots,x,(q_1,i_1)) \in \Delta_1,\ \tau_2 : (\dots,z,(q_2,i_1)) \in \Delta_1,\\
&\qquad\qquad \tau'_2 : (\dots,z,(p_1,i_2)) \in \Delta_2,\ \tau'_2 : (\dots,y,(p_2,i_2)) \in \Delta_2,\\
&\qquad\qquad \text{and } \mathrm{sync}(\{\tau_1,\tau_2\}) \text{ resp. } \mathrm{sync}(\{\tau'_1,\tau'_2\}) \text{ holds}\}\,,\ \text{and}\\
\mathfrak{F}' &:= \{(q_1, q_2p_1, p_2) \mid (q_1,q_2) \in \mathfrak{F}_1,\ (p_1,p_2) \in \mathfrak{F}_2\}
\end{aligned}
$$

for all $x \in \Sigma_1 \cup \{\varepsilon\}$, $y \in \Sigma_2 \cup \{\varepsilon\}$, $z \in \Gamma \cup \{\varepsilon\}$. For a clearer view we left out the left hand sides of transitions in the last two cases of the construction. They are analogously to the other cases. Please also note, that the construction is informal with respect to the combination of given states to new macro states, because macro states have to be disjoint according to Definition 5.3. This can be easily achieved by an appropriate renaming of states.

So, we obtain a separate-asynchronous automaton for $R \circledcirc S$. The projection on the first and the third component yields $R \circ S$ which is separate-rational by Lemma B.1. □

C Construction for Lemma 4.2(d)

Word relations can be seen as a special case of tree relations (see Definition 4.1), and we have mentioned a tight relationship in Lemma 4.2. Here we will give a formal construction for part (d) of this Lemma. Part (c) is only a slightly variant of (d). Parts (a) and (b) can be proved by similar automata construction [Rad07].

Proof (of Lemma 4.2(d)). In order to prove the direction from left to right, let $R \subseteq \Sigma_1^* \times \ldots \times \Sigma_n^*$ be a rational word relation, given by an asynchronous word automaton $\mathcal{A} = \langle Q, \Sigma_1, \ldots, \Sigma_n, q_0, \Delta, F\rangle$ recognizing R. Let $Q = \{q_0, \ldots, q_m\}$. We assume w. l. o. g. that $\mathcal{A}$ can read one symbol in each component at most, i. e. Δ only consists of transitions of the form $(p, x_1/\ldots/x_n, q)$ with $x_i \in \Sigma_i \cup \{\varepsilon\}$ $(1 \leq i \leq n)$. We define the corresponding asynchronous tree automaton $\mathcal{A}^{(1)}_{\mathrm{Rat}}(R) = \langle Q', \mathfrak{Q}', \mathrm{Var}', \Sigma_1 \cup \ldots \cup \Sigma_n \cup \{f^{(n)}, \$^{(0)}\}, \Delta', \mathfrak{F}'\rangle$ with

- $Q' = (Q \times \{1, \ldots, n\}) \uplus \{q_f\}$,
- $\mathfrak{Q}' = \{(q_0^1, \ldots, q_0^n), \ldots, (q_m^1, \ldots, q_m^n)\} \uplus \{(q_f)\}$,
- $\mathfrak{F}' = \{(q_f)\}$,
- $\mathrm{Var} = \{i\}$,
- and transitions

$$\begin{aligned}\Delta' = \{&(\$, (q_j^1, i)), \ldots, (\$, (q_j^n, i)) \mid q_j \in F\}\\ &\cup \{((q_k^1, i), x_1, (q_j^1, i)), \ldots, ((q_k^n, i), x_n, (q_j^n, i)) \mid (q_j, x_1/\ldots/x_n, q_k) \in \Delta\}\\ &\cup \{((q_0^1, i), \ldots, (q_0^n, i), f, (q_f, i))\}\end{aligned}$$

  for all $x_{i'} \in \Sigma_{i'} \cup \{\varepsilon\}, 1 \leq j, k \leq m$.

Then it holds $(u_1, \ldots, u_n) \in R \iff (u_1, \ldots, u_n) \in R(\mathcal{A}) \iff f(u_1\$, \ldots, u_n\$) \in R(\mathcal{A}^{(1)}_{\mathrm{Rat}}(R))$, hence $R(\mathcal{A}^{(1)}_{\mathrm{Rat}}(R)) = \mathrm{TreeLang}(R)$.

Now we prove the other direction. Given a word relation R by a asynchronous tree automaton $\mathcal{A}^{(1)} = \langle Q, \mathfrak{Q}, \mathrm{Var}, \Sigma_1 \cup \ldots \cup \Sigma_n \cup \{f^{(n)}, \$^{(0)}\}, \Delta, \mathfrak{F}\rangle$ which defines $\mathrm{TreeLang}(R)$. We construct an asynchronous word automaton $\mathcal{B} = \langle Q', \Sigma_1, \ldots, \Sigma_n, q_0', \Delta', F'\rangle$ with

- $Q' = (Q \uplus \{\top, \bot\})^n$,
- $q_0' = (\top, \ldots, \top)$,
- $F' = \{(\bot, \ldots, \bot)\}$,
- and transitions

$$\begin{aligned}\Delta' = \{&((\top, \ldots, \top), \varepsilon/\ldots/\varepsilon, (p_1, \ldots, p_n)) \mid \text{there exist } \{q_f\} \in \mathfrak{F},\ \Delta'' \subseteq \Delta,\\ &\quad ((p_1, i_1), \ldots, (p_n, i_n), f, (q_f, i')) \in \Delta'',\ \text{so that } \mathrm{sync}(\Delta'') \text{ holds}\\ &\quad \text{(according to Definition B.2)}\}\\ &\cup \{((q_1, \ldots, q_n), x_1/\ldots/x_n, (p_1, \ldots, p_n)) \mid \text{there exist } 1 \leq j_1, \ldots, j_m \leq n,\\ &\quad \text{so that } \mathfrak{q} = (q_{j_1}, \ldots, q_{j_m}) \in \mathfrak{Q},\ \Delta'' \subseteq \Delta, \text{ and there exist transitions}\\ &\quad ((p_{j_k}, i_{j_k}), x_{j_k}, (q_{j_k}, i')) \text{ in } \Delta'' \text{ (resp. transitions } (\$, (q_{j_k}, i')) \text{ in } \Delta''\\ &\quad \text{with } x_{j_k} = \varepsilon \text{ and } p_{j_k} = \bot) \text{ for } 1 \leq k \leq m, \text{ so that all in all}\\ &\quad \mathrm{sync}(\Delta'') \text{ holds (according to Definition B.2),}\\ &\quad \text{and for all } l \in \{1, \ldots, n\} \text{ holds: } l \notin \{j_1, \ldots, j_m\} \Rightarrow (p_l = q_l \wedge x_l = \varepsilon)\}\end{aligned}$$

  for all $x_i \in \Sigma_i \cup \{\varepsilon\}$.

Then it holds $(u_1, \ldots, u_n) \in R \iff f(u_1\$, \ldots, u_n\$) \in R(\mathcal{A}^{(1)}) \iff (u_1, \ldots, u_n) \in R(\mathcal{B})$, hence R is rational. $\square$

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from `http://aib.informatik.rwth-aachen.de/`. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: “Aachen Summer School Applied IT Security”
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks
2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12	Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13	Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14	Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15	Uwe Naumann: The Complexity of Derivative Computation
2005-16	Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17	Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18	Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”
2005-19	Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20	Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21	Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22	Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23	Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24	Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-01 *	Fachgruppe Informatik: Jahresbericht 2005
2006-02	Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03	Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler
2006-04	Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation
2006-05	Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F
2006-06	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color
2006-07	Thomas Colcombet, Christof Löding: Transforming structures by set interpretations
2006-08	Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09	Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12	Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning
2006-13	Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities
2006-14	Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"
2006-15	Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices
2006-16	Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness
2006-17	Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines
2007-01 *	Fachgruppe Informatik: Jahresbericht 2006
2007-02	Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations
2007-03	Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase
2007-04	Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation
2007-05	Uwe Naumann: On Optimal DAG Reversal
2007-06	Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking