Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 223 (2008) 103–117
www.elsevier.com/locate/entcs

The Compositional Method and Regular Reachability

Ingo Felscher[1]

Chair of Computer Science 7 – Logic and Theory of Discrete Systems
RWTH Aachen University
Aachen, Germany

Abstract

The compositional method, introduced by Feferman and Vaught in 1959, allows to reduce the model-checking problem for a product structure to the model-checking problem for its factors. It applies to first-order logic, and limitations for its use have recently been revealed by Rabinovich (2007). We sharpen the results of Rabinovich by showing that the composition method is applicable to the asynchronous product (and the finitely synchronized product) for an extended modal logic in which the reachability modality is enhanced by a (semi-linear) condition on path lengths. We show that a slight extension leads to the failure of the composition theorem. We add comments on extensions of the result and open questions.

Keywords: Feferman-Vaught theorem, compositional method, modal logic, regular reachability, asynchronous product, synchronized product

1 Introduction

In verification a central question is how to decide properties of products of systems using knowledge about the components (factors).

In the area of model theory Feferman and Vaught developed in [4] a compositional method which allows to do this. They proved that the first-order (FO) theory of a generalized product of structures is reducible to the first-order theory of the component structures and the monadic theory of the index structure (over which the product is formed). Many variants have been regarded since the original result of Feferman and Vaught; a good overview can be found in [10]. Further references are [6,9,13,15].

In computer science, one can use the method in infinite-state model-checking: When the composition method works, e. g. for a finite product of structures (so that

[1] Email:felscher@automata.rwth-aachen.de

doi:10.1016/j.entcs.2008.12.034

the monadic theory of the index set is decidable), then one can transfer decidability of the model-checking problem from the factor structures to the product structure.

In this context, three aspects are characteristic. First, the focus is on special kinds of relational structures, in particular labeled transition systems (i. e. directed graphs with labels on the edges and the vertices). Second, the product constructions are different from the classical work where direct, reduced, and ultraproducts [1] were the focus – in verification various versions of synchronized products are of interest. Third, first-order logic is replaced by logical systems in which (at least certain) reachability properties are definable.

In this framework a few composition results have been obtained, but also quite severe limitations of the compositional method have been shown. An obvious example is the MSO-theory [3,16] of the successor structure $\mathcal{S} = (\mathbb{N}, \mathrm{Succ})$ of the natural numbers (proved decidable by Büchi); the asynchronous product of two copies of $\mathcal{S}$ with itself is the infinite grid which has an undecidable MSO-theory [14]. So the composition method fails (at least in an effective way). It is also known that the method fails for computation tree logic (CTL).

In the present paper we start from the fact that for modal logic (ML) over transition systems, a Feferman-Vaught theorem holds (Rabinovich [11,12]), even for certain "generalized products". This result also holds for asynchronous products when modal logic is extended by the reachability quantifier "EF" of CTL (which gives us ML(R)-logic, "R" for "reachability"). As also shown in [11], the compositional method already fails for direct products and the logic ML(R). It also fails for asynchronous products and the extension of ML by one of the CTL quantifiers "EU" or "EG".

In this paper we study another borderline of the compositional method, which also illustrates the step from "EF" and "EG". We consider (first) asynchronous products and the extension of modal logic by a reachability operator that involves regular expressions for describing path properties (through the labels of edges). For instance, the operator $\Diamond_{(ab)^*}$ expresses the existence of a path whose sequence of edge labels is in the language $(ab)^*$. We call this the extension of ML by *regular reachability* and denote it by ML(RegR). (The logic ML(RegR) is expressively equivalent to propositional dynamic logic (PDL) without tests [7].) Our main result states that a composition theorem holds for the logic ML(Reg1R), i.e. for the case of one-letter alphabets in the regular reachability operators. This amounts to the presence of counting for lengths of paths by means of semi-linear sets of natural numbers. We then show (following a construction of [11]) that already for a two letter alphabet the composition theorem fails.

We also consider several variants of the main result. First, instead of asynchronous products we consider finitely synchronized products (in which synchronization can only happen via finitely many transitions): Extending a result of Wöhrle and Thomas [17] on the logic ML(R) over finitely synchronized products, we show that compositionality also holds for ML(Reg1R). It is known that for direct (i. e. fully synchronized) products the composition theorem fails. We also give brief comments on cases that work similarly but are quite technical and will not be developed

in detail here, in particular on the case of infinite products.

The paper is structured as follows: In the subsequent section, we introduce the structures, logics, and product constructions considered in this paper. Section 3 gives an outline of the composition technique in the form needed here. In Section 4 we show the main result. Section 5 addresses related results which are not given in detail in this paper and also open questions.

2 Technical Preliminaries

The structures in this paper are labeled graphs; we speak of "transition systems".

Definition 2.1 A *transition system* is a structure $K = (S, \{R_a \mid a \in \Sigma\}, \{P_v \mid v \in V\})$, where S is a set of states, R_a is the transition relation for the letter $a \in \Sigma$ and P_v is a unary predicate for every symbol $v \in V$.

In the subsequent sections we will first define the logics we use and afterwards present the product structures, namely the asynchronous product and the (finitely) synchronized product.

2.1 Logics

The logic ML(RegR) is defined as an extension to modal logic with reachability via a path that is labeled according to a regular expression.

Definition 2.2 Let K be a transition system. For every predicate P_v, p_v is a *ML formula*. Let φ, ψ be ML formulas, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\Diamond_a\varphi$ and $\Box_a\varphi$ are ML formulas. $\Diamond_a\varphi$ ($a \in \Sigma$) means there exists an a-labeled transition to a state, where φ holds and $\Box_a\varphi$ means the same for all a-labeled transitions.

The logic *ML(RegR)* additionally contains the formulas $\Diamond_\alpha\varphi$ and $\Box_\alpha\varphi$ with $\alpha \in \mathrm{Reg}(\Sigma)$ expressing the property "there exists a path π to a state where φ holds, such that the label $\mathrm{lab}(\pi)$ of the path is a word in the language of the regular expression α", respectively the same statement for all paths. Formally we write for $\Diamond_\alpha\varphi$: $\exists\pi : \mathrm{lab}(\pi) \in L(\alpha) \wedge (K, \pi[l]) \vDash \varphi$ where l is the length of the path. The logic *ML(Reg1R)* is defined analogously with the exception that the regular expression is built only over an one-element alphabet.

We now define the same extension to first-order (FO) logic.

Definition 2.3 Let x, y be variables. For every predicate P_v ($v \in V$) and every transition relation R_a ($a \in \Sigma$), $P_v(x)$ and $R_a(x, y)$ are *FO formulas*. Let φ, ψ be FO formulas, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\exists x\,\varphi$ and $\forall x\,\varphi$ are FO formulas.

For a regular expression $\alpha \in \mathrm{Reg}(\Sigma)$ the logic *FO(RegR)* contains the additional atomic formula $\mathrm{Reach}_\alpha(x, y)$ with the meaning "from x the state y is reachable via a path that is labeled by a word in the language of the regular expression α". As above, for the logic *FO(Reg1R)* we consider only regular expressions over an one-element alphabet.

2.2 Product structures

We look at different products of transition systems, namely the asynchronous and the synchronized product. The synchronized product is defined mainly according to [17], except the fact that we use the same labels for the local transitions in all components. The abbreviation $[n]$ is used for the set $\{1, \dots, n\}$.

Definition 2.4 Let $K_1, \dots, K_n$ $(n \in \mathbb{N})$ be transition systems with $K_i = (S_i, \{R^i_a \mid a \in \Sigma_l\} \cup \{R^i_c \mid c \in \Sigma^i_s\}, \{P^i_v \mid v \in V\})$ $(i \in [n])$, where Σ_l is an alphabet for *local* transitions with asynchronous behavior in the product and Σ^i_s are alphabets which are used to define *synchronized* transitions in the product. Let Σ_s be the set $\bigcup_{i\in[n]} \Sigma^i_s$, and let all Σ^i_s and Σ_l be pairwise disjoint.
A set $C \subseteq \prod_{i:=1}^n (\Sigma^i_s \cup \{\epsilon\})$ is called a *synchronizing constraint*. It defines which transitions are going to be synchronized. A vector $c = (c_1, \dots, c_n) \in C$ means that a c-transition exists in the product, if for every $i \in [n]$ with $c_i \neq \epsilon$ a c_i-transition exists in the component K_i.
The *synchronized product* of $K_1, \dots, K_n$ is the transition system $K = (S, \{R_a \mid a \in \Sigma_l\} \cup \{R_c \mid c \in C\}, \{\overline{P^i_v} \mid v \in V\})$ with

- $S := \prod_{i=1}^n S_i$,
- $R_a := \{((s_1, \dots, s_n), (s'_1, \dots, s'_n)) \mid \exists i : (s_i, s'_i) \in R^i_a \wedge \forall j \neq i : s_i = s'_i\}$ for $a \in \Sigma_l$,
- $R_c := \{((s_1, \dots, s_n), (s'_1, \dots, s'_n)) \mid \forall i : (s_i, s'_i) \in R^i_{c_i}$, if $c_i \neq \epsilon$ and $s_i = s'_i$ if $c_i = \epsilon\}$ for $c = (c_1, \dots, c_n) \in C$ and
- $\overline{P^i_v} := \{(s_1, \dots, s_n) \mid s_i \in P^i_v\}$.

A synchronized product is called *finitely synchronized* if the set of transitions R_c is finite for every symbol $c \in C$.
An *asynchronous* product is a synchronized product with $\Sigma^i_s = \emptyset$ and $C = \emptyset$.

Remark 2.5 We will refer to a variant of the asynchronous product which is discussed in [11]: the *n-ary asynchronous product*, in which one distinguishes between the a-transitions of the different components: instead of R_a, the n-ary asynchronous product contains the relations R^i_a $(i \in [n])$ defined as $\{((s_1, \dots, s_n), (s'_1, \dots, s'_n)) \mid (s_i, s'_i) \in R^i_a \wedge \forall j \neq i : s_j = s'_j\}$.

3 Composition Method

In the proof of the composition theorem (Theorem 3.3) we will – for any sentence φ to be interpreted in a product of transition systems – inductively compute a set of formulas φ^i_k $(k \in [j], i \in [n])$ for $j \in \mathbb{N}$ which are interpreted in the components. The composition theorem holds for φ and these φ^i_k if the following equivalence is true: The sentence φ holds in the product iff for some $k \in [j]$ in every component K_i the formula φ^i_k is satisfied. Let us make this precise:

Definition 3.1 Given a product K of components $K_1, \dots, K_n$ with $K_i = (S_i, \dots)$, a formula φ which is interpreted in the product, and formulas φ^i_k $(k \in [j], i \in [n])$ for

$j \in \mathbb{N}$ which are interpreted in the components. Let $s = (s_1, \ldots, s_n) \in S_1 \times \cdots \times S_n$ and $D := \{\varphi_1, \ldots, \varphi_j\}$ with $\varphi_k := \{\varphi_k^1, \ldots, \varphi_k^n\}$.
We say that φ_k is *satisfied* at s iff for all $i \in [n]$ $(K_i, s_i) \vDash \varphi_k^i$ and that D is satisfied at s iff for one $k \in [j]$ φ_k is satisfied.
We say that satisfaction of φ in (K, s) is *reducible* to the factor structures, if a set D exists, such that D is satisfied iff $(K, s) \vDash \varphi$, and write D_φ to indicate this.

Notice that D_φ can be seen as a set representation of a "disjunctive normal form of formulas, interpreted in the components $K_1, \ldots, K_n$". We will call D_φ the *component DNF* for the formula φ. Analogously, we will use the term K_φ with $K_\varphi := \{\bar\varphi_1, \ldots, \bar\varphi_j\}$ and $\bar\varphi_k := \{\bar\varphi_k^1, \ldots, \bar\varphi_k^n\}$ iff for all $k \in [j]$ there exists an $i \in [n]$, such that $(K_i, s_i) \vDash \varphi_k^i$. K_φ will be called *component KNF*.

A conversion between component KNF and component DNF is necessary for handling negation. It works like the normal conversion between KNF and DNF. However afterwards, one has to separate the formulas in the different components and potentially add formulas with the logical value "true" or "false" for some components, such that the format of a component DNF/KNF can be obtained. This is done as follows:

Theorem 3.2 *For every formula φ the component DNF D_φ can be converted into an equivalent component KNF K_φ and vice versa.*

Proof. Given the component DNF $D_\varphi = \{\varphi_1, \ldots, \varphi_j\}$ with $\varphi_k := \{\varphi_k^1, \ldots, \varphi_k^n\}$, we consider $\alpha_\varphi := (\varphi_1^1 \wedge \cdots \wedge \varphi_1^n) \vee \cdots \vee (\varphi_j^1 \wedge \cdots \wedge \varphi_j^n)$. With $N := [n]$ the conjunctive normal form[2] of α_φ is $\bigwedge_{(k_1,\ldots,k_j)\in N^j} (\varphi_1^{k_1} \vee \varphi_2^{k_2} \vee \cdots \vee \varphi_j^{k_j})$.
Let $h : \{1, \ldots, n^j\} \to N^j$ be a bijective mapping that assigns to every index k exactly one tuple $(k_1, \ldots, k_j)$ and let Z_k^i be the set $\{z \in [j] \mid h(k) = (k_1, \ldots, k_j) \wedge k_z = i\}$. We can then define α_ψ by $\alpha_\psi = \psi_1 \wedge \cdots \wedge \psi_{n^j}$ with $\psi_k = \psi_k^1 \vee \cdots \vee \psi_k^n$ for $k \in [n^j]$ where ψ_k^i is defined by:

$$\psi_k^i := \begin{cases} \bigvee_{x \in Z_k} \varphi_x^{k_x} & \text{if } Z_k \neq \emptyset \\ ff^i & \text{otherwise}^{3}. \end{cases}$$

The formula ψ_k^i combines all formulas that are interpreted in the i-th component. The component KNF is then $K_\varphi = \{\psi_1, \ldots, \psi_{n^j}\}$ with $\psi_k = \{\psi_k^1, \ldots, \psi_k^n\}$. The conversion of K_φ to D_φ is analogous. □

These preliminaries are used in the proof of the (known) composition theorem for modal logic and synchronized products.

Theorem 3.3 *(Composition theorem for synchronized products and ML)*

[2] if we see the φ_m^i as predicates
[3] The formulas tt^i and ff^i stand for statements that are interpreted in the component K_i and are always evaluated to true respectively to false.

Given the synchronized product $K = (S, \{R_a \mid a \in \Sigma_l\} \cup \{R_c \mid c \in C\}, \{\overline{P_v^i} \mid v \in V\})$ *of transition systems* $K_1, \ldots, K_n$ *with* $K_i = (S_i, \{R_a^i \mid a \in \Sigma_l\} \cup \{R_c^i \mid c \in \Sigma_s^i\}, \{P_v^i \mid v \in V\})$ *and the initial state* $s = (s_1, \ldots, s_n)$.
For each sentence φ *of ML which is interpreted in the product, there exists an (effectively computable) set of auxiliary formulas* φ_k^i $(k \in [j], i \in [n]$ *for a* $j \in \mathbb{N})$ *which are interpreted in the components, such that*

$$(K, s) \vDash \varphi \Leftrightarrow \exists k \in [j]\, \forall i \in [n] :\ (K_i, s_i) \vDash \varphi_k^i.$$

Proof. The proof of this theorem is an extended version of the proof in [11] for n-ary asynchronous products; it is done by structural induction. For any atomic formula $\overline{p_v^i}$ one uses the formulas φ^l $(l \in [n])$ with $\varphi^i = p_v^i$ and $\varphi^j = tt^j$ for $j \neq i$.
For the induction step, we assume that component DNFs for the subformulas ψ and θ are given and we construct the component DNF for φ for the cases[4] $\neg\psi$, $\psi \vee \theta$ and $\Diamond_a\psi$ $(a \in \Sigma_l \cup C)$:

- $\varphi = \neg\psi$: Let $D_\psi = \{\psi_1, \ldots, \psi_j\}$ with $\psi_k = \{\psi_k^1, \ldots, \psi_k^n\}$ be the given component DNF for ψ. We immediately get the component KNF $K_\varphi = \{\varphi_1, \ldots, \varphi_j\}$ with $\varphi_k = \{\neg\psi_k^1, \ldots, \neg\psi_k^n\}$ which can be converted[5] into a component DNF as mentioned above.
- $\varphi = \psi \vee \theta$: Let D_ψ and D_θ be the component DNFs for ψ and θ. We get $D_\psi \cup D_\theta$ for D_φ.
- $\varphi = \Diamond_a\psi$ $(a \in \Sigma_l)$: Given the component DNF $D_\psi = \{\psi_1, \ldots, \psi_j\}$ with $\psi_k = \{\psi_k^1, \ldots, \psi_k^n\}$ we will compute the component DNF for φ using the auxiliary sets D_φ^i for $i \in [n]$. D_φ^i is defined as $\{\varphi_{1;i}, \ldots, \varphi_{j;i}\}$ with $\varphi_{k;i} = \{\psi_k^1, \ldots, \psi_k^{i-1}, \Diamond_a^i\psi_k^i, \psi_k^{i+1}, \ldots, \psi_k^n\}$ for $k \in [j]$. D_φ^i expresses that in the i-th component of the product there exists an a-labeled transition to a state where ψ holds. D_φ is the disjunction over all these possibilities, so it can be defined as $D_\varphi = \bigcup_{i\in[n]} D_\varphi^i$.[6]
- $\varphi = \Diamond_c\psi$ $(c = (c_1, \ldots, c_n) \in C)$: Let $D_\psi = \{\psi_1, \ldots, \psi_j\}$ with $\psi_k = \{\psi_k^1, \ldots, \psi_k^n\}$ be the given component DNF for ψ, then the resulting component DNF for φ is $D_\varphi = \{\varphi_1, \ldots, \varphi_j\}$ with $\varphi_k = \{\varphi_k^1, \ldots, \varphi_k^n\}$ where $\varphi_k^i = \Diamond_{c_i}\psi_k^i$ if $c_i \neq \epsilon$ and $\varphi_k^i = \psi_k^i$ if $c_i = \epsilon$.

□

4 Main Result

In this section we study the composition theorem for the extension of modal logic by regular reachability. We treat the cases of asynchronous products, respectively

[4] We have to consider only these cases, because $\psi \wedge \theta = \neg(\neg\psi \vee \neg\theta)$ and $\Box_a\psi = \neg(\Diamond_a\neg\psi)$. Note that for complexity reasons it is better for the formula $\psi \wedge \theta$ to use the mechanism to multiply out the formulas with are represented by the according component DNFs.

[5] Note that the conversion between component KNF and DNF exponentially increases the size of the component DNF because of the underlying conversion between DNF and KNF.

[6] This construction uses the fact that $\Diamond_a\psi$ in the asynchronous product can be seen as $\Diamond_a^1\psi \vee \cdots \vee \Diamond_a^n\psi$ in the n-ary asynchronous product.

finitely synchronized products in subsections 4.1 and 4.2. Finally, we conclude with a generalization to FO-logic with regular reachability.

4.1 Asynchronous product

We prove that the composition theorem can be used for asynchronous products and the logic ML(Reg1R). Moreover, we show that it fails if the extension of modal logic is considered, where we allow reachability via a regular path property (according to a regular expression over an alphabet with two symbols).

Theorem 4.1 *(Composition theorem for asynchronous products and the logic ML(Reg1R))*
Given the asynchronous product $K = (S, \{R_a \mid a \in \Sigma_l\}, \{\overline{P_v^i} \mid v \in V\})$ *of transition systems* $K_1, \ldots, K_n$ *with* $K_i = (S_i, \{R_a^i \mid a \in \Sigma_l^i\}, \{P_v^i \mid v \in V\})$ *and the initial state* $s = (s_1, \ldots, s_n)$.
For each sentence φ *of ML(Reg1R) which is interpreted in the product, there exists an (effectively computable) set of auxiliary formulas* φ_k^i $(k \in [j], i \in [n]$ *for a* $j \in \mathbb{N})$ *of ML(Reg1R) which are interpreted in the components, such that*

$$(K, s) \vDash \varphi \Leftrightarrow \exists k \in [j]\, \forall i \in [n] :\ (K_i, s_i) \vDash \varphi_k^i.$$

For the proof of Theorem 4.1 we need a possibility to reduce a formula $\Diamond_\alpha \varphi$ with $\alpha \in \mathrm{Reg}(\{a\})$ to formulas in the components. Therefore, we will first show that it is sufficient to handle only the case of regular expressions $\alpha = \beta^*$ for $\beta = a^k$ with $k \in \mathbb{N}$ by using semi-linear sets. In the second step we will reduce a formula $\Diamond_{(a^k)^*}\varphi$ – which means that the length of the path is divisible by k – to formulas in the components.

Lemma 4.2 *For every regular expression* $\alpha \in \mathrm{Reg}(\{a\})$ *there exists an equivalent regular expression* $\bar{\alpha}$*, such that for all parts* β *of* $\bar{\alpha}$ *of the form* $\beta = \gamma^*$ *it holds that* $\gamma = a^k$ *for a* $k \in \mathbb{N}$.

Proof. We identify the word a^k with the number k. As is well-known [8, chap. 6.9] the language L defined by α is semi-linear, i. e. of the form $\psi(L) = \bigcup_{i\in[m]} M_i$ for a $m \in \mathbb{N}$ where $M_i = \{k_{i0} + n_{i1}k_{i1} + \cdots + n_{ir_i}k_{ir_i} \mid n_{ij} \geq 0 \text{ for } 1 \leq j \leq r_i\}$ with fixed $k_{ij} \in \mathbb{N}$. Thus α is equivalent to the regular expression $\bar{\alpha} = \alpha_1 + \cdots + \alpha_m$ where $\alpha_i = a^{k_{i0}}(a^{k_{i1}})^* \cdots (a^{k_{ir_i}})^*$. □

Remark 4.3 We can use the following algorithm to translate α to $t(\alpha)$ with the requirements of $\bar{\alpha}$ from lemma 4.2.

- $\alpha = \epsilon$: $t[\alpha] := \epsilon$
- $\alpha = a^k$ $(k \geq 1)$: $t[\alpha] := a^k$
- $\alpha = \alpha_1 + \alpha_2$ $(\alpha_1, \alpha_2 \neq \epsilon)$: $t(\alpha) := t[\alpha_1] + t[\alpha_2]$
- $\alpha = \alpha_1 \cdot \alpha_2$ $(\alpha_1, \alpha_2 \neq \epsilon)$: $t(\alpha) := t[\alpha_1] \cdot t[\alpha_2]$
- $\alpha = (\alpha_1 \cdot \alpha_2 \cdots \alpha_j)^*$ $(j \geq 1)$
  $\cdot\ \forall i \in [j]\ \alpha_i = a^{k_i} \Rightarrow t[\alpha] := (a^{k_1 + \cdots + k_j})^*$

- $\exists i \in [j]\ \alpha_i = \beta^*\ (\beta \neq \epsilon) \Rightarrow t[\alpha] := t[(\alpha_1 \cdots \alpha_{i-1} \cdot \alpha_{i+1} \cdots \alpha_j)^*] \cdot t[\beta^*]$ [7]
- $\exists i \in [j]\ \alpha_i = (\beta_1 + \beta_2)\ (\beta_1, \beta_2 \neq \epsilon) \Rightarrow t[\alpha] := t[(\alpha_1 \cdots \alpha_{i-1} \cdot \beta_1 \cdot \alpha_{i+1} \cdots \alpha_j)^*] \cdot t[(\alpha_1 \cdots \alpha_{i-1} \cdot \beta_2 \cdot \alpha_{i+1} \cdots \alpha_j)^*]$

It remains to check formulas of the type $\Diamond_{(a^k)^*}\varphi$, meaning that the path length is divisible by k. Intuitively, if we have e. g. two components K_1, K_2, the reason why the path length in the product is divisible by 3 can be that in both components the parts of the path are divisible by 3 without remainders or that the parts are divisible by 3 with remainders $(1,2)$ or $(2,1)$ in the components (K_1, K_2).

With these preparations we can now prove Theorem 4.1.

Proof. The atomic cases and the cases $\neg\psi$, $\psi \vee \theta$ and $\Diamond_a\psi$ are treated like in the proof of Theorem 3.3; for the case $\varphi = \Diamond_{(a^k)^*}\psi$ note that φ expresses that there exists a path π to a state s' where ψ holds, such that the length of π is l with $l = 0 \pmod k$ and $l \geq 0$. The label of this path is a^l. For these l a-labeled transitions exist different distributions over the components; let $\overline{l_i}$ denote the number of a-transitions chosen in the component K_i.
For the existence of this path $\sum_{i=1}^n \overline{l_i} = 0 \pmod k$ must hold for any distribution $(\overline{l_1}, \ldots, \overline{l_n})$ of a-transitions over the components $K_1, \ldots, K_n$. $\overline{l_i}$ is representable as $\overline{l_i} = l_i + t_i * k$ with $l_i \in \{0, \ldots, k-1\}$ and $t_i \in \mathbb{N}$. With this we also get $\sum_{i=1}^n l_i = 0 \pmod k$.
We can now express for fixed values of the l_i the distribution $(l_1 + t_1 * k, \ldots, l_n + t_n * k)$ by the tuple of regular expressions $(a^{l_1}(a^k)^*, \ldots, a^{l_n}(a^k)^*)$. Because of $l_i \in \{0, \ldots, k-1\}$, there exist k^n of such tuples.
Let $D_\psi = \{\psi_1, \ldots, \psi_j\}$ with $\psi_k = \{\psi_k^1, \ldots, \psi_k^n\}$. With the observation above we get $D_\varphi = \bigcup_{k \in [j]} D_{\varphi;k}$ with $D_{\varphi;k} = \{\{\Diamond_{a^{l_1}(a^k)^*}\psi_k^1, \ldots, \Diamond_{a^{l_n}(a^k)^*}\psi_k^n\} \mid \forall i \in [n] : l_i \in \{0, \ldots, k-1\} \wedge \sum_{i=1}^n l_i = 0 \pmod k\}$. □

The same proof covers the case that the reachability modality is used with a semi-linear constraint on the length of paths. Formally, this case is captured in the following way: For $\Sigma' \subseteq \Sigma$ we introduce a new label b such that $s \xrightarrow{b} s'$, if there exists an $a \in \Sigma'$ such that $s \xrightarrow{a} s'$.

From Theorem 4.1 we immediately obtain the following corollary on the decidability of the model checking problem in the product.

Corollary 4.4 *If the ML(Reg1R) model checking problem is decidable for each of the transition graphs $K_i = (S_i, \{R_a^i \mid a \in \Sigma_l\}, \{P_v^i \mid v \in V\})$ then it is decidable for the asynchronous product $K = (S, \{R_a \mid a \in \Sigma_l\}, \{P_v^i \mid v \in V\})$.*

For each of the composition theorems shown below, a corresponding corollary on decidability of model-checking over products can be inferred. We do not state these counterparts to Corollary 4.4 explicitly.

Let us now verify that the result cannot be strengthened to cover 2-letter alphabets. Therefore we first present a schema – developed in [11] – that can be used

[7] This is possible by the commutativity of regular expressions over the one-element alphabet and $L((\beta^*)^*) = L(\beta^*)$.

to prove that the composition theorem fails for a logic which can express a specific formula. It will be used afterwards to show that the composition theorem fails for the logic ML(RegR) and asynchronous products.

Lemma 4.5 *Given a formula ψ and two infinite sets of transition systems $\{C_k | k \in \mathbb{N}\}$ and $\{D_l | l \in \mathbb{N}\}$ with state sets S_k^C and S_l^D, a common state $s \in S_k^C \cap S_l^D$ and a product specification for $C_k \times D_l$ for all $k, l \in \mathbb{N}$. The composition theorem fails if the following properties hold: $\forall k \in \mathbb{N} : (C_k \times D_k, (s,s)) \vDash \psi$ and $\forall k, l \in \mathbb{N}, k \neq l : (C_k \times D_l, (s,s)) \nvDash \psi$.*

Proof. (essentially [11]) The proof is done by contradiction. Let $k, l \in \mathbb{N}$. We assume that the composition theorem holds for a logic which can express ψ. According to the composition theorem, for every sentence φ which is interpreted in the product $C_k \times D_l$ there exists a $j \in \mathbb{N}$ and formulas $\varphi_1^1, \varphi_1^2, \ldots, \varphi_j^1, \varphi_j^2$, interpreted in the components, such that

$$(C_k \times D_l, (s,s)) \vDash \varphi \Leftrightarrow \exists m \in [j] : (C_k, s) \vDash \varphi_m^1 \wedge (D_l, s) \vDash \varphi_m^2$$

We define an equivalence relation $\sim$ on $\{C_k \mid k \in \mathbb{N}\}$ which expresses that two transition systems are equivalent, if they satisfy the same formulas of $\{\varphi_m^1 \mid m \in [j]\}$.

$$C_k \sim C_l \Leftrightarrow (\forall m \in [j] : (C_k, s) \vDash \varphi_m^1 \Leftrightarrow (C_l, s) \vDash \varphi_m^1)$$

The set $\{\varphi_m^1 \mid m \in [j]\}$ is finite, so we get a partition of $\{C_k \mid k \in \mathbb{N}\}$ into finitely many equivalence classes. There must be at least one equivalence class which contains at least two (in fact infinitely many) transition systems, because $\{C_k \mid k \in \mathbb{N}\}$ is infinite. We now call these two classes C_k and C_l.
By the precondition $(C_l \times D_l, (s,s)) \vDash \psi$ holds and by the definition of the equivalence relation $(C_k, s) \vDash \varphi_m^1 \Leftrightarrow (C_l, s) \vDash \varphi_m^1$ holds, from which we get $(C_k \times D_l, (s,s)) \vDash \psi$ which is a contradiction to the assumption. □

Theorem 4.6
The composition theorem (Theorem 4.1) fails for the logic ML(RegR).

We will prove that the composition theorem already fails for a formula which expresses that there exists a path which is labeled alternatively with a and b: we use the formula $\Diamond_{(ab)^*}\varphi$.

Proof. Given two sets of transition systems $\{C_k \mid k \geq 2\}$ and $\{D_l \mid l \geq 2\}$ where C_k is defined as (S_k, R_a, R_b, Q) with $S_k := [k]$, $R_a = \emptyset$, R_b the successor relation, and $Q = \{k\}$. The transition system D_l has the same format with R_a and R_b switched. Let φ be the formula $q^1 \wedge q^2$ which holds only at the "last" state (k,l) of the asynchronous product $C_k \times D_l$ which is indicated by the filled states in figure 1. For $k = l$ the path $(1,1) \xrightarrow{a} (1,2) \xrightarrow{b} (2,2) \xrightarrow{a} (2,3) \xrightarrow{b} \cdots \xrightarrow{a} (k-1,k) \xrightarrow{b} (k,k)$ exists, whose label is in the language $L((ab)^*)$, so $(C_k \times D_k, (1,1)) \vDash \Diamond_{(ab)^*}\varphi$ holds. For $k = 4$ this situation is shown on the left side of the figure 1. For the case $k < l$ the only path starting at $(1,1)$ which is labeled alternating with a and b can only be

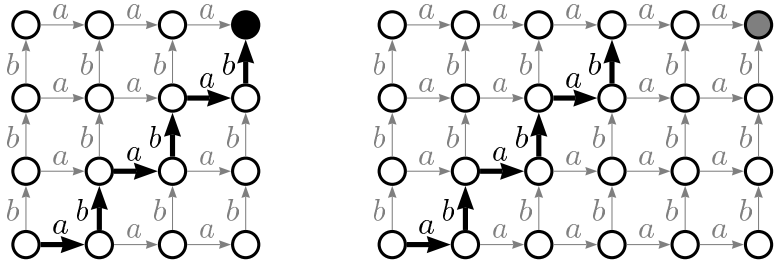

Fig. 1. The products $C_4 \times D_4$, $C_4 \times D_6$

extended by a's from the state (k,k) onwards, so $(C_k \times D_l, (1,1)) \not\models \Diamond_{(ab)^*}\varphi$. This is shown in the figure 1 for $k = 4$ and $l = 6$. The case $k > l$ is analogous.
So we meet the requirements for the lemma 4.5. As consequence the composition theorem fails for asynchronous products and the logic ML(RegR). □

4.2 Finitely synchronized product

We now show that Theorem 4.1 extends from asynchronous to finitely synchronized products (in the sense of [17]). Afterwards, we prove that the composition theorem also works for the logic FO(Reg1R), again for finitely synchronized products.

Theorem 4.7 *(Composition theorem for finitely synchronized products and the logic ML(Reg1R))*
Given the finitely synchronized product K of transition systems $K_1, \ldots, K_n$ from the definition 2.4 and the initial state $s = (s_1, \ldots, s_n)$.
For each sentence φ of ML(Reg1R) which is interpreted in the product, there exists an (effectively computable) set of auxiliary formulas φ^i_k ($k \in [j], i \in [n]$ for a $j \in \mathbb{N}$) of ML(Reg1R) which are interpreted in the components, such that

$$(K,s) \vDash \varphi \Leftrightarrow \exists k \in [j]\, \forall i \in [n] :\ (K_i, s_i) \vDash \varphi^i_k.$$

Proof. As the local and synchronized transitions are disjoint we only have the two cases $\alpha \in Reg(\{a\})$ for $a \in \Sigma_l$ respectively $a \in C$.

- For $a \in \Sigma_l$ we can restrict the transition structure of the product to the asynchronous transitions and treat the formula $\Diamond_\alpha \psi$ like above.
- For $a \in C$ we restrict the transition structure to the synchronized transitions. Since there are only finitely many transitions there are only finitely many reachable states. For every regular expression β^i, let $S(\beta^i, s)$ be the set of reachable states from the state s. For every β there exists a $k \in \mathbb{N}$, such that $S(\beta^k, s) = S(\beta^{k+1}, s)$, so β^* is equivalent to $\beta^0 \vee \beta^1 \vee \beta^2 \vee \cdots \vee \beta^k$.

□

We can strengthen Theorem 4.7 to the more interesting case that the counting extends over a full path, including the local and the synchronized transitions.

Theorem 4.8 *The composition theorem (Theorem 4.7) holds for finitely synchronized products and modal logic which is extended by reachability via paths where the length of the path is required to be a semi-linear set L.*

Proof. First we treat the typical case where the path length is required to be divisible by k with remainder r.
Let $\bigcup_{c\in C} R_c = \{r_1, \ldots, r_p\}$ be the set of synchronized edges. The theorem will be proven in three steps:

(i) We will inductively construct semi-linear sets which describe all paths using the synchronized edges $r_1, \ldots, r_m (m \leq p)$. This decomposition follows the pattern of Kleene's Theorem.

(ii) We show that for the calculation of the path lengths modulo a number k it is sufficient to use every synchronized edge only at most k times.

(iii) We use the idea of (i) together with the limitation (ii) to construct sets for each component and describe the distribution of the asynchronous and synchronized transitions over the components such that the length of the path in the product is divisible by k with a remainder r.

(i) Let s_m and t_m be the states of the edge $s_m \xrightarrow{r_m} t_m$ for $m \in [p]$, let z_0 be the initial state and z_f be a dummy state, which will only be used for notational ease. We inductively construct sets $L^m_{i,j}$ for $i, j \in \{s_1, t_1, \ldots, s_p, t_p, z_0, z_f\}$ which express "from i there exists a path to the state j using at most the synchronized transitions $r_1, \ldots, r_m$ such that the path is in the semi-linear set $L^m_{i,j}$". Finally we get $L^p_{z_0,z_f}$ which uses at most all synchronized transitions.
Let a be a new symbol such that $s \xrightarrow{a} s'$, if $s \xrightarrow{a'} s'$ for an $a' \in \Sigma_l$ and let $\alpha_{i,j}$ be a regular expression over the alphabet $\{a\}$.
Induction start ($m = 0$):

- $L^0_{i,j} = \alpha_{i,j}$ for $i \neq j$
- $L^0_{i,i} = \epsilon$
- $L_{s,t} = c$ for $s \xrightarrow{c} t$ for $c \in C$

Induction step ($m - 1 \to m$):

- $L^m_{i,j} = L^{m-1}_{i,j} \cup L^{m-1}_{i,s_m} \cdot L_{s_m,t_m} \cdot (L^{m-1}_{t_m,s_m} \cdot L_{s_m,t_m})^* \cdot L^{m-1}_{t_m,j}$
  For $m = 1$ the second part of the semi-linear set $L^m_{i,j}$ describes the path $i \rightsquigarrow s_m \to t_m (\rightsquigarrow s_m \to t_m)^* \to j$, where the arrow $\rightsquigarrow$ indicates reachability by asynchronous transitions. Note that $L^m_{i,j}$ is semi-linear, because the union $L_1 \cup L_2$, the concatenation $L_1 \cdot L_2$ and the Kleene star L_1^* of semi-linear images L_1, L_2 are semi-linear [8].

(ii) Note that it is sufficient to pass through each synchronized transition a finite number of times: For each synchronized transition we get all possible path lengths modulo k after passing through the transition at most k times. The worst case occurs if k is a prime number, we want to get a path length divisible by k with remainder 0 and for the synchronized transition $s \xrightarrow{c} t$ there exists only one path

from t back to s, which has the length k. Then we will have to use this path including the transition $s \xrightarrow{c} t$ k-times to get a path length divisible by k. So we can replace the star in $L^m_{i,j}$ by a disjunction of the repetitions up to k-times.
(iii) To build statements in the components, we use the following ideas:

- We guarantee that the corresponding parts $(c_1, \ldots, c_n)$ of the synchronized transition c are used in the same sequence in all components.
- We allow everywhere in between asynchronous transitions.
- We check that the number of the asynchronous transitions used in all components together with the number of synchronized transitions is divisible by k with remainder r.

Therefore we inductively define for each component K_h $(h \in [n])$ a function $M_{i,j}(J, \overline{X}(h))$ where $\overline{X}(h) = (X_{s_1,t_1}(h), \ldots, X_{z_0,z_f}(h))$ is a tuple of variables and $J = (j_1, \ldots, j_m)$ with $0 \leq j_g \leq k$ $(g \in [m])$ a tuple of indices for the repetitions of the loops of the synchronized transitions $r_1, \ldots, r_m$ $(m \leq p)$. The sets $M_{i,j}(J, \overline{X}(h))$ are defined analogously to the sets $L^m_{i,j}$.
Induction start:

- $M_{i,j}(\overline{X}(h), \emptyset) = X_{i,j}(h)$ for $i \neq j$
- $M_{i,i}(\overline{X}(h), \emptyset) = \epsilon$
- $M_{s,t}(h) = c_h$ for $s \xrightarrow{c} t$ for $c = (c_1, \ldots, c_h, \ldots, c_n) \in C$

Induction step $(m - 1 \to m)$:

- Let $J = (j_1, \ldots, j_m)$ and $J' = (j_1, \ldots, j_{m-1})$

$$M_{i,j}(\overline{X}(h), J) = \begin{cases} M_{i,j}(\overline{X}(h), J'), \text{ if } j_m = 0 \\ M_{i,s_m}(\overline{X}(h), J') \cdot M_{s_m,t_m}(h) \cdot [M_{t_m,s_m}(\overline{X}(h), J') \cdot M_{s_m,t_m}(h)]^{j_m - 1} \cdot \\ \qquad M_{t_m,j}(\overline{X}(h), J'), \text{ if } 1 \leq j_m \leq k \end{cases}$$

Now we can build a disjunction over all possible tuples $(Y_1, \ldots, Y_n)$ with $Y_h = M_{z_0,z_f}(J, \overline{X}(h))$ $(h \in [n])$ where $J = (j_1, \ldots, j_p)$ with $0 \leq j_g \leq k$ $(g \in [p])$ and all solutions for $\overline{X}(h)$ of the form $X_{i,j}(h) = a^{l_{i,j,h}}(a^k)^*$ $(0 \leq l_{i,j,h} < k)$ such that $\sum_{h \in [n]} \left| M_{z_0,z_f}(J, \overline{X}(h)) \right| = r \pmod k$ where $|M|$ denotes the length modulo k of the paths described by M. Note that for a fixed J there are only finitely many of such solutions and there are only finitely many J.
The general case where the length of the path shall be in a semi-linear set can be treated in the following way: Let L be a semi-linear set over an one-element alphabet. Then the semi-linear image $\psi(L)$ has the form $\bigcup_{i \in [u]} M_i$ for a $u \in \mathbb{N}$ where $M_i = \{k_{i0} + n_{i1}k_{i1} + \cdots + n_{it_i}k_{it_i} \mid n_{ij} \geq 0 \text{ for } 1 \leq j \leq t_i\}$ with fixed $k_{ij} \in \mathbb{N}$. For each of these sets M_i we define "modmin $\{k_{i1}, \ldots, k_{it_i}\}$" by: $x = r$ (modmin $\{k_{i1}, \ldots, k_{it_i}\}$) $:\Leftrightarrow x = \min\{r_s \mid x = r_s \pmod{k_s}, k_s \in \{k_{i1}, \ldots, k_{it_i}\}\}$. Note that modmin $\{k_{i1}, \ldots, k_{it_i}\}$ calculates the value of k_{i0}. The general case can now be solved for one M_i if we replace everywhere in the proof above "mod k" by "modmin $\{k_1, \ldots, k_{t_i}\}$". □

Before we generalize the result to FO-logic with regular reachability over an one-element alphabet, let us mention that Wöhrle and Thomas already treated in [17] the cases of asynchronous and finitely synchronized products for the logics FO(R) and FO(RegR) – in the first case, compositionality holds, in the second it fails. For reachability they used an extension of FO-logic by new atomic formulas $\mathrm{Reach}_{\Sigma'}(x, y)$ meaning that there is a path from x to y, carrying any letters of the alphabet $\Sigma' \subseteq \Sigma$.

We now adapt the proof of Theorem 4.8 to the logic FO(Reg1R), respectively FO-logic extended by reachability via paths where the length of the path is required to be a semi-linear set L. Here we admit atomic formulas of the form $\mathrm{Reach}_\alpha(x, y)$ where α is a regular expression over $\{b\}$ and b is a new edge label such that $s \xrightarrow{b} t$ if there exists an $a \in \Sigma' \subseteq \Sigma_l \cup C$ with $s \xrightarrow{a} t$. We obtain:

Theorem 4.9 *(Composition theorem for finitely synchronized products and the logic FO(Reg1R))*
Given the finitely synchronized product K of transition systems $K_1, \ldots, K_n$ from the definition 2.4.
For each sentence φ of FO(Reg1R) which is interpreted in the product, there exists an (effectively computable) set of auxiliary formulas φ^i_k ($k \in [j], i \in [n]$ for a $j \in \mathbb{N}$) of FO(Reg1R) which are interpreted in the components, such that

$$K \vDash \varphi \Leftrightarrow \exists k \in [j]\, \forall i \in [n] :\ K_i \vDash \varphi^i_k.$$

The analogous statement holds for finitely synchronized products and FO-logic which is extended by reachability via paths where the length of the path is required to be a semi-linear set L.

Proof. The (inductive) proof is analogous to the corresponding proofs for finitely synchronized products. The atomic cases of predicates and transition relations and the cases of the induction step (the existential quantifier, the negation, and the disjunction) are treated in the standard way. Let $\bar{x} = (x_1, \ldots, x_n)$, $\bar{y} = (y_1, \ldots, y_n)$ be variables which are interpreted in the product, let $a \in \Sigma_l$ and $c = (c_1, \ldots, c_n) \in C$ be transition labels of an asynchronous respectively a synchronized transition of the product, then the corresponding sets D_φ for the formulas φ are built as follows:

- $\varphi = (\bar{x} = \bar{y})$: The set D_φ is constructed as $\{\{\varphi^1, \ldots, \varphi^n\}\}$ with $\varphi^i = (x_i = y_i)$ for $i \in [n]$.
- $\varphi = P^k_v(\bar{x})$: The set D_φ is constructed as $\{\{\varphi^1, \ldots, \varphi^n\}\}$ with $\varphi^k = P^k_v(x_k)$ and $\varphi^i = tt^i$ for $i \in [n] \backslash \{k\}$.
- $\varphi = R_a(\bar{x}, \bar{y})$: We construct $D_\varphi = \{\varphi_1, \ldots, \varphi_n\}$ with $\varphi_j = \{\varphi^1_j, \ldots, \varphi^n_j\}$ for $j \in [n]$ where $\varphi^j_j = R_a(x_j, y_j)$ and $\varphi^i_j = (x_i = y_i)$ for $i \in [n] \backslash \{j\}$.
- $\varphi = R_c(\bar{x}, \bar{y})$: The set D_φ is constructed as $\{\{\varphi^1, \ldots, \varphi^n\}\}$ with $\varphi^i = R_{c_i}(x_i, y_i)$ if $c_i \neq \epsilon$ and $\varphi^i = (x_i = y_i)$ otherwise.
- $\varphi = \neg\psi$, $\varphi = \psi \vee \theta$: These cases are the same as in the composition theorem (Theorem 3.3).

- $\varphi = \exists \bar{x}\, \psi(\bar{x}_1, \ldots, \bar{x}_l, \bar{x})$: Given $D_\psi = \{\psi_1, \ldots, \psi_m\}$ with $\psi_j = \{\psi_j^1, \ldots, \psi_j^n\}$ for $j \in [m]$, the set D_φ is constructed as $\{\varphi_1, \ldots, \varphi_m\}$ with $\varphi_j = \{\exists x_1\, \psi_j^1, \ldots, \exists x_n\, \psi_j^n\}$ for $j \in [m]$.

Apart from that we have the additional atomic case that one state is reachable by another via a path that is labeled with a word of a regular expression α over the one-element alphabet $\{b\}$. By lemma 4.2 we can assume $\alpha = (b^k)^*$.

- $\varphi = \mathrm{Reach}_\alpha(\bar{x}, \bar{y})$: For the asynchronous product the set D_φ is constructed as $\{\{\mathrm{Reach}_{b^{l_1}(b^k)^*}(x_1, y_1), \ldots, \mathrm{Reach}_{b^{l_n}(b^k)^*}(x_n, y_n)\} \mid \forall i \in [n] : l_i \in \{0, \ldots, k-1\} \wedge \sum_{i=1}^n l_i = 0 \pmod k\}$ analogously to the proof of Theorem 4.1. For the finitely synchronized product we use reachability formulas for the tuples $(Y_1, \ldots, Y_n)$ computed in the proof of Theorem 4.8 above. The proof is analogously extendable to semi-linear sets.

□

5 Further Results and Conclusion

We have shown results that exhibit new cases where the composition technique (or Feferman-Vaught technique) can be applied in infinite-state model-checking. We showed that the composition theorem holds for an extension of modal logic by reachability together with regular constraints on path lengths – i. e., semi-linear properties of path lengths can be built into the reachability operator as an addendum to modal logic. Already the step to two-letter alphabets leads to a failure of the composition theorem. We generalized the results to FO-logic with regular reachability.

We add some remarks on further results and state some open questions. First, the complexity of the model-checking problem for a product structure is quite prohibitive. In [2] for FO a non-elementary lower bound was proven for the number of the auxiliary formulas. An open question is, if this also holds for the case of ML. In [5] we showed for which operators of the formulas the number of auxiliary formulas increases exponentially and that it depends not only on the alternation depth of $\Diamond$- and $\Box$-quantifiers, but also partly on the $\wedge$-/$\vee$-operators, depending on which is the next outer $\Diamond$- and $\Box$-quantifier.

Second, we mention that a more ambitious generalization of the present results which deals with the case of infinite products (with index set $\mathbb{N}$) is possible. Here the analogous statements to Theorems 4.1 on ML(Reg1R) and 4.6 on ML(RegR) hold, but for ML(Reg1R) involve quite more technical work. First, the "decomposition" of formulas can no more be done on the propositional level (as in our sets D_φ in Section 3) but on the level of MSO-logic over $\mathbb{N}$. The essential fact we use is the expressibility of semi-linear properties of natural numbers in the MSO-theory of $(\mathbb{N}, Succ)$, and that this theory is decidable.

An open question is whether other interesting fragments of the regular languages (beyond the case of one-letter alphabets) exist where a compositional approach to model checking is possible.

As another open problem we mention whether for ML the rather disappointing upper complexity bounds can be improved by more refined algorithms.

Acknowledgement

I wish to thank Wolfgang Thomas for many important suggestions and corrections. Thanks are also due to an anonymous referee for pointing out connections to PDL.

References

[1] Chang, C. C. and H. J. Keisler, *Model Theory*, "Studies in Logic And The Foundations Of Mathematics" **73** (1990), third edition, North Holland, Amsterdam

[2] Dawar, Anuj, Martin Grohe, Stephan Kreutzer and Nicole Schweikardt, *Model theory makes formulas large*, ICALP'07: 34th International Colloquium on Automata, Languages and Programming **2007**, pages 913–924, Springer-Verlag

[3] Ebbinghaus, H.-D., J. Flum and W. Thomas "Einführung in die mathematische Logik" (1996), fourth edition, Spektrum Akademischer Verlag, Heidelberg

[4] Feferman, S. and R. Vaught, *The first-order properties of products of algebraic systems*, Fundamenta Mathematicae **47** (1959), pages 57–103

[5] Felscher, Ingo, *Model-Checking über Produktstrukturen*, diploma thesis (2007), RWTH Aachen

[6] Gabbay, D.M. and V.B. Shehtman, *Products of modal logics, part 1*, Logic Journal of IGPL **6**/1 (1998), pages 73–146

[7] Harel, David, Dexter Kozen and Jerzy Tiuryn, "Dynamic Logic" (2002), MIT Press

[8] Harrison, Michael A., *Introduction to Formal Language Theory*, Series in Computer Science (1978), Addison-Wesley

[9] Hodges, Wilfrid, *Model theory*, "Encyclopedia of Mathematics and its Applications" **42** (1993), Cambridge University Press

[10] Mostowski, Andrzej, *On Direct Products of Theories*, The Journal of Symbolic Logic **17**/1 (1952), pages 1–31

[11] Rabinovich, Alexander, *On compositionality and its limitations*, ACM Transactions on Computational Logic **8**/1 (2007)

[12] Rabinovich, Alexander, *Selection and uniformization in generalized product*, Logic J. of IGPL **12**/2 (2004), pages 125–134

[13] Shelah, Saharon, *The Monadic Theory of Order*, The Annals of Mathematics **102**/3/2 (1975), pages 379–419

[14] Thomas, Wolfgang, *Automata on Infinite Objects*, In Jan van Leeuwen, editor, "Handbook of Theoretical Computer Science volume B: Formal Models and Semantics" (1990), pages 133–192

[15] Thomas, Wolfgang, *Ehrenfeucht Games, the Composition Method, and the Monadic Theory of Ordinal Words*, In Jan Mycielski and Grzegorz Rozenberg and Arto Salomaa, editors, "Structures in Logic and Computer Science, A Selection of Essays in Honor of A. Ehrenfeucht", Lecture Notes in Computer Science **1261** (1997), pages 118–143, Springer-Verlag

[16] Thomas, Wolfgang, *Languages, automata and logic*, In G. Rozenberg and A. Salomaa, editors, "Handbook of formal languages" **3** (1997), pages 389–455, Springer-Verlag

[17] Wöhrle, Stefan and Wolfgang Thomas, *Model checking synchronized products of infinite transition systems*, Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (2004), pages 2–11, Washington, DC, USA, IEEE Computer Society