Model Composition in Model-Checking

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Ingo Felscher
aus Münster

Berichter: Univ.-Prof. Dr. rer. nat. Dr. h. c. Dr. h. c. Wolfgang Thomas
Univ.-Prof. Dr. rer. nat. habil. Markus Lohrey

Tag der mündlichen Prüfung: 27. Januar 2014

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Abstract

Model-checking allows one to formally check properties of systems: these properties are modeled as logic formulas and the systems as structures like transition systems. These transition systems are often composed, i.e., they arise in form of products or sums. The composition technique allows us to deduce the truth of a formula in the composed system from "interface information": the truth of formulas for the component systems and information in which components which of these formulas hold. We are interested in identifying situations where the composition technique works in the context of model-checking (reachability properties, linear and branching time temporal logic) and, in these cases, how large the interface information can become.

In the first main part of this thesis, we extend known results for synchronized products of transition systems by showing a composition theorem for finitely synchronized products and first-order logic (or modal logic) extended by regular reachability over a unary alphabet. We further show that the composition technique fails for two generalizations of the logic: in the general case of regular reachability and in the case of propositional dynamic logic over a unary alphabet. Furthermore, it fails for synchronized products which are not finitely synchronized.

A known drawback of the composition technique is the growth of the size of the generated interface information in relation to the given formula. Göller, Jung and Lohrey have shown a non-elementary lower bound for this size in general for first-order logic (for products) and modal logic (for disjoint ordered sums). In the second and third part of this thesis, we look at combinations of logics and systems where we can avoid this. In the second part, we show a composition theorem for linear temporal logic (LTL) and disjoint ordered sums of words. A careful analysis shows that, here, the size of the interface information only grows at most exponential in the size of the given formula. In the third part, we generalize this composition technique to a disjoint ordered sum of trees and computation tree logic (CTL). Here, we deal with trees as components which are composed via a tree structure. We show a composition result for which the interface information grows exponentially in the size of the given formula and in the branching of the index tree structure.

Zusammenfassung

Model-Checking ermöglicht das formale Überprüfen von Eigenschaften von Systemen. Diese Eigenschaften werden als Logikformeln und die Systeme als Transitionssysteme modelliert. Die Transitionssysteme sind häufig zusammengesetzt als Produkte oder Summen. Die Kompositionstechnik leitet die Gültigkeit von Formeln im zusammengesetzten System aus "Interfaceinformationen" ab: der Gültigkeit von Formeln in den Komponenten des Systems sowie Informationen darüber, in welchen Komponenten welche Formel gilt. Wir interessieren uns für Situationen, in denen die Kompositionstechnik funktioniert und analysieren wie groß in diesen Situationen die Interfaceinformationen werden können.

Im ersten Teil der Arbeit erweitern wir bekannte Resultate für synchronisierte Produkte von Transitionssystemen indem wir einen Kompositionssatz für endlich synchronisierte Produkte und First-Order Logic (oder Modal Logic) erweitert mit regulärer Erreichbarkeit über einem einstelligen Alphabet zeigen. Weiter zeigen wir, dass die Kompositionstechnik für zwei Erweiterungen der Logik fehlschlägt: für reguläre Erreichbarkeit und für Propositional Dynamic Logic über einem einelementigen Alphabet. Außerdem schlägt sie für synchronisierte Produkte fehl, die nicht endlich synchronisiert sind.

Ein bekannter Nachteil der Kompositionstechnik ist das Wachstum der Größe der Interfaceinformationen in Bezug zur gegebenen Formel. Göller, Jung und Lohrey haben im Allgemeinen eine nicht-elementare untere Schranke für diese Größe gezeigt. Im zweiten und dritten Teil der Arbeit betrachten wir Kombinationen von Logiken und speziellen Systemen, bei denen wir dies verhindern können. Im zweiten Teil zeigen wir einen Kompositionssatz für Linear Temporal Logic (LTL) und disjunkte geordnete Summen von Wörtern. Eine gründliche Analyse zeigt, dass die Größe der Interfaceinformationen hier exponentiell zur Größe der gegebenen Formel wächst. Im dritten Teil verallgemeinern wir diese Kompositionstechnik auf eine disjunkte geordnete Summe von Bäumen und Computation Tree Logic (CTL). Hier ist jede Komponente ein Baum und im Allgemeinen auch die Struktur die zur Verknüpfung der Komponenten dient. Wir zeigen einen Kompositionssatz, bei dem die Interfaceinformationen exponentiell in der Länge der gegebenen Formel und in der Verzweigung der Indexbaumstruktur wachsen.

Contents

List of Figures

1 Introduction

Model-checking is a well-known formal method for hardware and software verification. It allows to formally check that a (hardware or software) systems meets given requirements. Often, these systems are composed, i.e., they consist of component systems which may work in sequence or in parallel and interact with each other. While model-checking can be performed efficiently on the small component systems, the composed system may become to large for efficient model-checking. A natural idea, called composition technique, is to reduce the model-checking problem for the composed system to model-checking problems for the components. This thesis investigates settings where this composition technique works and also shows where the boundaries for composition results are located.

In this introduction, we first give a more comprehensive overview of the model-checking background. Afterwards, we present more details on the composition technique and present our contribution to this approach. We conclude this chapter with an overview of the thesis.

As mentioned above, model-checking allows to check that a system satisfies desired properties. These properties are, e.g., deadlock freedom, invariants, reachability of some state of the system or request-response conditions. Over the last decades, formal methods for verification have become more and more important as the impact of the reliance on IT systems has grown. They not only affect daily life as, e.g., in mobile phones but also safety critical areas like medical systems, cars, planes or nuclear plants. Starting from a theoretical framework – invented by Emerson and Clarke in [CE81] and Queille and Sifakis in [QS82] – model-checking has become a practical solution to formally check the correctness of systems which is used in industry. An overview of the known results on the theoretical foundations can be found in [CGP99, CS01, Mer01, BK08].

The approach of model-checking is a model-based, i.e., we work with formal representations – models – of the system and the desired properties. As model for the system, we usually use a transition system which consists in the simplest case only of states and transitions between these states. A state encodes information like, e.g., the current value of variables and the current line in a program. A transition describes one possible continuation as time proceeds. Furthermore, the

desired properties have to be translated to a formal specification, described in a logic. Usually some kind of temporal logic is used for the specification to express correctness ("does the system do what it should?"), e.g., reachability ("is there a way to a deadlock?"), safety ("does the system never end in a bad state?"), liveness ("does the system finally end in a good state?") and fairness ("does an event occur repeatedly (under preconditions)?").

In the classical model-checking setting, transition systems have a finite number of states. A major task is the extension of model-checking to infinite transition systems. Two approaches which try to reduce the size of a transition system in general (and especially from infinite to finite) are abstraction and partial order reduction (see, e.g., Chapters 7 and 8 in [BK08]). Roughly speaking, abstraction combines equivalent states and partial order reduction reduces the number of transitions for sequences where the order of the transitions is irrelevant, i.e., the transitions are independent of each other.

We discuss here a third approach to reduce the size of a transitions system which goes back to Feferman and Vaught [FV59] and Shelah [She75] in the area of model theory. This approach has applications in computer science because large technical systems are often combined of several smaller ones. In the model-checking area, this means that we have (large) transition systems which consist of a composition of smaller ones. These components may "run" one after the other, in parallel and interact with each other. In this setting, a natural idea is to reduce the satisfaction of a formula in the composed transition system to the satisfaction of formulas in the components. This is the idea of the composition technique at which we look in this thesis.

We mainly consider here two different sorts of composition: synchronized products and disjoint ordered sums. Synchronized products allow to express concurrency and interaction of programs by asynchronous and synchronized transitions. (Asynchronous transitions are transitions in the product where only one component changes its state and for synchronized transitions in several components transitions are taken at the same time.) Disjoint ordered sums model the sequential execution of programs.

As logics for the specification, we discuss first-order logic (FO) with and without extensions, monadic second order logic (MSO) and temporal logics: linear-time temporal logic (LTL), computation tree logic (CTL) and extensions of both.

We say that we have a composition theorem if we are able to reduce the satisfaction of a formula in the product or sum of components to *interface information*. We use this term for the satisfaction of a set of *component formulas* – formulas which are interpreted in the components – and information about which of these component

formulas have to hold in which components. We are only interested in situations where we can also (algorithmically) compute the interface information. The aim of this thesis is to find settings in which a composition theorem holds.

We pose the following two questions:

1. For which logics and compositions in the form of products or sums does a composition theorem hold? To put it in another perspective: Where are the limits of the composition technique?

2. How much information about the components and their relation do we need? To be more precise: How "large" is the size of each component formula, the size of the set of these formulas and how "large" is the information which speaks about these formulas?

We now give an overview of the known results for both questions. As mentioned above the results presented here have a background in model theory. This goes back to work of Feferman and Vaught [FV59] in 1959 and Shelah [She75] in 1975. They introduced composition techniques for general products, respectively sums, of arbitrary (relational) structures (instead of only the special case of transition systems). Feferman and Vaught [FV59] showed that the satisfaction of a formula φ in a (generalized) product (or sum) can be deduced from FO formulas $\varphi_1, \ldots, \varphi_m$ for the component structures and a Boolean, respectively MSO formula $\beta(X_1, \ldots, X_m)$ interpreted in the index structure of the product (respectively sum). The formula $\beta(X_1, \ldots, X_m)$ describes in which of the components the formulas $\varphi_1, \ldots, \varphi_m$ have to hold. For the case of ordered disjoint sums of orderings, in [She75], Shelah showed a composition theorem which captures MSO formulas in the sum (using MSO logic for both the component formulas and the "index formula"). Many variants of these results have since been developed, e.g., Gurevich and Shelah found a composition theorem for sums of trees [GS85]. A good overview can be found in [Mak04]. Further references are [GS79, Gur85, GS98, Hod97, Mos52, Tho97a].

For the area of model-checking, we now discuss the two questions from above – first for products and then for sums of transition systems. For products, our focus lies on the first question.

Note that the MSO theory of the successor structure of the natural numbers is decidable but the MSO theory of the infinite binary grid – which can be seen as the asynchronous product of two copies of the natural numbers – is undecidable. Thus, the composition technique is not applicable for asynchronous products and MSO logic. The question is where the boundary for the application of the composition technique is located.

In the literature, various extensions of first-order logic (and modal logic) over transition systems have been discussed for the composition technique. We consider here the following extensions:

- reachability by a path (abbreviated as R) – "there exists a path to a node at which φ holds" (which corresponds to the CTL formula $EF\varphi$),
- "there exists a path which satisfies φ on every node" (which corresponds to the CTL formula $EG\varphi$),
- regular reachability (abbreviated as $RegR$) – "there exists a path to a node at which φ holds and the labeling sequence of this path is in a given regular language" and
- the special case of regular reachability over a unary alphabet (abbreviated as $Reg1R$) – this amounts to reachability augmented by modulo counting over the path lengths.

We now present the known results for these logics for asynchronous, synchronized and finitely-synchronized products. (The latter are synchronized products where the number of synchronous transitions in the product is finite.)

We start with asynchronous products which contain no synchronization at all. Rabinovich showed in [Rab07] a composition theorem for (multi-)modal logic extended by reachability and asynchronous[1] products. His proof also works for FO logic extended by reachability. Sadly, this result does not hold even for the simplest kind of (full) synchronization, namely the synchronous product of two components – also shown in [Rab07]. For this, Rabinovich developed a formal proof technique to show that there cannot be a composition theorem for a given logic formula and product. (By applying his technique he also showed that, even for asynchronous products, the composition technique fails for logics which can express the CTL quantifier EG.)

As synchronization transitions and reachability are crucial points in model-checking, attempts have been made to overcome the failure of the composition technique for special cases. In [WT07], Wöhrle and Thomas found a composition theorem for *finitely* synchronized products and FO(R) logic. We show here that their result can be extended to capture also regular reachability over a unary alphabet which is essentially the same as modulo counting over the path length. Furthermore, we show – using a unified proof schema developed by Rabinovich –

[1] Asynchronous products of n components are called n-products in [Rab07].

that the composition technique already fails for regular reachability[2] over a binary alphabet and also if we allow tests in the regular expressions over a unary alphabet. We look at the general setting where we allow both the components to be finite or infinite and also the number of components. A preliminary version of our results can also be found in [Fel08] and [FT09]. As a side effect we also extend the known results of [Rab07] and [WT07] from finite products of transition systems to infinite products.

Recall the two questions from page 3. With the results from above, we have a clearer view where the frontier of applicability of the composition technique lies for products. We now look at the size of the decomposition, i.e., the size of the interface information generated for a given formula by the composition technique. It is defined as the sum over size of the formulas, their number and the size of the formula for the index structure. The question of the size of the decomposition has been discussed by Dawar, Grohe, Kreutzer and Schweikardt in [DGKS07] and Göller, Jung and Lohrey in [GJL12]. In [GJL12], the authors found a non-elementary lower bound for the number of formulas in the composition theorems, even for products and modal logic, respectively for sums and first-order logic (with at least three variables). Their proof technique relies on the fact that each state of the underlying structures may have an unbounded out-degree of transitions.

The non-elementary size of the decomposition contrasts with an only exponential size of information needed in an automaton-based composition result on words. To check the acceptance of an ω-word w by a Büchi automaton $\mathcal{A}$, we observe that w must have the form $w = w_1 w_2 \dots$ with finite word segments $w_0, w_1, \dots$ which fulfill the following condition: With w_0, we get from the initial state of $\mathcal{A}$ to a final state and with each other w_i from a final back to a final state. Thus, whether $\mathcal{A}$ accepts w can be determined by the state transformations each w_i defines. This information is given by the set of all state pairs (p, q) of $\mathcal{A}$ such that $\mathcal{A}$ can go via w_i from p to q with, respectively without, passing a final state in-between. The number of possible state transformations is of order 2^{n^2} where n denotes the number of states of $\mathcal{A}$.

In this thesis, we get two similar composition results for logics (LTL, respectively CTL) instead of automata on words, respectively on trees. This means, by restricting the transition systems to words, respectively trees, and the logics from MSO to LTL, respectively CTL, we are able to overcome the non-elementary blow-up of the interface information (in relation to the size of the input formula) in the composition

[2] Note that the failure of the composition technique for regular reachability was already shown in [WT07] using a reduction from the reachability problem for universal pushdown automata with two stacks to the model-checking problem for FO(RegR).

technique. We get a composition theorem with a decomposition which has an exponential size for both cases – LTL on words and CTL on trees.

Instead of looking at the whole set of all theoretically possible Hintikka formulas as in the classical composition theorems of Feferman and Vaught, respectively Shelah, our composition technique uses the special structure of the logics LTL and CTL to reduce this number of formulas. We exploit that LTL and CTL can only "look in the future" and that information like the satisfaction of an "Until" condition may extend from the current component to "future" components.

We further discuss, why it will be either quite challenging or even impossible to extended the latter result to CTL* over trees.

Overview of the Thesis

The thesis is divided into four main parts. The first part (Chapter 3) introduces the composition method and reviews the classical results of Feferman and Vaught on the one hand and Shelah on the other hand as well as a discussion on the size of the decomposition. The second part (Chapter 4) contains a discussion about the possible extensions of FO logic for which the composition technique for products is applicable and where it fails. The third part (Chapter 5) presents the new composition method for LTL over disjoint ordered sums of words and the improvement in the size of the decomposition it involves. The fourth part (Chapter 6) generalizes this method to CTL and trees. All parts are presented in a way that they may be read independently of each other. However, for Chapter 6, it is useful to also have the examples from the previous chapter in mind.

We now give a detailed overview of the thesis. We begin in Chapter 2 with basic definitions. In Section 2.1, we define general structures and orderings, words, trees and transition systems as specializations of them. We present index structures and trees used for the definition of sums and products of structures. Then, we define disjoint ordered sums of orderings. In the easiest case, the orderings are labeled words and the sum can be seen as the concatenation of these words. We further define disjoint ordered sums of trees which give us again trees. Then, we define products of structures. In Section 2.2, we present the logics used in this thesis: We start with modal logic (ML), first order logic (FO) and monadic second order logic (MSO). We then consider the extensions of ML, respectively FO logic, by the relations "reachable by a path", "reachable by a modulo counting path" and "reachable by a path which labeling sequence is in a regular language". We further give a definition of linear-time temporal logic (LTL), computation tree logic (CTL and CTL*) and variations of them. In Section 2.3, we introduce the basic concept

"interface information tuples" to speak about the satisfaction of formulas in the components (called "component formulas") of a product or sum via a formula in the index structure (called "index formula"). This is done by describing, via the index formula, in which components the component formulas have to hold. The main usage of these tuples is in the composition theorems where they express the conditions when a formula in a product or sum holds. However, they can be also used to define further predicates and relations in the product. This is done afterwards for the product definition of transition systems.

In Chapter 3, we first present the model-theoretic background of the composition technique, followed by the classical composition theorems by Feferman and Vaught, respectively Shelah, and further, the known lower bounds for the size of the decomposition found in the literature. We start in Section 3.1 with elementary equivalence of structures and its description by Hintikka formulas for FO and MSO logic. In Section 3.2, we then present the composition theorem for direct products of (arbitrary relational) structures and FO logic and give a sketch of the proof. These results go back to Feferman and Vaught who also showed a composition theorem for a generalized version of product in [FV59]. We continue in Section 3.3 with the classical composition theorem for disjoint ordered sums of orderings and MSO logic by Shelah [She75]. For the composition theorems for products and sums, we follow the presentations in [Hod97], respectively [Tho97a].

In Chapter 4, we focus on the composition technique for synchronized products of transition systems. In Section 4.1, we first give a formal proof of the known composition theorem for synchronized products of transition systems and FO logic using our notation. We then present the necessary extensions of the structural induction (used in the proof) to capture reachability by paths which are "able to count modulo". This is first done in Section 4.2 for asynchronous products and afterwards, in Section 4.3, for finitely-synchronized products. These results extend work of Rabinovich [Rab07] and Wöhrle and Thomas [WT07]. In Section 4.4, we discuss the limits of the composition theorem for products. We give the proof schema – developed by Rabinovich in [Rab07] – to show the failure of the composition technique for given sets of transition systems and a formula. We then use his proof schema to show the failure of the composition technique in the following cases. On the one hand, we show it for asynchronous products if the logic can express reachability by a path with one of the following conditions: (a) the labels of the path are in a regular language (over at least a binary alphabet) or (b) the path fulfills a formula all the time (as in [Rab07]) or (c) the labels of the path are in a regular language over a unary alphabet but allow tests. On the other hand, we recap the proof of the failure for (unrestricted) synchronized products if the

logic can express reachability by a path from [Rab07]. In that sense, the results from Section 4.3 are tight. We conclude the chapter in Section 4.5 with a summary on the results of the composition technique for products. A preliminary version of the results of this chapter can be found in the papers [Fel08, FT09].

In the next two chapters, our focus lies on sums. In Chapter 5, we show a composition theorem for LTL over disjoint ordered sums of words. The main difference to the general composition theorem for MSO logic and a disjoint ordered sum of orderings is that we get a better complexity for the size of the interface information. Note that LTL has the property that each (sub-)formula evaluated at a current position in the word can only look at the positions following the current position. By using this property we can restrict the number of possible formulas for the current component as they depend on the formulas in the previous component. Furthermore, we use an enriched formula for the index structure in the sense that a part of the information of the given formula is also transferred to the formula for the index structure. For example, an "Until"-condition translates into "Until"-conditions over the components and an "Until"-condition over the index structure. Technically, we use an inductive construction over both the subformulas and smaller (sub-)sums – to be more precise, the sum starting from the next component. The component formulas (for the interface information tuples) are only an exponential extension of the closure of the given formula. We are able to show that the size of the interface information tuples is only exponential – in comparison to the non-elementary complexity in general – in the size of the closure of the given formula.

In Chapter 6, we generalize the results from the previous chapter to CTL and a disjoint ordered sum of trees. We use trees with special symbols $c_1, c_2, \ldots$ in the components. These marked trees are concatenated by replacing the symbols $c_1, c_2, \ldots$ again with (marked) trees of the next components. All nodes with the same special symbol are replaced by the same (or a CTL equivalent) tree. This setting subsumes the different approaches of tree concatenation found in literature and corresponds to the term "context" found, e.g., in [CDG+07]. We generalize the proof from the previous chapter to show a composition theorem for CTL and disjoint ordered sums of trees which has the same complexity. The main difference is now that CTL formulas can speak over all paths or express that there is a path.

Acknowledgement

I am very grateful to all people who contributed to this thesis.

First of all, I would like to thank my advisor Wolfgang Thomas for giving me the opportunity to work as a researcher and for proposing the topic of the thesis.

I am very grateful for his support and advice throughout the past years, both in personal and professional terms. I further like to thank Marcus Lohrey for his kind readiness to act as a co-examiner of this thesis and Joost-Pieter Katoen and Thomas Seidl for completing my thesis committee.

Moreover, special thanks go to my colleagues Marcus Gelderie, Martin Lang and Christof Löding for helpful discussions on the topic of this thesis. Furthermore, I like to thank all how helped me in proofreading and improving the readability of a first draft version of this thesis: Benedikt Brütsch, Marcus Gelderie, Simon Lessenich, Wied Pakusa, Stefan Repke and Daniel Neider.

I would like to thank all current and past members of the I7 & MGI chair for the pleasant working and learning environment and all the nice memories we shared together. Special thanks go to all my fellow sysadmins over the time: Marcus Gelderie, Martin Lang, Jörg Olschewski, Rolf Eschmann and Hartmut Eilers.

Furthermore, I am grateful to my friends for always lending a sympathetic ear to me and just being as they are. Finally, I would like to express my deep gratitude to my parents for all their constant support, encouragement and love (and their patience with my bad mood when a proof went wrong).

2 Technical Preliminaries

In this chapter, we present the technical preliminaries for this thesis: we define structures (Section 2.1) and logics (Section 2.2) and present "interface information tuples" (Section 2.3) used in the composition technique. Let $\mathbb{N}$ denote the natural numbers starting at 1 and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Furthermore, let $[n]$ denote the set $\{1, \ldots, n\}$ for $n \in \mathbb{N}$.

2.1 Structures

In this section, we give the basic definitions of the structures we use in this thesis. Much of this material can also be found in [Hod97].

We begin with the definition of general structures and structures with an ordering on their elements. Then, we consider special cases of structures: we first discuss word models for finite and infinite words, which are labeled ordered structures. Then, we look at trees and their models. Finally, we define transition systems which are labeled graphs, i.e., structures with transition relations and unary predicates. (The transition relations are binary relations between the elements (called states) of the domain and the unary predicates labels of the states.) We conclude with standard operations over structures: the disjoint ordered sum of words and trees and the direct product of transition systems.

Structures are parametrized by a signature: a *signature* is a tuple $\sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$ consisting of a set of constant symbols $\mathcal{C}$, function symbols $\mathcal{F}$ and relational symbols $\mathcal{R}$. These are the labels for the constants, functions and relations which are to be defined. We call σ finite, if $\mathcal{C}$, $\mathcal{F}$ and $\mathcal{R}$ are finite sets. Furthermore, a *relational signature* is a signature which has only relational symbols.

Definition 2.1. Given a signature $\sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$, a *$\sigma$-structure* $\mathcal{A}$ is a tuple $(A, \{c^{\mathcal{A}} \mid c \in \mathcal{C}\}, \{f^{\mathcal{A}} \mid f \in \mathcal{F}\}, \{R^{\mathcal{A}} \mid R \in \mathcal{R}\})$ with

- a set A of elements, called the *domain* and written $\mathrm{dom}(\mathcal{A}) = A$,
- a constant $c^{\mathcal{A}}$ for every constant symbol $c \in \mathcal{C}$,
- a function $f^{\mathcal{A}}$ for every function symbol $f \in \mathcal{F}$ and

- a relation $R^{\mathcal{A}}$ for every relation symbol $R \in \mathcal{R}$.

We denote elements of A by a and write $|A|$ for the cardinality of A. We call a structure *relational* if its signature is relational. Let V and Σ be alphabets. For a relational signature with only unary and binary relations, we use the notations P_v for $v \in V$ for the unary relations – also called predicates – and R_a for $a \in \Sigma$ for the binary relations.

To avoid misunderstandings, we use different constant, function and relation symbols for different arities. Note that for an n-ary function symbol f we can introduce a new $(n+1)$-ary relation symbol R_f with $R_f(x_1, \ldots, x_{n+1}) \Leftrightarrow f(x_1, \ldots, x_n) = x_{n+1}$. From now on, we consider only relational structures. If not stated explicitly, we assume finite signatures.

A linear order is a relation $<$ over the domain A which is irreflexive, total and transitive. An *ordered structure* or *ordering* is a structure with a linearly ordered domain.

To speak about the indices of a product or sum of structures, we introduce the notion *index structure*. Usually, we consider either the set of natural numbers $\mathbb{N}$ or the set $[n]$ for some $n \in \mathbb{N}$ as index set.

Definition 2.2. An *index structure* Ind is a structure with countable set I (called *index set*). An index ordering is an index structure $Ind = (I, <_{Ind})$ where $<_I nd$ is a linear order over the set I.

Words

We now define finite and infinite words and their word models for words over an (in most cases k-ary Boolean) alphabet.

Definition 2.3. Given a (finite) alphabet Σ, a *finite word* w over Σ is defined as $w = a_1 \ldots a_n$ for $n \in \mathbb{N}$ and $a_i \in \Sigma$. An *infinite word* w (also called ω-word) is defined as $a_1 a_2 \ldots$ with $a_i \in \Sigma$ for $i \in \mathbb{N}$. We write Σ^* and Σ^ω for the set of finite respectively infinite words.

Words over the alphabet $\mathbb{B}^k$ for some $k \in \mathbb{N}$ can be seen as ordered structures extended by a labeling of the elements by k symbols:

Definition 2.4. A *word model* of a finite word $w = a_1 \ldots a_n$ over the alphabet $\Sigma = \mathbb{B}^k$ for $k \in \mathbb{N}$ is an ordered structure $\underline{w} = (\mathrm{dom}_w, \mathrm{min}_w, \mathrm{max}_w, \mathrm{Suc}^w, <_w, P_1^w, \ldots, P_k^w)$ with

- domain $\mathrm{dom}_w = [n]$,

- minimal and maximal positions $\min_w = 1, \max_w = n$
- a binary successor relation $\mathrm{Suc}^w(i,j)$ that satisfies $(i,j) \in \mathrm{Suc}^w \Leftrightarrow j = i+1$,
- the natural linear order $i <_w j$ and
- the unary predicate $P_j^w(i)$ with $i \in P_j^w \Leftrightarrow a_i = (b_1, \ldots, b_k)^T$ with $b_j = 1$.

Word models for infinite words $\alpha \in \Sigma^\omega$ are defined analogously. For $i \in \mathrm{dom}_w$ we write $\underline{w}[i]$ and $\underline{w}[i..]$ to denote the letter (as word model) at position i, respectively the sub-word from position i onwards. We allow to omit the sub-, respectively superscript w if it is clear from the context.

Example 2.1. Consider the word $w = \binom{0}{1}\binom{0}{0}\binom{1}{1}\binom{0}{1}$, then the domain of $\underline{w}$ is $\{1, \ldots, 4\}$ with $\min = 1, \max = 4$ and $P_0(1) = P_0(2) = P_0(4) = 0$, $P_0(3) = 1$, $P_1(2) = 0$ and $P_1(1) = P_1(3) = P_1(4) = 1$. For the ω-word $\binom{0}{1}\binom{1}{1}^\omega$, we have as domain $\mathbb{N}$, $P_0(1) = 0, P_1(1) = 1$ and $P_0(i) = P_1(i) = 1$ for $i \geq 2$.

Trees

A tree is a generalization of a word. In a tree, we have several successors for a position (called node here). Given an alphabet Σ, a *tree over* Σ is a mapping $t : X \to \Sigma$ with $X \subseteq \mathbb{N}^*$, and X is closed under prefixes, i.e., $xi \in X \Rightarrow x \in X \wedge \forall j < i : xj \in X$. A tree has at most n successors if $X \subseteq [n]^*$.

Definition 2.5. Let t be a tree over Σ with domain X and at most n successors. The *tree model of* t over Σ is the structure $\underline{t} = (\mathrm{dom}_t, \sqsubseteq_t, \mathrm{Suc}^t, \{\mathrm{Suc}_i^t \mid i \in [n]\}, \mathrm{val}_t)$ with

- (prefix-closed) domain $\mathrm{dom}_t = X \subseteq [n]^*$,
- successor relation $\mathrm{Suc}^t(x,y)$ that satisfies $(x,y) \in \mathrm{Suc}^t \Leftrightarrow \exists i : y = xi$,
- i-th successor relation $\mathrm{Suc}_i^t(x,y)$ that satisfies $(x,y) \in \mathrm{Suc}_i^t \Leftrightarrow y = xi$,
- $x \sqsubseteq^t y$ that satisfies $x \sqsubseteq^t y \Leftrightarrow y = xw$ for some $w \in [n]^*$ and
- the value (or labeling) function $\mathrm{val}_t : \mathrm{dom}_t \to \Sigma$ with $\mathrm{val}_t(x) = t(x)$.

For a tree model over $\Sigma = \mathbb{B}^k$, we often use predicates $P_1^t, \ldots, P_k^t$ instead of the value function where $P_j^t(x)$ holds iff $t(x) = (b_1, \ldots, b_k)^T$ and $b_j = 1$. A *tree model for a tree with an unrestricted number of successors* is defined with $\mathrm{dom}_t \subseteq \mathbb{N}^*$ and an ordering $\prec$ over the successors of a node instead of the relations Suc_i^t: $x \prec^t y$ iff $x = ui, y = uj$ for a $u \in \mathrm{dom}_t$ and $i < j$.

Again, we allow to omit the sub- and superscript t. To simplify notation, from now on, we also allow to write "tree" instead of tree model. Note that in contrast to common definitions as, e.g., in [CDG+07], in our definition, the arity of a tree node does not depend on the label of the node.

To prepare the composition of trees out of given trees, we introduce trees which may have leaf nodes with labels of a special alphabet C_i with $|C_i| = i$ for $i \in \mathbb{N}$. (The trees may also have normal leaves from Σ.) We call these trees *marked* by the symbols $c_j \in C_i$. The idea of the composition of a marked tree with trees over Σ is to replace all nodes labeled with letters from C_i with the Σ-labeled trees. We force that each letter of C_i has to be used at least at one leaf and that leaves with the same letter $c \in C_i$ have to be replaced with the same tree. Note that this is a generalization of "special tree" and "context" found in literature (compare e.g. [CDG+07]). Special trees (as used e.g. in the definition of regular tree languages) only allow exactly one node labeled by one special symbol, whereas contexts allow each symbol of C to occur only exactly once. We now give the definition of a marked tree:

Definition 2.6. Let $C := \{c_i \mid i \in \mathbb{N}\}$ be an infinite alphabet of special symbols and C_i denote the initial fragment $\{c_1, \ldots, c_i\}$ for $i \in \mathbb{N}$ and $C_0 := \emptyset$. A *C_i-marked tree* $\underline{t}$ (also called *i-marked tree*) is a tree over the alphabet $\Sigma \dot\cup C_i$ with the following conditions:

- only leaf nodes can be labeled by C_i, i.e., $\forall u\, \mathrm{val}_t(u) \in C_i \Rightarrow \neg\exists j\, uj \in \mathrm{dom}_t$,
- each $c \in C_i$ occurs at least at one leaf, i.e., $\forall c \in C_i\, \exists u\, \mathrm{val}_t(u) = c$ and
- the root is not labeled with a special letter, $\mathrm{val}_t(\epsilon) \notin C_i$.

A *marked tree* is a C_i-marked tree for some $i \in \mathbb{N}_0$. Note that by definition a tree over Σ is also a marked tree; it is $\emptyset$-marked. (We call a tree *strictly marked* if we want to exclude this case.) For the alphabet $\mathbb{B}^k \dot\cup C_i$ (of a C_i-marked tree), we use the set of predicates $\{P_1, \ldots, P_k\} \cup \{P_c \mid c \in C_i\}$ instead of the value function val_t.

Definition 2.7. We define $\mathrm{T}[\Sigma]$ as the *set of all trees* over the alphabet Σ, $\mathrm{T}_i[\Sigma; C]$ as the *set of all C_i-marked trees* with $C_i \subseteq C$ and $\mathrm{T}[\Sigma; C]$ as the *set of all marked trees*, i.e., $\mathrm{T}[\Sigma; C] = \bigcup_{i:=0}^{\infty} \mathrm{T}_i[\Sigma; C]$.

Example 2.2. We consider the following trees $\underline{t}_\epsilon, \underline{t}_1, \underline{t}_2, \underline{t}_{21} \in \mathrm{T}[\Sigma; C]$ from Figure 2.1. The tree t_ϵ is in $\mathrm{T}_2[\Sigma; C]$ because it contains leaves which are labeled with c_1 and c_2. The tree t_2 is in $\mathrm{T}_1[\Sigma; C]$ and the trees t_1, t_{21} are in $\mathrm{T}_0[\Sigma; C]$.

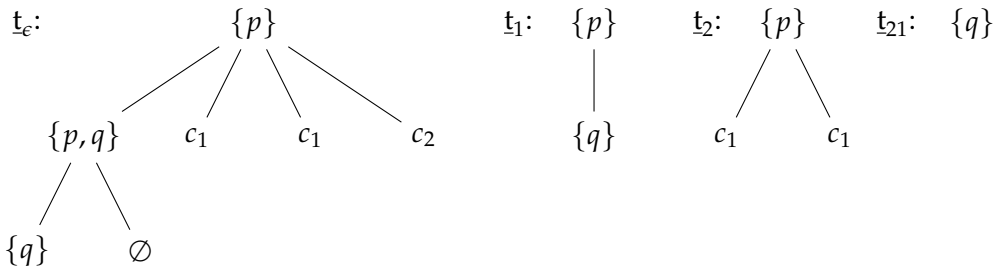


Figure 2.1: Trees from Example 2.2

Given a C_i-marked tree $\underline{t}$ and i trees $\underline{t}_1, \ldots, \underline{t}_i \in T[\Sigma]$, we get a new Σ-labeled tree by replacing all c_j-labeled nodes in $\underline{t}$ with the tree $\underline{t}_j$ for all $j \in [i]$ (see also Figure 2.2). We call this process tree concatenation.

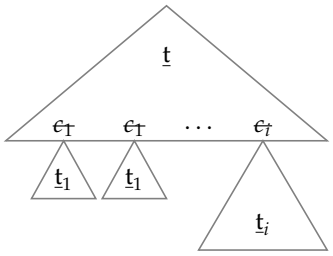


Figure 2.2: Example for Tree Concatenation

To describe tree concatenation iteratively, we use a special kind of index structure: an *index tree* which is a tree of trees – each node is labeled *with a marked tree* and a node u has i successors iff the tree at node u is C_i-marked. (The leaves are labeled with C_0-marked trees which are Σ-labeled trees as $C_0 = \emptyset$.) In Example 2.2 from above, we get an index tree with the nodes $\epsilon, 1, 2, 21$. The node ϵ is labeled with the tree $\underline{t}_\epsilon$, the node 1 with $\underline{t}_1$ and so on.

Formally, an index tree is defined as follows:

Definition 2.8 (Index tree)**.** A tree $\underline{t}_I$ over the alphabet $T[\Sigma; C]$ of marked trees is called *index tree* iff the following compatibility condition holds:

$$uj \in \mathrm{dom}_{\underline{t}_I} \text{ iff the tree at node } u \text{ is } C_i\text{-marked and } j \leq i,$$
$$\text{i.e., iff } u \in \mathrm{dom}_{t_I}, \mathrm{val}_{t_I}(u) \in T_i[\Sigma; C] \text{ and } j \leq i$$

We call a set $T \subseteq T[\Sigma; C]$ of (marked) trees compatible with a tree $\underline{t}_I$ over the alphabet $T[\Sigma; C]$ if we have for each index exactly one tree in T (and there are no other trees in T) and $\underline{t}_I$ forms an index tree.

Example 2.3. In Figure 2.3, we see a short notation for the index tree from above using the trees from Figure 2.1. The compatibility condition is fulfilled as the node ϵ has two successors and the tree $\underline{t}_\epsilon$ uses exactly the symbols c_1 and c_2. Furthermore, the node 2 has one successor and the associated tree uses only c_1 and the other nodes have no successors and the associated are from $\mathrm{T}[\Sigma] = \mathrm{T}_0[\Sigma; C]$. The concatenation of the trees (the "sum tree") defined via this index tree is shown in Figure 2.4.

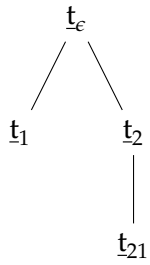


Figure 2.3: Index Tree for the Trees from Figure 2.1

We introduce a variant of an index tree: a tree which has sets of marked trees as labels.

Definition 2.9 (Index tree over sets of trees)**.** A tree $\underline{t}_I$ over the alphabet $\mathcal{P}(\mathrm{T}[\Sigma; C])$ of sets of marked trees is called *index tree (over sets of trees)* iff the following compatibility conditions hold:

- $\forall u \in \mathrm{dom}_{\underline{t}_I} \exists i \in \mathbb{N}$ s.t. all $t \in \mathrm{val}_{\underline{t}_I}(u)$ are C_i-marked and
- $\forall u \in \mathrm{dom}_{\underline{t}_I} : uj \in \mathrm{dom}_{\underline{t}_I}$ iff the trees at node u are C_i-marked and $j \leq i$.

We now define transition systems, which are basically structures with only unary and binary relations. The unary relations (also called predicates) are seen as a labeling of the states and the binary relations as transitions between the states.

Definition 2.10. Given two alphabets Σ and V, a *(labeled) transition system* – also called *Kripke structure* – is a structure $\mathcal{K} = (S, \{R_a \mid a \in \Sigma\}, \{P_v \mid v \in V\})$ where

- S is the domain of $\mathcal{K}$, called state space, and the elements $s \in S$ are called states,
- R_a are (binary) transition relations for $a \in \Sigma$ and
- P_v are (unary) predicates for $v \in V$.

A *pointed transition system* $(\mathcal{K}, s)$ is a transition system $\mathcal{K}$ with one marked initial state s.

Paths

Intuitively, a path is a sequence of states that are connected via the transition relations. We mainly follow [BK08] in the formal definition:

Definition 2.11. Let S denote the domain of a transition system, respectively of a tree, and let R denote the union over the transition relations R_a ($a \in \Sigma$) , respectively the union over the successor relations Suc_i ($i \in [n]$).

A *finite path fragment* is a sequence $\pi = (s_1, \ldots, s_k)$ of states from S such that $(s_i, s_{i+1}) \in R$ for $i \in [k-1]$. An *infinite path fragment* is a sequence $\pi = (s_1, s_2, \ldots)$ of states with $(s_i, s_{i+1}) \in R$ for $i \in \mathbb{N}$. We use the notation $\pi[i..]$ to denote the path fragment starting at s_i and $\pi[i, j]$ to denote the fragment from s_i to s_j.

A *maximal path fragment* is a path fragment that cannot be extended, i.e., it is either infinite or a finite path segment $(s_1, \ldots, s_k)$ and there is no outgoing transition from s_k. Given a state s, a *path* is a maximal path fragment starting in s. If we do not mention the state s, we mean the initial state of the (pointed) transition system, respectively the root of the tree.

Definition 2.12. For a transition system, the *(transition) labeling sequence* $l(\pi)$ of a path fragment π is defined as the sequence of labels of the transition between the states of π. For a finite path fragments, we have $l(\pi) = a_1 \ldots a_{n-1}$ if $\pi = (s_1, \ldots, s_n)$ and $(s_i, s_{i+1}) \in R_{a_i}$ for $i \in [n-1]$ and for infinite fragments the same condition for $i \in \mathbb{N}$. Furthermore, the *(predicate) labeling sequence* $L(\pi)$ of a path fragment π is defined as the sequence of labels of the predicates at the states of π. For finite path fragments, we have $L(\pi) = v_1 \ldots v_n$ if $\pi = (s_1, \ldots, s_n)$ and $P_{v_i}(s_i)$ holds. The definition for infinite path fragments is analogous.

Sums and products of structures

We now consider two standard operations on structures: the (ordered) disjoint sum of orderings and trees and the direct product of general structures.

Definition 2.13 (Sum of orderings)**.** We assume that we are given an index structure $Ind = (I, <_I)$ with infinite index set I (usually $\mathbb{N}$) and words w_i with word models $\underline{w_i} = (\mathrm{dom}_i, <_i, \mathrm{min}_i, \mathrm{max}_i, \{P_v^i \mid v \in V\})$ – the components – with finite domain dom_i for $i \in I$. Then, the *(infinite) ordered disjoint sum over* $\underline{w_i}$ *for* $i \in I$ is defined as $\underline{w} = (\mathrm{dom}, <, \mathrm{min}, \{P_v \mid v \in V\})$ where $\mathrm{dom} = \dot{\bigcup}_{i \in I} \mathrm{dom}_i$ (in the order of the indices), $\mathrm{min} = \mathrm{min}_1$ (where 1 denotes the first element from I), $x < y$ if there is either an $i \in I$ with $x <_i y$ or there are $i, j \in I$ with $i <_I j$ and x belongs to $\underline{w_i}$ and y to $\underline{w_j}$ and $P_v(x)$ holds if $P_v^i(x)$ holds and x belongs to $\underline{w_i}$.

The *(finite) ordered disjoint sum over* $w_i, i \in I = [n]$ (where the last w_i may be infinite) is defined analogously.

For a given index tree (with marked trees for the nodes), we now give the definition for the sum of the trees which is constructed by concatenating the marked trees which are assigned to the nodes of the index tree.

Definition 2.14 (Sum of trees)**.** Let $\underline{t}_I$ be an index tree over the alphabet of marked trees $\mathrm{T}[\Sigma; C]$ with trees $\underline{t}_z := \mathrm{val}_{\underline{t}_I}(z) \in \mathrm{T}_i[\Sigma; C]$ for a $i \in \mathbb{N}$ given for every $z \in \mathrm{dom}_{\underline{t}_I}$. The *sum tree* $\underline{t}_S$ of the trees $\underline{t}_z, z \in \mathrm{dom}_{\underline{t}_I}$ in the order given by the index tree is now constructed as follows:

Intuitively, each c_j-labeled node of the tree $\underline{t}_z$ is replaced with the tree of the j-th successor. Formally, we first define a set Y of all pairs of a node z in the index tree and a node u of the tree $\underline{t}_z$. Then, we define a function d which maps each of these tuples to a subset of $\mathbb{N}^*$. The image of this function is the domain of the sum tree.

Let $Y := \{(z, u) \mid z \in \mathrm{dom}_{\underline{t}_I}, u \in \mathrm{dom}_{\underline{t}_z}\}$. The function $d : Y \to \mathcal{P}(\mathbb{N}^*)$ is defined as

- $d(\epsilon, u) := \{u\}$ and
- $d(zj, u) := \bigcup\{v \cdot u \mid \exists w \in \mathrm{dom}_{\underline{t}_z} : v \in d(z, w) \wedge \mathrm{val}_{\underline{t}_z}(w) = c_j\}$, i.e., there is a node w in the tree at the position z of the index tree that is labeled by c_j and v is the corresponding node to w in the already defined part of the sum tree

This defines the domain of the sum tree as $\mathrm{dom}_{\underline{t}_S} := d(Y)$. The labels of the sum tree are defined in the natural way:

- If $u \in \mathrm{dom}_{\underline{t}_S}$ with $\{u\} = d(\epsilon, u)$ and $\mathrm{val}_{t_\epsilon} \notin C$, then $\mathrm{val}_{\underline{t}_S}(u) = \mathrm{val}_{\underline{t}_\epsilon}(u)$.
- If $w \in \mathrm{dom}_{\underline{t}_S}$ with $w \in d(z, u)$ for a $z \in \mathrm{dom}_{\underline{t}_I}$ and $u \in \mathbb{N}^*, \mathrm{val}_{t_z} \notin C$ then $\mathrm{val}_{\underline{t}_S}(w) = \mathrm{val}_{t_z}(u)$.

The sum tree for an index tree over sets of trees at each component is defined analogously and yields a set of trees as sum – the *sum tree set*.

Example 2.4. We look at the sum tree that we get from the concatenation of the trees according to the index tree $\underline{t}_I$ from Figure 2.3. The domain of $\underline{t}_S$ contains all nodes from $\underline{t}_\epsilon$ because of $d(\epsilon, u) = \{u\}$. Starting at the nodes $2, 3$, it further contains the tree t_1 as $d(1, \epsilon) = \{2, 3\}$ and $d(1, 1) = \{21, 31\}$ and at node 4 it contains the tree $\underline{t}_2$. The sum tree contains the labels of associated trees where each $c \in C$ is replaced with the root of the tree which is inserted at this point. We get the sum tree $\underline{t}_S$ of Figure 2.4.

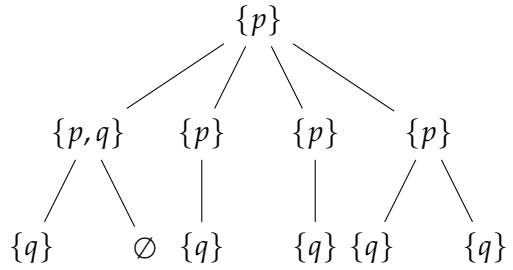


Figure 2.4: Sum Tree of the Index Tree from Figure 2.3

Definition 2.15 (Product of structures)**.** Given a (relational) signature $\sigma = (\emptyset, \emptyset, \mathcal{R})$, an index structure Ind with index set I and σ-structures $\mathcal{A}_i$, $i \in I$, the *direct σ-product* $\prod_{i \in I} \mathcal{A}_i$ – also called the *Cartesian σ-product* – is defined as the structure $(A, \{R^{\mathcal{A}} \mid R \in \mathcal{R}\})$ with

- domain $A := \{\bar{a} \mid \bar{a} : I \to A\}$, $\bar{a}[i]$ denotes the state of the i-th component and
- n-ary relation symbol $R^{\mathcal{A}}$ with $R^{\mathcal{A}}(\bar{a}_1, \ldots \bar{a}_n) \Leftrightarrow \forall i \in I : R^{\mathcal{A}_i}(\bar{a}_1[i], \ldots \bar{a}_n[i])$.

For the composition theorem, one can look at various other definitions of products. In [Hod97], there are various definitions which generalize the concept of direct products.

In the field of model checking, we are mainly interested in products of transition systems. Here, the definition from above means that we have an a-labeled transition in the product if we have an a-labeled transition in *every* component. We generalize this definition: a transition is in the product if there is a subset of the components and each of these takes a transition. To express this information, we use "interface information tuples" which define a relation in the product by giving conditions expressing which formulas have to hold in which components. The formal definition of synchronized products (and interface information tuples) can be found in Section 2.3.

2.2 Logics

In this section, we present the logics we use in this thesis. We begin with the commonly used logics: first-order (FO) logic and monadic second order (MSO) logic. Then, we present the definition of logics used in model checking: modal logic (ML), linear temporal logic (LTL) and computation tree logic (CTL). For convenience, we present the definitions with all common operators in these logics even if shorter

definitions are possible. We use the standard definitions of satisfaction and validity as defined, e.g., in [EF05, EFT96].

First order and monadic second order logic

For the definition of first-order and monadic second order logic, we follow the presentation in [EF05]. Given a finite (relational) signature σ with relation symbols $R_1, \dots, R_m$ of arities $n_1, \dots, n_m$, we define *first order logic over σ (FO(σ) logic)* as follows: let $x_1, x_2, \dots$ denote variables for elements of a σ-structure, then $R_i(x_1, \dots, x_{n_i})$, tt, ff and $x_1 = x_2$ are (atomic) FO formulas. Furthermore, for FO formulas φ, ψ, inductively, we get the formulas $\neg\varphi, \varphi \vee \psi, \varphi \wedge \psi, \varphi \to \psi, \varphi \leftrightarrow \psi$ and the existential and universal quantification $\exists x\varphi(\dots, x, \dots), \forall x\varphi(\dots, x, \dots)$ with the usual semantics as, e.g., in [EF05]. For FO formulas over the signature σ, we also write FO σ-formulas. We write $FO(<)$ to explicitly state that $<$ has to be in the signature σ.

Variables which are not in scope of a quantifier are called *free* variables, the other ones are *bounded*. A formula in which all occurrences of variables are bounded is called *sentence*. We write $\varphi(x_1, \dots, x_n)$ if φ contains *at most* $x_1, \dots, x_n$ as free variables.

A formula φ is a called *(proper) subformula of δ* (written $\varphi \lhd \delta$) if φ is used in the inductive definition of δ. We write $\varphi \trianglelefteq \delta$ if φ may also be δ. The formula φ is a *direct subformula of δ* if it is used in the inductive definition in the last step. The set of all subformulas of a formula δ is called the *closure of δ* and denoted as $\mathrm{CL}(\delta)$. The *size $s(\delta)$ of a formula δ* is defined as $\mathrm{CL}(\delta)$, i.e., $s(\delta) := |\mathrm{CL}(\delta)|$.

The *quantifier depth* $\mathrm{qd}(\varphi)$ of an FO formula φ is defined as follows: For atomic formulas φ, we have $\mathrm{qd}(\varphi) = 0$, for negation $\mathrm{qd}(\neg\varphi) = \mathrm{qd}(\varphi)$, for conjunction and disjunction $\mathrm{qd}(\varphi \vee \psi) = \mathrm{qd}(\varphi \wedge \psi) = \max\{\mathrm{qd}(\varphi), \mathrm{qd}(\psi)\}$ and for existential and universal quantification $\mathrm{qd}(\exists x\varphi) = \mathrm{qd}(\forall x\varphi) = \mathrm{qd}(\varphi) + 1$. The *quantifier alternation depth* counts only the alternations between existential and universal quantifications.

Monadic second order logic (MSO logic) is defined as an extension of first order logic. Let $X_1, X_2, \dots$ be variables for sets of elements. We add (atomic) formulas $X(y)$ stating that the variable y belongs to the set variable X and formulas $\exists X\varphi(\dots, X, \dots)$, $\forall X\varphi(\dots, X, \dots)$ which quantify over set variables with the usual semantics as, e.g., in [EF05]. The closure and size of an MSO formula are defined analogously the ones for FO formulas. For the quantifier depth of an MSO formula, we count both the normal and the set quantifiers.

One should note that there is a fragment of MSO($<$) logic which uses only set variables and quantification over them and is logically equivalent to MSO($<$)

logic, see, e.g., [Tho97a]. The main idea is the following: For set variables X, Y we introduce new atomic formulas $\text{Nonempty}(X \cap Y)$ with the meaning X and Y have a nonempty intersection, $X \subseteq Y$, $X < Y$ meaning there are elements x in X and y in Y with $x < y$ and the formula $X_1 \cup \cdots \cup X_k = \text{All}$ which expresses that the union of the sets $X_1, \ldots, X_k$ is the universe.

We define a *partition formula* that expresses that set variables $X_1, \ldots, X_k$ form a partition of a set variable X, i.e., $(\bigcup_{i=1}^{k} X_i = X) \wedge \bigwedge_{i,j \in [k], i \neq j} (X_i \cap X_j = \emptyset)$.

Definition 2.16. The *partition formula* $\beta_{\text{Partition}}(X_1, \ldots, X_k; X)$ is defined as

$$\forall x (X(x) \to \bigvee_{i \in [k]} X_i(x)) \wedge \bigwedge_{i,j \in [k], i \neq j} \neg \exists x (X_i(x) \wedge X_j(x)).$$

We now discuss extensions of FO and MSO logic interpreted over transition systems. A main drawback of FO logic in comparison to MSO logic is the lack of being able to express reachability via paths. We consider several variants of reachability added to FO logic.

Definition 2.17. Let σ be a signature with binary relations R_a for $a \in \Sigma$ and let $R := \bigcup_{a \in \Sigma} R_a$. Let R^* be a symbol denoting the binary relation of all state pairs (x, y) such that y is reachable from x via a path fragment of transitions from R. Then, we define *FO logic with reachability (FO(R))* by FO logic over the signature $\sigma \cup R^*$.

Given a regular expression α, we define Reach_α by the set of all pairs (x, y) such that there exists a path fragment π from x to y and $l(\pi)$ is in the language of the regular expression α over Σ. *FO logic with regular reachability FO(RegR)* is defined as FO logic over the signature σ extended by the relations $\text{Reach}_\alpha(x, y)$ for regular expressions α over Σ. If the regular expressions only use a unary alphabet, we call this extension *FO(Reg1R)*.

We show that every relation Reach_α over a unary alphabet can be expressed by relations which express reachability where the path length is in a semilinear set.

Lemma 2.18. For $l, k \in \mathbb{N}$, let $\text{Path}_{l,k}$ be the set of all pairs (x, y) such that there exists a path fragment π from x to y and the length of π is divisible by k with remainder l. Every relation Reach_α with a regular expression α over a unary alphabet can be expressed by a formula using atomic formulas $\text{Path}_{l,k}$ with $k \in \mathbb{N}$ and $l \in [k] - 1$. Thus, the logic FO(Reg1R) can be described as FO logic with the additional relations $\text{Path}_{l,k}$ for $k \in \mathbb{N}$ and $l \in [k] - 1$.

Proof. Let L be the regular language defined by α. The language L has a semilinear image $\psi(L) = \bigcup_{i \in [m]} M_i$ with $m \in \mathbb{N}$ and $M_i = \{k_{i_0} + n_{i_1} \cdot k_{i_1} + \cdots + n_{i_r} \cdot k_{i_r} \mid$

$n_{i_j} \geq 0$ for $1 \leq j \leq r_i\}$. We have $k_{i_j} \in \mathbb{N}$ because α is a regular expression over a unary alphabet. It is well-known that Presburger arithmetic is decidable and that the family of semilinear sets is equivalent to the family of Presburger sets in $\mathbb{N}$. Thus, each M_i from above can be described as a union of a finite set R and a set $\{z \mid z = l + n \cdot k, z > m, n \in \mathbb{N}\}$ for some $k, l \in \mathbb{N}$ and m is the maximum of the elements from R. The results used in this proof can be found in [Har78]. □

We now look at an extension of MSO logic (without $<$) that allows counting over the number of elements in a set (variable). Afterwards, we show that this logic can be expressed in MSO($<$) logic.

Definition 2.19. For a set variable X, let $\mathrm{Card}_{j,k}(X)$ denote that X contains a number of elements which is divisible by k with remainder j. *Counting monadic second order logic (CMSO)* is defined as the extension of MSO logic by the predicates $\mathrm{Card}_{j,k}(X)$.

Lemma 2.20. Over linearly ordered structures, CMSO logic is equivalent to MSO($<$) logic. In general, CMSO logic is strictly more expressive than MSO logic.

A proof of the second statement can be found in [GR08]. The proof idea of the first statement is just the standard argument to express counting in MSO($<$): for every formula $\mathrm{Card}_{j,k}(X)$, we force that there exist $k-1$ disjoint subsets $X_0, \ldots, X_{k-1}$ of X, the first element of X is in X_0, the last is in X_{j-1} (respectively X_{k-1} for $j = 0$) and two successive elements from X are in the sets X_j, X_{j+1} for $j \in \{0, \ldots, k-2\}$ (respectively X_{k-1}, X_0). For example, for the formula $\mathrm{Card}_{0,2}(X)$ we get

$$\forall x \forall y [X(x) \wedge X(y) \rightarrow \exists X_0 (\mathrm{first}(x, X) \rightarrow X_0(x) \wedge \mathrm{last}(x, X) \rightarrow \neg X_0(x) \\ \mathrm{succ}(x, y, X) \rightarrow (X_0(x) \leftrightarrow \neg X_0(y)))]$$

with $\mathrm{first}(x, X) := \neg \exists y (y < x \wedge X(y))$, $\mathrm{last}(x, X)$ analogously and $\mathrm{succ}(x, y, X) := \neg \exists z (x < z \wedge z < y \wedge X(z))$.

Modal logic and propositional dynamic logic

In this section, we present *(multi-)modal logic* (ML) and propositional dynamic logic according to [BvBW07, HKT02]. These logics can be interpreted in any labeled transition system together with an (initial) state. Modal logic extends predicate logic: it not only allows to check predicates of states (of a given transition system) and Boolean combinations of formulas, but also local modalities between neighbor states. Note that in contrast to FO and MSO logic, ML does not contain variables.

We present the syntax of multi-modal logic for a given signature with (unary) predicates P_v for $v \in V$ and (binary) transition relations R_a for $a \in \Sigma$: we have

the atomic formulas tt, ff and for every P_v the atomic formula p_v. For ML formulas φ, ψ we get the formulas $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi$. Moreover, for every $a \in \Sigma$ we get the formulas $\langle a \rangle \varphi$ and $[a]\varphi$ – meaning that there is an outgoing a-labelled transition to a state where φ holds, respectively for all outgoing a-labelled transitions φ holds.

Strictly speaking, modal logic is the fragment of multi-modal logic which does not distinguish between the different transition relations (and uses simply $\Diamond\varphi$ and $\Box\varphi$ instead of $\langle a \rangle \varphi$ and $[a]\varphi$). As we are only interested in multi-modal logic, we simply write modal logic if we mean multi-modal logic. In the literature [HKT02], multi-modal logic is also called dynamic logic.

We look at the same extensions of modal logic that we discussed for FO logic: *ML(R)* denotes the extension by reachability, *ML(RegR)* reachability by path fragments described by regular expressions and *ML(Reg1R)* reachability by path fragments described by regular expressions over a unary alphabet. We use the notations $EF\varphi$ respectively $\langle \alpha \rangle \varphi$ for "there is a path (which satisfies α) to a state where φ holds".

(Regular) propositional dynamic logic (PDL) extends modal logic by allowing more complex conditions instead of the modalities $\langle a \rangle$ and $[a]$. We only give here the syntax and an intuition of the semantics – a comprehensive discussion can be found in [HKT02]. In PDL, we distinguish between formulas and programs. For the signature of a transition system with (unary) predicates and binary transition relations, the predicates are the atomic formulas and the transition relations the atomic programs. For formulas φ, ψ and a program α, inductively, we get the formulas $\neg\varphi, \varphi \vee \psi, \varphi \wedge \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi$ and $\langle \alpha \rangle \varphi, [\alpha]\varphi$. For a formula φ and programs α, β, we inductively get the programs $\alpha;\beta$ (sequential composition of the programs), $\alpha + \beta$ (non-deterministic choice between the programs), α^* (arbitrary iteration of the program α) and $\varphi?$ (test if the formula φ holds). Note that the first three operations for programs are similar to the standard operations of regular expressions: concatenation, disjunction and iteration. Thus, the variant "PDL without tests" coincides with modal logic with regular reachability (ML(RegR)) from above. We call PDL over a unary alphabet *1PDL*. Note that 1PDL without tests is equivalent to ML(Reg1R).

Linear Temporal Logic

Linear temporal logic (LTL) allows specifying linear time properties. It was introduced into computer science by Pnueli [Pnu77] and can be interpreted over linear structures (words) or *over all paths* of a branching structure (transition systems).

LTL extends predicate logic by modalities which allow to express global properties, i.e., properties over whole path(s).

We mainly follow Baier and Katoen [BK08] in the presentation of LTL and also the logics defined in the next sections. However, we use the letters X, F, G (instead of $\bigcirc, \Diamond, \Box$) for "next", "finally" and "globally" to avoid confusion with modal logic.

Given a signature σ and V an alphabet for the unary predicates, the syntax of linear temporal logic is defined as follows.

- We have the atomic formulas $\mathtt{tt}, \mathtt{ff}$ and p for $p \in V$.
- Given LTL formulas φ and ψ, we get the LTL formulas $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $X\varphi$, $F\varphi$, $G\varphi$, $\varphi U \psi$ and $\psi R \varphi$.

The *closure* and the *size* of an CTL or LTL formula are defined as for FO formulas. The *modal depth* $\mathrm{md}(\varphi)$ of an LTL formula φ is defined analogously to the quantifier depth of an FO formula: $\mathrm{md}(\varphi) = 0$ for atomic formulas $\varphi \in \{p, \mathtt{tt}, \mathtt{ff}\}$, $\mathrm{md}(X\varphi) = \mathrm{md}(\varphi) + 1$, $\mathrm{md}(\varphi \wedge \psi) = \mathrm{md}(\varphi \vee \psi) = \max(\mathrm{md}(\varphi), \mathrm{md}(\psi))$ and $= \mathrm{md}(\varphi U \psi) = \mathrm{md}(\varphi R \psi) = \max(\mathrm{md}(\varphi), \mathrm{md}(\psi)) + 1$.

Before we define the semantics of LTL, we first give the intuition of the operators:

- $X\varphi$: at the **next** state φ holds,
- $F\varphi$: eventually (**finally**) there is a state where φ holds,
- $G\varphi$: **globally** (on all states) we have φ,
- $\varphi U \psi$: we have φ on all states **until** ψ holds at a state and
- $\psi R \varphi$: we have φ on all states or there is a state where ψ holds and this state **releases** φ, i.e., φ does not have to hold after this state.

For shorter notation, we further use the Weak-Until operator $\varphi W \psi$ as an abbreviation for $(\varphi U \psi) \vee G\varphi$.

Formally, the semantics of LTL is defined as follows. Let π be a path or a word model. Then, we have:

$$
\begin{array}{lcl}
\pi[i..] \models \mathtt{tt} & :\Leftrightarrow & i \text{ is a position of } \pi \\
\pi[i..] \models \mathtt{ff} & :\Leftrightarrow & i \text{ is no position of } \pi \\
\pi[i..] \models p & :\Leftrightarrow & P_p(\pi[i]) \text{ holds} \\
\pi[i..] \models \varphi \wedge \psi & :\Leftrightarrow & \pi[i..] \models \varphi \text{ and } \pi[i..] \models \psi \\
\pi[i..] \models \varphi \vee \psi & :\Leftrightarrow & \pi[i..] \models \varphi \text{ or } \pi[i..] \models \psi \\
\pi[i..] \models \neg\varphi & :\Leftrightarrow & \pi[i..] \not\models \varphi \\
\pi[i..] \models X\varphi & :\Leftrightarrow & \pi[i{+}1..] \models \varphi \\
\pi[i..] \models F\varphi & :\Leftrightarrow & \exists k \geq i\, (\pi[k..] \models \varphi) \\
\pi[i..] \models G\varphi & :\Leftrightarrow & \forall j \geq i\, (\pi[j..] \models \varphi) \\
\pi[i..] \models \varphi U \psi & :\Leftrightarrow & \exists k \geq i\, (\pi[k..] \models \psi \text{ and } \forall j, i \leq j < k : \pi[j..] \models \varphi) \\
\pi[i..] \models \psi R \varphi & :\Leftrightarrow & \forall j \geq i\, (\pi[j..] \models \varphi) \text{ or} \\
 & & \exists k \geq i\, (\pi[k..] \models \varphi \wedge \psi \text{ and } \forall j, i \leq j < k : \pi[j..] \models \varphi)
\end{array}
$$

For a finite path π (respectively word models), we can define the last state of π by stating that the next position satisfies ff. Thus, we define the formula last as $X\mathtt{ff}$.

We now present some useful transformations for LTL formulas (see also [BK08]). Most of these transformations are used in later chapters.

Duality laws:

$$
\begin{array}{ll}
\neg X\varphi \equiv X\neg\varphi & \neg(\varphi U \psi) \equiv (\neg\varphi) R (\neg\psi) \\
\neg F\varphi \equiv G\neg\varphi & \neg(\varphi W \psi) \equiv (\varphi \wedge \neg\psi) U (\neg\varphi \wedge \neg\psi)
\end{array}
$$

Expansion laws:

$$
\begin{array}{ll}
\varphi U \psi \equiv \psi \vee (\varphi \wedge X(\varphi U \psi)) & \psi R \varphi \equiv (\varphi \wedge \psi) \vee (\varphi \wedge X(\psi R \varphi)) \\
\varphi W \psi \equiv \psi \vee (\varphi \wedge X(\varphi W \psi)) &
\end{array}
$$

Distributive laws:

$$
\begin{array}{ll}
X(\varphi U \psi) \equiv (X\varphi) U (X\psi) & F(\varphi \vee \psi) \equiv F\varphi \vee F\psi \\
 & G(\varphi \wedge \psi) \equiv G\varphi \wedge G\psi
\end{array}
$$

Note that $F(\varphi \wedge \psi) \neq F\varphi \wedge F\psi$ and, dually, $G(\varphi \vee \psi) \neq G\varphi \vee G\psi$.

We now introduce two negation normal forms (NNF) of an LTL formula, where the negation sign only occurs in front of the atomic formulas. The two versions

use the Until and the Release operator, respectively the Until and the Weak-Until operator. They are called Release positive normal form (Release PNF) and Weak-Until PNF in the literature [BK08].

We only introduce the Release PNF – here simply called NNF – as the transformation of an LTL formula into Release PNF produces an LTL formula which size is only linear in the size of the original formula. (In the other case, the size grows exponential.)

Definition 2.21. An LTL formula is in negation normal form (NNF) if it can be constructed from the atomic and negated atomic formulas $\mathtt{tt}, \mathtt{ff}, p, \neg p$ and inductively from $\varphi \wedge \psi, \varphi \vee \psi, X\varphi, \varphi U \psi, \psi R \varphi$ for formulas φ, ψ in NNF.

Lemma 2.22. Every LTL formula φ can be transformed into an equivalent LTL formula ψ in NNF with size $|\psi| \in O(|\varphi|)$.

Proof sketch. To prove this lemma, we can use the duality laws for X, F, G, U and R operators. These laws (as well as the duality of $\wedge$ and $\vee$) increase the size only by a constant factor. Thus, by inductively using them, the size of φ grows only linearly. □

Computation tree logic

In this subsection, we present *computation tree logic (CTL)* which is – in contrast to linear time logic – a branching time logic. CTL formulas are interpreted at the states of a transition system. CTL allows to specify conditions at nodes about the existence of a path or about all paths starting at a node. It does only allow "next", "until" and "release" conditions over these paths where the subformulas are again interpreted at states. In the next section we introduce the logic CTL* in which this restriction is lifted.

The name "computation tree logic" stems from the fact that instead of looking at all paths starting in the initial node of the transition system, CTL looks at the (usually infinite) tree which is obtained by "unfolding" the transition system and each traversal through this tree represents a possible computation. CTL was introduced by Clarke and Emerson [CES86]. This subsection follows mainly the (more detailed) presentation in [BK08].

We first introduce the syntax of CTL. For this, we inductively define CTL path and state formulas which are interpreted over a path respectively at a state.

- We have $\mathtt{tt}, \mathtt{ff}$ and the predicates p as (atomic) CTL state formulas.

- For CTL state formulas φ, ψ and a CTL path formula $\bar{\varphi}$, we get the CTL state formulas $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, E\bar{\varphi}$ and $A\bar{\varphi}$

- For CTL state formulas φ, ψ, we get CTL path formulas $X\varphi$, $F\varphi$, $G\varphi$, $(\varphi U \psi)$, and $(\psi R \varphi)$.

We give the intuition for these formulas: $E\bar{\varphi}$ stands for "there exists a path starting at the current node which satisfies $\bar{\varphi}$" and $A\bar{\varphi}$ stands for "all paths starting at the current node satisfy $\bar{\varphi}$". Note that state and path formulas have to alternate, e.g. $AFAGp$ is a CTL formula but $AFGp$ not.

Formally, the semantics of CTL is defined as follows:

$$
\begin{array}{lcl}
s \models \mathrm{tt} & & \text{for all states } s \\
s \models \mathrm{ff} & & \text{for no state} \\
s \models p & :\Leftrightarrow & P_p(s) \text{ holds} \\
s \models \neg\varphi & :\Leftrightarrow & s \not\models \varphi \\
s \models \varphi \wedge \psi & :\Leftrightarrow & s \models \varphi \text{ and } s \models \psi \\
s \models \varphi \vee \psi & :\Leftrightarrow & s \models \varphi \text{ or } s \models \psi \\
s \models E\bar{\varphi} & :\Leftrightarrow & \text{there exists a path } \pi \text{ starting in } s \text{ which satisfies } \pi \models \bar{\varphi} \\
s \models A\bar{\varphi} & :\Leftrightarrow & \text{all paths } \pi \text{ starting in } s \text{ satisfy } \pi \models \bar{\varphi} \\
\pi[i..] \models X\varphi & :\Leftrightarrow & \pi[i{+}1] \models \varphi \\
\pi[i..] \models F\varphi & :\Leftrightarrow & \exists k \geq i\, (\pi[k] \models \varphi) \\
\pi[i..] \models G\varphi & :\Leftrightarrow & \forall j \geq i\, (\pi[j] \models \varphi) \\
\pi[i..] \models \varphi U \psi & :\Leftrightarrow & \exists k \geq i\, (\pi[k] \models \psi \text{ and } \forall j, i \leq j < k : \pi[j] \models \varphi) \\
\pi[i..] \models \psi R \varphi & :\Leftrightarrow & \forall j \geq i\, (\pi[j] \models \varphi) \text{ or} \\
 & & \exists k \geq i\, (\pi[k] \models \varphi \wedge \psi \text{ and } \forall j, i \leq j < k : \pi[j] \models \varphi)
\end{array}
$$

We further define a Weak-Until in both version $E(\varphi W \psi)$ and $A(\varphi W \psi)$ with an analogous meaning as the Weak-Until in LTL. We cannot use the definition from LTL as $E((\varphi U \psi) \vee G\varphi)$ is not a valid CTL formula. However, we can use the duality law for the LTL Weak-Until to define the CTL Weak-Until and get:

$$
\begin{aligned}
E(\varphi W \psi) &\equiv \neg A((\varphi \wedge \neg\psi) U (\neg\varphi \wedge \neg\psi)) \\
A(\varphi W \psi) &\equiv \neg E((\varphi \wedge \neg\psi) U (\neg\varphi \wedge \neg\psi))
\end{aligned}
$$

As in the case of LTL, we present useful transformations for CTL formulas (according to [BK08]) which we use later.

Duality laws:

$$\begin{aligned} \neg AF\varphi &\equiv EG\neg\varphi & \neg A(\varphi U\psi) &\equiv E((\neg\varphi)R(\neg\psi)) \\ \neg EF\varphi &\equiv AG\neg\varphi & \neg E(\varphi U\psi) &\equiv A((\neg\varphi)R(\neg\psi)) \\ \neg AX\varphi &\equiv EX\neg\varphi & \neg A(\varphi W\psi) &\equiv E(\varphi\wedge\neg\psi)U(\neg\varphi\wedge\neg\psi) \\ & & \neg E(\varphi W\psi) &\equiv A(\varphi\wedge\neg\psi)U(\neg\varphi\wedge\neg\psi) \end{aligned}$$

Expansion laws:

$$\begin{aligned} AF\psi &\equiv \psi\vee AX\,AF\psi & A(\varphi U\psi) &\equiv \psi\vee(\varphi\wedge AX\,A(\varphi U\psi)) \\ EF\psi &\equiv \psi\vee EX\,EF\psi & E(\varphi U\psi) &\equiv \psi\vee(\varphi\wedge EX\,E(\varphi U\psi)) \\ AG\varphi &\equiv \varphi\wedge AX\,AG\varphi & A(\psi R\varphi) &\equiv (\varphi\wedge\psi)\vee(\varphi\wedge AX\,A(\psi R\varphi)) \\ EG\varphi &\equiv \varphi\wedge EX\,EG\varphi & E(\psi R\varphi) &\equiv (\varphi\wedge\psi)\vee(\varphi\wedge EX\,E(\psi R\varphi)) \end{aligned}$$

Distributive laws:

$$AG(\varphi\wedge\psi)\equiv AG\varphi\wedge AG\psi \qquad EF(\varphi\vee\psi)\equiv EF\varphi\vee EF\psi$$

Note that $AF(\varphi\vee\psi)\neq AF\varphi\vee AF\psi$ and dually $EG(\varphi\vee\psi)\neq EG\varphi\vee EG\psi$.

Again, we introduce a negation normal form (NNF) for CTL formulas. As in the case of LTL, we only consider the negation normal form which uses the Until- and Release-operators.

Definition 2.23. A CTL formula is in negation normal form (NNF) if it can be constructed from the atomic and negated atomic (state) formulas $\mathtt{tt}$, $\mathtt{ff}$, p, $\neg p$ and inductively from state formulas φ,ψ and a path formula $\bar{\varphi}$ in NNF, giving the state formulas $\varphi\wedge\psi$, $\varphi\vee\psi$, $E\bar{\varphi}$ and $A\bar{\varphi}$ and path formulas $X\varphi$, $F\varphi$, $G\varphi$, $(\varphi U\psi)$, and $(\psi R\varphi)$.

As in the case of LTL, the translation into NNF is linear. The proof again uses the duality laws for the X, F, G, U and R operators.

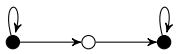
Lemma 2.24. Every CTL formula φ can be transformed into an equivalent CTL formula ψ in NNF with size $|\psi|\in O(\varphi)$.

Comparison between LTL and CTL

In this subsection, we discuss the relationship between LTL and CTL. For this, we evaluate an LTL formula over all paths of a tree. We further introduce the logic CTL* which allows arbitrary nesting of state and path formulas like $AFGp$ (and includes the logics LTL and CTL). Then, we present the restrictions ACTL* and ECTL* of CTL* which only use universal, respectively existential, quantified path formulas. We finish with CTL extended by a past version of the next operator.

Theorem 2.25. The logics LTL and CTL (both interpreted as tree logics) are incomparable in expressive power, i.e., there are LTL formulas which are not expressible in CTL and vice verse.

The proof of this theorem is shown in [BK08]. The authors give two formula pairs: the LTL formula FGp and the CTL formula $AFAGp$ and the pair $F(p \wedge Xp)$ and $AF(p \wedge AXp)$. They show that there is no CTL formula which is equivalent to the LTL formula FGp and no LTL formula which is equivalent to the CTL formula $AFAGp$. Here, we just give the intuition why FGp and $AFAGp$ are not equivalent. For this, consider the following transition system where p holds at the black nodes:



We look at the unravelling of this transition system shown in Figure 2.5 to determine the truth of the LTL formula FGp (over all paths) and the CTL formula $AFAGp$. For FGp, we have to check if there is a node on all paths from which on we have p on all nodes. The right outermost path satisfies this condition as it has only black nodes. On all other paths, we have a white node on the left successor of the nodes of the right outermost path and after this node only black nodes follow. Thus, FGp holds on all paths. For the formula $AFAGp$, we have to find on all paths a node such that all paths starting at this node satisfy Gp. For the right outermost path, we cannot find such a node as all nodes have a black and a white successor. Thus, $AFAGp$ does not hold. The same counter example can be used to show that $F(p \wedge Xp)$ and $AF(p \wedge AXp)$ are not equivalent.

We now introduce the logic *CTL**. The syntax is the same as for CTL only without the restriction that path formulas can only use state formulas as subformulas.

- As atomic CTL* formulas we have tt, ff and the predicates p for $p \in V$ are (atomic) CTL* state formulas.
- For CTL* state formulas φ, ψ and a CTL* path formula $\bar{\varphi}$, we get the CTL* state formulas $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, E\bar{\varphi}$ and $A\bar{\varphi}$.

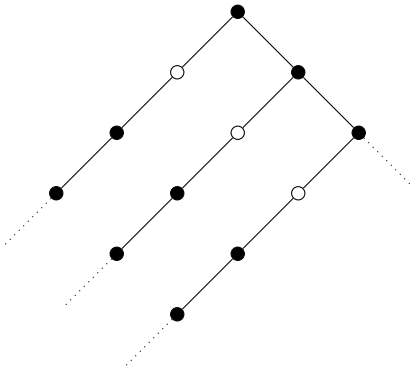


Figure 2.5: Counter Example Tree for FGp and $AFAGp$

- For a CTL* state formula φ and CTL* path formulas $\bar{\varphi}, \bar{\psi}$, we get CTL* path formulas φ, $X\bar{\varphi}$, $F\bar{\varphi}$, $G\bar{\varphi}$, $\bar{\varphi}U\bar{\psi}$, and $\bar{\psi}R\bar{\varphi}$.

The semantics of CTL* is defined as follows:

$$
\begin{array}{lcl}
s \models \mathtt{tt} & & \text{for all states } s \\
s \models \mathtt{ff} & & \text{for no state} \\
s \models p & :\Leftrightarrow & P_p(s) \text{ holds} \\
s \models \neg\varphi & :\Leftrightarrow & s \not\models \varphi \\
s \models \varphi \wedge \psi & :\Leftrightarrow & s \models \varphi \text{ and } s \models \psi \\
s \models \varphi \vee \psi & :\Leftrightarrow & s \models \varphi \text{ or } s \models \psi \\
s \models E\bar{\varphi} & :\Leftrightarrow & \text{there exists a path } \pi \text{ starting in } s \text{ with } \pi \models \bar{\varphi} \\
s \models A\bar{\varphi} & :\Leftrightarrow & \text{all paths } \pi \text{ starting in } s \text{ satisfy } \pi \models \bar{\varphi}
\end{array}
$$

$$
\begin{array}{lcl}
\pi[i..] \models \varphi & :\Leftrightarrow & s_i \models \varphi \text{ and } \pi = (s_0, s_1, \dots) \\
\pi[i..] \models X\bar{\varphi} & :\Leftrightarrow & \pi[i{+}1..] \models \bar{\varphi} \\
\pi[i..] \models F\bar{\psi} & :\Leftrightarrow & \exists k \geq i\, (\pi[k..] \models \bar{\psi} \\
\pi[i..] \models G\bar{\varphi} & :\Leftrightarrow & \forall j \geq i\, (\pi[j..] \models \bar{\varphi}) \\
\pi[i..] \models \bar{\varphi}U\bar{\psi} & :\Leftrightarrow & \exists k \geq i\, (\pi[k..] \models \bar{\psi} \text{ and } \forall j, i \leq j < k : \pi[j] \models \bar{\varphi}) \\
\pi[i..] \models \bar{\psi}R\bar{\varphi} & :\Leftrightarrow & \forall j \geq i\, (\pi[j..] \models \bar{\varphi}) \text{ or} \\
 & & \exists k \geq i\, (\pi[k..] \models \bar{\varphi} \wedge \bar{\psi} \text{ and } \forall j, i \leq j < k : \pi[j..] \models \bar{\varphi})
\end{array}
$$

CTL* extends both CTL and LTL. *ACTL** is the fragment of CTL* which contains all CTL* formulas which only allow universal quantification over the paths, i.e., for ACTL* state formulas φ, ψ and ACTL* path formula $\bar{\varphi}$ it contains $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$ and $A\bar{\varphi}$ as state formulas. The ACTL* path formulas and the semantics are defined as above. ECTL* is defined analogously to ACTL* by using only existential quantification over the paths.

All these logics only allow quantification over "future" states and paths. There is a theory which extends these results to the past by introducing past quantifiers for X, F, G, U and R, see [LS95]. From these logics, we only use the simplest extension of CTL by the past variant of X: the previous operator X^{-1}. Let $CTL{+}X^{-1}$ be the extension of CTL by X^{-1} defined by adding the state formula $X^{-1}\varphi$ for a state formula to the syntax of CTL with the semantics: $\pi[i] \models X^{-1}\varphi :\Leftrightarrow \pi[i-1] \models \varphi$. As in [LS95], we consider the past as fixed. Thus, we can simply write X^{-1} instead of EX^{-1} or AX^{-1}.

2.3 Interface Information

The aim of the composition technique is to reduce the truth of a formula in a product or sum to information about the truth values of formulas in the components of the product or sum. To express this information formally, we introduce interface information tuples.

An interface information tuple – also called determining sequence [Rab07] or MSO profile [FT09] in the literature – consists of formulas which are interpreted in the components and a formula which speaks about the component indices in which these formulas should hold. Formally, it is described as follows

Definition 2.26 (Interface information)**.** Let σ be a (relational) signature and Ind an index structure with index set I for the indices of the component σ-structures of a sum or product. An *interface information tuple* is a tuple $\rho = \langle \alpha_1, \ldots, \alpha_k; \beta \rangle$ of formulas α_j ($j \in [k]$) and a formula β. The formulas α_j are interpreted in the components and the formula β is interpreted in the index structure. The formula β allows to describe conditions in which components these formulas should hold. To achieve this, for every formula α_j ($j \in [k]$) the formula β speaks about an additional predicate which is interpreted by the set of indices of the components in which α_j holds.

If β is an MSO formula, we use free set variables $X_1, \ldots, X_k$ for these predicates. Formally, for given σ-components $\mathcal{A}_i$ ($i \in I$), we write $(Ind, I_1, \ldots, I_k) \models \beta(X_1, \ldots, X_n)$ with $I_j := \{i \in I \mid \mathcal{A}_i \models \alpha_j\}$ to denote that β holds in the index structure if the sets X_j are interpreted with the component indices in which α_j holds. If β is an LTL, CTL or CTL* formula, we use the predicates P_j as atomic formulas and P_j is interpreted by $\{i \in I \mid \mathcal{A}_i \models \alpha_j\}$. For easier notation, we also allow to use the formula α_j instead of the index j, i.e., we allow to write X_{α_j} for X_j and P_{α_j} for P_j.

We assume that we are given an interpretation of the predicates and relations in the components. Then, an interface information tuple $\rho = \langle \alpha_1, \ldots, \alpha_k; \beta \rangle$ with FO-

or MSO-formulas α_j defines a relation R_ρ of arity l in a product or a sum where l is the number of different variables all α_j use.

For a sum with $|I|$ components, we want to ensure that $(s_1, \ldots, s_l) \in R_\rho$ holds in the sum iff $(Ind, I_1, \ldots, I_k) \models \beta(X_1, \ldots, X_k)$ where I_j contains all components which satisfy α_j, i.e., $I_j := \{i \in I \mid \mathcal{A}_i \models \alpha_j(s_1, \ldots, s_l)\}$. For a product with $|I|$ components, we have $(\bar{s}_1, \ldots, \bar{s}_l) \in R_\rho$ in the product iff $(Ind, I_1, \ldots, I_k) \models \beta(X_1, \ldots, X_k)$ where I_j contains all components which satisfy the formula α_j for the i-th component of the states $\bar{s}_m$ ($m \in [l]$), i.e., $I_j := \{i \in I \mid \mathcal{A}_i \models \alpha_j(\bar{s}_1[i], \ldots, \bar{s}_l[i])\}$.

On the one hand, interface information tuples can be used to specify relations in the product as we will see below in the examples and in the definition of a synchronized product of transition systems. On the other hand, in the composition theorem, we inductively construct an interface information tuple to describe when a formula $\varphi(x_1, \ldots, x_l)$ holds in the sum or product, i.e., it is used to describe the l-ary relation which is defined by $\varphi(x_1, \ldots, x_l)$.

We consider two examples with FO logic over the components and MSO logic over the index structure.

Example 2.5. In the following examples, we consider FO logic interpreted over component transition systems which only have one (unary) predicate P and one binary (transition) relation R. In the (direct) product, the interface information tuple $\rho_1 = \langle P(x); \exists i X_1(i) \rangle$ expresses that there is at least one component where $P(x)$ holds. In a product, this is the set $\{\bar{x} \mid \exists i : P(\bar{x}[i]) \text{ holds}\}$. For a product of transition systems, the interface information tuple $\rho = \langle R(x,y), x = y; \exists i\, X_1(i) \wedge \forall j (j \neq i) \rightarrow X_2(j) \rangle$ expresses that there is exactly one component in which a transition is taken and in the other components the state stays the same. Thus, this interface information tuple describes an asynchronous transition relation in the product, i.e., the set $\{(\bar{x}, \bar{y}) \mid R(\bar{x}[i], \bar{y}[i]) \wedge \bar{x}[j] = \bar{y}[j] \text{ for all } j \neq i\}$.

We consider a third example with LTL over the components.

Example 2.6. We look at an infinite disjoint ordered sum of (finite) words with predicate r and LTL over the components and FO logic over the indices. We consider the interface information tuple $\langle r, G(\neg \text{last} \rightarrow Xr); \beta \rangle$ with $\beta = \forall i (P_2(i) \wedge \exists j (\text{Suc}(i,j) \wedge P_1(j)))$. This expresses that we have Xr on all states of the sum which is equivalent to a predicate with "GXr" at the first state of the sum. Note that we can express the same condition with LTL over the indices by using $\beta = G(P_2 \wedge XP_1)$.

In the literature, there are several ways to describe the "size" of an interface information tuple. We take the size of the formulas $\alpha_1, \ldots, \alpha_k$ for the components, their number k and the size of the index formula into account.

Definition 2.27. The *size of an interface information tuple* $\langle \alpha_1, \ldots, \alpha_k; \beta(X_1, \ldots, X_k) \rangle$ is defined as $\sum_{j=1}^{k} s(\alpha_j) + s(\beta)$ where $s(\alpha_j)$ and $s(\beta)$ denote the size of α_j, respectively β.

The composition theorems in the later chapters describe the truth value of a formula in the sum or product by an interface information tuple. The *size of the decomposition* of a given formula is defined as the size of the generated interface information tuple.

Product Definition via Interface Information

The next definition introduces synchronized products. For this definition, we consider "local" and "synchronous" transitions *in the components* for which we use two disjoint subsets Σ_l and Σ_s of alphabets. In the product, the local transitions result in asynchronous transitions, i.e., only one component takes a transition and the other components stay at the same state. Moreover, we use interface information tuples to describe "synchronized" transitions in the product where a subset of the components takes a synchronous transition at the same time. (Formally, this is done via a mapping τ from the alphabet for the synchronized transitions to interface information tuples.)

Definition 2.28. Let Ind be an index structure for the components (of a product) with index set I. Moreover, let $\Sigma := \Sigma_l \dot\cup \Sigma_s$ and V be alphabets and σ be a signature (for the components) with transition relations R_a for $a \in \Sigma$ and (unary) predicates P_v for $v \in V$. Furthermore, let C be another alphabet and σ' be a signature (for the product) with transition relations $\bar{R}^i_a$ for $a \in \Sigma_l$, $\bar{R}_c$ for $c \in C$ and predicates $\bar{P}^i_v$ for $v \in V$ and $i \in I$.

We further assume that we are given for every $c \in C$ an interface information tuple $\langle R_{a_1}(x,y), \ldots, R_{a_m}(x,y), x = y; \beta(X_1, \ldots, X_{m+1}) \rangle$ with an MSO formula $\beta(X_1, \ldots, X_{m+1})$ that specifies the conditions in which component which transition has to be taken. Furthermore, it satisfies $\beta_{\text{Partition}}(X_1, \ldots, X_{m+1}; I)$ (i.e., it assures that every component takes at most one synchronous transition).

We call these tuples *disjoint* and use a mapping τ from C to these interface information tuples to denote the corresponding interface information tuples.

For σ-transition systems $\mathcal{K}_i = (S_i, \{R^i_a \mid a \in \Sigma\}, \{P^i_v \mid v \in V\})$ for $i \in I$, the synchronized (σ', τ)-product $\prod_{i \in I} \mathcal{K}_i$ is the structure $\mathcal{K} = (\bar{S}, \{\bar{R}_a \mid a \in \Sigma_l\}, \{\bar{R}_c \mid c \in C\}, \{\bar{P}^i_v \mid v \in V\})$ with

- domain $\bar{S} := \{\bar{s} \mid \bar{s} : I \dashrightarrow \bigcup_{i \in I} S_i \text{ with } \bar{s}[i] \in S_i\}$ where $\bar{s}[i]$ denotes the state of the i-th component,

- asynchronous transition relations $\bar{R}_a$ ($a \in \Sigma_l$), i.e., $(\bar{x}, \bar{y}) \in \bar{R}_a \Leftrightarrow \exists i : (\bar{x}[i], \bar{y}[i]) \in R_a^i \wedge \forall j \neq i : \bar{x}[j] = \bar{y}[j]$,
- synchronized transition relations $\bar{R}_c$ ($c \in C$) defined by the given interface information tuples $\tau(c)$ and
- predicates $\bar{P}_v^i = \{\bar{s} \mid \bar{s}[i] \in P_v^i\}$.

We call a product *asynchronous* if we have no synchronized transitions in the product, i.e., if C is empty. Furthermore, if the transition relation $\bar{R}_c$ is finite for every $c \in C$, the product is called *finitely-synchronized*.

Note that the asynchronous transition relation $\bar{R}_a$ of the product can be described by the interface information tuple $\langle R_a(x, y), x = y; \beta(X_1, X_2)\rangle$ with $\beta(X_1, X_2) = \exists i : (X_1(i) \wedge \forall j : j \neq i \rightarrow X_2(j))$. Furthermore, the predicate $\bar{P}_v^i$ can be described by $\langle P_v(x); \beta(X_1)\rangle$ with $\beta(X_1) = X_1(i)$.

3 The Composition Method

In this chapter, we first repeat (in Section 3.1) the model theoretic background which is used in the classical composition theorems of Feferman and Vaught and Shelah. In Sections 3.2 and 3.3, we present the composition theorems and give a short sketch of the proofs. We conclude the chapter in Section 3.4 with known results on the complexity of these composition approaches.

3.1 Model Theoretic Background

In this section, we first introduce Hintikka formulas, which are FO formulas that describe all possible structures for a given signature. Then, we discuss a similar definition for MSO logic. These formulas are used in the following sections in the proofs of the classical composition theorems of Feferman and Vaught for products of structures and of Shelah for sums of orderings.

Hintikka Formulas

A common way to describe how similar two structures are, is to look at the FO sentences which they satisfy: two structures $\mathcal{A}, \mathcal{B}$ are *elementary k-equivalent* if no FO formula of quantifier rank k can distinguish between them. They are called *elementary equivalent* if they are elementary k-equivalent for all $k \in \mathbb{N}$.

For a given signature, the class of all structures which are elementary k-equivalent to a given structure $\mathcal{A}$ can be described by a special FO formula: all structures with m marked elements that are elementary k-equivalent to $\mathcal{A}$ with marked elements $a_1, \ldots, a_m$ are described by the FO formula $\varphi^k_{\mathcal{A},\bar{a}}(\bar{x})$ with $\bar{a} = a_1, \ldots, a_m$ and $\bar{x} = x_1, \ldots, x_m$. It is defined, in Definition 3.1, by induction over the number of quantifiers k. In the literature, these formulas are called *Hintikka formulas* [EF05] or *game-normal formulas* [Hod97].

For quantifier rank 0, the formula $\varphi^0_{\mathcal{A},\bar{a}}(\bar{x})$ states all atomic (and negated atomic) properties which hold between the elements of $\bar{a}$. Formally, this is simply a conjunction of all atomic or negated atomic formulas with free variables $x_1, \ldots, x_m$ which hold for $a_1, \ldots, a_m$. In the induction step for quantifier rank $k+1$, the formula

considers all possible valuations of the new variable y by elements a of $\mathcal{A}$. On the one hand, it states that there is an element which behaves like a and on the other hand, that there are no other than these elements.

Definition 3.1 (Hintikka formulas for a structure $\mathcal{A}$)**.** For a structure $\mathcal{A}$ with domain A, the *Hintikka formulas for* $\mathcal{A}$ are inductively defined as follows.

$$\varphi^0_{\mathcal{A},\bar{a}}(\bar{x}) = \bigwedge\{\varphi(\bar{x}) \mid \varphi \text{ atomic or negated atomic formula, } (\mathcal{A},\bar{a}) \models \varphi(\bar{x})\}$$
$$\varphi^{k+1}_{\mathcal{A},\bar{a}}(\bar{x}) = \bigwedge\{\exists y\, \varphi^k_{\mathcal{A},\bar{a},a}(\bar{x},y) \mid a \in A\} \wedge \forall y \bigvee\{\varphi^k_{\mathcal{A},\bar{a},a}(\bar{x},y) \mid a \in A\}$$

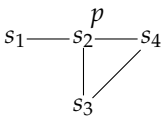


Figure 3.1: Transition System from Example 3.1

Example 3.1. We consider the transition system $\mathcal{K}$ with state set $S = \{s_1, \ldots, s_4\}$, transition relation R and unary predicate P as shown in Figure 3.1. We assume that we always have $\neg R(s_i, s_i)$ for all $i \in [4]$ and that $R(s_i, s_j)$ holds iff $R(s_j, s_i)$ holds for all $i, j \in [4]$. To simplify notation, we shorten the Hintikka formulas by leaving out the formulas which become redundant under these preliminaries. For the element $\bar{a} = (s_1, s_2)$, we get the formula $\varphi^0_{\mathcal{K},\bar{a}}(x_1, x_2) = \neg(x_1 = x_2) \wedge R(x_1, x_2) \wedge \neg P(x_1) \wedge P(x_2)$. Furthermore, we get $\varphi^0_{\mathcal{K},(s_1,s_3)}(x_1, x_2) = \varphi^0_{\mathcal{K},(s_1,s_4)}(x_1, x_2) = \neg(x_1 = x_2) \wedge \neg R(x_1, x_2) \wedge \neg P(x_1) \wedge \neg P(x_2)$. With these formulas, $\varphi^1_{\mathcal{K},s_1}(x_1)$ is defined as $\exists x[x_1 \neq x \wedge R(x_1, x) \wedge \neg P(x_1) \wedge P(x)] \wedge \exists x\,[x_1 \neq x \wedge \neg R(x_1, x) \wedge \neg P(x_1) \wedge \neg P(x)] \wedge \forall x\,[(x_1 \neq x \wedge R(x_1, x) \wedge \neg P(x_1) \wedge P(x)) \vee (x_1 \neq x \wedge \neg R(x_1, x) \wedge \neg P(x_1) \wedge \neg P(x))]$.

As a second example we consider the finite word $w = 10111$ with its word model $\underline{w} = (\{1, \ldots, 5\}, <, Q)$, such that $Q(i)$ holds for all positions except the second one. As above we shorten the notation by skipping formulas of the form $\neg(x_i < x_i)$. Then, $\varphi^0_{\underline{\mathbf{W}},(3,1)} = Q(x_1) \wedge Q(x_2) \wedge x_2 < x_1 \wedge \neg(x_1 < x_2)$, $\varphi^0_{\underline{\mathbf{W}},(3,2)} = Q(x_1) \wedge \neg Q(x_2) \wedge x_2 < x_1 \wedge \neg(x_1 < x_2)$ and $\varphi^0_{\underline{\mathbf{W}},(3,4)} = \varphi^0_{\underline{\mathbf{W}},(3,5)} = Q(x_1) \wedge Q(x_2) \wedge x_1 < x_2 \wedge \neg(x_1 < x_2)$. The formula $\varphi^1_{\underline{\mathbf{W}},3}$ is a conjunction over the existence of these possibilities and states that these are the only possible elements.

So far, we only considered Hintikka formulas for given structures. *General Hintikka formulas* sum up the Hintikka formulas for all structures: for a fixed

signature they consider all theoretically possible formulas which can be constructed with the given signature. There are only finitely many different formulas (up to equivalence) because the signature is finite. Consider, e.g., the signature of the transition system from the example above, then we get for $k = 0$ and two variables all possible conjunctions of $x_i = x_j$, $R(x_i, x_j)$ and $P(x_i)$ for $i, j \in \{1, 2\}$ and their negations. For each quantifier rank k and free variables $x_1, \ldots, x_m$, these formulas are stored as a set Φ^k_m. Note that in general, we may also produce unsatisfiable formulas like $\neg(x_1 = x_1)$ in the other example above. These formulas cannot be satisfied by any structure, but they do not harm in the set of all possible formulas. The sets Φ^k_m are formally defined as follows.

Definition 3.2 (General Hintikka formulas)**.** Let $\{\varphi_1(\bar{x}), \ldots, \varphi_l(\bar{x})\}$ with the abbreviation $\bar{x} = (x_1, \ldots, x_m)$ denote the set of all (positive) atomic formulas with m free variables. Then, Φ^0_m is the set of all conjunctions of these formulas, respectively their negations: $\Phi^0_m := \{\tau(1)\varphi_1(\bar{x}) \wedge \cdots \wedge \tau(l)\varphi_l(\bar{x}) \mid \tau : \{1, \ldots, l\} \to \{\epsilon, \neg\}\}$. Given $\Phi^k_{m+1} = \{\psi_1, \ldots, \psi_j\}$, the set Φ^{k+1}_m is defined as $\{\bigwedge_{i \in S} \exists x\, \psi_i(\bar{x}, x) \wedge \forall x \bigvee_{i \in S} \psi_i(\bar{x}, x) \mid S \subseteq \{1, \ldots, j\}\}$.

We state known facts about these general Hintikka formulas, known as the Fraïssé-Hintikka theorem (see, e.g., [Hod97, Theorem 3.3.2]). For every given signature σ, quantifier rank k and number of free variables m, we have:

- Every σ-structure $\mathcal{A}$ together with m elements of $\mathcal{A}$ satisfies exactly one formula ψ of the general Hintikka formulas from Φ^k_m if the free variables in ψ are interpreted by the given m elements.

- Two σ-structures $\mathcal{A}, \mathcal{B}$ together with m-tuples $\bar{a}$, respectively $\bar{b}$, of elements from A, respectively B, are k-equivalent iff they satisfy the same general Hintikka formula (interpreted with the elements from $\bar{a}$, respectively $\bar{b}$).

- Every FO formula of quantifier rank $\leq k$ with m free variables is equivalent to a disjunction of Hintikka formulas from the set Φ^k_m.

In the next sections, we use an enumeration of these general Hintikka formulas as α_j with $j \in [k]$ in interface information tuples $\langle \alpha_1, \ldots, \alpha_k; \beta(X_1, \ldots, X_k) \rangle$. We show that the truth of a formula in a product or sum can be reduced to the "satisfaction" of such an interface information tuple, i.e., that the MSO formula holds, if the sets X_j are interpreted by the component indices for which the components satisfy the formulas α_j.

Hintikka Formulas for MSO Logic

The notion of Hintikka formulas can be generalized from FO logic to MSO logic. This is done according to [Tho97a]. Technically, MSO formulas which contain only set variables are considered. Recall that each MSO formula can be translated into one with only set variables (see Section 2.2). For the inductive construction of the MSO Hintikka formulas, we use the quantifier *alternation* depth as induction parameter, i.e., we use blocks of only existential, respectively only universal[1], (set) quantifiers in one step.

With these preliminaries, the definition of the MSO Hintikka formulas follows straight-forwardly from the definition of the FO Hintikka formulas. For the definition, we use some abbreviations: let $()$ denote the empty tuple. Given a tuple $\bar{l} = (l_1, \ldots, l_k)$ of natural numbers l_j, let $(\bar{l}, l_{k+1})$ denote the tuple $(l_1, \ldots, l_k, l_{k+1})$. Furthermore, let $\bar{P}$ be an abbreviation for the tuple $\bar{P}_1, \ldots, \bar{P}_m$ and $\bar{X}$ for the tuple $\bar{X}_1, \ldots, \bar{X}_m$.

Definition 3.3 (MSO Hintikka formulas for a structure $\mathcal{A}$)**.** For a structure $\mathcal{A}$ with domain A, the *MSO Hintikka formulas for* $\mathcal{A}$ are inductively defined as follows.

$$\varphi^{()}_{\mathcal{A},\bar{P}}(\bar{X}) = \bigwedge\{\varphi(\bar{X}) \mid \varphi \text{ atomic or negated atomic formula, } (\mathcal{A}, \bar{P}) \models \varphi(\bar{X})\}$$

$$\varphi^{\bar{l},l_{k+1}}_{\mathcal{A},\bar{P}}(\bar{X}) = \bigwedge\{\exists \bar{Y}\, \varphi^{\bar{l}}_{\mathcal{A},\bar{P},\bar{Q}}(\bar{X},\bar{Y}) \mid \bar{Q} \in \mathcal{P}(A)^{l_{k+1}}\} \wedge \forall \bar{Y} \bigvee\{\varphi^{\bar{l}}_{\mathcal{A},\bar{P},\bar{Q}}(\bar{X},\bar{Y}) \mid \bar{Q} \in \mathcal{P}(A)^{l_{k+1}}\}$$

Example 3.2. We consider a labeling of the natural numbers: the structure $(\mathcal{A}, \bar{P})$ with $\mathcal{A} = (\omega, <)$, $\bar{P} = (P_1, P_2)$ and predicates $P_1 = \{7\}$, $P_2 = \{7, 8\}$. For $\varphi^{()}_{\mathcal{A},P_1,P_2}(X_1, X_2)$, we get a concatenation of the atomic formulas which hold for P_1 and P_2: $\text{Nonempty}(X_1), \text{Nonempty}(X_2), \text{Nonempty}(X_1 \cap X_2), X_1 \subseteq X_2, \neg X_2 \subseteq X_1, \neg X_1 < X_1, X_2 < X_2, X_1 < X_2, \neg X_2 < X_1, \neg X_1 = \text{All}, \neg X_2 = \text{All}, \neg X_1 \cup X_2 = \text{All}$. We get different combinations of these formulas for the following choices of P_2: $\{6\}$, $\{7\}$, $\{8\}$, $\{5,6\}$, $\{6,7\}$, $\{6,8\}$, $\{7,8\}$, $\{8,9\}$, $\{6,7,8\}$, $\emptyset$, $\mathbb{N}$, $\mathbb{N} \setminus \{7\}$. All other choices are equivalent to one of these combinations. We name these formulas $\varphi_1, \ldots, \varphi_{12}$. Now we consider the MSO Hintikka formula $\varphi^{(1)}_{\mathcal{A},P_1}$: we get $\varphi^{(1)}_{\mathcal{A},P_1} = \bigwedge_{j=1}^{12} \exists Y \varphi_j(X_1, Y) \wedge \forall Y \bigvee_{i=1}^{12} \varphi_j(X_1, Y)$.

[1] To be more precise, by using the equivalence $\forall X \varphi(X) = \neg\exists X \neg\varphi(X)$, we consider only blocks of existential quantifiers, but have to start with a new block when a negation appears.

General MSO Hintikka formulas are defined as the general Hintikka formulas and as in the case of (FO) Hintikka formulas, every MSO formula can be written as a disjunction over elements of the set of all possible general MSO Hintikka formulas.

There is a more compact representation of the (general) MSO Hintikka formulas which avoids mentioning each possible formula $\varphi^{\bar{l}}_{\mathcal{A},\bar{P},\bar{Q}}(\bar{X},\bar{Y})$ in the inductive Definition 3.3 twice: the representation by $\bar{l}$-types, where the MSO Hintikka formulas are represented inductively as sets of previous sets. The formal definition is as follows:

Definition 3.4. The *$\bar{l}$-type $T^{\bar{l}}(\mathcal{A},\bar{P})$ of a structure $\mathcal{A}$ with predicates $\bar{P}=\{P_1,\ldots,P_m\}$* is defined by

- $T^{()}(\mathcal{A},\bar{P}) = \{\varphi(\bar{X}) \mid \varphi(\bar{X}) \text{ atomic and } (\mathcal{A},\bar{P}) \models \varphi(\bar{X})\}$
- $T^{\bar{l},l_{k+1}}(\mathcal{A},\bar{P}) = \{T^{\bar{l}}(\mathcal{A},\bar{P},\bar{Q}) \mid \bar{Q} \in \mathcal{P}(\mathcal{A})^{l_{k+1}}\}$

The induction definition starts for the $()$-type with all possible sets of atomic formulas with m free second-order variables which hold in the structure $\mathcal{A}$ with the predicates $P_1,\ldots,P_m$. In the induction step, all possible subsets of the last set with $m+l_{k+1}$ free variables are taken.

Example 3.3. Let us consider again the structure $(\mathcal{A},\bar{P})$ with $\mathcal{A}=(\omega,<)$, $\bar{P}=(P_1,P_2)$ and predicates $P_1=\{7\}$, $P_2=\{7,8\}$. Then, the type $T^{()}(\mathcal{A},\bar{P})$ is the set $\{\text{Nonempty}(X_1),\text{Nonempty}(X_2),\text{Nonempty}(X_1\cap X_2),X_1\subseteq X_2,X_2<X_2,X_1<X_2\}$. As $T^{(1)}(\mathcal{A},(P_1))$, we get a set which contains the sets $T^{()}(\mathcal{A},\bar{P})$ for $\bar{P}=(P_1,P_2)$ with the 12 possible choices for P_2 from the last example.

As in the generalization from Hintikka to general Hintikka formulas, we introduce *general $\bar{l}$-types* for all possible structures. For this, we consider all possible atomic formulas with m free second-order variables in the base of the inductive definition and all possible subsets of the previous sets.

Definition 3.5. The *general $\bar{l}$-type* $T^{\bar{l}}(m)$ is defined by

- $T^{()}(m) := \mathcal{P}(\{\varphi(X_1,\ldots,X_m) \mid \varphi(X_1,\ldots,X_m) \text{ atomic}\})$
- $T^{\bar{l},l_{k+1}}(m) := \mathcal{P}(T^{\bar{l}}(m+l_{k+1}))$

This set representation of the general Hintikka formulas is used in the composition theorem for disjoint ordered sums by Shelah.

3.2 Composition Theorem for Products

In this section, we present the classical Feferman-Vaught composition theorem for (direct) products and FO logic (in our notation). It allows to deduce the truth of an FO formula in a product of structures from information about the components: which component satisfies which of the general Hintikka formulas. This information can be expressed by a Boolean formula (as in the original paper) or by an MSO formula (expressing the Boolean conditions). We present it here as an MSO formula as we use this format also in our own results.

Theorem 3.6. For a given signature σ, let $\alpha_1(\bar{x}), \ldots, \alpha_k(\bar{x})$ be the general Hintikka formulas for quantifier rank r and with a tuple $\bar{x}$ of free variables.

For a σ-FO formula $\varphi(\bar{x})$ with quantifier rank r, we can compute an MSO formula $\beta_\varphi(X_1, \ldots X_k)$ – interpreted in the index structure Ind – such that for every direct σ-product $\prod_{i\in I} A_i$ of σ-components A_i the following holds:

$$\prod_{i\in I} A_i \models \varphi(\bar{a}) \Leftrightarrow (I, I_1, \ldots, I_k) \models \beta_\varphi(X_1, \ldots, X_k)$$

where I_j is the set $\{i \in I : A_i \models \alpha_j(\bar{a}[i])\}$.

We want to remark that Feferman and Vaught also showed a composition theorem for more general products.

We now give a sketch of the proof idea of the composition theorem for direct products. The theorem is proven by an induction over the structure of the given FO formula $\varphi(\bar{x})$.

For the induction base, $\varphi(\bar{x})$ is an atomic (or negated atomic) formula and the general Hintikka formulas are conjunctions of atomic formulas. We consider the subset of the general Hintikka formulas which contain $\varphi(\bar{x})$ as a conjunct. The MSO formula for φ expresses that every component satisfies one of these formulas.

In the induction step for the conjunction and disjunction of FO formulas φ, ψ of the same quantifier rank, the MSO formula is $\beta_\varphi \wedge \beta_\psi$ respectively $\beta_\varphi \vee \beta_\psi$. If the formulas φ, ψ have different quantifier rank, the one with lower quantifier rank is transformed into an equivalent formula with the higher quantifier rank and the Hintikka formulas are adjusted accordingly. For negation, we take the negation of the MSO formula.

We now consider the existential quantification $\exists y \varphi(\bar{x}, y)$. Let $\alpha_1(\bar{x}, y), \ldots, \alpha_k(\bar{x}, y)$ be the given general Hintikka formulas (of quantifier rank $r-1$) for $\varphi(\bar{x}, y)$. Furthermore, let $\mu_j(\bar{x})$, $j \in [2^k]$ denote the general Hintikka formulas (of quantifier rank r) for $\exists y \varphi(\bar{x}, y)$. Recall that $\mu_j(\bar{x})$ is defined as $\exists y \bigwedge_{h\in S_j} \alpha_h(\bar{x}, y) \wedge \forall y \bigvee_{h\in S_j} \alpha_h(\bar{x}, y)$

where S_j denotes the j-th set in the list of all subsets of $[k]$. From the induction hypothesis, we are given the MSO formula β_φ over the sets $Y_1, \ldots, Y_k$. The MSO formula $\beta_{\exists y \varphi(\bar{x},y)}$ over the sets $X_1, \ldots, X_{2^k}$ expresses that the index set I can be partitioned into sets $Y_1, \ldots, Y_k$, such that $\beta_\varphi(Y_1, \ldots, Y_k)$ holds and for all $h \in [k]$: $Y_h \subseteq \bigcup_{h \in S_j} X_j$. In [Hod97] it is shown that this choice actually works.

3.3 Composition Theorem for Sums

In this section, we show the composition theorem for ordered disjoint sums and MSO logic by Shelah. The theorem is presented according to [Tho97a] and works on the general $\bar{l}$-types as defined in Section 3.1. It reduces the general $\bar{l}$-type of the sum to general types of the index structure which is expanded by sets I_{τ_j} containing the components which satisfy the $\bar{r}$-type τ_j. Note that this allows to reduce the truth of any formula in the sum to corresponding general types of the index structure and the components because every formula can be represented by the types it satisfies.

Theorem 3.7. Let $(I, <)$ be an index ordering and $(\mathcal{A}_i, \bar{P}_i)$ for $i \in I$ be component structures with predicates $\bar{P}_i = \{P_1^i, \ldots, P_m^i\}$. Furthermore, let $\tau_1, \ldots, \tau_s$ denote an enumeration of the $\bar{l}$-types of the components (which are all contained in the general $\bar{l}$-type). Given $\bar{l} = (l_1, \ldots, l_k)$ and $m \in \mathbb{N}$, we can compute $\bar{r} = (r_1, \ldots, r_k)$ such that for every ordered disjoint sum $\sum_{i \in I}(\mathcal{A}_i, \bar{P}_i)$ the $\bar{l}$-type of the sum is determined by the $\bar{r}$-type of the index structure $(I, <)$ which is expanded by the sets $I_j = \{i \in I \mid (\mathcal{A}_i, \bar{P}_i)$ has the type $\tau_j\}$ for $j \in [s]$.

For the proof sketch of the theorem, we first show how $\bar{r}$ is constructed from $\bar{l}$ and then show the construction of the types for the index structure and the components by induction over the general types in the sum.

The value of $\bar{r}$ is inductively defined as follows: for empty sequences $\bar{l} = ()$, we have $\bar{r} = ()$. For non-empty sequences $\bar{l}' = (\bar{l}, l_{k+1})$ and m free MSO variables, we have $\bar{r}' = (\bar{r}, r')$ where $\bar{r}$ is the sequence assigned to $\bar{l}$ for $m + l_{k+1}$ free MSO variables and r' is $|T^{\bar{l}}(m + l_{k+1})|$.

Note that $\bar{l}$ and $\bar{r}$ have the same *number of entries* whereas the single entries of $\bar{r}$ can be larger than those of $\bar{l}$. This is the reason why the quantifier alternation depth (and not the quantifier depth) was used in the definition of MSO Hintikka formulas respectively their type-representation.

The proof of the theorem uses an induction over k. For $k = 0$, we have $\bar{l} = ()$ and atomic formulas. The kinds of atomic formulas are Nonempty$(X_i \cap X_j)$, $X_i \subseteq X_j$, $X_i < X_j$ and $X_{i_1} \cup \cdots \cup X_{i_j} = \text{All}$.

- The formula $\text{Nonempty}(X_i \cap X_j)$ holds in the ordered disjoint sum – i.e., we have $\text{Nonempty}(X_i \cap X_j) \in T^{()}(\sum_{h \in I}(\mathcal{A}_h, \bar{P}_h))$ – if and only if there is at least one component index l such that in this component $\text{Nonempty}(X_i \cap X_j)$ holds, i.e., $\text{Nonempty}(X_i \cap X_j) \in T^{()}(\mathcal{A}_l, \bar{P}_l)$. Thus, there has to be (at least) one non-empty set I_k for a type τ_k with $\text{Nonempty}(X_i \cap X_j) \in \tau_k$. This is equivalent to that there exists a k with $\text{Nonempty}(X_k) \in T^{()}(I, <, I_1, \ldots, I_s)$ and $\text{Nonempty}(X_i \cap X_j) \in \tau_k$.

- The formula $X_i \subseteq X_j$ holds in the ordered disjoint sum if and only if only components are taken where $X_i \subseteq X_j$ holds, i.e., $X_i \subseteq X_j \in \tau_k \Leftrightarrow \text{Nonempty}(X_k) \in T^{()}(I, <, I_1, \ldots, I_s)$.

- The formula $X_i < X_j$ holds in the ordered disjoint sum if it either holds in one component or if X_i and X_j are used in two components where X_i is used in a component before that where X_j is used (and X_i and X_j are not empty). Thus, we have $X_i < X_j \in T^{()}(\sum_{h \in I}(\mathcal{A}_h, \bar{P}_h))$ iff either $X_i < X_j \in \tau_k$ and $\text{Nonempty}(X_k) \in T^{()}(I, <, I_1, \ldots, I_s)$ or $\text{Nonempty}(X_i) \in \tau_k$ and $\text{Nonempty}(X_j) \in \tau_{k'}$ and $X_k < X_{k'} \in T^{()}(I, <, I_1, \ldots, I_s)$.

- The formula $X_{i_1} \cup \cdots \cup X_{i_j} = \text{All}$ holds in the sum iff it holds in all components, i.e., $X_{i_1} \cup \cdots \cup X_{i_j} = \text{All} \in \tau_k \Leftrightarrow \text{Nonempty}(X_k) \in T^{()}(I, <, I_1, \ldots, I_s)$.

We now consider the induction step. Recall that $T^{(\bar{l}, l_k+1)}(\sum_{h \in I}(\mathcal{A}_h, \bar{P}_h))$ contains all types of $T^{\bar{l}}(\sum_{h \in I}(\mathcal{A}_h, \bar{P}_h, \bar{R}_h))$ for all possibilities of $\bar{R}_h := R^h_1, \ldots, R^h_{l_k+1}$. By induction hypothesis, we can compute each $T^{\bar{l}}(\sum_{h \in I}(\mathcal{A}_h, \bar{P}_h, \bar{R}_h)$ for a specify choice of $\bar{R}_h$ from an expanded index structure $(I, <, I_1, \ldots, I_r)$ where $\tau_1, \ldots, \tau_r$ is the enumeration of all types from the general type $T^{\bar{l}}(m + l_{k+1})$ which hold for the components for this choice of $\bar{R}_h$.

The goal is to collect all these $\bar{r}$-types for the index structure which are induced by all possible choices of $\bar{R}_h$. By definition, the type $T^{(\bar{r}, r_{k+1})}(I, <, I_1, \ldots, I_s)$ (with $I_h = \{i \in I \mid T^{(\bar{l}, l_{k+1})}(\mathcal{A}_i, \bar{P}_i) = \tau_h\}$) contains types $T^{\bar{r}}(I, <, I_1, \ldots, I_s, J_1, \ldots, J_t)$ where $t = r_{k+1} = |T^{\bar{l}}(m + l_{k+1})|$ and $\sigma_1, \ldots, \sigma_t$ are all types from the general type $T^{\bar{l}}(m + l_{k+1})$.

The main proof idea is to establish compatibility conditions over these $\sigma_1, \ldots, \sigma_t$ and to choose only those $\bar{r}$-types which satisfy them. Let $X_1, \ldots, X_s$, $Y_1, \ldots, Y_t$ denote the variables used in the types for the sets $I_1, \ldots, I_s$, $J_1, \ldots, J_t$. Then, the compatibility conditions are:

- Each component $(\mathcal{A}_i, \bar{P}_i, \bar{R}_i)$ has to satisfy exactly one[2] of the types $\sigma_1, \ldots, \sigma_t$. This is achieved by forcing that the sets $J_1, \ldots, J_t$ form a partition, formally: $Y_1 \cup \cdots \cup Y_t = \text{All}$ and $Y_i \cap Y_j = \emptyset$ for all i, j.
- For each $i \in I_h$, i.e., a component of type $\tau_h = T^{(\bar{l}, l_{k+1})}(\mathcal{A}_h, \bar{P}_h)$, the index i also has to be in all J_j with $\sigma_j = T^{(\bar{l})}(\mathcal{A}_j, \bar{P}_j, \bar{R}_j)$ for any $\bar{R}_j$. This is captured by $\text{Nonempty}(X_h \cap Y_j)$.

It can be shown that these conditions are not only necessary but also sufficient to describe the type $T^{(\bar{l}, l_{k+1})}(\sum_{i \in I}(\mathcal{A}_i, \bar{P}_i))$. For details of the proof see [Tho97a].

3.4 Size of the Decomposition

In the last sections, we have only seen the classical composition theorems for products and ordered sums. There are several variants of composition theorems, especially composition theorems for less expressive logics like modal logic, see, e.g., Rabinovich [Rab07]. A good overview of the results can be found in Makowsky [Mak04].

Remember that the size of the decomposition for a given formula was defined as the size of the generated interface information tuple, see Definition 2.27. Note that both the constructions for products and sums in the last two sections lead to a non-elementary size of the decomposition, because in the induction step for the existential quantification we consider all combinations of the (general) Hintikka formulas from the previous step. Thus, every existential quantification leads to an exponential increase of the number of formulas, resulting, in total, in a non-elementary size of the decomposition. In the literature [DGKS07, GJL12], it has been shown that this is unavoidable for almost all combinations of logics and products or sums: Dawar, Grohe, Kreutzer and Schweikardt showed a non-elementary lower bound for the size of the decomposition for FO logic and products. This result has been extended by Göller, Jung and Lohrey to modal logic and transferred to disjoint ordered sums and FO logic. We briefly present here their results and give a sketch of the proof. Note that the proof uses that the structures – here: transition systems – may have an unbounded degree of outgoing transitions.

Theorem 3.8. In general, for asynchronous products and modal logic the size of the decomposition is non-elementary in the size of the given formula. Furthermore, this holds also for any logic which is at least as expressive as modal logic.

[2] Note that on the other hand there may be types which are not satisfied by any component. (There are even types which are not satisfiable at all, e.g., if $X_i < X_j$ is in a type but $\text{Nonempty}(X_i)$ is missing.)

We only present a sketch of the proof here. Let $\mathrm{Tower}(l, n)$ be defined by $\mathrm{Tower}(0, n) = n$ and $\mathrm{Tower}(l, n) = 2^{\mathrm{Tower}(l-1,n)}$. The main idea of the proof is to encode large structures by small formulas. For this, trees are constructed in which each node at height l has $\mathrm{Tower}(l, n)$ successors. To be more precise, a (l, n)-treelike structure is defined where the root has $\mathrm{Tower}(l, n)$ successors which are all $(l-1, n)$-treelike. These trees are an extension of the trees defined in [DGKS07] by additional predicates. In particular, a predicate P_b is added which is used to assign a value to each of these trees. To be more precise, the predicate P_b at all successors together encodes a natural number in binary which is defined as the value of the tree. Then, formulas are defined which compare the predicates of two such trees for each level (l, n). Afterwards, using these formulas, a formula $\varphi_{l,n}$ is defined which checks that two such trees have the same value (via the P_b). This can be used in a binary asynchronous product to check that both components have the same value.

Now, the proof that the size of the decomposition is non-elementary is shown by contradiction. One uses an elementary function from the size of the formula $\varphi_{l,n}$ to the number of formulas for the components. Then, there is some (l, n) from which on there are less formulas for the components than there are different trees. Thus, two trees satisfy the same formulas. Assuming that the composition theorem holds, this would mean that $\varphi_{l,n}$ holds for these trees although they have different values, which contradicts the definition of $\varphi_{l,n}$.

The result from above is used to derive a non-elementary lower bound for the composition theorem of FO logic and disjoint ordered sums of two components. Furthermore, it is also shown that this holds if FO logic with only three variables – denoted as FO^3 logic – is used.

Theorem 3.9. In general, for disjoint ordered sums and FO^3 logic the size of the decomposition is non-elementary in the size of the given formula.

We want to mention that this result does not hold for FO^2 logic. In [GJL12], the authors found a double exponential upper bound for the case of FO^2.

4 Composition for Products and FO Logic extended by Paths with Counting

In Sections 3.2 and 3.3, we have seen composition theorems for products and sums of general structures. In the field of model-checking, we are mainly interested in special structures – transition systems. Instead of looking at direct products of arbitrary relational structures, in this chapter, we look at synchronized products of transition systems. In these products, we may have both asynchronous and synchronous behaviour between the components. To be more precise, we may have an a-transition between two states in the product if exactly one component takes an a-transition (asynchronous behaviour) or all components take an a-transition (synchronous behaviour) or a mixture of both called synchronized behaviour, where a fixed subset of components takes an a-transition and the other components stay at the same state. A special case of a synchronized product is a finitely-synchronized product which contains only a finite number of synchronized transitions.

In Chapter 3, we have seen that the composition theorem for disjoint ordered sums is stated for MSO logic while the composition theorem for products is stated for FO logic. It is easy to see that we can not generalize the composition theorem for products to MSO logic. For this, consider an asynchronous product of two copies of the natural numbers with successor. It is well-known that the MSO theory of the natural numbers with successor relation is decidable, but the MSO theory of their product – which is the infinite grid – is undecidable.

This raises two questions:

1. Does the composition technique for products work for stronger logics than FO logic?

2. Which logical properties cause the technique to fail?

Both questions have been addressed in the literature. On the one hand, Wöhrle and Thomas [WT07] looked at the extension of FO logic by (unconditioned) reachability, i.e., by relations $\mathrm{Reach}(x, y)$ which express that there exists a path fragment from x to y. For arbitrary synchronized products, already this extension leads to a failure of the composition theorem, but for finitely-synchronized products the

composition theorem is applicable for this logic. Note that (unconditioned) reachability corresponds to the CTL quantifier EF. On the other hand, Rabinovich [Rab07] showed that the composition theorem fails even for asynchronous products if the logic can express the CTL quantifier EG. Furthermore, in [WT07], the failure of the composition technique for asynchronous products and regular reachability – i.e., reachability by path fragments which are labeled according to a regular language – described by a regular expression – was shown.

In this chapter, we take a deeper look at where the actual limit for the composition theorem for synchronized products is. We show that for the subclass of finitely synchronized products we can extend FO logic by a conditioned version of reachability: FO(Reg1R) – FO logic augmented by modulo counting over the length of the paths. We also consider general regular reachability (over any finite alphabet). Here, we present an alternative proof to [WT07] which shows the failure of the composition technique for regular reachability over an alphabet of two labels and asynchronous products.

For the result that the composition theorem is also applicable for FO logic extended by modulo counting over the path length, we generalize the proof of the composition theorem by Rabinovich – shown in [Rab07] – which is a slight variation of the original proof by Feferman and Vaught. (The original result by Feferman and Vaught has a lot of redundancy as it considers all theoretically possible (Hintikka) formulas and does not take the current input formula into account. For example, also if a predicate P is not used in the input formula, the atomic Hintikka formulas contain all conditions with p as well as with $\neg p$ as a conjunct. The construction of Rabinovich avoids this and uses only the formulas which are relevant for the given input formula.)

We begin this section by adapting the proof of Rabinovich to FO logic. (He considers modal logic.) Then, we present our extension of the proof to capture FO logic with predicates which allow to express reachability with modulo counting over the length of the path fragment. This will be shown first for asynchronous products and afterwards for finitely-synchronized products. We continue with the extension to regular reachability and show the failure of the composition technique in this case. We conclude with a summary and an outlook. As we only speak about path fragments in this chapter, we simply write path instead of path fragment to ease notation.

4.1 Composition Theorem for Synchronized Products of Transition Systems

For the following theorem, we use as in Definition 2.28 an index structure Ind with index set I. Moreover, let $\Sigma := \Sigma_l \dot{\cup} \Sigma_s$ be an alphabet for the local and synchronous relations in the components, V for the (unary) predicates and C for the synchronized transitions in the product. Furthermore, let σ be the signature which contains the relation symbols R_a for $a \in \Sigma$ and P_v for $v \in V$ and $\bar{\sigma}$ the signature which contains the relation symbols $\bar{R}_a$ for $a \in \Sigma_l$, $\bar{R}_c$ for $c \in C$ and $\bar{P}_v^i$ for $v \in V$ and $i \in I$. Moreover, let τ denote a mapping from C to disjoint interface information tuples.

Theorem 4.1. For every given $\bar{\sigma}$-FO formula $\gamma(\bar{x}_1, \ldots, \bar{x}_r)$, we can effectively compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(Z_1, \ldots, Z_m) \rangle$ with σ-FO formulas $\gamma_j(x_1, \ldots, x_r)$ for $j \in [m]$ and an MSO formula $\beta(Z_1, \ldots, Z_m)$ interpreted in the index structure Ind, such that for all synchronized $(\bar{\sigma}, \tau)$-products $K = (\bar{S}, \{\bar{R}_a \mid a \in \Sigma_l\}, \{\bar{R}_c \mid c \in C\}, \{\bar{P}_v^i \mid v \in V\})$ of transition systems $K_i = (S_i, \{R_a^i \mid a \in \Sigma\}, \{P_v^i \mid v \in V\})$ $(i \in I)$ and for every state tuple $(\bar{s}_1, \ldots, \bar{s}_r)$ with $\bar{s}_j \in \bar{S}$:

$$(K, \bar{s}_1, \ldots, \bar{s}_r) \models \gamma(\bar{x}_1, \ldots, \bar{x}_r) \iff Ind \models \beta(I_1, \ldots, I_m)$$

with $I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \gamma_k(x_1, \ldots, x_r)\}$ for $k \in [m]$.

We preprocess the formula by replacing every universal quantifier by $\forall x \varphi = \neg \exists x \neg \varphi$. Then, the proof uses a structural induction over the input formula.

Induction base:
For the induction base, we have to find interface information tuples which describe the atomic formulas, i.e., the asynchronous and synchronized transition relations $\bar{R}_a, \bar{R}_c$, the predicates $\bar{P}_v^i$ and the equivalence $\bar{x} = \bar{y}$.

The asynchronous transition relation $\bar{R}_a$ for $a \in \Sigma_l$ can be described by the interface information tuple $\langle R_a(x, y), x = y; \beta(X_1, X_2) \rangle$ with $\beta(X_1, X_2) = \exists i : (X_1(i) \wedge \forall j (j \neq i \rightarrow X_2(j)))$. For the synchronized transition relations $\bar{R}_c$ with $c \in C$, the interface information tuples are given by τ. The interface information tuple $\langle P_v(x); \beta(X_1) \rangle$ with $\beta(X_1) = X_1(i)$ defines the predicate $\bar{P}_v^i$ for $v \in V$ and $i \in I$. Furthermore, the equivalence relation $\bar{x} = \bar{y}$ can be described by the interface information tuple $\langle x = y; \forall i X_1(i) \rangle$.

Induction hypothesis:
The induction hypothesis states that for every subformula $\varphi(\bar{x}_1, \ldots, \bar{x}_{r'})$ of the

current formula $\delta(\bar{x}_1, \ldots, \bar{x}_r)$ we have an interface information tuple $\langle \varphi_1, \ldots, \varphi_m; \beta_\varphi(X_1, \ldots, X_m) \rangle$ for φ such that for every state tuple $(\bar{s}_1, \ldots, \bar{s}_{r'})$ with $\bar{s}_j \in \bar{S}$:

$$(K, \bar{s}_1, \ldots, \bar{s}_{r'}) \models \varphi(\bar{x}_1, \ldots, \bar{x}_{r'}) \iff Ind \models \beta_\varphi(I_1, \ldots, I_m)$$

with $I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_{r'}[i]) \models \varphi_k(x_1, \ldots, x_{r'})\}$ for $k \in [m]$.

Induction step:
In the induction step, we have to consider negation, disjunction and existential quantification.

For the negation of the given formula, we simply have to negate the MSO formula of the interface information tuple. So for $\delta(\bar{x}_1, \ldots, \bar{x}_r) = \neg\varphi(\bar{x}_1, \ldots, \bar{x}_r)$ and a given interface information tuple $\langle \varphi_1, \ldots, \varphi_m; \beta_\varphi(X_1, \ldots, X_m) \rangle$ for φ, we get $\langle \delta_1, \ldots, \delta_m; \beta_\delta(X_1, \ldots, X_m) \rangle$ with $\delta_j = \varphi_j$ for $j \in [m]$ and $\beta_\delta(X_1, \ldots, X_m) = \neg\beta_\varphi(X_1, \ldots, X_m)$ as the interface information tuple for δ.

For the disjunction of two given formulas φ and ψ, we take the disjunction of the index formula of their interface information tuples with a small preparation to ensure that φ and ψ use the same free variables. Consider a free variable x which is used in ψ but not in φ. We can simply add the formula $x = x$ to the list of component formulas of the interface information tuple of φ and shift the indices of the sets $X_1 \ldots, X_m$ according to the insert position without changing its satisfiability. So, w.l.o.g., we can assume that φ and ψ have the same free variables. Let $\langle \varphi_1, \ldots, \varphi_m; \beta_\varphi(X_1, \ldots, X_m) \rangle$ be the interface information tuple for $\varphi(\bar{x}_1, \ldots, \bar{x}_r)$ and $\langle \psi_1, \ldots, \psi_{m'}; \beta_\psi(X_1, \ldots, X_{m'}) \rangle$ for $\psi(\bar{x}_1, \ldots, \bar{x}_r)$. The interface information tuple for δ is $\langle \varphi_1, \ldots, \varphi_m, \psi_1, \ldots, \psi_{m'}; \beta_\delta(Z_1, \ldots, Z_m, Z_{m+1}, \ldots, Z_{m+m'}) \rangle$ with $\beta_\delta = \beta_\varphi(Z_1, \ldots, Z_m) \vee \beta(Z_{m+1}, \ldots, Z_{m+m'})$. The construction for the conjunction of two given formulas is analogous.

For the existential quantification, we need a preparation of the given interface information tuple. In order to find existentially quantified formulas for the components, we guarantee that each component satisfies exactly one of the formulas. This will be done by ensuring that the conjunction of two of the formulas for the components is unsatisfiable and the disjunction over all of the formulas is true.

Lemma 4.2. For every interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(X_1, \ldots, X_m) \rangle$, there exists an equivalent interface information tuple $\langle \delta_1, \ldots, \delta_{2^m}; \beta(Y_1, \ldots, Y_{2^m}) \rangle$ that fulfills the following properties: $\delta_i \wedge \delta_j$ is unsatisfiable for all $i, j \in [2^m]$ and $\bigvee_{i=1}^{2^m} \delta_i$ is valid, i.e., the sets $Y_1, \ldots, Y_{2^m}$ have to form a partition.

Proof. To show the lemma, we start by defining the interface information tuple $\langle \delta_1, \ldots, \delta_{2^m}; \beta(Y_1, \ldots, Y_{2^m}) \rangle$. For a subset $H \subseteq [m]$, let $\delta'_H := \bigwedge_{i \in H} \gamma_i \wedge \bigwedge_{i \notin H} \neg\gamma_i$. We

use a listing of the formulas δ'_H ($H \subseteq [m]$) for the formulas $\delta_1, \ldots, \delta_{2^m}$. This listing is chosen by an arbitrary bijective mapping $f : \mathcal{P}([m]) \to [2^m]$ from the subsets to the indices of the formulas δ_i. Let $\delta_i := \delta'_H$ iff $f(H) = i$. In $\beta(Y_1, \ldots, Y_{2^m})$, we guarantee that there are sets $X_1, \ldots, X_m$ such that the original formula holds for these sets and (by using the mapping f) that $z \in X_i$ iff $z \in Y_H$ for one of the sets H which contain i:

$$\beta(Y_1, \ldots, Y_{2^m}) = \exists X_1 \ldots \exists X_m [\bigwedge_{i=1}^{m} \forall z (X_i(z) \leftrightarrow \bigvee_{\{k \mid i \in f^{-1}(k)\}} Y_k(z)) \wedge \beta(X_1, \ldots, X_m)]$$

The interface information tuple $\langle \delta_1, \ldots, \delta_{2^m}; \beta(Y_1, \ldots, Y_{2^m}) \rangle$ defines the same relation as $\langle \gamma_1, \ldots, \gamma_m; \beta(X_1, \ldots, X_m) \rangle$: we have for all $i \in [m]$ that $\bigvee_{\{H \subseteq [m] \mid i \in H\}} \delta'_H = \gamma_i$, because of $\bigvee_{\{H \subseteq [m] \mid i \in H\}} \delta'_H = \bigvee_{\{H \subseteq [m] \mid i \in H\}} (\bigwedge_{j \in H} \gamma_j \wedge \bigwedge_{j \notin H} \neg \gamma_j)$ which is equal to $\bigwedge_{j \in [m], j \neq i} (\bigvee_{\{H \subseteq [m] \mid j \in H\}} \gamma_j \vee \bigvee_{\{H \subseteq [m] \mid j \notin H\}} \neg \gamma_j) \wedge \bigvee_{\{H \subseteq [m] \mid i \in H\}} \gamma_i = \mathrm{tt} \wedge \gamma_i$.

It remains to show the properties: $\delta_i \wedge \delta_j$ is unsatisfiable for all $i, j \in [2^m]$ and $\bigvee_{i=1}^{2^m} \delta_i$ is valid. We first show that $\bigvee_{H \subseteq [m]} \delta'_H$ is valid. As we consider all subsets of H, there is a subset $G \subseteq [m]$ such that exactly the formulas γ_i with $i \in G$ are satisfied and the formulas γ_i with $i \notin G$ not. Thus, by definition δ'_G holds. Now consider two different subsets $G_1, G_2 \in [m]$. As they are different, there exists at least one index $i \in G_1$ and $i \notin G_2$ (or vice verse). Thus, $\delta'_{G_1} \wedge \delta'_{G_2}$ can not be satisfied because it contains γ_i and $\neg \gamma_i$ in the conjunction. □

We continue with the existential quantification in the inductive proof of the composition theorem. For $\delta(\bar{x}_1, \ldots, \bar{x}_r) = \exists \bar{x}_{r+1} \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x}_{r+1})$, we have to ensure that the formulas $\exists x \varphi_j(x_1, \ldots, x_r, x)$ for the components are unambiguous, i.e., that each component satisfies exactly one of the formulas. This is done as follows: we first apply Lemma 4.2 to ensure that every component satisfies one of the formulas $\varphi_j(x_1, \ldots, x_r, x_{r+1})$. Let $\langle \varphi_1, \ldots, \varphi_m; \beta_\varphi(Y_1, \ldots, Y_m) \rangle$ be the converted interface information tuple for φ which we get by applying the lemma. Then, we have $\varphi_i \wedge \varphi_j$ is unsatisfiable for all $i, j \in [m]$ and $\bigvee_{i=1}^{m} \varphi_i$ is valid. (Note that by applying the lemma, we get an exponential growth of the number of formulas.)

If we now consider the formulas $\exists x \varphi_1(x_1, \ldots, x_r, x), \ldots, \exists x \varphi_m(x_1, \ldots, x_r, x)$ we may no longer have $\exists x \varphi_i \wedge \exists x \varphi_j$ is unsatisfiable and $\bigvee_{i=1}^{m} \exists x \varphi_i$ is valid. Thus, we have to assure these conditions again by a partition condition over these formulas. This is done as follows: we define the interface information tuple for $\delta(\bar{x}_1, \ldots, \bar{x}_r) = \exists \bar{x} \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$ as $\langle \delta_1, \ldots, \delta_m; \beta_\delta(X_1, \ldots, X_m) \rangle$ with $\delta_i := \exists x \varphi_i(x_1, \ldots, x_r, x)$ and $\beta_\delta(X_1, \ldots, X_m)$ as the formula $\exists Z_1 \ldots \exists Z_m [\bigwedge_{k \in [m]} Z_k \subseteq X_k \wedge \beta_{\text{Partition}}(Z_1, \ldots, Z_m; I) \wedge \beta_\varphi(Z_1, \ldots, Z_m)]$.

We now show that this choice for the existential quantification is correct. For this, let $(K, \bar{s}_1, \ldots, \bar{s}_r) \models \delta(\bar{x}_1, \ldots, \bar{x}_r) = \exists \bar{x} \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$. We show that the index structure fulfills β_δ of the interface information tuple $\langle \delta_1, \ldots, \delta_m; \beta_\delta(X_1, \ldots, X_m) \rangle$ as defined above if X_k is interpreted as $I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \exists x \varphi_k(x_1, \ldots, x_r, x)\}$. From $(K, \bar{s}_1, \ldots, \bar{s}_r) \models \exists \bar{x} \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$, we get that there exists a state $\bar{s}$ in the product such that $(K, \bar{s}_1, \ldots, \bar{s}_r, \bar{s}) \models \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$ holds. By induction hypothesis, we know that this holds iff the MSO formula $\beta_\varphi(Y_1, \ldots, Y_m)$ of the interface information tuple for φ (which is already in the converted form by Lemma 4.2) is satisfied for $J_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots \bar{s}_r[i], \bar{s}[i]) \models \varphi_k(x_1, \ldots, x_r, x)\}$. Note that $J_k \subseteq I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \exists x \varphi_k(x_1, \ldots, x_r, x)\}$ holds. By Lemma 4.2, we also have that $J_k \cap J_{k'} = \emptyset$ for every $k, k' \in [m]$ and $\bigcup_{i=1}^{m} J_i = I$, i.e., that $\beta_{\text{Partition}}(J_1, \ldots, J_m; I)$ holds. Thus, the sets J_k for $k \in [m]$ can be used as interpretation for the sets Z_k in $\beta_\delta(X_1, \ldots, X_m)$ as they satisfy all the demanded properties. We conclude that $\beta_\delta(X_1, \ldots, X_m)$ is satisfied for I_k as defined above.

For the other direction, let the MSO formula $\beta_\delta(X_1, \ldots, X_m)$ of the interface information tuple $\langle \delta_1, \ldots, \delta_m; \beta_\delta(X_1, \ldots, X_m) \rangle$ be satisfied for the sets I_k, again, defined as $\{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \exists x \varphi_k(x_1, \ldots, x_r, x)\}$. By definition of $\beta_\delta(X_1, \ldots, X_m)$, there are sets J_k with $J_k \subseteq I_k$ and $\beta_{\text{Partition}}(J_1, \ldots, J_k; I)$ that satisfy the condition $\beta_\varphi(Y_1, \ldots, Y_m)$ from the interface information tuple for φ. From $J_k \subseteq I_k$ we know that also for all $i \in J_k$ we have $(K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \exists x \varphi_k(x_1, \ldots, x_r, x)$. So for every $k \in [m]$ and every $i \in J_k$ there exists a state $s_i \in S_i$ such that $(K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i], s_i) \models \varphi_k(x_1, \ldots, x_r, x)$. From $\beta_{\text{Partition}}(J_1, \ldots, J_k; I)$ we get that for every $i \in I$ there is exactly one $k \in [m]$ with $(K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i], s_i) \models \varphi_k(x_1, \ldots, x_r, x)$. So together, these states s_i for $i \in I$ define a state $\bar{s}$ with $\bar{s}[i] = s_i$ such that the formula $\beta_\varphi(Y_1, \ldots, Y_m)$ of the interface information tuple $\langle \varphi_1, \ldots, \varphi_m; \beta_\varphi(Y_1, \ldots, Y_m) \rangle$ is satisfied for $\{i \in I \mid (K_i, \bar{s}_1[i], \ldots \bar{s}_r[i], \bar{s}[i]) \models \varphi_k(x_1, \ldots, x_r, x)\}$. By induction hypothesis this holds iff $(K, \bar{s}_1, \ldots, \bar{s}_r, \bar{s}) \models \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$. Thus, $(K, \bar{s}_1, \ldots, \bar{s}_r) \models \exists \bar{x} \varphi(\bar{x}_1, \ldots, \bar{x}_r, \bar{x})$ holds.

4.2 Composition Theorem for Asynchronous Products and Counting over the Path Length

In this and the next section, we discuss modulo counting over the path length. We first generalize the composition theorem from the last section to capture this property for the special case of asynchronous products.

Let Ind be an index structure with index set I. Moreover, let Σ be an alphabet for the local relations in the components and V for the (unary) predicates. Furthermore,

let σ be the signature which contains the relation symbols R_a for $a \in \Sigma$ and P_v for $v \in V$ and $\bar{\sigma}$ the signature which contains the relation symbols $\bar{R}_a$ for $a \in \Sigma_l$ and $\bar{P}_v^i$ for $v \in V$ and $i \in I$.

Theorem 4.3. For every $\bar{\sigma}$-FO(Reg1R) formula $\gamma(\bar{x}_1, \ldots, \bar{x}_r)$, we can effectively compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(Z_1, \ldots, Z_m) \rangle$ with σ-FO(Reg1R) formulas $\gamma_j(x_1, \ldots, x_r)$ for $j \in [m]$ and a CMSO formula $\beta(Z_1, \ldots, Z_m)$ interpreted in the index structure Ind, such that for the asynchronous $\bar{\sigma}$-product $K = (\bar{S}, \{\bar{R}_a \mid a \in \Sigma\}, \{\bar{P}_v^i \mid v \in V\})$ of σ-transition systems $K_i = (S_i, \{R_a^i \mid a \in \Sigma\}, \{P_v^i \mid v \in V\})$ $(i \in I)$ and for every state tuple $(\bar{s}_1, \ldots, \bar{s}_r)$ we have:

$$(K, \bar{s}_1, \ldots, \bar{s}_r) \models \gamma(\bar{x}_1, \ldots, \bar{x}_r) \iff Ind \models \beta(I_1, \ldots, I_m)$$

with $I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \gamma_k(x_1, \ldots, x_r)\}$ for $k \in [m]$.

As the asynchronous product is a special case of a finitely-synchronized product, we extend the inductive proof of Theorem 4.1 by the case for the additional atomic formulas $\mathrm{Path}_{l,k}(\bar{x}, \bar{y})$. Recall that FO(Reg1R) is equivalent to FO logic with additional predicates $\mathrm{Path}_{l,k}$. So we have to find interface information tuples for the atomic formulas $\mathrm{Path}_{l,k}(\bar{x}, \bar{y})$ in the product. Note that for the special case of an asynchronous product the length of a path in the product is simply the sum of the lengths of the segments of this path in the components.

The main idea is as follows: we use formulas $\mathrm{Path}_{0,k}(x, y), \ldots, \mathrm{Path}_{k-1,k}(x, y)$ in every component to count the fragment of the path length modulo k in this component. Note that $\mathrm{Path}_{0,k}(x, y)$ can be used for components with a path length divisible by k and for components with no path between x and y at all, i.e., for $x = y$. With these formulas, we can count via the CMSO formula of the interface information tuple the number of components which have a path length $0, \ldots, k-1$ modulo k. We ensure that the components with path lengths $1, \ldots, k-1$, together, create a path of length l modulo k via a formula β' by expressing the solutions of an equation in MSO logic.

We now present the formal definition. Let $|Y|_k$ denote the number of elements of Y modulo k and let $[k]-1 := \{0, \ldots, k-1\}$. For every $l \in [k]-1$ the interface information tuple has the following form: $\langle \mathrm{Path}_{0,k}(x, y), \ldots, \mathrm{Path}_{k-1,k}(x, y);$

$\beta(X_0, \ldots, X_{k-1})\rangle$ with

$$\beta(X_0, \ldots, X_{k-1}) := \exists Y_0 \ldots \exists Y_{k-1} : \bigwedge_{j \in [k]-1} Y_j \subseteq X_j \wedge \beta_{\text{Partition}}(Y_0, \ldots, Y_{k-1}; I) \wedge \beta'(Y_1, \ldots, Y_{k-1}) \wedge \bigwedge_{j \in \{1,\ldots,k-1\}} \bigvee_{r \in [k]-1} \text{Card}_{r,k}(Y_j)$$

where β' (defined later) describes the condition that the path lengths for the components from $Y_1, \ldots, Y_{k-1}$ sum up to l.

We explain the other parts of β first: If X_j is interpreted by $\{i \in I \mid (K_i, \bar{x}[i], \bar{y}[i]) \models \text{Path}_{j,k}(x, y)\}$ for all $j \in [k] - 1$, then the formula means that there are (sub-)sets Y_j which satisfy $\text{Path}_{j,k}(x, y)$ such that every component is in exactly one of these sets, the formula $\beta'(Y_1, \ldots, Y_{k-1})$ holds and the sets Y_j are finite for $j \in 1, \ldots, k-1$. (The set Y_0 does not need to be finite because it includes the possibility that $x = y$.)

To count the path length modulo k in the product, we observe that for $j \in \{1, \ldots, k-1\}$, each component with a path segment of length j modulo k adds $j \pmod k$ to the whole path. We have to ensure that these segment lengths sum up to $l \pmod k$. In other words, we have to ensure $\sum_{j \in \{1,\ldots,k-1\}} j \cdot y_j = l \pmod k$ with $y_j = |Y_j|_k \pmod k$.

Note that the variables y_j are in the range $\{0, \ldots, k-1\}$. So we get finitely many solutions for the equation. Let L be the set of solutions

$$L := \{(z_1, \ldots, z_{k-1}) \mid 1 * z_1 + \cdots + (k-1) * z_{k-1} = l \pmod k \wedge \forall j : z_j \in [k-1]\}$$

We use a description of these solutions in MSO logic as formula $\beta'(Y_1, \ldots, Y_{k-1})$:

$$\bigvee_{(z_1,\ldots,z_{k-1}) \in L} (\text{Card}_{z_1,k}(Y_1) \wedge \cdots \wedge \text{Card}_{z_{k-1},k}(Y_{k-1}))$$

4.3 Composition Theorem for Finitely Synchronized Products and Counting over the Path Length

In this section, we generalize the result from the last section to finitely synchronized products.

Again, let Ind be an index structure with index set I. Moreover, let $\Sigma := \Sigma_l \dot{\cup} \Sigma_s$ be an alphabet for the local and synchronous relations in the components, V for the (unary) predicates and C for the synchronized transitions in the product. Furthermore, let σ be the signature which contains the relation symbols R_a for $a \in \Sigma$ and

P_v for $v \in V$ and $\bar{\sigma}$ the signature which contains the relation symbols $\bar{R}_a$ for $a \in \Sigma_l$, $\bar{R}_c$ for $c \in C$ and $\bar{P}_v^i$ for $v \in V$ and $i \in I$. Furthermore, let τ denote a mapping from C to disjoint interface information tuples.

Theorem 4.4. For every $\bar{\sigma}$-FO(Reg1R) formula $\gamma(\bar{x}_1, \ldots, \bar{x}_r)$ and $q \in \mathbb{N}$, we can effectively compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(Z_1, \ldots, Z_m) \rangle$ with σ-FO(Reg1R) formulas $\gamma_j(x_1, \ldots, x_r)$ $(j \in [m])$ and a CMSO formula $\beta(Z_1, \ldots, Z_m)$ interpreted in the index structure Ind, such that we have: If the synchronized $(\bar{\sigma}, \tau)$-product $K = (\bar{S}, \{\bar{R}_a \mid a \in \Sigma_l\} \cup \{\bar{R}_c \mid c \in C\}, \{\bar{P}_v^i \mid v \in V\})$ of transition systems $K_i = (S_i, \{R_a^i \mid a \in \Sigma\}, \{P_v^i \mid v \in V\})$ $(i \in I)$ is finitely synchronized with q synchronized transitions, then for every state tuple $(\bar{s}_1, \ldots, \bar{s}_r)$, the following equivalence holds:

$$(K, \bar{s}_1, \ldots, \bar{s}_r) \models \gamma(\bar{x}_1, \ldots, \bar{x}_r) \iff Ind \models \beta(I_1, \ldots, I_m)$$

with $I_k = \{i \in I \mid (K_i, \bar{s}_1[i], \ldots, \bar{s}_r[i]) \models \gamma_k(x_1, \ldots, x_r)\}$ for $k \in [m]$.

Note that the generated interface information tuple for γ depends on the number q of synchronized transitions in the product. Before we show the theorem, we first explain the proof ideas: The proof works by induction over the number of synchronized transitions in the product. For one synchronized transition in the product, we consider the paths in the components and split each path into asynchronous parts and the usage of this synchronized transition. We count the length of the asynchronous segments (before and after taking the synchronized transition) in each component modulo k and create for each component their sum (modulo k). Then, we add these sums for all components and add the number of times the synchronized transition has been taken. It has to be counted only once for every repetition as it is taken together in all components. The idea is the same for a fixed number of synchronized transitions. Here, we observe that the usage of r synchronized transitions can be described by segments using only $r - 1$ synchronized transitions and the usage of the r-th synchronized transition.

Proof. Induction base:
We show the theorem for the case of only one synchronized transition in the product. Let $C = \{\bar{c}_1\}$ and $\{(\bar{s}_1, \bar{t}_1)\} = R_{\bar{c}_1}$ be the set which contains this transition. To simplify notation, we assume that $\bar{c}_1$ is defined as a synchronization *between all components* and the transitions in the components are labeled by c_1. (The generalization to only a subset of components which are synchronized via potentially different symbols is straightforward.)

For asynchronous products, we already have an MSO formula over the index structure that can express that two states are reachable in the product via a path with a length $l \pmod k$ (by summing up the paths in the components). In a synchronized product with one synchronized transition[1], we have the following cases: First, we may also have in all components paths which use only asynchronous transitions which sum up to a path of length $l \pmod k$. Second, we may have in all components paths which use exactly once a c_1-transition. Third, we may have in all components paths which use a c_1-transition more than once but in all components the same number of times. In the later two cases, we have to ensure that the c_1-transitions are only counted once (per repetition) for the whole path length as all components take them "at the same time".

We take a deeper look at the paths in all components with the c_1-transitions (see also Figure 4.1). In the second case where at least one c_1-transition (from state s_1 to t_1) is taken, we must have the following situation in each component: y is reachable from x iff there is a path from x to s_1, then the c_1 transition is taken which leads us to t_1 and there is a path from t_1 to y.

This must also be the case in the third case, but here we must also have a path from t_1 back to s_1. To simplify notation, we consider here only the case that there is at most one path from t_1 back to s_1. This situation is shown in Figure 4.1. To capture the general situation "more paths from t_1 back to s_1" (or to be more precise paths of different lengths), the following construction has to be generalized which we skip to improve readability. However, the generalization is straightforward and we give hints at the corresponding points how it can be realized. Note that for the paths from t_1 back to s_1, we may have at most k different path lengths because we count modulo k.

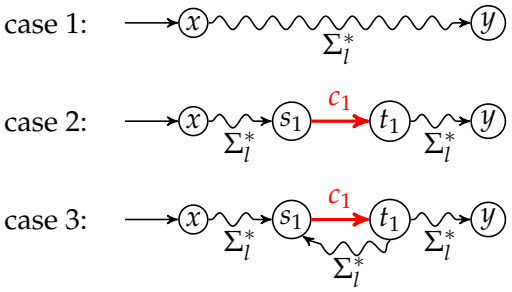


Figure 4.1: Path Using One Synchronization Transition per Component

For a component, let $\mathrm{Path}_{i,k}^{l}(x, y)$ denote that there is a path from x to y which uses only asynchronous transitions (i.e., from Σ_l) and which has the length $i \pmod k$.

[1] Recall that we have a $\bar{c}_1$-transition in the product if all components take a c_1-transition.

We can express the cases 1,2 and 3 from above by FO(Reg1R)-formulas $\alpha_{i_1}(x,y)$, $\alpha_{i_1 i_2}(x,y)$ and $\alpha_{i_1 i_2 i_3}(x,y)$ for $i_1, i_2, i_3 \in \{0,\ldots,k-1\}$:

- $\alpha_{i_1}(x,y) := \mathrm{Path}^l_{i_1,k}(x,y)$,
- $\alpha_{i_1 i_2}(x,y) := \exists s_1 \exists t_1 (\mathrm{Path}^l_{i_1,k}(x,s_1) \wedge R_{c_1}(s_1,t_1) \wedge \mathrm{Path}^l_{i_2,k}(t_1,y))$ and
- $\alpha_{i_1 i_2 i_3} := \exists s_1 \exists t_1 (\mathrm{Path}^l_{i_1,k}(x,s_1) \wedge R_{c_1}(s_1,t_1) \wedge \mathrm{Path}^l_{i_3,k}(t_1,s_1) \wedge \mathrm{Path}^l_{i_2,k}(t_1,y))$.

For the generalization to more paths from t_1 back to s_1, we would have here $k-1$ additional cases for all paths of different path lengths modulo k.

The interface information for $\mathrm{Path}_{l,k}(\bar{x},\bar{y})$ in the product is defined analogously to the last section by an interface information tuple with the formulas $\alpha_{i_1}(x,y)$, $\alpha_{i_1 i_2}(x,y)$ and $\alpha_{i_1 i_2 i_3}(x,y)$ and an MSO formula $\beta(X_0,\ldots,X_{k-1},X_{00},\ldots,X_{k-1k-1},$ $X_{000},\ldots,X_{k-1k-1k-1})$ expressing that there are subsets $Y_0,\ldots,Y_{k-1k-1k-1}$ that form a partition and (by β' defined below) that the path lengths sum up correctly:

$$\exists Y_0 \ldots \exists Y_{k-1k-1k-1} : Y_0 \subseteq X_0 \wedge \cdots \wedge Y_{k-1k-1k-1} \subseteq X_{k-1k-1k-1} \wedge \\ \beta_{\mathrm{Partition}}(Y_0,\ldots,Y_{k-1k-1k-1};I) \wedge \beta'(Y_0,\ldots,Y_{k-1k-1k-1})$$

(We consider here only the case that *all* components are synchronized. For the general case where τ defines the components which are synchronized the condition β_τ also has to be satisfied.)

For the definition of the formula $\beta'(Y_0,\ldots,Y_{k-1k-1k-1})$, we use functions f_1, f_2 and f_3 which map tuples (i_1), (i_1,i_2) and $(i_1,i_2,i_3;j)$ for the length of the segments of the asynchronous paths in a component and the repetition j of the path from t_1 back to s_1 to the sum of these paths (without counting the repetition of the synchronized c_1-transition – this is done later together for all components). The first index i_1 is used for the path x to s_1 (respectively y), the index i_2 for t_1 to y and i_3 for t_1 to s_1.

For $i_1, i_2, i_3 \in \{0,\ldots,k-1\}$ and $j \in \{1,\ldots,k-1\}$, the functions f_1,f_2 and f_3 are defined as:

- $f_1(i_1) := i_1 \pmod k$,
- $f_2(i_1,i_2) := i_1 + i_2 \pmod k$ and
- $f_3(i_1,i_2,i_3;j) := i_1 + i_2 + (i_3 * j \pmod k)$.

Because we count modulo k we get all possible path lengths by repeating the path from t_1 to s_1 at most $k-1$ times, so $j \leq k-1$ is sufficient. Furthermore, j

cannot be 0 as the case that there is no path back from t_1 to s_1 or there is one, but it is not taken is already covered by the second case.

For the three cases, we get the three following solutions. The intuition for $l_0, \ldots, l_{k-1k-1k-1}$ is that these are the number of components which have a path length of $f_1(0), \ldots, f_3(k-1, k-1, k-1; j)$:

- $L_1 = \{(l_0, \ldots, l_{k-1}) \mid f_1(0) \cdot l_0 + \cdots + f_1(k-1) \cdot l_{k-1} = l \pmod k\}$
- $L_2 = \{(l_{00}, \ldots, l_{k-1k-1}) \mid f_2(0,0) \cdot l_{00} + \cdots + f_2(k-1, k-1) \cdot l_{k-1k-1} + 1 = l \pmod k\}$
- $L_3 = \{(l_{000}, \ldots, l_{k-1k-1k-1}) \mid \exists j : f_3(0,0,0;j) \cdot l_{000} + \cdots + f_3(k-1, k-1, k-1; j) \cdot l_{k-1k-1k-1} + j = l \pmod k\}$

The "$+1$" in the second case and the "$+j$" in the third one count the number of times the c_1-transition is taken – once for all components. Note that the usage of only $f_3(i_1, i_2, i_3; j)$ for the same j ensures that the c_1-transition is taken the same number of times in all components.

We build the CMSO formula β' as in the asynchronous case:

$$\begin{aligned}
&\beta'(Y_0, \ldots, Y_{k-1}) = \\
&\bigvee_{(l_0,\ldots,l_{k-1}) \in L_1} \mathrm{Card}_{l_0,k}(Y_0) \wedge \cdots \wedge \mathrm{Card}_{l_{k-1},k}(Y_{k-1}) \vee \\
&\bigvee_{(l_{00},\ldots,l_{k-1k-1}) \in L_2} \mathrm{Card}_{l_{00},k}(Y_{00}) \wedge \cdots \wedge \mathrm{Card}_{l_{k-1k-1},k}(Y_{k-1k-1}) \vee \\
&\bigvee_{(l_{000},\ldots,l_{k-1k-1k-1}) \in L_3} \mathrm{Card}_{l_{000},k}(Y_{000}) \wedge \cdots \wedge \mathrm{Card}_{l_{k-1k-1k-1},k}(Y_{k-1k-1k-1})
\end{aligned}$$

Induction step:

We now discuss how the construction works for more than one synchronized transition. Recall that q denotes the total number of synchronized transitions in the product. Furthermore, recall the cases from Figure 4.1. For r synchronized transitions labeled by $\bar{c}_1, \ldots, \bar{c}_{r-1}, \bar{c}_r$ in the product for labels $c_1, \ldots, c_{r-1}, c_r$ in the components, we have that y is reachable from x if either we have a path from x to y using at most the $r-1$ transitions (labeled by $c_1, \ldots, c_{r-1}$) or we have paths from x to s_r, t_r to y and potentially from t_r back to s_r which use at most these $r-1$ transitions. Let Π_{r-1} denote the paths using the at most the $r-1$ synchronized transitions and Π_0 asynchronous paths, then we get the general inductive description for $\Pi_r, r \in [q]$ shown in Figure 4.2.

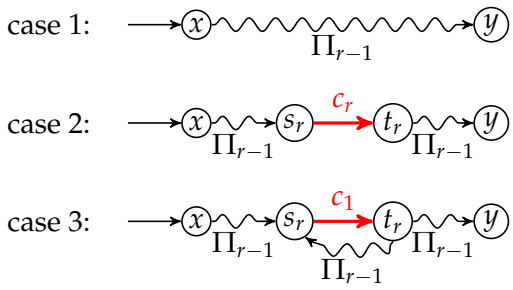


Figure 4.2: Path Using r Synchronization Transitions per Component

Note that, as for only one synchronized transition, this is a simplified view which considers only one path back from t_r to s_r for all $r \in [q]$. The generalization is analogous to the case in the induction base. Thus, by induction, it is possible to generate an interface information tuple for all paths using at most q synchronized transitions of the product by the technique as described above. □

4.4 Limits of the Composition Technique for Products

In this section, we show the limits of the composition technique for products. For this, let us recall the products and logics we already discussed. We present an overview of the combinations of these logics for which the composition technique is applicable in Table 4.1 in the summary of this chapter.

We introduced *synchronized products* with asynchronous transitions (where all components act independently of each other) and synchronized transitions where (some of) the components act at the same time. We considered the restriction to *finitely synchronized products* in which the number of synchronized transitions in the product is finite. Furthermore, as a special case (of both) we presented *asynchronous products* with no synchronized transitions at all. In all these cases, the product had a finite or an infinite number of components.

As logics, we introduced modal logic and first-order logic and the following extensions thereof (starting with the least expressive one):

- reachability via any path (R),
- regular reachability over a unary alphabet (Reg1R) which amounts to modulo counting over the path length or
- regular reachability (RegR), i.e., reachability by paths which are labeled by any word in the language of a regular expression.

Now, we also discuss the extension of modal logic by the CTL quantifier EG and propositional dynamic logic over a unary alphabet (1PDL) which can be seen as an extension of ML(Reg1R) by tests.

In the previous section, we showed that the composition technique is applicable for FO(Reg1R) and finitely synchronized products (in which the product may have a finite or an infinite number of components). We now show that both the generalization on the product level to unrestricted (i.e., not necessarily finitely) synchronized products and on the logic level to either FO(RegR) logic or 1PDL fail.

For the synchronized product, we recapitulate a proof by Rabinovich in [Rab07] in which he showed that the direct product – a special case of the synchronized product – leads to a failure of the composition technique if the logic can express reachability, e.g., for ML(R) and FO(R). For the logics ML(RegR) and 1PDL, we adapt his technique to show that the composition theorem fails even for asynchronous products.

The remainder of this subsection is structured as follows. We first introduce the schema to proof the failure of the composition technique. Then, we apply it to generate counterexamples for the cases described above.

Lemma 4.5 (Schema to prove the failure of the composition technique [Rab07])**.** The composition technique fails for a type[2] of products and a logic $\mathcal{L}$ if we can find a formula ψ in $\mathcal{L}$ and two (infinite) families of transition systems $\mathcal{C} = \{C_k \mid k \in \mathbb{N}\}$ and $\mathcal{D} = \{D_l \mid l \in \mathbb{N}\}$ with common initial state s_0 such that we have $\forall k \in \mathbb{N} : (C_k \times D_k, (s_0, s_0)) \models \psi$ and $\forall k, l \in \mathbb{N}$ with $k \neq l : (C_k \times D_l, (s_0, s_0)) \not\models \psi$.

We now present the proof of this schema by adapting[3] the proof of Rabinovich shown in [Rab07] to interface information tuples.

For easier presentation, we show the proof for binary products and $\mathcal{C} = \mathcal{D}$. The main ideas of the proof are the following.

- Towards a contradiction, we assume that a composition theorem holds which assigns an interface information tuple to ψ.
- We define an equivalence relation on the family $\mathcal{C}$ of transition systems: all transitions systems which satisfy the same set of component formulas from the interface information tuple are in the same equivalence class. As there are only finitely many component formulas (and thus also finitely many sets of component formulas) one class contains at least two elements – in fact, it contains infinitely many elements.

[2] Type means asynchronous, direct or (finitely) synchronized.
[3] We have to adapt his schema as he uses a simpler form of interface information which can only be used for finite products.

- Thus, two different transition systems satisfy the same formulas. Using the composition technique their product has to satisfy ψ. This gives a contradiction to the precondition.

Proof (by contradiction). We assume that we have a composition theorem for a binary product with index set $I = \{i_1, i_2\}$ out of the family $\mathcal{C}$ – i.e., $C_{i_1} \times C_{i_2}$ – and a logic which can express ψ. Then, this theorem states that ψ holds if we have for the interface information tuple $\langle \psi_1, \ldots, \psi_l; \beta(X_1, \ldots, X_l) \rangle$ that $(I, I_1, \ldots, I_l) \models \beta(X_1, \ldots, X_l)$ holds with $I_k = \{i \in I \mid C_i \models \psi_k\}$.

As $\beta(X_1, \ldots, X_l)$ specifies for each $i \in I$ whether ψ_j, $j \in [l]$, has to hold, we can rewrite this interface information to a simpler form: there exist formulas $\psi_1^1 \ldots, \psi_m^1$ and $\psi_1^2, \ldots, \psi_m^2$ with $m = 2^l$ and ψ_k^1, ψ_k^2 are Boolean combinations of the $\{\psi_j \mid j \in [m]\}$ such that we have:

$$\bigvee_{j \in [2^m]} (C_{i_1} \models \psi_j^1 \wedge C_{i_2} \models \psi_j^2)$$

We define an equivalence relation on the set $\mathcal{C}$ as follows: $C_i \sim C_{i'} :\Leftrightarrow \forall j \in [m] : ((C_i, s_0) \models \psi_j^1 \Leftrightarrow (C_{i'}, s_0) \models \psi_j^2)$. As the set $[m]$ is finite, we have only finitely many equivalence classes and thus, one equivalence class contains at least two elements C_k and $C_{k'}$.

From the precondition, we have $(C_k \times C_k, (s_0, s_0)) \models \psi$ and $(C_{k'} \times C_k, (s_0, s_0)) \models \neg\psi$. However, from the assumption that we have a composition theorem in this case and $C_{k'} \sim C_k$, we get also $(C_{k'} \times C_k, (s_0, s_0)) \models \psi$. This is a contradiction. Thus, the assumption was wrong, i.e., the composition technique fails for this binary product $C_{i_1} \times C_{i_2}$.

□

We now consider the mentioned extensions of the products and the logics. First, we discuss the unrestricted synchronized product. The simplest form of a synchronized product is the direct product. We show the failure of the composition technique in this case if the logic can express reachability.

Theorem 4.6. The composition technique fails for direct products and any logic which can express reachability.

Proof (by Rabinovich). The proof is shown by giving a formula ψ and transition systems which meet the requirements of Theorem 4.5. Let $\psi = EF\langle b \rangle \mathrm{tt}$. We define $\mathcal{C}$ as the set of transition systems of the form $1 \xrightarrow{a} 2 \xrightarrow{a} \ldots \xrightarrow{a} k \xrightarrow{b} k+1$. It is formally defined as $\mathcal{C} = \{C_k \mid k \in \mathbb{N}\}$ with $C_k = \{S_k, R_a^k, R_b^k\}$ and $S_k := [k+1]$, $R_a = \{(i, i+1) \mid i \in \{1, \ldots, k-1\}\}$ and $R_b := \{(k, k+1)\}$. In any direct product

$C_k \times C_k$, we have exactly one path $(1,1) \xrightarrow{a} (2,2) \xrightarrow{a} \ldots \xrightarrow{a} (k,k) \xrightarrow{b} (k+1,k+1)$. For any direct product $C_k \times C_{k'}$ with $k \neq k'$, we can assume w.l.o.g. $k' > k$. Then, we have a path $(1,1) \xrightarrow{a} (2,2) \xrightarrow{a} \ldots \xrightarrow{a} (k,k)$ which cannot be extended because we have $k \xrightarrow{b} k+1$ in the first but only $k' \xrightarrow{b} k'+1$ in the second component. Thus, $EF\langle b\rangle \mathrm{tt}$ holds for $C_k \times C_k$ and does not hold for $C_k \times C'_k$ for $k \neq k'$ with $k, k' \in \mathbb{N}$. This meets the requirements of Theorem 4.5. Thus, the composition technique fails. An example for $k = 3$ and $k' = 5$ is shown in Figure 4.3. □

We immediately get the following corollary.

Corollary 4.7. The composition technique fails for ML(R) and FO(R) for synchronized products.

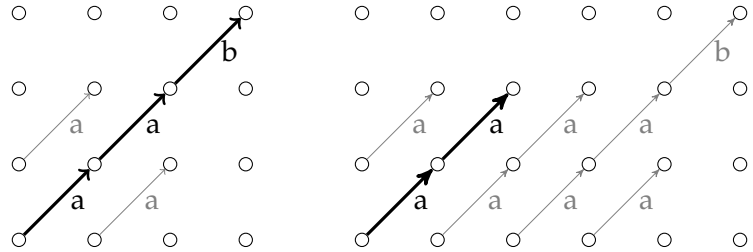


Figure 4.3: Products $C_3 \times C_3$ and $C_3 \times C_5$ for the Proof that the Composition Technique Fails for Direct Products and Reachability

We now consider the extensions on the logic level: reachability by words in the language of arbitrary regular expressions and afterwards logic 1PDL which extends Reg1R by tests.

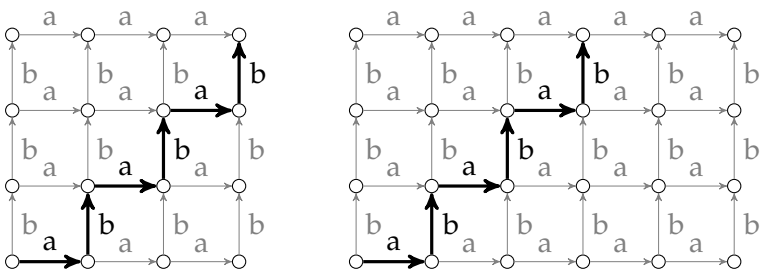
Theorem 4.8. The composition technique fails for any logic which can express regular reachability over an alphabet with at least two elements and asynchronous products.

Proof. We use the formula $\langle (ab)^* \rangle (p_1 \wedge p_2)$, which expresses that there exists a path to a state where $(p_1 \wedge p_2)$ holds and this path is either empty or labelled with a and b in alternation and ends with a b-transition.

The theorem is again proven by applying Lemma 4.5. We use a formula ψ which expresses that there exists a path which is labeled with a and b in alternation and that ends in a state with no outgoing transitions. We define two families of transition systems $\mathcal{C} = \{C_k \mid 2 \leq k \in \mathbb{N}\}$ and $\mathcal{D} = \{D_l \mid 2 \leq l \in \mathbb{N}\}$, where C_k and D_l are simply a copy of the natural numbers up to k, respectively l with the successor

relation labeled by a, respectively b. They are defined as $C_k := \{[k], R_a^k\}$ and $D_l := \{[l], R_b^l\}$ with $R_a^k := \{(i, i+1) \mid i \in [k-1]\}$ and $R_b^l := \{(i, i+1) \mid i \in [l-1]\}$. The formula ψ is defined as $\langle (ab)^* \rangle \varphi$ where φ expresses that there are no outgoing transitions: $\varphi = [a]\mathrm{ff} \wedge [b]\mathrm{ff}$.

By definition the asynchronous product $C_k \times D_l$ forms a grid as shown in Figure 4.4. The formula φ holds only at the last state (k, l). In Figure 4.4, we see that there is a path labeled with a and b in alternation to the last state for the product $C_4 \times D_4$, but not in $C_4 \times D_6$. This is now shown for all $k, l \geq 2$.



(1) Same index – $C_4 \times D_4$ (2) Different indices – $C_4 \times D_6$

Figure 4.4: Products $C_4 \times D_4$ and $C_4 \times D_6$ for the Proof that the Composition Technique Fails for Asynchronous Products and Regular Reachability

For $k \geq 2$, we observe that there is a path $(1,1) \xrightarrow{a} (1,2) \xrightarrow{b} (2,2) \xrightarrow{a} \ldots \xrightarrow{a} (k-1,k) \xrightarrow{b} (k,k)$ in the product $C_k \times D_k$. This path ends in the last state (k,k) of the product. Thus, the formula ψ holds for $C_k \times D_k$, $k \geq 2$.

W.l.o.g., we only look at the product $C_k \times D_l$ for $l > k$. Here, we observe that the path from above also exists, but at the state (k,k) we only have an outgoing a-transition to $(k, k+1)$. At this state we either have no outgoing transition (for $l = k+1$) or only a-transitions (for $l > k+1$). Thus, there is no path starting in $(1,1)$ with a and b in alternation which ends with a b in the last state (k,l) of the product.

Thus, we have proven $C_k \times D_k \models \psi$ for all $k \geq 2$ and $C_k \times D_l \not\models \psi$ for all $k \neq l$ with $k, l \geq 2$. By applying Theorem 4.5, we have shown that the composition technique fails for asynchronous products and any logic which can express regular reachability. □

We immediately get the following corollary.

Corollary 4.9. The composition technique fails for ML(RegR) and FO(RegR) for asynchronous products.

We now know that the composition technique is applicable for ML(Reg1R) (and FO(Reg1R)) but not for ML(RegR). We show that allowing simple tests in the regular expressions over a unary alphabet – i.e., 1PDL – also leads to a failure of the composition technique.

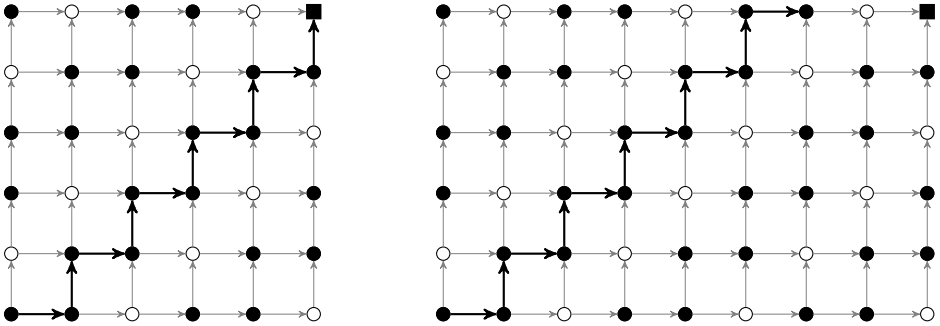
Theorem 4.10. The composition technique fails for 1PDL and asynchronous products.

Again, we use Lemma 4.5 to show the proof. We reuse the construction of Rabinovich which he used to show that the composition technique fails for any logic which can express the CTL quantifier EG and asynchronous products.

Proof. We define a family of transition systems $C_k = \{[3 \cdot k], R_a, Q_0, Q_1, Q_2\}$ with transition relation R_a defined as the successor relation and predicates Q_r which hold at all states that are divisible by 3 with remainder r. As product $C_k \times C_l$, we get a grid with states (i, j) for $i \in [k], j \in [l]$ and predicates $\bar{Q}_r^1, \bar{Q}_r^2$ which state that the first, respectively second component is in a state which is divisible by 3 with remainder r (see also Figure 4.5). We define a formula $\mathsf{last} := \neg\langle a \rangle \mathsf{tt}$ and a formula φ which defines the black diagonals in the figure by forcing that either both components have the same remainder r or that the first has remainder r and the second $r + 1 (\mathrm{mod}\ 3)$. This condition is defined as $\varphi = (p_0^1 \wedge (p_0^2 \vee p_1^2)) \vee (p_1^1 \wedge (p_1^2 \vee p_2^2)) \vee (p_2^1 \wedge (p_2^2 \vee p_0^2))$. We now use the 1PDL-formula $\psi = \langle (a\varphi?)^* \rangle \mathsf{last}$ which describes that there is an (a-labeled) path of black states to the last state. Note that ψ simulates the CTL-condition $E(\varphi U(\varphi \wedge \mathsf{last}))$ and (as we have finite transition systems) also $EG\varphi$. For the product $C_k \times C_k$ we have a path of black states from $(1, 1)$ to the last state (k, k). Thus, the formula ψ holds. We now look at $C_k \times C_l$ with $k \neq l$. W.l.o.g., we can assume $k < l$. In this product, we also have a path of black states from $(1, 1)$ to (k, k). This path can be extended to $(k, k + 1)$. However, from $(k, k + 1)$ we have no outgoing transition to a black state and $(k, k + 1)$ is not the last state. (Both situations are shown in Figure 4.5 for $k = l = 2$, respectively $k = 2$ and $l = 3$.) Thus, the formula ψ is not satisfied at the initial state $(1, 1)$. We meet the requirements for Lemma 4.5 and can conclude that the composition technique fails for asynchronous products and 1PDL. □

4.5 Summary

In this chapter, we looked at the composition technique for synchronized products and various extensions of ML and FO logic. For the products, we looked at the special cases asynchronous products and finitely synchronized products. On the logic



(1) Same index – $C_2 \times C_2$ (2) Different indices – $C_2 \times C_3$

Figure 4.5: Products $C_2 \times C_2$ and $C_2 \times C_3$ for the Proof that the Composition Technique Fails for Asynchronous Products and 1PDL

side, we discussed (normal) reachability (R), reachability with modulo counting over the path length (Reg1R) and regular reachability (RegR). An overview of the results can be found in Table 4.1.

On the positive side, i.e., where the composition technique is applicable, we extended results of Feferman and Vaught, Rabinovich and Wöhrle and Thomas. Feferman and Vaught showed the applicability for all types of products and FO logic (which includes ML). Rabinovich extended this for asynchronous products to ML(R) and Wöhrle and Thomas extended the results to finitely synchronized products and FO(R). Our contribution in this chapter was on the one hand the extension to finitely synchronized products and ML(Reg1R) or FO(Reg1R). On the other hand, we allowed the number of components in our products to be finite or infinite.

On the negative side, we extended results of Rabinovich. He showed the failure in the following cases: direct products (which are a special case of synchronized products) and logics which can express reachability, and asynchronous products and logics which can express the CTL-quantifier EG. We used his proof schema to show that both the extension of ML(Reg1R) or FO(Reg1R) to ML(RegR) or FO(RegR) with alphabets with at least two letters and the extension of 1PDL without tests – which is the same as ML(Reg1R) – to include tests lead to a failure of the composition technique.

Thus, in this chapter, we have given an overview about the frontiers of the composition technique for products – i.e., where the composition technique is still applicable and where it fails. This overview is also shown in Table 4.1. However, in the positive cases, a major drawback of the composition technique remains: as shown in Section 3.4 both for sums and products, already in the simplest cases, the

	ML or FO logic extended by					1PDL
	–	R	Reg1R	RegR	EG	
asynchronous	$\surd^1/\surd^1$	$\surd^2/\surd^4$	$\surd^4/\surd^4$	$-^4/-$	$-^2/-$	$-^4/-$
finitely synchronized	$\surd^1/\surd^1$	$\surd^3/\surd^4$	$\surd^4/\surd^4$	$-/-$	$-/-$	$-/-$
synchronized	$\surd^1/\surd^1$	$-^2/-$	$-/-$	$-/-$	$-/-$	$-/-$

Table 4.1: Overview over results for products with finite/infinite number of components

[1] proven 1959 by Feferman and Vaught [FV59]
[2] proven 2007 by Rabinovich [Rab07]
[3] proven 2004 by Wöhrle and Thomas [WT07]
[4] proven 2009 by the author of this thesis and Thomas [FT09]

size of the decomposition – especially the number of formulas for the components – grows non-elementary in the size of the input formula and it has been shown that this is a lower bound. The proof in [GJL12] mainly relies on the fact that the structures can have an unbounded out-degree of transitions at each state.

In the case of (disjoint ordered) *sums*, we overcome this situation in the next chapters for special structures, namely words and finite branching trees and LTL/CTL over these structures.

5 Composition for Sums of Words and LTL

In Section 3.3, we discussed Shelah's composition theorem [She75, Tho97a] for MSO logic over disjoint ordered sums of labeled orderings[1] which yields a decomposition with a non-elementary size. In Section 3.4, we presented the result of Göller, Jung and Lohrey, that this complexity is in general unavoidable – even for FO^3 logic and finite sums. The goal of this chapter is to improve the complexity of the composition theorem for a special logic: We consider linear temporal logic (LTL) over ordered disjoint sums of labeled orderings, i.e., models of words. It is well-known that in this case LTL is expressively equivalent to FO($<$) logic (Kamp's Theorem [Kam68]). We show a composition theorem for words and LTL which has only exponential complexity. To be more precise, the size of the decomposition of a formula is in $O(2^{n^2})$ if n is the size of the formula. Our result is an improved version of [Fel12].

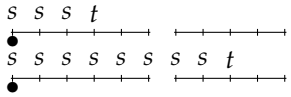
We are mainly interested in decomposing LTL formulas on ω-words. However, our theorem allows the composed word to be finite or infinite as it works with a finite or infinite number of components. Each component must be a finite word with one exception: the last component may be infinite in the case where we have a finite number of components.

5.1 Composition Idea

We explain the idea of the composition in a simple setting: we consider the disjoint ordered sum of two components which simply amounts to the concatenation of two words. We look at the formulas sUt and Gr in this setting. For sUt in the sum, obviously, one of the following two cases has to hold: either sUt holds in the first component or we have Gs in the first component and sUt in the second component (see also Figure 5.1).

For Gr in the sum, we must have Gr in both components. These simple ideas form the basis of the composition theorem. However, for general formulas $\varphi U \psi$

[1] Recall that building an ordered disjoint sum over labeled orderings can be seen as the concatenation of words over a k-ary Boolean alphabet.



(sUt, tt)

(Gs, sUt)

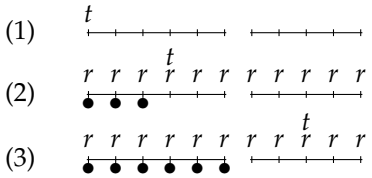
Figure 5.1: The Formula sUt in a Sum of Two Components

the idea from sUt is not sufficient. If we consider the formula φUt where φ is a composed formula, in general, φ has to hold at a set T of states. As an example, we consider the formula Gr. (This is also shown in Figure 5.2 in which we use black dots to mark the positions where φ has to hold.)

1. If t holds directly at the first state of the first component, we should not have any further conditions as φUt is already fulfilled in the sum.

2. If t holds at some later state of the first component, we must have Gr for all states before this state and Gr at the second component.

3. As in the previous example, we may have $G\varphi = (G)Gr$ in the first component and $GrUt$ in the second one.



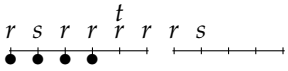
(t, tt)

$(GrUt, Gr)$

$((G)Gr, GrUt)$

Figure 5.2: The Formula $GrUt$ in a Sum of Two Components

Note that in the cases (2) and (3), the subformula φ (here Gr) has to hold at sets T_2, respectively T_3, of states in the first component which are indicated by the black dots in Figure 5.2. We now consider another formula φ in $\delta = \varphi Ut$: $\varphi = rUs$. Here, for each of the states from T_2, respectively T_3, we must have rUs in the sum. For the decomposition into components, this gives us for each of these states the possibilities from Figure 5.1: either rUs holds in the first component or we have Gr in the first and rUs in the second one. Thus, for the set of these states we may have for all states the first possibility or the second possibility or a combination as shown in Figure 5.3.



$(G((rUs) \vee Gr)Ut, rUs)$

Figure 5.3: One Case for $(rUs)Ut$ in a Sum of Two Components

This leads to an inductive construction: If we consider the general case $\delta = \varphi U \psi$ and assume that we already know that $\bar{\varphi}, \bar{\psi}$ are possible formulas for the first component for the subformulas φ, ψ of δ, we get the formulas $\bar{\psi}$, $\bar{\varphi} U \bar{\psi}$, $G\bar{\varphi}$ and $(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}$ for the first component. We will see that these cases are sufficient. This idea is now used to develop an algorithm for the composition theorem.

5.2 Technical Preliminaries

We use interface information tuples $\langle \delta_1, \ldots, \delta_m; \beta(X_1, \ldots, X_m) \rangle$ to express the conditions which have to hold in the components. For this, we first introduce how the formulas $\delta_1, \ldots, \delta_m$ for the components are defined and afterwards, when they have to hold via the formula $\beta(X_1, \ldots, X_m)$.

We inductively define the *set of component formulas* for a given formula δ – denoted by $\mathrm{cf}(\delta)$. For "atomic" formulas $\varphi \in \{p, \neg p \mid p \in V\}$, we simply get $\mathrm{cf}(\varphi) = \{\varphi\}$. For $\delta = \varphi U \psi$, these are the four possibilities from above for given $\bar{\varphi} \in \mathrm{cf}(\varphi)$ and $\bar{\psi} \in \mathrm{cf}(\psi)$. The intuition for the set $\mathrm{cf}(\delta)$ is that it sums up all possibilities for the component with index i why the formula δ holds in the sum starting at index i if the "correct"[2] formulas hold in the later components. As mentioned above, we have:

- $\mathrm{cf}(\varphi U \psi) = \{\bar{\psi}, \bar{\varphi} U \bar{\psi}, G\bar{\varphi}, (\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$

For "Finally" as special case of "Until", we get $\mathrm{cf}(F\varphi) = \{\mathrm{tt}\} \cup \{F\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$. For $G\varphi$, we simply have $\mathrm{cf}(G\varphi) = \{G\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$. The idea for $\varphi \vee \psi$ is quite similar: If $\delta = \varphi \vee \psi$ should hold at a set of states (e.g. as in $\delta U t$), we can either have φ at all of these states, ψ at all of these states or at some φ and at some ψ. For $\varphi \wedge \psi$ we simply must have $\varphi \wedge \psi$ at all of the states. Thus, we get:

- $\mathrm{cf}(\varphi \vee \psi) = \{\bar{\varphi}, \bar{\psi}, \bar{\varphi} \vee \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$ and
- $\mathrm{cf}(\varphi \wedge \psi) = \{\bar{\varphi} \wedge \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$.

With the equivalence $\psi R \varphi = \varphi U (\varphi \wedge \psi) \vee G\varphi$, we get:

- $\mathrm{cf}(\psi R \varphi) = \{\bar{\varphi} U (\bar{\varphi} \wedge \bar{\psi}), G\bar{\varphi}, (\bar{\varphi} U (\bar{\varphi} \wedge \bar{\psi})) \vee G\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$.

Finally, we consider $\delta = X\varphi$: For δ at one state or a set of states, we have to distinguish if the last state is among those states (1) or not (2) or if the last state is the only state of the current component (3). For (1), we get the formula $\neg \mathrm{last} \rightarrow X\bar{\varphi}$

[2] The condition which are the "correct" formulas will be defined later via β.

which forces $X\varphi$ for all but the last state and we have to ensure that φ also holds at the first state of the next component. For (2), we take the formula $\neg\mathsf{last} \wedge X\bar{\varphi}$ which excludes the last state and forces $X\varphi$ on the other states. For (3), we simply take the formula last.

- $\mathrm{cf}(X\varphi) = \{\neg\mathsf{last} \wedge X\bar{\varphi}, \neg\mathsf{last} \to X\bar{\varphi}, \mathsf{last} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$

For the definition of the formula β, we will inductively define conditions in which components these formulas have to hold. For this, we also have to refer to the component formulas of the subformulas of δ. We define $\mathrm{ecf}(\delta)$ – called the extended component formulas of δ – as the set which contains the component formulas for all subformulas of δ. Formally, we have $\mathrm{ecf}(\delta) = \bigcup_{\hat{\delta} \in \mathrm{CL}(\delta)} \mathrm{cf}(\hat{\delta})$.

Note that the modal depth of the formulas in $\mathrm{cf}(\delta)$ (and thus, also in $\mathrm{ecf}(\delta)$) is at most the modal depth of δ.

5.3 LTL Composition Theorem

We now present the composition theorem for LTL. It reduces the truth of an LTL formula γ in an ordered disjoint sum to the truth values of the formulas of $\mathrm{ecf}(\gamma)$ in the components and information about which of these formulas hold in which components.

Theorem 5.1. Given a signature σ and an index ordering $(I, <)$ with index set I (usually $\mathbb{N}$), the following statement holds: For every σ-LTL formula γ of modal depth r, we can compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(X_1, \ldots, X_m) \rangle$ with σ-LTL formulas γ_j of modal depth at most r – interpreted in the components – and an FO formula $\beta_\gamma(X_1, \ldots, X_m)$ – interpreted in the index ordering – such that for every disjoint sum $\underline{w}$ of σ-components $\underline{w}_i$ ($i \in I$):

$$\underline{w} \models \gamma \Leftrightarrow (I, <, I_1, \ldots, I_m) \models \beta_\gamma(X_1, \ldots, X_m)$$

where $I_k = \{i \in I \mid \underline{w}_i \models \gamma_k\}$ for $k \in [m]$.
Let s be the size of γ, then the size of the decomposition is at most 2^{s^2}.

We want to emphasize that we only use an FO-formula for the index structure for didactic reasons: on one hand, to use the same notation as in the previous chapters and, on the other hand, to make it easier for the reader to distinguish between component formulas and formulas over the index structure. The theorem also holds if we exchange the FO formula by an LTL formula, in particular, also the size of the decomposition does not change. The reason is that, in fact, the FO formula just expresses Until-, Release- and Next-conditions in FO logic.

We first show the construction and estimate the size of the generated decomposition afterwards. Instead of proving Theorem 5.1 directly, we show a more general result. For this, let h be a component index and T an arbitrary set of states in this component. We consider the part of the ordered sum that starts at h and a state $u \in T$. We show that the equivalence from Theorem 5.1 holds for this part of the sum if we adapt the interpretation of the sets I_k. For this, we need additional notations:

Let $\underline{w}^{\geq h}[u]$ denote the ordered disjoint sum starting at the index $h \in I$ and the state u. For a set T of states in the current component, let $I_k^{\geq h,T}$ be the set I_k restricted to the components from the $h+1$-th component onwards and from all $u \in T$ in the h-th component onwards, i.e., $I_k^{\geq h,T} := \{i \in I \mid i > h+1 \wedge \underline{w}_i \models \gamma_k\} \cup \{i \in I \mid i = h \wedge \forall u \in T : (\underline{w}_h, u) \models \gamma_k\}$. The generalization of the equivalence in Theorem 5.1 is that we have for any set T of states in the current component:

$$\forall u \in T : \underline{w}^{\geq h}[u] \models \gamma \Leftrightarrow (I, <, I_1^{\geq h,T}, \ldots, I_m^{\geq h,T}) \models \beta_\gamma(X_1, \ldots, X_m)$$

For the proof, we construct interface information tuples $\langle \delta_1, \ldots, \delta_k; \beta_\delta(X_1, \ldots, X_k)\rangle$, inductively for all subformulas δ of the given formula γ. The formula $\beta_\delta(X_1, \ldots, X_k)$ will be an FO formula with free MSO variables $X_1, \ldots, X_k$. We use an enumeration of the set $\mathrm{ecf}(\delta)$ as the formulas $\bar{\delta}_1, \ldots, \bar{\delta}_k$. We allow to use $X_{\bar{\delta}}$ instead of X_j if $\bar{\delta}$ is the j-th formula in that enumeration. To define $\beta_\delta(X_1, \ldots, X_k)$, we use a parametrized formula $\beta_\delta(X_1, \ldots, X_k, h)$ (or shortly $\beta_\delta(h)$) and auxiliary formulas $\alpha_{\delta,\bar{\delta}}(X_1, \ldots, X_k, h)$ (abbreviated by $\alpha_{\delta,\bar{\delta}}(h)$) and then set $\beta_\delta(X_1, \ldots, X_m) := \beta_\delta(X_1, \ldots, X_m, 0)$.

- The *formula* $\beta_\delta(h)$ will be satisfied if the sum satisfies δ from the component with index h onwards. It expresses LTL conditions *over the index structure*. For compatibility with the previous chapters, we use FO logic to express this condition.
- It states that one $\bar{\delta}$ of the set $\mathrm{cf}(\delta)$ of component formulas has to hold in the current component and that the correct conditions for this component formula have to hold at the successor component. These conditions are stored in the *auxiliary formulas* $\alpha_{\delta,\bar{\delta}}(h)$. Formally, we have $\beta_\delta(h) = \bigvee_{\bar{\delta} \in \mathrm{cf}(\delta)} (X_{\bar{\delta}}(h) \wedge \alpha_{\delta,\bar{\delta}}(h))$.

Example 5.1. For better understanding, we present these formulas for the case $\delta = rUs$. The formula $\beta_{rUs}(h)$ expresses an Until-condition over the component indices, namely that from h onwards, we have components where Gr (or $(rUs) \vee Gr$)

holds until a component for which rUs (or s) holds. The formulas $\alpha_{\delta,s}(h)$ and $\alpha_{\delta,rUs}(h)$ are simply tt because the subformulas r, s are atomic and thus, have no conditions for later components. The formulas $\alpha_{\delta,Gr}(h)$ and $\alpha_{\delta,(rUs)\vee Gr}(h)$ state that rUs still has to be satisfied later.

The generalization of Theorem 5.1 is now proven by a simultaneous induction over the subsums starting at any component and the subformulas of the given formula. Note that we add components to the sum at the front, i.e., we reduce the truth of a formula δ in the sum starting at the h-th component to the truth values of subformulas of it at components with an index $\geq h$.

To get an inductive proof, we show that the induction base holds for the atomic formulas and any subsum starting at any component.

Induction base:
The induction base states for any component index h and any set of states T in this component, that the atomic formula p holds in the (infinite) sum starting from h and a set of states T iff it holds in the component with index h at the set of states T. This is obviously true. Thus, we set $\beta_p(X_p, h) = X_p(h)$, $\alpha_{p,p} = \mathrm{tt}$. Formally, we have for any set T in the current component:

$$\forall u \in T : \underline{w}^{\geq h}[u] \models p \Leftrightarrow (I, <, I_p^{\geq h,T}) \models \beta_p(X_p).$$

Induction hypothesis:
The induction hypothesis states for any set T of states of the current component (with index h) that for all $u \in T$ the sum starting from h satisfies the formula φ iff the formula $\beta_\varphi(X_1, \ldots, X_m)$ holds in the index structure with the sets $I_l^{\geq h,T}$ for $l \in [m]$. Formally, we have for any set T of states in the component with index h:

$$\forall u \in T : \underline{w}^{\geq h}[u] \models \varphi \Leftrightarrow (I, <, I_1^{\geq h,T}, \ldots, I_m^{\geq h,T}) \models \beta_\varphi(X_1, \ldots, X_m).$$

To simplify notation, from now on, we use $\mathcal{I}^{\geq h,T}$ as abbreviation for the sets $I_1^{\geq h,T}, \ldots, I_m^{\geq h,T}$. Furthermore, we omit the MSO variables, e.g., we simply write β_φ instead of $\beta_\varphi(X_1, \ldots, X_m)$.

Induction step:
As mentioned above, in the induction step for a formula δ and a component index h, we assume that the induction hypothesis holds for any proper subformula φ of δ and the sum starting from index $\geq h$. Formally, we have the order $(h', \varphi) < (h, \delta) \Leftrightarrow h' \geq h \wedge \varphi \sqsubset \delta$ for the induction. (Note that the order of h is reversed as we add components at the front of the sum.)

$\varphi U \psi$

We start with a detailed construction and proof for the formula $\delta = \varphi U \psi$. The other cases will be presented with less detail as they are quite similar. We first introduce the formulas β_δ and $\alpha_{\delta,\bar{\delta}}$ and prove correctness and completeness of the construction afterwards. For $\delta = \varphi U \psi$, by definition of the Until-operator, we have for any state u which lies in the current component of the sum that $\varphi U \psi$ is satisfied iff there is a state v where ψ holds and on all states up to (but excluding) v the formula φ holds. The state v may be either in the current component or in a later component.

For the first case, we either have ψ directly at u or later (but still in the current component). This gives us the cases $\bar{\psi}$, $\bar{\varphi} U \bar{\psi}$ of $\mathrm{cf}(\delta)$ for the current component.

For the second case, we must either have φ on all states of the sum which belong to the current component or – for δ on more than one state u – for some states $\varphi U \psi$ is satisfied directly and for some at a later component. This gives us the cases $G\bar{\varphi}$, $(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}$ of $\mathrm{cf}(\delta)$ for the current component. (Furthermore, $\varphi U \psi$ still has to be satisfied from the next component onwards.)

The formula $\beta_\delta(h)$ states that we must have the second case for all components with the indices from h up to z where the first case holds. Additionally, the conditions $\alpha_{\varphi,\bar{\varphi}}, \alpha_{\psi,\bar{\psi}}$ for the subformulas $\bar{\varphi}$ and $\bar{\psi}$ have to hold for the component indices where they are used.

$$
\begin{aligned}
\beta_\delta(h) = \exists z \geq h [[& \beta_\psi(z) \\
& \vee \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{\bar{\varphi} U \bar{\psi}}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))] \\
& \wedge \forall x, h \leq x < z : [\\
& \quad \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x)) \\
& \vee \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x) \wedge \alpha_{\psi,\bar{\psi}}(x))]]
\end{aligned}
$$

For δ itself as subformula, we get the conditions $\alpha_{\delta,\bar{\delta}}$:

$$
\begin{aligned}
\alpha_{\delta,\bar{\psi}}(h) &= \alpha_{\psi,\bar{\psi}}(h) \\
\alpha_{\delta,\bar{\varphi} U \bar{\psi}}(h) &= \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)
\end{aligned}
$$

$$\alpha_{\delta,G\bar{\varphi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\delta(h+1)$$
$$\alpha_{\delta,(\bar{\varphi}U\bar{\psi})\vee G\bar{\varphi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h) \wedge \beta_\delta(h+1)$$

Note that by this construction $\beta_\delta(h) = \bigvee_{\bar{\delta}\in \mathrm{cf}(\delta)}(X_{\bar{\delta}}(h) \wedge \alpha_{\delta,\bar{\delta}}(h))$ holds. To show this, consider the cases $z = h$ and $h > z$ in the equation for $\beta_\delta(h)$. Before we show the correctness and completeness of the construction of $\beta_\delta(h)$, we discuss two examples using only the Until-operator:

Example 5.2. We first consider $\delta = rUs$. As component formulas, we have $\mathrm{cf}(rUs) = \{s, Gr, rUs, (rUs) \vee Gr\}$. For atomic propositions, we have $\alpha_{p,p}(h) = \mathrm{tt}$ and get

$$\beta_\delta(h) = \exists z \geq h[[X_s(z) \vee X_{rUs}(z)] \wedge \forall x, h \leq x < z[X_{Gr}(x) \vee X_{(rUs)\vee Gr}(x)]].$$

This means, that we have components with Gr (or $(rUs) \vee Gr$) until we have a component with rUs (or s). Obviously, this is the case iff we have rUs in the sum. (In this case the formulas in brackets are superfluous.) The conditions for δ as a subformula of some other formula are as follows:

$$\alpha_{\delta,s}(h) = \alpha_{\delta,rUs}(h) = \mathrm{tt} \text{ and } \alpha_{\delta,Gr}(h) = \alpha_{\delta,(rUs)\vee Gr}(h) = \beta_\delta(h+1)$$

In other words, δ has to be satisfied at the next component iff we have Gr or $(rUs) \vee Gr$ at the current component.
We consider another example: We use the formula δ from above inside another formula. Let $\gamma = \delta Ut = (rUs)Ut$. Here, we get as component formulas $\mathrm{cf}(\gamma) = F_1 \cup F_2 \cup F_3 \cup F_4$ with

- $F_1 = \{t\}$,
- $F_2 = \{sUt, (rUs)Ut, GrUt, ((rUs) \vee Gr)Ut\}$,
- $F_3 = \{Gs, G(rUs), G(Gr), G((rUs) \vee Gr)\}$ and
- $F_4 = \{(sUt) \vee Gs, ((rUs)Ut) \vee G(rUs), ((Gr)Ut) \vee G(Gr), (((rUs) \vee Gr)Ut) \vee G((rUs) \vee Gr)\}$.

The formula $\beta_\gamma(h)$ states that we have component indices x where a formula from F_3 or F_4 holds until we have a component index z where a formula from F_1 or F_2 holds. Furthermore, the components following after x, respectively z, have to

satisfy the conditions for the subformulas, e.g., if $GrUt$ holds at the component z, rUs still has to be satisfied from $z+1$ onwards. Thus, we get

$$\begin{aligned}\beta_\gamma(h) = \exists z \geq h[[&X_t(z) \vee X_{sUt}(z) \vee X_{(rUs)Ut}(z) \vee (X_{(Gr)Ut}(z) \wedge \beta_\delta(z+1)) \\ &\vee (X_{((rUs)\vee Gr)Ut}(z) \wedge \beta_\delta(z+1))] \\ \wedge \forall x, h \leq x < z[&X_{Gs}(x) \vee X_{G(rUs)}(x) \vee (X_{G(Gr)}(x) \wedge \beta_\delta(x+1)) \\ &\vee (X_{G((rUs)\vee Gr)}(x) \wedge \beta_\delta(x+1)) \vee \alpha_{F_4}]]\end{aligned}$$

where α_{F_4} contains the sets for the formulas of F_4.

We now show the correctness and completeness of the choice of the interface information tuple for $\delta = \varphi U \psi$. Let s_z denote the size of the component with index z. Let h be a component of $\underline{w}$ and T a set of states in this component. We first look at *δ at the first state* of the current component. Taking into account that $\underline{w}^{\geq h}$ is a composed word, we get the following situation:

- There exists a component index $z \geq h$ and a state $k \leq s_z$ in that component such that the sum $\underline{w}^{\geq z}$ satisfies ψ at state k and all states before that state in the same component satisfy φ.

- Furthermore, for all components with index x up to z (and $x \geq h$), we have for all states in these components that φ holds.

Formally, this is the following statement:

$$\begin{aligned}\exists z \geq h[\ \ &(\exists k \leq s_z : \underline{w}^{\geq z}[k] \models \psi \wedge \forall i, i < k : \underline{w}^{\geq z}[i] \models \varphi) \\ &\wedge \forall x, h \leq x < z : \forall i \leq s_x : \underline{w}^{\geq x}[i] \models \varphi]\end{aligned}$$

We can apply the induction hypothesis for φ and ψ as these are proper subformulas of δ. Using $\beta_\varphi(z) = \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)}(X_{\bar{\varphi}}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z))$ and $\beta_\psi(z) = \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)}(X_{\bar{\psi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))$, we get:

$$\begin{aligned}\exists z \geq h[(&\exists k\, (I, <, \mathcal{I}^{\geq z, \{k\}}) \models \beta_\psi(z) \wedge (I, <, \mathcal{I}^{\geq z, J_z^1}) \models \beta_\varphi(z)) \\ &\wedge \forall x, h \leq x < z : \forall i\, (I, <, \mathcal{I}^{\geq x, J_x^2}) \models \beta_\varphi(x)]\end{aligned}$$

with $J_z^1 := \{i \in \mathrm{dom}_{\underline{w}_z} \mid i < k\}$ and $J_x^2 := \{i \in \mathrm{dom}_{\underline{w}_x} \mid i \leq s_x\}$. Separating $k = 1$ and $k > 1$, we get:

$$\begin{aligned}\exists z \geq h[\ \ &((I,<,\mathcal{I}^{\geq z,\{1\}}) \models \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (X_{\bar{\psi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z)) \\ &\vee (I,<,\mathcal{I}^{\geq z,\{1\}}) \models \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (X_{\bar{\varphi}U\bar{\psi}}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))) \\ &\wedge \forall x, h \leq x < z : (I,<,\mathcal{I}^{\geq x,\{1\}}) \models \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x))]\end{aligned}$$

Note that this is equivalent to $(I,<,\mathcal{I}^{\geq z,\{1\}}) \models \beta_\delta(h)$. On the one hand, if the formula from above holds, also $\beta_\delta(h)$ holds as it only has one other further possible condition in the disjunction. On the other hand, if $\beta_\delta(h)$ holds and we have for one x with $h \leq x < z$ the possibility $\bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} X_{(\bar{\varphi}U\bar{\psi})\vee G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x) \wedge \alpha_{\psi,\bar{\psi}}(x)$ then also $\bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x))$ holds or we already have $\bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(X_{\bar{\varphi}U\bar{\psi}}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))$.

We now look at the subwords starting at any $u \in T$, i.e., at all words $\underline{w}^{\geq h}[u]$. The formula $\varphi U \psi$ holds for $\underline{w}^{\geq h}[u]$ for all $u \in T$ iff for every single $u \in T$, one of the following cases hold:

- We already have $\varphi U \psi$ in the current component, i.e., there is a state $k \geq u$ such that ψ holds and on all states from u up to k the formula φ holds or
- from u onwards, we have φ on all states of the current component and the formula δ holds from the next component onwards.

Formally, this means that we have:

$$\begin{aligned}&(\exists k, u \leq k \leq s_h : \underline{w}^{\geq h}[k] \models \psi \wedge \forall i, u \leq i < k : \underline{w}^{\geq h}[i] \models \varphi) \\ \vee &(\forall u \leq i \leq s_h : \underline{w}^{\geq h}[i] \models \varphi \wedge \underline{w}^{\geq h+1} \models \delta).\end{aligned}$$

We apply the induction hypothesis for φ and ψ and the sum from index h onwards. Furthermore, we use the result proven above for δ at the first state of the sum starting at the next component. With $\beta_\varphi(z) = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(X_{\bar{\varphi}}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z))$ and $\beta_\psi(z) = \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(X_{\bar{\psi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))$, we get:

$$\begin{aligned}&(\exists k, u \leq k \leq s_h\, (I,<,\mathcal{I}^{\geq h,\{k\}}) \models \beta_\psi(h) \wedge (I,<,\mathcal{I}^{\geq h,J_h^1}) \models \beta_\varphi(h)) \\ \vee &\ ((I,<,\mathcal{I}^{\geq h,J_h^2}) \models \beta_\varphi(h) \wedge (I,<,\mathcal{I}^{\geq h+1,\{1\}}) \models \beta_\delta(h+1))\end{aligned}$$

with $J_h^1 := \{i \in \mathrm{dom}_{\underline{w}_h} \mid u \leq i < k\}$ and $J_h^2 := \{i \in \mathrm{dom}_{\underline{w}_h} \mid u \leq i \leq s_h\}$. Separating $k = u$ and $k > u$, we get:

$$\begin{aligned}
(I, <, \mathcal{I}^{\geq h,\{u\}}) &\models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{\bar{\psi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)) \\
\vee \ (I, <, \mathcal{I}^{\geq h,\{u\}}) &\models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{\bar{\varphi} U \bar{\psi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)) \\
\vee \ [(I, <, \mathcal{I}^{\geq h,\{u\}}) &\models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\delta(h+1))
\end{aligned}$$

Recall that these conditions have to hold for all $u \in T$. If we now consider all these states, i.e., the set T, we may have any combination of the disjuncts from above (including the case that all satisfy the same disjunct). Thus, we get:

$$\begin{aligned}
(I, <, \mathcal{I}^{\geq h,T}) &\models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{\bar{\psi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)) \\
\vee (I, <, \mathcal{I}^{\geq h,T}) &\models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{\bar{\varphi} U \bar{\psi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)) \\
\vee (I, <, \mathcal{I}^{\geq h,T}) &\models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\delta(h+1)) \\
\vee (I, <, \mathcal{I}^{\geq h,T}) &\models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (X_{(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h) \wedge \beta_\delta(h+1))
\end{aligned}$$

As $\beta_\delta(h+1)$ has to hold at the first state of the $(h+1)$-th component, we can use the formula for $\beta_\delta(h+1)$ from above. Now we can use the expansion law for "Until" (see Section 2.2) and take the four conditions from above into the Until-condition over the index structure in $\beta_\delta(h+1)$ and thus get $(I, <, \mathcal{I}^{\geq h,T}) \models \beta_\delta(h)$.

We continue the induction with the other cases of subformulas. As the proof principle is similar in these cases, we skip the formal proof and only present the construction of the formulas β_δ and $\alpha_{\delta,\bar{\delta}}$.

$G\varphi$

For $\delta = G\varphi$, we must have for all components the formula $G\bar{\varphi}$ and everywhere the conditions for φ for the next component. We get

$$\beta_\delta(h) = \forall x \geq h : \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x)$$

and $\alpha_{\delta,G\bar{\varphi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\delta(h+1)$.

$\psi R \varphi$

We could use the equivalence $\psi R \varphi = \varphi U (\varphi \wedge \psi) \vee G\varphi$ for the Release operator. However, this would introduce a preprocessing of the given formula which would increase the formula size. Thus, we give a direct construction analogously to the Until-case:

$$\begin{aligned}
\beta_\delta(h) = \exists z \geq h[[& \bigvee_{\substack{\bar{\varphi}\in \mathrm{cf}(\varphi)\\ \bar{\psi}\in \mathrm{cf}(\psi)}} (X_{\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi})}(z) \wedge \alpha_{\varphi,\bar{\varphi}}(z) \wedge \alpha_{\psi,\bar{\psi}}(z))] \\
\wedge \forall x, h \leq x < z : [& \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x)) \\
& \vee \bigvee_{\substack{\bar{\varphi}\in \mathrm{cf}(\varphi)\\ \bar{\psi}\in \mathrm{cf}(\psi)}} X_{(\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi}))\vee G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x) \wedge \alpha_{\psi,\bar{\psi}}(x))]] \\
\vee \forall x \geq h : [& \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (X_{G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x)) \\
& \vee \bigvee_{\substack{\bar{\varphi}\in \mathrm{cf}(\varphi)\\ \bar{\psi}\in \mathrm{cf}(\psi)}} X_{(\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi}))\vee G\bar{\varphi}}(x) \wedge \alpha_{\varphi,\bar{\varphi}}(x) \wedge \alpha_{\psi,\bar{\psi}}(x))]
\end{aligned}$$

As auxiliary formulas, we get:

$$\begin{aligned}
\alpha_{\delta,\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi})}(h) &= \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h) \\
\alpha_{\delta,G\bar{\varphi}}(h) &= \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\delta(h+1) \\
\alpha_{\delta,(\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi}))\vee G\bar{\varphi}}(h) &= \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h) \wedge \beta_\delta(h+1)
\end{aligned}$$

$\varphi \vee \psi$

For the disjunction $\delta = \varphi \vee \psi$ at a set T of states of the current component, we have to consider three cases: φ on all of these states, ψ on all of these states and that there are states in T where φ holds and on the other ones ψ holds. Thus, we get:

$$\beta_\delta(h) = \beta_\varphi(h) \vee \beta_\psi(h) \vee (\bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} X_{\bar{\varphi}\vee\bar{\psi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h))$$

and $\alpha_{\delta,\bar{\varphi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h)$, $\alpha_{\delta,\bar{\psi}}(h) = \alpha_{\psi,\bar{\psi}}(h)$ and $\alpha_{\delta,\bar{\varphi}\vee\bar{\psi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)$.

$\varphi \wedge \psi$

For the conjunction $\delta = \varphi \wedge \psi$, we have to satisfy both the conditions for φ and ψ. We get:

$$\beta_\delta(h) = (\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} X_{\bar{\varphi} \wedge \bar{\psi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h))$$

and $\alpha_{\delta,\bar{\varphi}\wedge\bar{\psi}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \alpha_{\psi,\bar{\psi}}(h)$.

$X\varphi$

For the Next-operator $\delta = X\varphi$, note that there are two cases why a formula φ holds at the next state of the sum:

- Either the current state u is the last state of the current component, then φ has to hold at the sum starting from the next component
- or the current state is not the last one, then φ simply has to hold at the next state of the current component.

For a set T of states, again, we have to consider the combination of these cases, i.e., φ holds at the next component and for all states in T excluding the last one, φ has to hold at the next state. Formally, we have:

$$\begin{aligned}\beta_\delta(h) =& (\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (X_{\neg \mathsf{last} \wedge X\bar{\varphi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h))) \\ & \vee (X_{\mathsf{last}}(h) \wedge \beta_\varphi(h+1)) \\ & \vee (\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} X_{\neg \mathsf{last} \to X\bar{\varphi}}(h) \wedge \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\varphi(h+1))\end{aligned}$$

As auxiliary formulas, we get $\alpha_{\delta,X_{\neg \mathsf{last}\wedge X\bar{\varphi}}}(h) = \alpha_{\varphi,\bar{\varphi}}(h)$, $\alpha_{\delta,\mathsf{last}}(h) = \beta_\varphi(h+1)$ and $\alpha_{\delta,X_{\neg \mathsf{last}\to X\bar{\varphi}}}(h) = \alpha_{\varphi,\bar{\varphi}}(h) \wedge \beta_\varphi(h+1)$.

5.4 Size of the Decomposition

We now show that the size of the decomposition is indeed exponential. For this, we have to estimate the size of the generated LTL component formulas, their number and the size of the FO formula.

Size of the component formulas:
From the definition of the set $\mathrm{cf}(\delta)$, we get $s(\bar{\delta}) \leq s(\bar{\varphi}) + c$ for the cases of $\bar{\varphi}, X\bar{\varphi}, F\bar{\varphi}$

and $G\bar{\varphi}$ and $s(\bar{\delta}) \leq s(\bar{\varphi}) + s(\bar{\psi}) + c$ for the other cases and some $c \in \mathbb{N}$. Thus, for the input formula γ, we have $s(\bar{\gamma}) \leq c * s(\gamma)$ for a $c \in \mathbb{N}$ and all $\bar{\gamma} \in \text{ecf}(\gamma)$.

Number of the component formulas:
From the construction of the set $\text{cf}(\delta)$, we get $|\text{cf}(\delta)| \leq c * |\text{cf}(\varphi)|$ and $|\text{cf}(\delta)| \leq c * |\text{cf}(\varphi)| * |\text{cf}(\psi)|$ for $\delta \in \{X\varphi, F\varphi, G\varphi\}$, respectively $\delta \in \{\varphi U \psi, \psi R \varphi, \varphi \wedge \psi, \varphi \vee \psi\}$. This also holds for the set of extended component formulas. Thus, we have $|\text{ecf}(\gamma)| \in O(2^{s(\gamma)})$.

Size of the MSO formula:
We first observe that in all cases of (direct) subformulas φ, ψ of δ, we have for the auxiliary formulas $\alpha_{\delta,\bar{\delta}}$ that $|\alpha_{\delta,\bar{\delta}}| \leq |\alpha_{\varphi,\bar{\varphi}}| + |\alpha_{\psi,\bar{\psi}}| + |\beta_\delta| \leq 2 * |\beta_\delta|$ (with $|\alpha_{\psi,\bar{\psi}}| := 0$ for $\delta \in \{X\varphi, F\varphi, G\varphi\}$). For $\delta = \varphi U \psi$, we get the following size by the definition of β_δ: $|\beta_\delta| \leq c * |\beta_\psi| + 2 * \sum_{\bar{\varphi} \in \text{cf}(\varphi), \bar{\psi} \in \text{cf}(\psi)} (|\beta_\varphi| + |\beta_\psi| + 1) + \sum_{\bar{\varphi} \in \text{cf}(\varphi)} (|\beta_\varphi| + 1)$ for a $c \in \mathbb{N}$. Obviously, this is in $O(c * |\text{cf}(\varphi)| * |\text{cf}(\psi)| * (|\beta_\varphi| + |\beta_\psi|))$ for a $c \in \mathbb{N}$. By using similar arguments, we get the same bound for the other cases of (direct) subformulas. Thus, the size of β_γ is at most $|\text{cf}(\gamma)|^{|s(\gamma)|} \in O(2^{|s(\gamma)| * \log_2(|\text{cf}(\gamma)|)})$. With $|\text{cf}(\gamma)| \in O(2^{s(\gamma)})$ from above, we get $|\beta_\gamma| \in O(|2^{(s(\delta))^2}|)$.
We conclude, that the size of the decomposition is $s(\bar{\gamma}) * |\text{cf}(\gamma)| + |\beta_\gamma| \in O(2^{(s(\gamma))^2})$. As the theorem works only for LTL formulas in NNF, we need a preprocessing for general LTL formulas to convert it. However, this conversion is known to be linear. Thus, in total, we have the same complexity.

5.5 Summary and Further Remarks

In this chapter, we have seen a composition technique for LTL over composed words. On the one hand, we looked at the setting in which all components were finite words and there were finitely or infinitely many components. On the other hand, we considered a finite number of components where the last component was an infinite word. We did not consider that the composed words may have a special structure. Especially in the second setting, one may look at the case where the composed word is in a regular language. Thus, one could use that there is at least one word which can be written as uv^ω for finite words u, v [Büc62, Tho97b].

We have shown in the last section that the size of the decomposition is at most exponential in the size of the given formula. A natural question which arises is whether this upper bound can be further improved.

One direction to generalize the LTL composition technique is to look at more general index structures and components. A natural idea is the generalization to trees instead of words. Here, we may either allow one successor component or

several. The second possibility results in a tree as index structure. This setting is considered in the next chapter with CTL over these sum trees. We show a composition theorem which generates – as the one in this chapter – a decomposition which is exponential in the size of the given formula.

6 Composition for Sums of Trees and CTL

In this chapter, we generalize the composition theorem of the last chapter to CTL over a sum of trees. We look at a sum tree, defined via an index tree with (marked) trees as labels for the nodes, as in Definition 2.6. We reduce the truth of a CTL formula in the sum to truth values of CTL formulas for the components (the "component formulas") and a CTL+X^{-1} formula in the index structure which expresses which component formulas have to hold at which component (described by an interface information tuple).

The idea of the composition theorem is a generalization of the idea for the case of LTL formulas over a disjoint ordered sum of words as considered in the last chapter: Again, we first convert the given formula to negation normal form and inductively define a set of component formulas. For paths in the sum tree, this set expresses all possibilities of how the fragments of these paths can look like in the components. Generalizing the set for the LTL formulas is straight-forward: They are simply adapted to compensate for the fact that paths may go in more than one direction and that we have either existential (E) or universal (A) quantification over these possibilities.

Then, we inductively define formulas for the index structure which express the allowed combinations of these fragments of the paths for the different components for all types of subformulas. Here, we need some more effort than in the LTL case – in particular we need a logic for the index tree which is more expressive than CTL, namely CTL+X^{-1}. For notational convenience, we explain the idea of the construction for the formulas $E(rUs)$ and $A(rUs)$ in a simplified setting: a sum tree for an index tree of height 1 with index nodes $\epsilon, 1$ and 2 and corresponding trees $\underline{t}_\epsilon, \underline{t}_1, \underline{t}_2$.

6.1 Composition Idea

Idea for $E(rUs)$

We first look at the formula $E(rUs)$ at the root of the sum tree. Here, we may have the following cases, also shown in Figure 6.1(1)–(3):

(1) The formula $E(rUs)$ may hold at $\underline{t}_\epsilon$.

(2) There may be a path fragment in $\underline{t}_\epsilon$ labeled with r which ends in a leaf labeled with c_1 – i.e., $E(rUc_1)$ holds in $\underline{t}_\epsilon$ – and which is continued in $\underline{t}_1$ by $E(rUs)$. (Recall that for the construction of the sum, the c_1-labeled node in $\underline{t}_\epsilon$ is replaced with the tree $\underline{t}_1$.)

(3) We may have the analogous situation of case (2) but with $E(rUc_2)$ at $\underline{t}_\epsilon$ and $E(rUs)$ in $\underline{t}_2$.

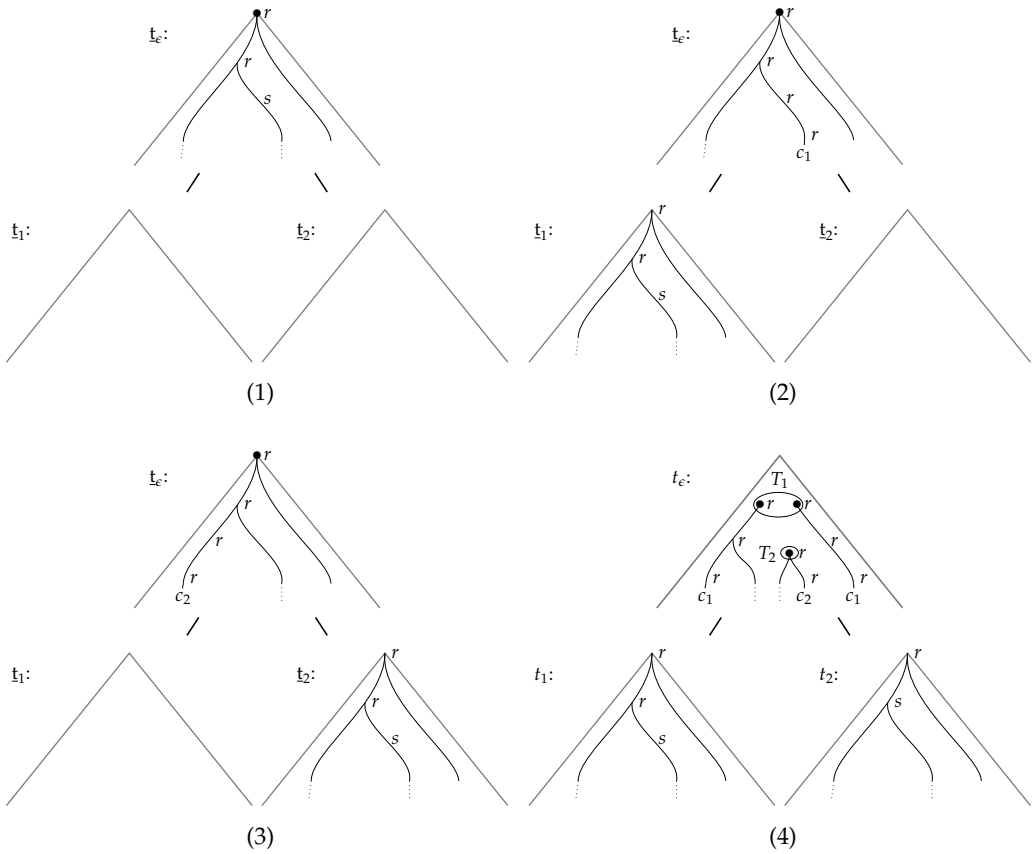


Figure 6.1: $E(rUs)$ at One Node – Cases (1)–(3) and Example for Set of Nodes – (4)

As in the last chapter, we have to consider that $\delta := E(rUs)$ is used inside another formula as, e.g., in $AX\delta$ or in $E(\delta Ut)$. Here, in general, $E(rUs)$ has to hold at more than one node of $\underline{t}_\epsilon$. If $E(rUs)$ should hold at a set T of nodes of the tree $\underline{t}_\epsilon$, we get all combinations of the cases above. For example, T may be partitioned into two subsets T_1, T_2, where we have $E(rUc_1)$ for T_1 and $E(rUc_2)$ for T_2 as shown in Figure 6.1(4), which amounts to $E(rU(c_1 \vee c_2))$ in $\underline{t}_\epsilon$ for T. Furthermore, both conditions for the successors have to be fulfilled, in this example, we must have $E(rUs)$ at $\underline{t}_1$ and $\underline{t}_2$. Considering all combinations of the cases (1)–(3) gives us the following component formulas for $\underline{t}_\epsilon$: $E(rUs), E(rUc_1), E(rUc_2), E(rU(c_1 \vee s)), E(rU(c_2 \vee s)), E(rU(c_1 \vee c_2)), E(rU(c_1 \vee c_2 \vee s))$. (We must have $E(rUs)$ for the successor $\underline{t}_i$ if c_i occurs in the component formula for $\underline{t}_\epsilon$ for $i \in \{1, 2\}$.) We further add the formula s to the list of these formulas. The reason is that we want to have a uniform construction which also works for formulas like $E(\varphi Us)$ instead of $E(rUs)$. (For $E(\varphi Us)$ we have: If s holds already at the root, we should not force inductively any conditions for φ at the next component(s).)

So far, this is a straight-forward generalization of the case rUs in LTL. We now look at $E(rUs)$ for an index tree of arbitrary height. Here, for $E(rUs)$ in the sum tree, we have to describe that there exists an index path (fragment) such that for each index the tree at this index satisfies one of the component formulas from above. The first idea is that there is an index path and an index on this path for which the component has to fulfill $E(rUs)$ (or s) and the components before have to satisfy one of the other component formulas. So far, this describes an EU-condition over the index tree. However, this condition is not sufficient as the index path always has to continue in "correct" direction: If we are able to get to a c_1 node in a component on the index path, the index path has to continue to the left (and analogously for c_2 to the right).

We explain this compatibility condition in more detail by an example shown in Figure 6.2. For simplicity, we consider $E(rUs)$ only at one node, i.e., we only need the component formulas $s, E(rUs), E(rUc_1)$ and $E(rUc_2)$. A path (fragment) in the sum tree which satisfies rUs is divided into parts in the components which satisfy rUc_1 or rUc_2 and a part which satisfies s or rUs. So all components along the path in the index tree satisfy $E(rUc_1)$ or $E(rUc_2)$ up to a component in which s or $E(rUs)$ holds as shown in Figure 6.2.

Additionally, we must have $E(rUc_1)$, $E(rUc_2)$ or $E(rUs)$ at the *right* successor $\underline{t}_2$ of the root as we have $E(rUc_2)$ at the root itself. Analogous conditions have to hold for the whole path fragment: To express conditions like "the index path continues to the *right* with $E(rUc_1)$ or $E(rUc_2)$ if we have $E(rUc_2)$ at the tree of the current index" over *the whole index path fragment* (up to the component where

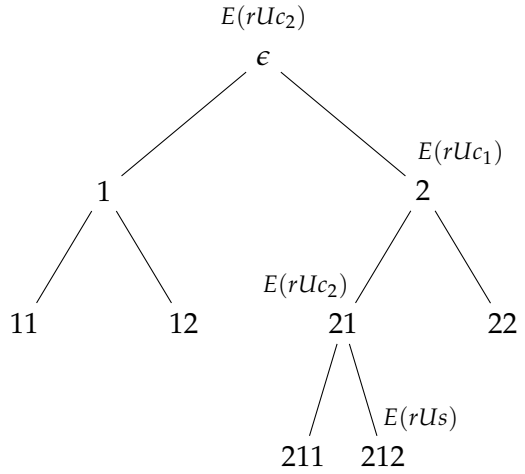


Figure 6.2: Example of an Index Tree with Annotated Formulas for $E(rUs)$

$E(rUs)$ holds), we use the past operator X^{-1} (from CTL+X^{-1}). Because of the tree structure the past operator X^{-1} allows to address the unique previous node for the nodes on the index path fragment described by the EU-condition. Note that it is not possible to access the next node *on a given index path fragment* by using EX or AX quantifiers. We use a CTL+X^{-1} formula which expresses conditions like: If the current index h (on the index path fragment) is a right successor and the tree at the predecessor fulfills $E(rUc_2)$, then we must have one of the component formulas $E(rUc_1)$ or $E(rUc_2)$ in the tree at the current index h.

Idea for $A(rUs)$

We now explain the idea for $A(rUs)$ in the sum tree. Again, we first consider the setting that the index tree only contains the nodes ϵ, 1 and 2. There are three reasons, why a single path fulfills rUs in the sum: It may lie completely in $\underline{t}_\epsilon$ or it may be spread over $\underline{t}_\epsilon$ and $\underline{t}_1$, respectively $\underline{t}_\epsilon$ and $\underline{t}_2$. If we express these reasons in CTL *for all paths* in the sum tree, we get the following cases for the components:

(1) We may have rUs for all paths of $\underline{t}_\epsilon$.

(2) All paths of $\underline{t}_\epsilon$ may satisfy rUc_1 and $A(rUs)$ holds at the root of $\underline{t}_1$.

(3) All paths of $\underline{t}_\epsilon$ may satisfy rUc_2 and $A(rUs)$ holds at the root of $\underline{t}_2$.

(4) Furthermore, we may have any combination of these cases where the set of paths is divided into subsets which fulfill one of the conditions above. Here the conditions for the successors have to be combined.

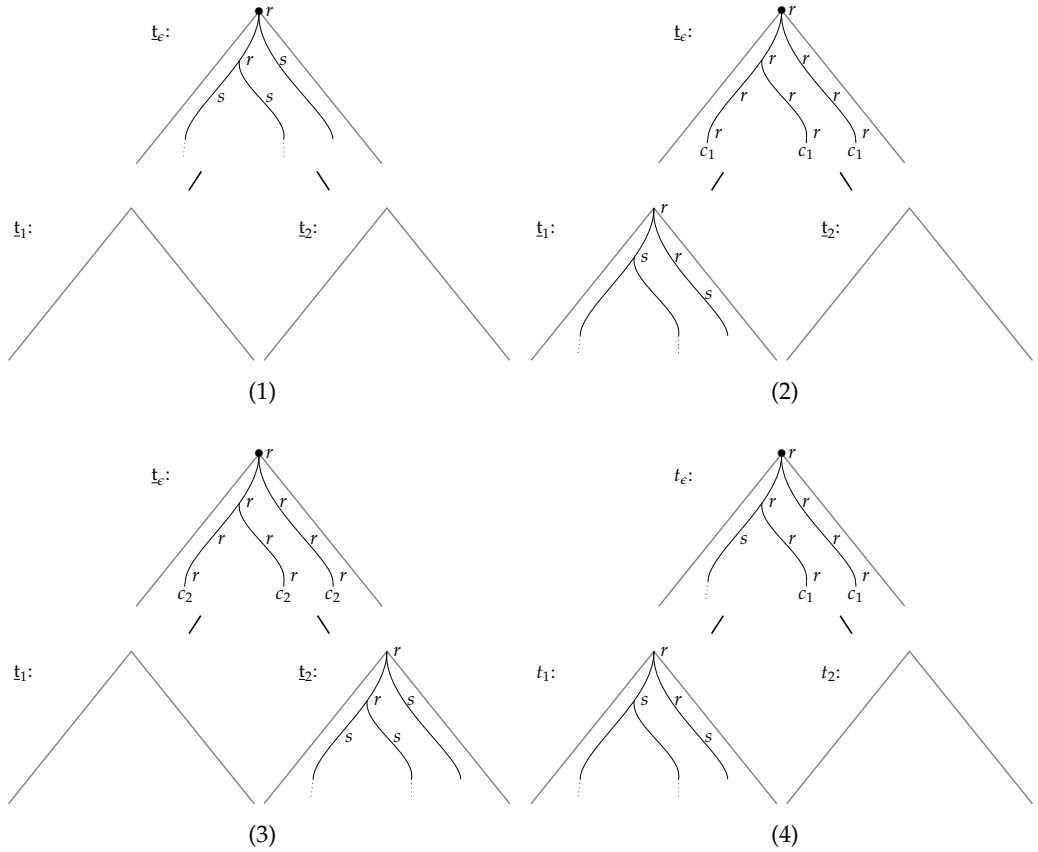


Figure 6.3: $A(rUs)$ Cases (1)–(3) and Example for Combination – (4)

In Figure 6.3(1)–(3), we show the first three cases. Furthermore, in Figure 6.3(4), we see an example for the last case: In $\underline{t}_\epsilon$, we have on some paths rUs and on the other paths rUc_1 (expressed by $A(rU(s \vee c_1))$). Here, we must have the combination of the conditions for the successors, which means here $A(rUs)$ at $\underline{t}_1$.

Together, this gives us the following component formulas: s, $A(rUs)$, $A(rUc_1)$, $A(rUc_2)$, $A(rU(c_1 \vee s))$, $A(rU(c_2 \vee s))$, $A(rU(c_1 \vee c_2))$, $A(rU(c_1 \vee c_2 \vee s))$. (Again we added s to the list – the reason is the same as for $E(rUs)$.) We can use the same component formulas if we consider $\delta = A(rUs)$ at more than one node as in the formula $AX\delta$. Consider, e.g., that δ shall hold at two nodes u_1, u_2 of the sum tree. One reason for this is $A(rUs)$ at u_1 in $\underline{t}_\epsilon$ and $A(rUc_1)$ at u_2 (and $A(rUs)$ at the root of $\underline{t}_2$). This is already captured by $A(rU(s \vee c_1))$ both at u_1 and u_2 as it results in the same conditions for the successor trees $\underline{t}_1$ and $\underline{t}_2$.

Let us now consider the formula $A(rUs)$ for an index tree of arbitrary height. We have to ensure that every single path in the sum tree on which rUs holds

corresponds to path fragments with rUc_1 or rUc_2 and a path fragment rUs in the components. Our task is to describe this condition for the set of all paths using the component formulas from above.

We call the component formulas s and $A(rUs)$ complete. The other formulas (which contain at least one c_1 or c_2) are called incomplete. To satisfy rUs on all paths, it is sufficient if we have on all index paths incomplete formulas up to a component which satisfies a complete formula. However, this is not the only possible reason why $A(rUs)$ holds in the sum tree.

Consider the following example (shown in Figure 6.4): $A(rU(c_2 \vee s))$ holds at $\underline{t}_2$. Here, it suffices to check conditions only for the right successor tree $\underline{t}_{22}$. (Note that in a tree where $A(rU(c_2 \vee s))$ holds, there may be a path with rUs which can be extended to a c_1 node. Thus, we may have a left successor tree.) In general, whenever a component index on an index path satisfies $A(rU(c_2 \vee s))$ or $A(rUc_2)$ the left successor tree does not need to fulfill any further condition.

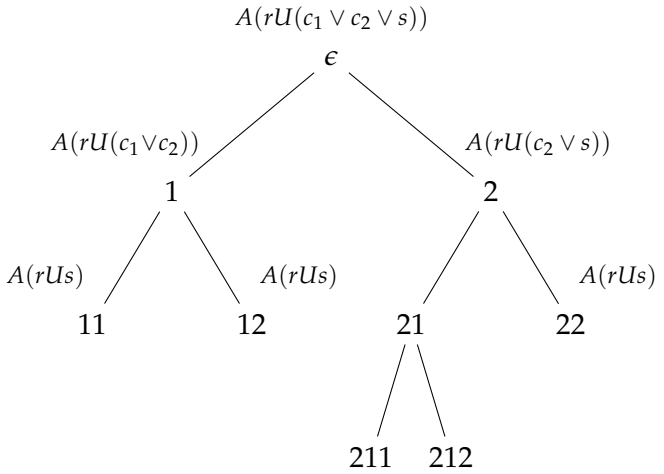


Figure 6.4: Example of an Index Tree with Annotated Formulas for $A(rUs)$

So basically, we allow to abort index paths earlier for a left successor if we have $A(rU(c_2 \vee s))$ or $A(rUc_2)$ at the tree for the current index. This and the dual case (right successor and c_1) can be expressed using CTL+X^{-1} logic: We allow incomplete formulas until we either have a component which satisfies a complete formula or we are at an index which is a left successor and we have $A(rU(c_2 \vee s))$ or $A(rUc_2)$ at the predecessor (like $\underline{t}_{21}$ with the predecessor $\underline{t}_2$ in Figure 6.4).

Outline

Having shown the basic ideas for $A(rUs)$ and $E(rUs)$, we now present the outline of this chapter:

1. We first present the general inductive definition of the component formulas, i.e., for atomic and negated atomic formulas and for $AX\varphi$, $EX\varphi$, $A(\varphi U\psi)$, $E(\varphi U\psi)$, $A(\psi R\varphi)$ and $E(\psi R\varphi)$. (Note that we need both the E- and A-quantified formulas as we use negation normal form.) As in last chapter, we first define the component formulas and then the extended component formulas which inductively contain the component formulas for the subformulas of the given formula.

2. We present the composition theorem, which allows to reduce the truth of a CTL formula in the sum tree to truth values of the (CTL) component formulas and a CTL+X^{-1} formula over the index tree. For this, we show the construction of the index formula and present detailed constructions for every type of CTL formula δ and a formal proof for the cases $E(\varphi U\psi)$ and $A(\varphi U\psi)$. As in the previous chapter, the index formula β_δ is defined by inductively using auxiliary formulas $\alpha_{\delta,\bar{\delta}}$. Recall that the auxiliary formula $\alpha_{\delta,\bar{\delta}}$ contains the conditions why δ holds in the sum tree for the case that we have the component formula $\bar{\delta}$ at the current component. Furthermore, β_δ is equivalent to the disjunction over "$\bar{\delta}$ at the current component" and $\alpha_{\delta,\bar{\delta}}$ holds.

3. We estimate the size and the number of component formulas and the size of the CTL+X^{-1} formula in the composition theorem.

For better readability, we restrict ourselves to sum trees build by an index tree with at most $\{c_1, c_2\}$-marked trees, i.e., the index tree is at most binary branching. The generalization to at most n-marked trees as components is straight-forward.

6.2 Technical Preliminaries

We present the set of component formulas $\mathrm{cf}(\delta)$ for a given CTL formula δ. We always use $\bar{\delta}$ to denote a formula from $\mathrm{cf}(\delta)$. For atomic formulas $\delta = r$, we define $\mathrm{cf}(\delta) = \{r\}$ and for $\delta = \neg r$, we set $\mathrm{cf}(\delta) = \{\neg r\}$. Using the given component formulas for φ and ψ, we get for $E(\varphi U\psi)$ and $A(\varphi U\psi)$ an inductive version of the component formulas from the examples $E(rUs)$, respectively $A(rUs)$:

- $\mathrm{cf}(E(\varphi U\psi)) = \{\bar{\psi}, E(\bar{\varphi}U\bar{\psi}), E(\bar{\varphi}Uc_1), E(\bar{\varphi}Uc_2), E(\bar{\varphi}U(c_1 \vee \bar{\psi})), E(\bar{\varphi}U(c_2 \vee \bar{\psi})),$ $E(\bar{\varphi}U(c_1 \vee c_2)), E(\bar{\varphi}U(c_1 \vee c_2 \vee \bar{\psi})) \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$

- $\mathrm{cf}(A(\varphi U \psi)) = \{\bar{\psi}, A(\bar{\varphi} U \bar{\psi}), A(\bar{\varphi} U c_1), A(\bar{\varphi} U c_2), A(\bar{\varphi} U (c_1 \vee \bar{\psi})), A(\bar{\varphi} U (c_2 \vee \bar{\psi})), A(\bar{\varphi} U (c_1 \vee c_2)), A(\bar{\varphi} U (c_1 \vee c_2 \vee \bar{\psi})) \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$

We now look at the cases $E(\psi R \varphi)$ and $A(\psi R \varphi)$. Before we begin, we repeat some useful formula transformations that introduce negation, but which we only use *for the component formulas*. Recall the Weak-Until operator W: $E(\bar{\varphi} W \bar{\psi})$ holds if "$E((\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi})$" holds. The latter is no CTL formula, but it can[1] be defined as $\neg A((\bar{\varphi} \wedge \neg\bar{\psi}) U (\neg\bar{\varphi} \wedge \neg\bar{\psi}))$ (see Section 2.2, e.g., [BK08]). Furthermore, the R-operator can be expressed in terms of the W-operator: $E(\bar{\psi} R \bar{\varphi})$ is equivalent to $E(\bar{\varphi} W (\bar{\varphi} \wedge \bar{\psi}))$. The dual transformations hold for the A-quantified formulas.

We start with the component formulas for $E(\psi R \varphi)$ and explain the idea for $\varphi = r$ and $\psi = s$. We only look at the component formulas for the current component. In general, these formulas (and their subformulas) imply conditions for the next components (as already shown in the introduction for $E(rUs)$) – this will be checked later via the formulas for the index tree. We first consider $E(\psi R \varphi)$ at one node in the current component and discuss the possible cases which are also shown in Figure 6.5.

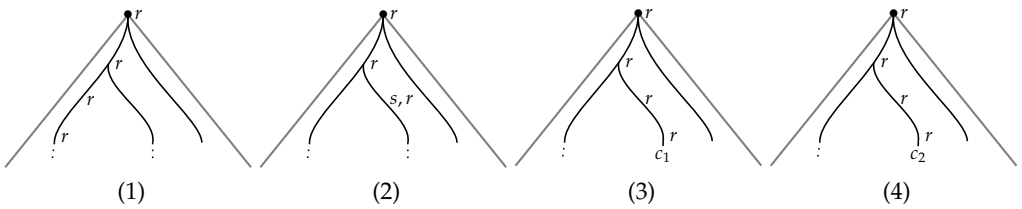


Figure 6.5: $E(sRr)$ at One Node

First, we may have a path with r on all nodes which is completely in the current component. Thus, we get the formula $EG\bar{\varphi}$. Second, we may have sRr on a path completely inside the current component, resulting in $E(\bar{\psi} R \bar{\varphi})$. Third and fourth, we may have a path labeled by r which ends either in c_1 or in c_2 (and which will be continued in the left, respectively the right successor). Thus, we get the formulas $E(\bar{\varphi} U c_1)$ and $E(\bar{\varphi} U c_2)$. If we look at more than one node, we get all combinations of the four cases above. We have to check that these combinations are still expressible in CTL. For this, we consider two representative examples. For $EG\bar{\varphi}$ on some and $E(\bar{\varphi} U c_1)$ on the other nodes, we can use the W-operator to combine these cases and get $E(\bar{\varphi} W c_1)$. For the combination $E(\bar{\psi} R \bar{\varphi})$ and $E(\bar{\varphi} U c_1)$, we use $E(\bar{\psi} R \bar{\varphi}) = E(\bar{\varphi} W (\bar{\varphi} \wedge \bar{\psi}))$ and get $E(\bar{\varphi} W ((\bar{\varphi} \wedge \bar{\psi}) \vee c_1))$.

In total, we get the following component formulas:

[1] Note that we can use negation on the level of component formulas.

- $\mathrm{cf}(\psi R\varphi) = \{EG\bar{\varphi}, E(\bar{\psi}R\bar{\varphi}), E(\bar{\varphi}Uc_1), E(\bar{\varphi}Uc_2), E(\bar{\varphi}Wc_1), E(\bar{\varphi}Wc_2), E(\bar{\varphi}U(c_1 \vee c_2)), E(\bar{\varphi}W(c_1 \vee c_2)), E(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_1)), E(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_2)), E(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_1 \vee c_2)) \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$.

We continue with the component formulas for $A(\psi R\varphi)$ and show the first four examples in Figure 6.6. The first case is that we may have r on all paths of the current tree. This gives us the component formula $AG\bar{\varphi}$. Second, we may have sRr on all paths of the current tree, which gives us $A(\bar{\psi}R\bar{\varphi})$ and is equivalent to $A(\bar{\varphi}W(\bar{\varphi} \wedge \bar{\psi}))$. In the cases that all paths have to be continued either only in the left or only in the right successor, we get $A(\bar{\varphi}Uc_1)$, respectively $A(\bar{\varphi}Uc_2)$.

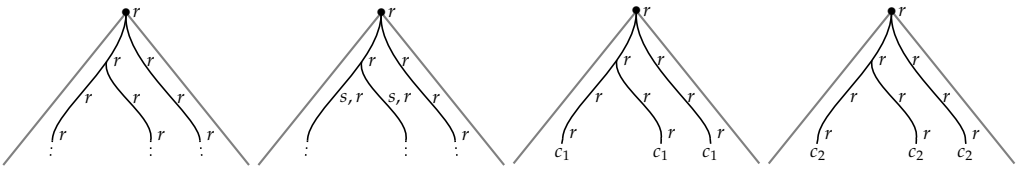


Figure 6.6: $A(sRr)$ – First Four Cases

Furthermore, we can have any combination of these cases, e.g., some paths may have Gr and some rUc_1. Using the W-operator, we get $A(\bar{\varphi}Wc_1)$ in this case. With the equivalence $A(\bar{\psi}R\bar{\varphi}) = A(\bar{\psi}W(\bar{\varphi} \wedge \bar{\psi}))$, the combination of "some paths sRr and some rUc_1" can be expressed by $A(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_1))$. In total, we get the following component formulas for $A(\psi R\varphi)$:

- $\mathrm{cf}(\psi R\varphi) = \{AG\bar{\varphi}, A(\bar{\psi}R\bar{\varphi}), A(\bar{\varphi}Uc_1), A(\bar{\varphi}Uc_2), A(\bar{\varphi}Wc_1), A(\bar{\varphi}Wc_2), A(\bar{\varphi}U(c_1 \vee c_2)), A(\bar{\varphi}W(c_1 \vee c_2)), A(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_1)), A(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_2)), A(\bar{\varphi}W((\bar{\varphi} \wedge \bar{\psi}) \vee c_1 \vee c_2)) \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$.

The constructions for the formulas $EX\varphi$ and $AX\varphi$ are straight-forward: for EXs at one node, we may either have a successor directly labeled with s or with c_1 or with c_2 (and in the latter two cases s at the tree of the left, respectively the right successor of the index tree). At more than one node we may have all possible combinations of these cases. For AXs at one node, we either get directly s on all successors – resulting in the component formula $AX\bar{\varphi}$ – or on all successors c_1, respectively c_2 – resulting in AXc_1, respectively AXc_2 – or any combination, e.g., $AX(\bar{\varphi} \vee c_2)$. The same conditions occur for more than one node. Thus, we get:

- $\mathrm{cf}(EX\varphi) = \{EX\bar{\varphi}, EXc_1, EXc_2, EX(\bar{\varphi} \vee c_1), EX(\bar{\varphi} \vee c_2), EX(c_1 \vee c_2), EX(\bar{\varphi} \vee c_1 \vee c_2) \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$ and

- $\mathrm{cf}(AX\varphi) = \{AX\bar{\varphi}, AXc_1, AXc_2, AX(\bar{\varphi} \vee c_1), AX(\bar{\varphi} \vee c_2), AX(c_1 \vee c_2), AX(\bar{\varphi} \vee c_1 \vee c_2) \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$.

For disjunction and conjunction, we get analogous conditions as in the last chapter:

- $\mathrm{cf}(\varphi \wedge \psi) = \{\bar{\varphi} \wedge \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$ and
- $\mathrm{cf}(\varphi \vee \psi) = \{\bar{\varphi}, \bar{\psi}, \bar{\varphi} \vee \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$.

The extended component formulas inductively sum up the component formulas for the subformulas and are defined as in the last chapter: $\mathrm{ecf}(\delta) = \bigcup\{\mathrm{cf}(\varphi) \mid \varphi \in \mathrm{CL}(\delta)\}$.

6.3 CTL Composition Theorem

We now present the composition theorem for CTL.

Theorem 6.1. Given a signature σ, the following statement holds: For every σ-CTL formula γ of modal depth r, we can compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta \rangle$ with CTL σ-formulas γ_j of modal depth at most r – interpreted in the components – and a CTL+X^{-1} formula β – interpreted in the index tree – such that for every sum tree $\underline{t}_S$ of an index tree $\underline{t}_I$ with (at most n-marked) σ-trees $\underline{t}_z$ (:=$\mathrm{val}_{\underline{t}_I}(z)$) for $z \in \mathrm{dom}_{\underline{t}_I}$ as components, we have:

$$\underline{t}_s \models \gamma \Leftrightarrow (\underline{t}_I, \tilde{S}_1, \ldots, \tilde{S}_n, I_1, \ldots, I_m) \models \beta$$

where β contains the following predicate symbols:

- $S_1, \ldots, S_n$ where S_j is interpreted by $\tilde{S}_j$ which expresses that the current node is a j-th successor of its parent node and
- $P_{\gamma_1}, \ldots, P_{\gamma_m}$ where P_{γ_k} is interpreted by the set $I_k := \{z \in \mathrm{dom}_{\underline{t}_I} \mid \underline{t}_z \models \gamma_k\}$.

The proof of this composition theorem is structured analogously to the one from the last chapter. We use an induction over the subformulas of the given formula γ and over the "sub-sums" – the sums of trees starting from an index of the original sum – to construct interface information tuples. In the induction base, we give interface information tuples for the atomic and negated atomic formulas at any sub-sum (as these formulas only talk about the current component). In the induction step, we use the induction hypothesis for a subformula and the same or a smaller

sum. We give formal proofs for the cases $E(\varphi U \psi)$ and $A(\varphi U \psi)$. Furthermore, we present detailed explanations for all types of subformulas. We conclude this section by estimating the size of the constructed interface information tuples.

Before we start with the induction, we present some notation: For the inductive construction of the interface information tuples $\langle \delta_1, \ldots, \delta_k; \beta_\delta \rangle$ ($\delta \trianglelefteq \gamma$), we use an enumeration of the (extended) component formulas $\mathrm{ecf}(\delta)$ from the previous section as $\delta_1, \ldots, \delta_k$. To ease notation, we write $P_{\bar{\delta}}$ for $\bar{\delta} \in \mathrm{ecf}(\delta)$ instead of P_{δ_j}, $j \in [k]$.

We need a more general version of the equivalence in Theorem 6.1 in order to use it in the induction: We show the equivalence for all parts of the sum tree starting at the current component (with index h) and any set[2] T of nodes in the component $\underline{t}_h$. For every $u \in T$, we consider the part of the sum tree which starts in the h-th component at the node u. To express the equivalence for these parts, we have to adapt the interpretation of the sets $I_{\bar{\delta}}$ s.t. they also "start" at T. This is formally done as follows.

Let $\underline{t}_I^{\sqsupseteq h}$ denote the part of the index tree which starts at component $\underline{t}_h$, defined by $\mathrm{dom}_{\underline{t}_I^{\sqsupseteq h}} := \{v \mid hv \in \mathrm{dom}_{\underline{t}_I}\}$, $\mathrm{val}_{\underline{t}_I^{\sqsupseteq h}}(v) := \mathrm{val}_{\underline{t}_I}(hv)$ and $v \sqsupseteq v'$ in $\underline{t}_I^{\sqsupseteq h} :\Leftrightarrow hv \sqsupseteq hv'$ in $\underline{t}_I$. Furthermore, let $\underline{t}_s^{\sqsupseteq h}$ be the part of the sum tree $\underline{t}_s$ which starts at the h-th component, i.e., $\underline{t}_s^{\sqsupseteq h}$ is the sum tree of the trees $\mathrm{val}_{\underline{t}_I^{\sqsupseteq h}}(z), z \in \mathrm{dom}_{\underline{t}_I^{\sqsupseteq h}}$. Using the abbreviation $\underline{t}_z$ for $\mathrm{val}_{\underline{t}_I}(z)$, we define the set $I_{\bar{\delta}}^{\sqsupseteq h,T}$ as $\{z \sqsupset h \mid \underline{t}_z \models \bar{\delta}\} \cup \{h \mid \forall u \in T : \underline{t}_h[u] \models \bar{\delta}\}$, i.e., it contains all indices after h of trees which satisfy the formula $\bar{\delta}$ and the index h if $\bar{\delta}$ holds for all states of the set T in the h-th tree from the node u onwards. We show the following generalization of the equivalence in Theorem 6.1 for any set T of nodes in the current component:

$$\forall u \in T : \underline{t}_s^{\sqsupseteq h}[u] \models \delta \Leftrightarrow (\underline{t}_I, \tilde{S}_1, \tilde{S}_2, \{I_{\bar{\delta}}^{\sqsupseteq h,T} \mid \bar{\delta} \in \mathrm{ecf}(\delta)\}) \models \beta_\delta.$$

For the construction of the formula β_δ in the interface information tuples, we use (as in the last chapter) auxiliary formulas $\alpha_{\delta,\bar{\delta}}$ for $\bar{\delta} \in \mathrm{ecf}(\delta)$ that store the conditions for the case that $\bar{\delta}$ holds at the current component and get $\beta_\delta = \bigvee_{\bar{\delta} \in \mathrm{cf}(\delta)} (P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$. We now show that this generalized equivalence holds by induction.

Induction base:
For a predicate $\delta = r$ or negated predicate $\delta = \neg r$, we have $\mathrm{cf}(\delta) = \{\delta\}$ and set $\beta_\delta = P_\delta$. We have for all nodes $u \in T$: $\underline{t}_s^{\sqsupseteq h}[u] \models \delta$ iff for all $u \in T$: $\underline{t}_h[u] \models \delta$. This is equivalent to $(\underline{t}_I, \{\tilde{S}_i \mid i \in [n]\}, I_{\bar{\delta}}^{\sqsupseteq h,T}) \models P_\delta$.

[2] The equivalence is shown for any set T of nodes in the current component of the tree, because the "context" may, e.g., force δ at a path like in $E(\delta U s)$ or at several successors like in $AX\delta$.

Induction hypothesis:
Let $\underline{t}_h$ be the current component, T a set of nodes in $\underline{t}_h$ and δ the current formula. The induction hypothesis states that the equivalence

$$\forall u \in T : \underline{t}_s^{\sqsupseteq y}[u] \models \varphi \Leftrightarrow (\underline{t}_I, \tilde{S}_1, \tilde{S}_2, \{I_{\bar{\varphi}}^{\sqsupseteq y,T} \mid \bar{\varphi} \in \mathrm{ecf}(\varphi)\}) \models \beta_\varphi$$

holds in the following setting: φ is a proper subformula of δ and the part of the sum may be the same or smaller, i.e.[3] $h \sqsubseteq y$ holds

Induction step:

$E(\varphi U \psi)$

We first discuss that $E(\varphi U \psi)$ holds in the sum at one node u which belongs to the current component $\underline{t}_h$ and afterwards at a set T of nodes which belong to the current component. The following cases were already shown in Figure 6.1 for $\varphi = r$ and $\psi = s$. For the first case, by definition of $E(\varphi U \psi)$, we either have ψ directly at the node u (a special case of case 1) or a path starts in u and there is a node v on that path where ψ holds and on all nodes up to v the formula φ holds. The node v lies in the sum tree at a node which belongs to either the current component $\underline{t}_h$ (case 1) or some component after it, starting in either the left or the right successor (cases 2 and 3).

We are given the component formulas $\bar{\varphi} \in \mathrm{cf}(\varphi)$ and $\bar{\psi} \in \mathrm{cf}(\psi)$ and the formulas $\beta_\varphi = \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$, respectively $\beta_\psi = \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$. We now sum up the possibilities for the components for $E(\varphi U \psi)$ *at one node* u in the sum. We define β_δ^1 as the disjunction over these possibilities.

For the special case of (1), we can directly apply the induction hypothesis as ψ is a proper subformula of δ. Thus, the formula β_ψ is satisfied with the interpretation $I_{\bar{\psi}}^{\sqsupseteq h,\{u\}}$ and we have that $(\underline{t}_I, \tilde{S}_1, \tilde{S}_2, \{I_{\bar{\delta}}^{\sqsupseteq h,\{u\}} \mid \bar{\delta} \in \mathrm{ecf}(\delta)\}) \models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$. Thus, we get $\bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ as first possibility for β_δ^1.

For case (1), the path fragment from u to v lies completely in (the part of the sum which belongs to) the current component. We can use the induction hypothesis for ψ at $\{v\}$ and the induction hypothesis for φ at the set of all nodes x from u up to v. Let J_1 denote this set. Then, we have that $\beta_\psi = \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ is satisfied for the interpretation $I^{\sqsupseteq h,\{v\}}$ and $\beta_\varphi = \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ for $I^{\sqsupseteq h,J_1}$. This gives us $\bar{\varphi} U \bar{\psi}$ on the path for the current component, respectively $E(\bar{\varphi} U \bar{\psi})$ as component formula and both $\alpha_{\varphi,\bar{\varphi}}$ and $\alpha_{\psi,\bar{\psi}}$ have to hold. Thus, we have that $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{E(\bar{\varphi} U \bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$ holds. This is the second possibility for β_δ^1.

[3] If y is later in the index tree, the sum tree starting at y is smaller.

In case (2), the node v (of the path) lies in the left successor component (with index $h1$) or in a later component. In particular, this means that (in the sum) we have φ on the part of the path which belongs to the h-th component. Using the induction hypothesis for the set of nodes of this part, we get $E(\bar{\varphi} U c_1)$ as component formula (and $\alpha_{\varphi,\bar{\varphi}}$ has to hold) because in the h-th component the part of the path ends with a c_1-marked node. Additionally, the remainder of the path starting at the first node of the remainder sum from index $h1$ onwards has to fulfill $\varphi U \psi$. Thus, we must have $E(\varphi U \psi)$ at the root of this remainder sum. If we can show that this is equivalent to "β^1_δ holds at the left successor $\underline{t}_{h1}$", this gives us $\bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{E(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta))$. Case (3) is analogous. The cases (2) and (3) give us the third and fourth possibility for β^1_δ.

Note that we still have to give a construction for β^1_δ. The ideas for this construction were already shown in the introduction: The path fragment from u to v is distributed over several components whose indices form a path fragment in the index tree. The first condition is that each component on this index path (fragment) has to fulfill a component formula which allows the path to be continued – i.e., which contains a c_1 or a c_2 – until a component where either $\bar{\psi}$ or $E(\bar{\varphi} U \bar{\psi})$ holds. We describe conditions $\gamma_{\delta,\mathrm{incomplete}}$ for the formulas which still can be continued and $\gamma_{\delta,\mathrm{complete}}$ for the other formulas. Both are defined via the following γ-formulas.

$$\gamma_{\delta,\psi} = \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$$

$$\gamma_{\delta,\varphi\psi} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (P_{E(\bar{\varphi} U \bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$$

$$\gamma_{\delta,1} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}})$$

$$\gamma_{\delta,2} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_2)} \wedge \alpha_{\varphi,\bar{\varphi}})$$

We set $\gamma_{\delta,\mathrm{incomplete}} := \gamma_{\delta,1} \vee \gamma_{\delta,2}$ and $\gamma_{\delta,\mathrm{complete}} := \gamma_{\delta,\psi} \vee \gamma_{\delta,\varphi\psi}$. The intuitive description of "an index path labeled with incomplete formulas until a complete formula" is simply $E(\gamma_{\delta,\mathrm{incomplete}} U \gamma_{\delta,\mathrm{complete}})$. However, as mentioned in the introduction, this index path has to fulfill a second condition up to the complete formula: It has to proceed to the left (respectively the right) if the path in the current component ends with a c_1 (respectively with c_2). This can be achieved by forcing conditions on the path fragment like "if the current index is a left successor then the predecessor has to fulfill a component formula which can end with c_1" defined as $\gamma_{\delta,\mathrm{back1}} := S_1 \to X^{-1}\gamma_{\delta,1}$, respectively $\gamma_{\delta,\mathrm{back2}} := S_2 \to X^{-1}\gamma_{\delta,2}$. Due to the past quantifier, these conditions have to start at a successor of the current index. Thus,

in total, we get:

$$\begin{aligned}\beta^1_\delta =& \gamma_{\delta,\text{complete}} \vee [\gamma_{\delta,\text{incomplete}} \wedge \\ & EX[E(\gamma_{\delta,\text{incomplete}} \wedge \gamma_{\delta,\text{back1}} \wedge \gamma_{\delta,\text{back2}}) U (\gamma_{\delta,\text{complete}} \wedge \gamma_{\delta,\text{back1}} \wedge \gamma_{\delta,\text{back2}})]]\end{aligned}$$

We now consider $E(\varphi U \psi)$ at a set T of nodes. Recall the cases (1)–(3) from above. At a set T of nodes, we get any combination of these cases. We discuss only one combination as the others are analogous: for some nodes of T the case (1) and for the other the case (3) holds. Then, all nodes of T satisfy the component formula $E(\bar{\varphi} U (\bar{\psi} \vee c_2))$ and we have both auxiliary formulas $\alpha_{\varphi,\bar{\varphi}}$ and $\alpha_{\psi,\bar{\psi}}$: $\bigvee_{\bar{\varphi} \in \text{cf}(\varphi)} \bigvee_{\bar{\psi} \in \text{cf}(\psi)} (P_{E(\bar{\varphi} U (\bar{\psi} \vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta^1_\delta))$.

In total, for $E(\varphi U \psi)$ at one node in the sum tree (especially at the root), we get $\beta^1_\delta = \bigvee_{\bar{\delta} \in \text{cf}^1(\delta)} (P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ with $\text{cf}^1(\delta) = \{\bar{\psi}, E(\bar{\varphi} U \bar{\psi}), E(\bar{\varphi} U c_1), E(\bar{\varphi} U c_2)\}$ and the auxiliary formulas defined below. Furthermore, for a set T of nodes in the sum tree, we get $\beta_\delta = \bigvee_{\bar{\delta} \in \text{cf}(\delta)} (P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ and again the auxiliary formulas from below. In both cases β^1_δ is defined as above.

$$\begin{aligned}\alpha_{\delta,\bar{\psi}} &= \alpha_{\psi,\bar{\psi}} \\ \alpha_{\delta,E(\bar{\varphi} U \bar{\psi})} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \\ \alpha_{\delta,E(\bar{\varphi} U c_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U c_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U (c_1 \vee \bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U (c_2 \vee \bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U (c_1 \vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U (c_1 \vee c_2 \vee \bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta^1_\delta) \wedge EX(S_2 \wedge \beta^1_\delta)\end{aligned}$$

Formal proof for $E(\varphi U \psi)$. Also for the formal proof, we stick to at most $\{c_1, c_2\}$-marked trees. The generalization to marked trees with at most n special symbols is straight-forward.

Let h be an index and T be a set of nodes in the marked tree $\underline{t}_h$. Recall that $\underline{t}_s^{\sqsupseteq h}$ is the part of the sum tree $\underline{t}_s$ which starts at the h-th component. For a node $u \in T$, we denote the set of paths starting in u by Π_u. We use the abbreviation $\tilde{S} = \{\tilde{S}_1, \tilde{S}_2\}$ for the interpretation of the predicate symbols S_1, S_2 by the left and right successors $\tilde{S}_1, \tilde{S}_2$.

Recall that $I_{\bar{\delta}}^{\sqsupset h,T}$ is defined as the set $\{z \sqsupset h \mid \underline{t}_z \models \bar{\delta}\} \cup \{h \mid \forall u \in T : \underline{t}_h[u] \models \bar{\delta}\}$, i.e., it contains all component indices after h which satisfy the formula $\bar{\delta}$ and h if $\bar{\delta}$ holds in it at the node u.

We have to show that the formula $E(\varphi U \psi)$ holds at a set T of nodes of the sum tree starting at component $\underline{t}_h$ iff the index tree with the interpretation $I_{\bar{\delta}}^{\sqsupset h,T}$ ($\bar{\delta} \in \mathrm{cf}(\delta)$) satisfies the formula β_δ. This is done in two steps: We first show that $E(\varphi U \psi)$ holds at the root of the sum tree starting at component $\underline{t}_h$ iff $I_{\bar{\delta}}^{\sqsupset h,\epsilon}$ ($\bar{\delta} \in \mathrm{cf}(\delta)$) satisfies β_δ^1. Afterwards, we show the general case for an arbitrary set T in the h-th component for which we use the result from the first step for the direct successor components $\underline{t}_{h1}$ and $\underline{t}_{h2}$.

We have $t_{\bar{s}}^{\sqsupset h}[\epsilon] \models E(\varphi U \psi)$ iff there is a node k such that $t_{\bar{s}}^{\sqsupset h}[k] \models \psi$ and for all i, $i \sqsubset k$ $t_{\bar{s}}^{\sqsupset h}[k] \models \varphi$ holds. Using the induction hypothesis for φ and ψ, we get:

- $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupset h,\{\epsilon\}} \mid \bar{\psi} \in \mathrm{ecf}(\psi)\}) \models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ or
- there is a node k in the current tree $\underline{t}_h$, such that $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupset h,\{k\}} \mid \bar{\psi} \in \mathrm{ecf}(\psi)\}) \models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ and we have $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupset h,J_1} \mid \bar{\varphi} \in \mathrm{ecf}(\varphi)\}) \models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ for $J_1 := \{i \in \mathrm{dom}_{\underline{t}_h} \mid i \sqsubset k\}$ or
- there is an index $z \sqsupseteq h1$ and a node k in the tree $\underline{t}_z$ s.t. $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupset z,\{k\}} \mid \bar{\psi} \in \mathrm{ecf}(\psi)\}) \models \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ and $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupset z,J_2} \mid \bar{\varphi} \in \mathrm{ecf}(\varphi)\}) \models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ for $J_2 := \{i \in \mathrm{dom}_{\underline{t}_z} \mid i \sqsubset k\}$ and
  for all indices y with $h \sqsubseteq y \sqsubset z$, we have a last c_i-labeled node k' and for $J_3 := \{i \in \mathrm{dom}_{\underline{t}_y} \mid i \sqsubset k'\}$: $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupset y,J_3} \mid \bar{\varphi} \in \mathrm{ecf}(\varphi)\}) \models \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ holds or
- we have the same situation as above but with $h2$ instead of $h1$.

Recall the definition of $\gamma_{\delta,\mathrm{complete}} = \gamma_{\delta,\psi} \vee \gamma_{\delta,\varphi\psi}$ and $\gamma_{\delta,\mathrm{incomplete}} = \gamma_{\delta,1} \vee \gamma_{\delta,2}$ with $\gamma_{\delta,\psi}$, $\gamma_{\delta,\varphi\psi}$, $\gamma_{\delta,1}$ and $\gamma_{\delta,2}$ from above. From the four cases above, we first observe that $\gamma_{\delta,\mathrm{complete}} \vee \gamma_{\delta,\mathrm{incomplete}}$ has to hold at the current component. Furthermore, there has to be an index path fragment with components in which $\gamma_{\delta,\mathrm{incomplete}}$ holds until a component where $\gamma_{\delta,\mathrm{complete}}$ holds. Moreover, on this path the left successor has to lie on the path if the formula $\gamma_{\delta,1}$ holds at the current tree and the right successor if $\gamma_{\delta,2}$ holds. This is equivalent to the statement that we are at the left (respectively right) successor, i.e., S_1 (respectively S_2) holds, and at the predecessor of this successor – i.e., at the current index – we have the formula $\gamma_{\delta,1}$ (respectively $\gamma_{\delta,2}$).

Thus, we get conditions $\gamma_{\delta,\mathrm{back1}} = S_1 \rightarrow X^{-1}\gamma_{\delta,1}$ and $\gamma_{\delta,\mathrm{back2}} = S_2 \rightarrow X^{-1}\gamma_{\delta,2}$ on the path. Together, we get $\gamma_{\delta,\mathrm{complete}} \vee [\gamma_{\delta,\mathrm{incomplete}} \wedge EX(E(\gamma_{\delta,\mathrm{incomplete}} \wedge \gamma_{\delta,\mathrm{back1}} \wedge$

$\gamma_{\delta,\text{back2}})U(\gamma_{\delta,\text{complete}} \wedge \gamma_{\delta,\text{back1}} \wedge \gamma_{\delta,\text{back2}}))]$. Thus, we have $(\underline{t}_I, \tilde{S}, \{I_{\bar{\delta}}^{\sqsupseteq h,\{\epsilon\}} \mid \bar{\delta} \in \text{ecf}(\delta)\}) \models \beta_\delta^1$.

It remains to show the result for a set T of nodes. We have $\forall u \in T : t_s^{\sqsupseteq h}[u] \models E(\varphi U \psi)$ iff $\forall u \in T$ there is a node $k \sqsupseteq u$ such that $t_s^{\sqsupseteq h}[k] \models \psi$ and for all i, $u \sqsubseteq i \sqsubset k$: $t_s^{\sqsupseteq h}[i] \models \varphi$ holds.

Using the induction hypothesis, we get for every $u \in T$:

- $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupseteq h,\{u\}} \mid \bar{\psi} \in \text{ecf}(\psi)\}) \models \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ or
- there is a node $k \sqsupseteq u$ in the current tree $\underline{t}_h$ s.t. $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupseteq h,\{k\}} \mid \bar{\psi} \in \text{ecf}(\psi)\}) \models \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ and $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupseteq h,J_1} \mid \bar{\varphi} \in \text{ecf}(\varphi)\}) \models \bigvee_{\bar{\varphi}\in\text{cf}(\varphi)}(P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ for $J_1 := \{i \in \text{dom}_{\underline{t}_h} \mid u \sqsubseteq i \sqsubset k\}$ or
- we have $E(\varphi U \psi)$ at the root of the sum starting at index $h1$ – thus, from above, we know that $(\underline{t}_I, \tilde{S}, \{I_{\bar{\delta}}^{\sqsupseteq h1,\{\epsilon\}} \mid \bar{\delta} \in \text{ecf}(\delta)\}) \models \beta_\delta^1$ holds – and we have a maximal c_1-labeled node k' and $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupseteq z,J_1} \mid \bar{\varphi} \in \text{ecf}(\varphi)\}) \models \bigvee_{\bar{\varphi}\in\text{cf}(\varphi)}(P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ with $J_1 := \{i \in \text{dom}_{\underline{t}_h} \mid u \sqsubseteq i \sqsubset k'\}$ or
- we have the same situation as above but with $h2$ instead of $h1$.

These cases are equivalent to: $(\underline{t}_I, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupseteq h,T} \mid \bar{\psi} \in \text{ecf}(\psi)\})$ satisfies the disjunction over the following cases.

1. $\bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$
2. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)} \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{E(\bar{\varphi}U\bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
3. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)}(P_{E(\bar{\varphi}Uc_1)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta^1))$
4. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)}(P_{E(\bar{\varphi}Uc_2)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\delta^1))$
5. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)} \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{E(\bar{\varphi}U(c_1\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta^1))$
6. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)} \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{E(\bar{\varphi}U(c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta_\delta^1))$
7. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)}(P_{E(\bar{\varphi}U(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta^1) \wedge EX(S_2 \wedge \beta_\delta^1))$
8. $\bigvee_{\bar{\varphi}\in\text{cf}(\varphi)} \bigvee_{\bar{\psi}\in\text{cf}(\psi)}(P_{E(\bar{\varphi}U(c_1\vee c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta^1) \wedge EX(S_2 \wedge \beta_\delta^1))$

Note that at the next component(s) only β_δ^1 has to hold as δ has to hold at the first node of the sum tree from the component $\underline{t}_{h1}$, respectively $\underline{t}_{h2}$. Thus, we can use the result from the first step.

□

$A(\varphi U \psi)$

For $A(\varphi U \psi)$, the possible cases were already shown in the example in Figure 6.3 for $\varphi = r$ and $\psi = s$. Again, we first consider $A(\varphi U \psi)$ at one node u of the sum. We either have ψ directly at u (a special case of (1)) or for all paths of the sum starting at u there exists a node v at which ψ holds and up to this node φ holds. These paths – or, to be more precise, the fragments of these paths on which $\varphi U \psi$ holds – may all lie completely in the part of the sum which belongs to the current component (case 1). They may also all end in the left successor or a later component of it (case 2). Analogously, they may all end in the right or a later component (case 3). Furthermore, some may end in the current component, some in a later component to the left and some in a later component to the right. This gives us all combinations of the cases (1),(2) and (3).

The special case of (1) is the same as above for $E(\varphi U \psi)$: We get $\bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$ as possibility for β_δ. For case (1), for all paths starting at u the node v (where ψ holds in the sum) lies in the current component. Let J denote the set of these nodes v. We can apply the induction hypothesis for J. So β_ψ holds for the interpretation $I_{\bar{\psi}}^{\exists h,J}$. Furthermore, for the set X of nodes x until the nodes of J, the formula φ holds in the sum and we can apply the induction hypothesis for φ and this set in $\underline{t}_h$. Together, we have $\bar{\varphi} U \bar{\psi}$ for every path in the component $\underline{t}_h$ and get $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi} U \bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$ as possibility for β_δ.

In case (2), all paths proceed to the left successor. In particular, this means that we have φ on the parts of the paths of the sum which belong to the current component and can use the induction hypothesis for φ and the set of nodes of these parts in $\underline{t}_h$. Apart from this, every[4] path in the component $\underline{t}_h$ has to end in a c_1 as the path in the sum continues to the left successor. Thus, we get the component formula $A(\bar{\varphi} U c_1)$. Furthermore, as every path continues in (the root of) the left successor, we must have $A(\varphi U \psi)$ in the remainder sum from the root of the component with index $h1$ onwards. Assuming that we have already shown that this means β_δ holds from $\underline{t}_{h1}$ onwards, we get for this case $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{A(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta))$. The case (3) is analogous.

We consider only one of the remaining cases as the arguments are similar for all combinations of the cases (1), (2) and (3). If the set of paths is divided into paths which go to the left and right successor, we have a combination of the cases (2) and (3), i.e., $\bar{\varphi} U c_1$ or $\bar{\varphi} U c_2$ holds for the parts of all the paths in the current component, giving the component formula $A(\bar{\varphi} U (c_1 \vee c_2))$. Furthermore, both

[4] Recall that we only consider paths starting at u.

successor components have to satisfy β_δ, which gives us $\bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}U(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta))$.

Looking at the formula $A(\varphi U\psi)$ at a set of nodes gives no further combinations as we already have all combinations of the cases (1), (2) and (3) and, furthermore, combining the special case of (1) with one of the others leads to a known combination. This can be seen by considering, e.g., the situation "the special case of (1) on some nodes and (2) on the other nodes". This gives the component formulas $\bar{\psi}$ and $A(\bar{\varphi}Uc_1)$ for which we can use the component formula $A(\bar{\varphi}U(c_1 \vee \bar{\psi}))$ which has the same conditions for the successor components.

In total, we get $\beta_\delta = \bigvee_{\bar{\delta}\in\mathrm{cf}(\delta)}(P_\delta \wedge \alpha_{\delta,\bar{\delta}})$ with the following auxiliary formulas:

$$
\begin{aligned}
\alpha_{\delta,\bar{\psi}} &= \alpha_{\psi,\bar{\psi}} \\
\alpha_{\delta,A(\bar{\varphi}U\bar{\psi})} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \\
\alpha_{\delta,A(\bar{\varphi}Uc_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}Uc_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}U(c_1\vee\bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}U(c_2\vee\bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}U(c_1\vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}U(c_1\vee c_2\vee\bar{\psi}))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta)
\end{aligned}
$$

We still have to give the construction for β_δ (without referring to β_δ itself at the next components). As for the formula $E(\varphi U\psi)$, we define formulas $\gamma_{\delta,\mathrm{incomplete}}$ and $\gamma_{\delta,\mathrm{complete}}$ with the same meanings as above. For their definition, we use the following formulas:

$$
\begin{aligned}
\gamma_{\delta,\psi} &= \bigvee_{\bar{\psi}\in\mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}}) \\
\gamma_{\delta,\varphi\psi} &= \bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in\mathrm{cf}(\psi)} (P_{A(\bar{\varphi}U\bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}}) \\
\gamma_{\delta,1} &= \bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)} (P_{A(\bar{\varphi}Uc_1)} \wedge \alpha_{\varphi,\bar{\varphi}}) \\
\gamma_{\delta,2} &= \bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)} (P_{A(\bar{\varphi}Uc_2)} \wedge \alpha_{\varphi,\bar{\varphi}}) \\
\gamma_{\delta,1\psi} &= \bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in\mathrm{cf}(\psi)} (P_{A(\bar{\varphi}U(c_1\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})
\end{aligned}
$$

$$\gamma_{\delta,2\psi} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi}U(c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$$

$$\gamma_{\delta,12} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} (P_{A(\bar{\varphi}U(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}})$$

$$\gamma_{\delta,12\psi} = \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi}U(c_1\vee c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$$

We set $\gamma_{\delta,\mathrm{incomplete}} := \gamma_{\delta,1} \vee \gamma_{\delta,2} \vee \gamma_{\delta,12} \vee \gamma_{\delta,1\psi} \vee \gamma_{\delta,2\psi} \vee \gamma_{\delta,12\psi}$ and $\gamma_{\delta,\mathrm{complete}} := \gamma_{\delta,\psi} \vee \gamma_{\delta,\varphi\psi}$. As explained in the introduction, it suffices if all index paths have components in which incomplete formulas hold until a complete formula holds. However, an index path may "end" earlier if a component formula does not force a continuation in this direction of the index path as shown in Figure 6.4. This is, e.g., the case if we have $A(\bar{\varphi}U(c_1 \vee \bar{\psi}))$ at a component and go to the right successor in the index tree. We express these additional abort conditions by $\gamma_{\delta,\mathrm{back1}} := S_1 \wedge X^{-1}(\gamma_{\delta,2} \vee \gamma_{\delta,2\psi})$ and $\gamma_{\delta,\mathrm{back2}} := S_2 \wedge X^{-1}(\gamma_{\delta,1} \vee \gamma_{\delta,1\psi})$. (We start checking these conditions only at the successor of the current component because we use the past quantifier.) In total, we get:

$$\beta_\delta = \gamma_{\delta,\mathrm{complete}} \vee [\gamma_{\delta,\mathrm{incomplete}} \wedge AX[A(\gamma_{\delta,\mathrm{incomplete}} U (\gamma_{\delta,\mathrm{complete}} \vee \gamma_{\delta,\mathrm{back1}} \vee \gamma_{\delta,\mathrm{back2}}))]]$$

Formal proof for case $A(\varphi U\psi)$. For this formal proof, we use the same notation as in the case $E(\varphi U\psi)$: We denote the current component by $\underline{t}_h$ and use T for a set of nodes in $\underline{t}_h$. Furthermore, we use $\underline{t}_s^{\sqsupset h}$ for the part of the sum tree $\underline{t}_s$ from index h onwards. We use Π_u for the set of paths starting in u and $\tilde{S}$ for $\{\tilde{S}_i \mid i \in [2]\}$. Recall again the definition of $I_{\bar{\delta}}^{\sqsupset h,T}$ as $\{z \sqsupset h \mid \underline{t}_z \models \bar{\delta}\} \cup \{h \mid \forall u \in T : \underline{t}_z[u] \models \bar{\delta}\}$.

We now show that for all nodes $u \in T$ of the current component (with index h), the sum tree $\underline{t}_s^{\sqsupset h}$ starting at the node u of the component $\underline{t}_h$ satisfies the formula $\delta = A(\varphi U\psi)$ iff the formula β_δ is satisfied in the index tree with the interpretation $I_{\bar{\delta}}^{\sqsupset h,T}$.

We have:

$$\begin{aligned}
&\forall u \in T : \underline{t}_s^{\sqsupset h}[u] \models A(\varphi U\psi)\\
\Leftrightarrow &\forall u \in T\ \forall \pi \in \Pi_u : (\underline{t}_s^{\sqsupset h}[u], \pi) \models \varphi U\psi\\
\Leftrightarrow &\forall u \in T\ \forall \pi \in \Pi_u\ \exists k \text{ on } \pi : \underline{t}_s^{\sqsupset h}[k] \models \psi \wedge \forall i \text{ on } \pi \text{ with } u \sqsubseteq i \sqsubset k : \underline{t}_s^{\sqsupset h}[i] \models \varphi
\end{aligned}$$

Using the induction hypothesis for the h-th component for the subformulas φ, ψ, this is equivalent to: For all $u \in T$ and for all paths $\pi \in \Pi_u$, we have:

1. For the node u, we directly have $(\underline{t}_I^{\sqsupseteq h}, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupseteq h,\{u\}} \mid \bar{\psi} \in \mathrm{cf}(\psi)\}) \models \beta_\psi$ or
2. (for all these paths) there exists a node k which lies in $\underline{t}_h$ s.t. $(\underline{t}_I^{\sqsupseteq h}, \tilde{S}, \{I_{\bar{\psi}}^{\sqsupseteq h,K} \mid \bar{\psi} \in \mathrm{cf}(\psi)\}) \models \beta_\psi$ where K is the set of these nodes k. Furthermore, we have $(\underline{t}_I^{\sqsupseteq h}, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupseteq h,J} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}) \models \beta_\varphi$ holds for $J := \{j \mid \exists k \in K : u \sqsubseteq j \sqsubset k\}$ or
3. (for all these paths π) there exists a k on π which lies in a tree $\underline{t}_z$ for a $z \sqsupseteq h1$. Then, for all π there is a last node k' in the tree $\underline{t}_h$ with $\underline{t}_h[k'] \models P_{c_1}$. Let K' denote the set of these k'. Then, for J defined as $\{i \mid \exists k' \in K' : u \sqsubseteq i \sqsubset k'\}$ we have $(\underline{t}_I, \tilde{S}, \{I_{\bar{\varphi}}^{\sqsupseteq h,J} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}) \models \beta_\varphi$. Furthermore, for the remainder of the path π (starting from the next component $\underline{t}_{h1}$) $\varphi U \psi$ holds. With π_0 denoting this remainder, we have $(\underline{t}_{\tilde{s}}^{\sqsupseteq h1}[\epsilon], \pi_0) \models \varphi U \psi$ or
4. the case (3) with $z \sqsupseteq h2$ and c_2 instead of c_1 or,
5. as every path (starting from any node $u \in T$) may satisfy one of the four conditions, we get also all combinations of these conditions[5] for all paths starting at all $u \in T$.

We now consider all these paths and assume that we have already shown that β_δ holds from $\underline{t}_{h1}$ onwards. Recall that $\beta_\varphi = \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$ and $\beta_\psi = \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$. We get that $(\underline{t}_I^{\sqsupseteq h}, \tilde{S}, \{I_{\bar{\delta}}^{\sqsupseteq h,T} \mid \bar{\delta} \in \mathrm{ecf}(\delta)\})$ satisfies one of the following conditions.

- $\bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi} U \bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{A(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta))$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{A(\bar{\varphi} U c_2)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\delta))$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{A(\bar{\varphi} U (c_1 \vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta))$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi} U (c_1 \vee \bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta))$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi} U (c_2 \vee \bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta_\delta))$
- $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{A(\bar{\varphi} U (c_1 \vee c_2 \vee \bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta))$

[5] Recall that combining the first condition with the third and/or fourth condition leads to the same as combining the second with the third and/or fourth condition. Thus, it suffices to consider the first condition alone and any combination of the other three.

We still have to show the construction of β_δ. For this, recall the definition of the γ-formulas.

- $\gamma_{\delta,\psi} := \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{\bar{\psi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,\varphi\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}U\bar{\psi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,1} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Uc_1)} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,2} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Uc_2)} \wedge \alpha_{\varphi,\bar{\varphi}}))$
- $\gamma_{\delta,12} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}U(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,1\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}U(c_1\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,2\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}U(c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,12\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}U(c_1\vee c_2\vee\bar{\psi}))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$

We use $\gamma_{\delta,\mathrm{incomplete}} := \gamma_{\delta,1} \vee \gamma_{\delta,2} \vee \gamma_{\delta,12} \vee \gamma_{\delta,1\psi} \vee \gamma_{\delta,2\psi} \vee \gamma_{\delta,12\psi}$ and $\gamma_{\delta,\mathrm{complete}} := \gamma_{\delta,\psi} \vee \gamma_{\delta,\varphi\psi}$. For $A(rUs)$ in the sum tree, we get an universal "Until" condition of the form "incomplete formulas on all paths until complete formulas" over the index tree. We first observe that $\gamma_{\delta,\mathrm{complete}} \vee \gamma_{\delta,\mathrm{incomplete}}$ has to hold at the first component (with index h). Furthermore, we must still have δ at the sum tree from the left successor onwards (but not necessary from the right one) if we have $\gamma_{\delta,1}$ or $\gamma_{\delta,1\psi}$ at the current component. Analogously, we must have δ at the right successor sum if we have $\gamma_{\delta,2}$ or $\gamma_{\delta,2\psi}$ and δ has to hold in both sums for $\gamma_{\delta,12}$ or $\gamma_{\delta,12\psi}$.

If we now consider the left, respectively the right, successor as the current tree, the conditions above are equivalent to:

- We need no condition, because we are at a right successor and $\gamma_{\delta,1}$ or $\gamma_{\delta,1\psi}$ holds at the predecessor
- or we need no condition, because we are we are at a left successor and $\gamma_{\delta,2}$ or $\gamma_{\delta,2\psi}$ holds at the predecessor
- or δ still has to hold.

We get the following universal "Until" condition: $\beta_\delta = \gamma_{\delta,\mathrm{complete}} \vee [\gamma_{\delta,\mathrm{incomplete}} \wedge AX\, A(\gamma_{\delta,\mathrm{incomplete}} U(\gamma_{\delta,\mathrm{complete}} \vee \gamma_{\delta,\mathrm{back1}} \vee \gamma_{\delta,\mathrm{back2}}))]$ with $\gamma_{\delta,\mathrm{back1}} = S_1 \wedge X^{-1}(\gamma_{\delta,2} \vee \gamma_{\delta,2\psi})$ and $\gamma_{\delta,\mathrm{back2}} = S_2 \wedge X^{-1}(\gamma_{\delta,1} \vee \gamma_{\delta,1\psi})$. Thus, we have $(\underline{\mathfrak{t}}_{\bar{I}}^{\sqsupseteq h}, \tilde{S}, \{I_{\bar{\delta}}^{\sqsupseteq h,T} \mid \bar{\delta} \in \mathrm{ecf}(\delta)\}) \models \beta_\delta$. □

$E(\psi R \varphi)$

The basic ideas for $E(\psi R \varphi)$ and $A(\psi R \varphi)$ in the sum are quite similar to the ideas from $E(rUs)$ and $A(rUs)$. For $\delta = E(\psi R \varphi)$ at one node u, a path must start in the current component at u which satisfies one of following cases in the sum tree (see also Figure 6.6):

(1) All nodes of the path lie in (the part of the sum which belongs to) the current component and satisfy φ,

(2) all nodes of the path[6] lie in the current component and the path satisfies $\psi R \varphi$,

(3) the path continues in the left successor component, all nodes of the path satisfy φ in the part of the sum belonging to the current component and $E(\psi U \varphi)$ is fulfilled at the root of the remainder sum or

(4) the third case but for the right successor.

Again, we can use for cases (1) and (2) the induction hypothesis for the subformulas φ, respectively ψ, for all nodes that lie on the path in the current component. For cases (3) and (4), we can use the induction hypothesis for φ for the nodes on the part of the path in the current component. Furthermore, at the remainder sum, we also have δ at one node – the root. If we can show that this is equivalent to β^1_δ, we get for β^1_δ the recursive definition:

(1) $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{EG\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$

(2) $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{E(\bar{\psi} R \bar{\varphi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$

(3) $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta))$

(4) $\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_2)} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta^1_\delta))$

We still have to find a definition of β^1_δ without using β^1_δ itself. Again, we use γ-formulas to define β^1_δ:

- $\gamma_{\delta,\varphi} := \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{EG\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,\varphi\psi} := \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} (P_{E(\bar{\psi} R \bar{\varphi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,1} := \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_1)} \wedge \alpha_{\varphi,\bar{\varphi}})$

[6] To be more precise if $\psi R \varphi$ (but not $G\varphi$) holds, only the fragment of the path which satisfies this formula has to lie in the current component.

- $\gamma_{\delta,2} := \bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} (P_{E(\bar{\varphi} U c_2)} \wedge \alpha_{\varphi,\bar{\varphi}})$

Let $\gamma_{\delta,\text{complete}}$ be $\gamma_{\delta,\varphi} \vee \gamma_{\delta,\varphi\psi}$ and $\gamma_{\delta,\text{incomplete}} := \gamma_{\delta,1} \vee \gamma_{\delta,2}$. Recall that $E(\psi R \varphi)$ means that there is a path which is either labeled with φ until $\varphi \wedge \psi$ holds or it is completely labeled with φ. For the first case, we know how to define β^1_δ as an Until-condition over the index tree from the already discussed case $E(\varphi U \psi)$. We have to add the possibility that the path fulfills φ all the time. This can have two reasons: either there is an index path which finally goes to (or through) a component where $EG\varphi$ holds – this is already captured by the Until-condition over the index tree – or there is an infinite index path which allows on every component that the path is continued via an incomplete formula. The second condition can be expressed using the W-operator. Thus, we get the following formula for β^1_δ:

$$\begin{aligned}\beta^1_\delta = &\gamma_{\delta,\text{complete}} \vee [\gamma_{\delta,\text{incomplete}} \wedge \\ &EX[E(\gamma_{\delta,\text{incomplete}} \wedge \gamma_{\delta,\text{back1}} \wedge \gamma_{\delta,\text{back2}}) W (\gamma_{\delta,\text{complete}} \wedge \gamma_{\delta,\text{back1}} \wedge \gamma_{\delta,\text{back2}})]]\end{aligned}$$

with $\gamma_{\delta,\text{back1}} := S_1 \to X^{-1}\gamma_{\delta,1}$ and $\gamma_{\delta,\text{back2}} := S_2 \to X^{-1}\gamma_{\delta,2}$.

At a set T of nodes, again, we may have all combinations of the cases (1)–(4) from above. For example, for the combination (1) and (3), for some $u \in T$ the path satisfies φ on all nodes and lies completely in the h-th component and for the other nodes of the set T the paths continue in the left successor. Thus, we have to express for the current component the CTL*-condition $A(G\bar{\varphi} \vee (\bar{\varphi} U c_1))$ which is equivalent to $A(\bar{\varphi} W c_1)$ and can be converted to a CTL formula as explained in Section 2.2. Again, the conditions for the next components have to be combined. For combinations like (2) and (3), we further use the equivalence of $E(\bar{\psi} R \bar{\varphi})$ and $E(\bar{\varphi} W (\bar{\varphi} \wedge \bar{\psi}))$.

Thus, in total, we get $\beta_\delta = \bigvee_{\bar{\delta} \in \mathrm{cf}(\delta)} (P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ with the following auxiliary formulas:

$$\begin{aligned}\alpha_{\delta,EG\bar{\varphi}} &= \alpha_{\varphi,\bar{\varphi}} \\ \alpha_{\delta,E(\bar{\psi} R \bar{\varphi})} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \\ \alpha_{\delta,E(\bar{\varphi} U c_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U c_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} W c_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} W c_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} U (c_1 \vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \wedge EX(S_2 \wedge \beta^1_\delta) \\ \alpha_{\delta,E(\bar{\varphi} W (c_1 \vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta^1_\delta) \wedge EX(S_2 \wedge \beta^1_\delta)\end{aligned}$$

$$\begin{aligned}
\alpha_{\delta,E(\bar{\varphi}W(c_1\vee(\bar{\varphi}\wedge\bar{\psi})))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta^1_\delta) \\
\alpha_{\delta,E(\bar{\varphi}W(c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta^1_\delta) \\
\alpha_{\delta,E(\bar{\varphi}W(c_1\vee c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta^1_\delta) \wedge EX(S_2 \wedge \beta^1_\delta)
\end{aligned}$$

$A(\psi R \varphi)$

For $\delta = A(\psi R \varphi)$, we must have one of the following cases in the sum tree starting at the current component *at node u*:

(1) All paths of the sum lie in the current component and satisfy φ on all nodes,

(2) all paths of the sum[7] lie in the current component and satisfy $\psi R \varphi$,

(3) all paths continue to the left successor component and all nodes on these paths in the part of the sum belonging to the current component satisfy φ and δ is fulfilled at the root of the remainder sum,

(4) the third case but for the right successor or

(5) any combination of the cases (1)–(4) for the set of all paths.

Again, we can use for cases (1) and (2) the induction hypothesis for the subformulas φ, respectively ψ, for all nodes that lie on the paths (fragments). For cases (3) and (4), we can use the induction hypothesis for φ for the nodes on the part of the paths in the current component. Furthermore, if we assume that β_δ holds at the next component iff δ holds in the remainder sum, we get that $\beta_\delta = \bigvee_{\bar{\delta}\in \mathrm{cf}(\delta)}(P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ holds with the following auxiliary formulas:

$$\begin{aligned}
\alpha_{\delta,AG\bar{\varphi}} &= \alpha_{\varphi,\bar{\varphi}} \\
\alpha_{\delta,A(\bar{\psi}R\bar{\varphi})} &= \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \\
\alpha_{\delta,A(\bar{\varphi}Uc_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}Uc_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}Wc_1)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}Wc_2)} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}U(c_1\vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta) \\
\alpha_{\delta,A(\bar{\varphi}W(c_1\vee c_2))} &= \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta)
\end{aligned}$$

[7] To be more precise, again, for $\psi R \varphi$ and not $G\varphi$, only the fragments of the paths which satisfy this formula have to lie in the current component.

$$\alpha_{\delta,A(\bar{\varphi}W(c_1\vee(\bar{\varphi}\wedge\bar{\psi})))} = \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta)$$
$$\alpha_{\delta,A(\bar{\varphi}W(c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} = \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_2 \wedge \beta_\delta)$$
$$\alpha_{\delta,A(\bar{\varphi}W(c_1\vee c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} = \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}} \wedge EX(S_1 \wedge \beta_\delta) \wedge EX(S_2 \wedge \beta_\delta)$$

Analogously to the case $A(\varphi U \psi)$, we define γ-formulas:

- $\gamma_{\delta,\varphi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{AG\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,\varphi\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\psi}R\bar{\varphi})} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,1} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Uc_1)} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,2} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Uc_2)} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,12} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}U(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,1\varphi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Wc_1)} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,2\varphi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}Wc_2)} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,12\varphi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)}(P_{A(\bar{\varphi}W(c_1\vee c_2))} \wedge \alpha_{\varphi,\bar{\varphi}})$
- $\gamma_{\delta,1\varphi\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}W(c_1\vee(\bar{\varphi}\wedge\bar{\psi})))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,2\varphi\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}W(c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$
- $\gamma_{\delta,12\varphi\psi} := \bigvee_{\bar{\varphi}\in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi}\in \mathrm{cf}(\psi)}(P_{A(\bar{\varphi}W(c_1\vee c_2\vee(\bar{\varphi}\wedge\bar{\psi})))} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$

We set $\gamma_{\delta,\mathrm{complete}} := \gamma_{\delta,\varphi} \vee \gamma_{\delta,\varphi\psi}$ and $\gamma_{\delta,\mathrm{incomplete}}$ as the disjunction over the other formulas. We use the abort conditions $\gamma_{\delta,\mathrm{back1}} := S_1 \wedge X^{-1}(\gamma_{\delta,2} \vee \gamma_{\delta,2\varphi} \vee \gamma_{\delta,2\varphi\psi})$ and $\gamma_{\delta,\mathrm{back2}} := S_2 \wedge X^{-1}(\gamma_{\delta,1} \vee \gamma_{\delta,1\varphi} \vee \gamma_{\delta,1\varphi\psi})$. In total, we get:

$$\beta_\delta = \gamma_{\delta,\mathrm{complete}} \vee [\gamma_{\delta,\mathrm{incomplete}} \wedge AX[A(\gamma_{\delta,\mathrm{incomplete}} W (\gamma_{\delta,\mathrm{complete}} \vee \gamma_{\mathrm{back1}} \vee \gamma_{\mathrm{back2}}))]]$$

$\varphi \vee \psi$

As in the LTL composition theorem, for the disjunction $\delta = \varphi \vee \psi$ at a set T of states of the current component, we have to consider three cases: φ on all of these states, ψ on all of these states and that there are states in T where φ holds and on the other ones ψ holds. Thus, we get:

$$\beta_\delta = \beta_\varphi \vee \beta_\psi \vee (\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} P_{\bar{\varphi} \vee \bar{\psi}} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$$

and $\alpha_{\delta,\bar{\varphi}} = \alpha_{\varphi,\bar{\varphi}}$, $\alpha_{\delta,\bar{\psi}} = \alpha_{\psi,\bar{\psi}}$ and $\alpha_{\delta,\bar{\varphi}\vee\bar{\psi}} = \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}}$.

$\varphi \wedge \psi$

For the conjunction $\delta = \varphi \wedge \psi$, we have to satisfy both the conditions for φ and ψ. We get:

$$\beta_\delta = (\bigvee_{\bar{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\bar{\psi} \in \mathrm{cf}(\psi)} P_{\bar{\varphi} \wedge \bar{\psi}} \wedge \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}})$$

and $\alpha_{\delta,\bar{\varphi}\wedge\bar{\psi}} = \alpha_{\varphi,\bar{\varphi}} \wedge \alpha_{\psi,\bar{\psi}}$.

$EX\varphi$

We first discuss $EX\varphi$ at one node u of the sum. Here, we must have a successor v at which φ holds. The node v (of the sum) may either belong to the current component (case (1)) or the left or the right successor component (cases (2) and (3)). For the case (1), we can use the induction hypothesis for φ at the current component and simply get $\bigvee_{\bar{\varphi}\in\mathrm{cf}(\varphi)}(P_{EX\bar{\varphi}} \wedge \alpha_{\varphi,\bar{\varphi}})$. For the case (2), we can use the induction hypothesis for φ at the (root of the) left successor. Furthermore, we must have a connection point c_1 in the current component at a successor of the node u, i.e., EXc_1 has to hold. We get $P_{EXc_1} \wedge EX(S_1 \wedge \beta_\varphi)$. The case (3) is analogous. For $EX\varphi$ at a set T of nodes, we may also have any combination of the cases (1), (2) and (3). Thus, we get $\beta_\delta = \bigvee_{\bar{\delta}\in\mathrm{cf}(\delta)}(P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ with the auxiliary formulas:

- $\alpha_{\delta,EX\bar{\varphi}} = \alpha_{\varphi,\bar{\varphi}}$
- $\alpha_{\delta,EXc_1} = EX(S_1 \wedge \beta_\varphi)$
- $\alpha_{\delta,EXc_2} = EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,EX(c_1\vee c_2)} = EX(S_1 \wedge \beta_\varphi) \wedge EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,EX(c_1\vee\bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\varphi)$
- $\alpha_{\delta,EX(c_2\vee\bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,EX(c_1\vee c_2\vee\bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\varphi) \wedge EX(S_2 \wedge \beta_\varphi)$

$AX\varphi$

For $AX\varphi$ at one node u in the sum tree, we have one of the following situations:

(1) All successors of u lie in the part of the sum which belongs to the current component and fulfill φ,

(2) all successors of u lie in the part of the sum which belongs to the left successor and fulfill φ,

(3) the second case for the right successor or

(4) any combination of these cases for the set of all successors.

We get the same cases for $EX\varphi$ at a set T of nodes. Using the induction hypothesis for φ in the current component, respectively the successor components, we get that $\beta_\delta = \bigvee_{\bar{\delta}\in \mathrm{cf}(\delta)}(P_{\bar{\delta}} \wedge \alpha_{\delta,\bar{\delta}})$ is fulfilled with the following auxiliary formulas:

- $\alpha_{\delta,AX\bar{\varphi}} = \alpha_{\varphi,\bar{\varphi}}$
- $\alpha_{\delta,AXc_1} = EX(S_1 \wedge \beta_\varphi)$
- $\alpha_{\delta,AXc_2} = EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,AX(c_1\vee c_2)} = EX(S_1 \wedge \beta_\varphi) \wedge EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,AX(c_1\vee \bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\varphi)$
- $\alpha_{\delta,AX(c_2\vee \bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_2 \wedge \beta_\varphi)$
- $\alpha_{\delta,AX(c_1\vee c_2\vee \bar{\varphi})} = \alpha_{\varphi,\bar{\varphi}} \wedge EX(S_1 \wedge \beta_\varphi) \wedge EX(S_2 \wedge \beta_\varphi)$

6.4 Size of the Decomposition

We now show that the size of the decomposition is indeed exponential in the size of the input formula. For this, we have to estimate the size of the generated CTL component formulas, their number and the size of the CTL+X^{-1} formula. We show the complexity for the general case where the branching of the index tree is at most n, i.e., each tree has at most the special symbols $c_1, \ldots, c_n$. Note that this leads in the induction step to component formulas which distinguish between each possible subset of $c_1, \ldots, c_n$: e.g., for the formula $A(\varphi U \psi)$ the paths in the sum which have to satisfy $\varphi U \psi$ may go to any combination of the next n components.

Size of the component formulas:
From the definition of the set $\mathrm{cf}(\delta)$, we get in all cases $s(\bar{\delta}) \leq c * s(\bar{\varphi}) + c * s(\bar{\psi}) +$

$c + n$ for some constant $c \in \mathbb{N}$ (and $s(\psi) = 0$ if it does not occur in δ). Thus, for the input formula γ, we have $s(\bar{\gamma}) \in O(n * s(\gamma))$ for $\bar{\gamma} \in \mathrm{cf}(\gamma)$.

Number of the component formulas:
From the construction of the set $\mathrm{cf}(\delta)$, we get in the induction step $|\mathrm{cf}(\delta)| \leq c * 2^n * |\mathrm{cf}(\varphi)| * |\mathrm{cf}(\psi)|$ in all cases of subformulas (and $|\mathrm{cf}(\psi)| = 0$ if it does not occur in δ). Thus, we have $|\mathrm{ecf}(\gamma)| \in O(2^{n*s(\gamma)})$.

Size of the CTL+X^{-1} formula:
We first observe that in all cases of (direct) subformulas φ, ψ of δ, we have for the auxiliary formulas $\alpha_{\delta,\bar{\delta}}$ that $|\alpha_{\delta,\bar{\delta}}| \leq |\alpha_{\varphi,\bar{\varphi}}| + |\alpha_{\psi,\bar{\psi}}| + |\beta_\delta| \leq 2 * |\beta_\delta|$.

We estimate the size of the CTL+X^{-1} formula only for the case $\delta = E(\varphi U \psi)$. The other cases are quite similar. Note that in general (for at most n successors), β_δ is defined as $\gamma_{\text{complete}} \vee [\gamma_{\text{incomplete}} \wedge EX[E(\gamma_{\text{incomplete}} \wedge \gamma_{\text{back}}) U \gamma_{\text{complete}} \wedge \gamma_{\text{back}}]]$ with $\gamma_{\text{back}} = \gamma_{\text{back1}} \wedge \cdots \wedge \gamma_{\text{back}n}$ and $\gamma_{\text{back}i}$ is defined analogously to γ_{back1} and γ_{back2}.

Let $\gamma_{\text{any}} := \gamma_{\text{complete}} \vee \gamma_{\text{incomplete}}$. We get $|\beta_\delta| \leq c * (|\gamma_{\text{any}}| + n * |\gamma_{\text{back}i}|)$ with some constant $c \in \mathbb{N}$. We first observe that γ_{any} is a disjunction over $2 * 2^n$ formulas, as we have the formula γ_ψ and at most the formulas $\gamma_{i_1 \ldots i_k}$ and $\gamma_{i_1 \ldots i_k \psi}$ for every[8] subset $\{i_1, \ldots, i_k\} \subseteq \mathcal{P}([n])$. Thus, we can estimate $|\gamma_{\text{any}}|$ by 2^{n+1} times the largest of these formulas which has the size $|\mathrm{cf}(\varphi)| * |\mathrm{cf}(\psi)| * (n + |\alpha_{\varphi,\bar{\varphi}}| + |\alpha_{\psi,\bar{\psi}}|)$. As $\gamma_{\text{back}i}$ is defined by $S_i \to X^{-1}(\lambda)$ where λ is a disjunction over all formulas with i fixed, we have $\gamma_{\text{back}i} \leq c * |\gamma_{\text{any}}|$.

We get $|\beta_\delta| \in O(2^{n+2} * |\mathrm{cf}(\varphi)| * |\mathrm{cf}(\psi)| * (|\beta_\varphi| + |\beta_\psi|))$ for the induction step. In total, we have $|\beta_\gamma| \in O((|\mathrm{cf}(\gamma)| * 2^{n+2})^{s(\gamma)})$ and by using $|\mathrm{cf}(\gamma)| \in O(2^{n*s(\gamma)})$ that $|\beta_\gamma| \in O(2^{n*s(\gamma)^2})$.

6.5 Summary and Further Results

In this section, we have shown a generalization of the LTL composition theorem to CTL over composed trees. Although the proof is much more involved, the results are quite similar. We also get a decomposition which is exponential in the size of the given formula. The main difference is that we need the more powerful logic CTL+X^{-1} in the index tree and that the decomposition is also exponential in the branching of the index tree, i.e., in the maximal number of successor indices of the index tree.

The proof of the CTL composition theorem was shown for the setting where we concatenated the same special symbols c_i of the marked trees with the same

[8] To be more precise, for the empty set, we only have $\gamma_{\varphi\psi}$.

successor. Note that we get the same result if we allow different trees at the c_i positions which are CTL equivalent as we only check CTL formulas on these trees.

We take a deeper look at the composition technique for generalizations of the logic CTL. We consider the logics ACTL* and ECTL* which are the fragments of CTL* that only allow universal, respectively existential, quantification over paths. We discuss the natural approach to generalize the results from Chapters 5 and 6 to ACTL* and ECTL* by using the composition technique on a single path and writing "A", respectively "E" in front of the component formulas.

Note that a fixed path in the sum tree actually describes a word. Thus, for each single path, we have a composition theorem which is analogous to the LTL case. (Formally, we have to take into account that the definition of the sum tree uses the special symbols c_1 and c_2 to concatenate the component trees. Thus, e.g., for a formula $\varphi U \psi$ instead of the component formula $G\bar{\varphi}$ we get $\bar{\varphi} U c_1, \bar{\varphi} U c_2$. The given path decides whether the formulas with c_1 or c_2 have to be taken.)

We first look at the extension ACTL*. If we look at universal quantification over a (composed) path formula like $(rUs)Ut$ – i.e., in this example, at the formula $A((rUs)Ut)$ – the component formulas have to be state formulas. The natural approach is to simply use the "A"-quantified version of the given path formulas for each single path. We present arguments why this natural generalization fails (at least) for infinite sums of trees. (There may be a more intelligent way to generalize the composition technique of CTL to ACTL*.) Recall the tree from Figure 2.5. This tree can be represented as the sum tree for the index tree which is simply a copy of the natural numbers where each node is labeled with the following tree: It consists of the leftmost path of the tree from Figure 2.5 and a c_1 successor at the right successor of the root. Recall that the formula $AFGp$ holds in the tree from Figure 2.5, but the formula $AFAGp$ not. The reason is that the rightmost path satisfies FGp but not $FAGp$ because every node on this path has a successor with p and one with $\neg p$. This path is now distributed over all (infinitely many) components. Intuitively, the composition approach from above fails because every component formula is quantified over all paths and this does not allow to express that we have (finally) only p on the rightmost branch. Note that this argument depends on the infinite index structure. However, for finite index structure, we may be able to use a formula from [Boj08] to also show the failure for this approach. (This formula expresses that all paths are labeled by $(ab)^*a(ab)^*c^\omega$. It is used to show that there are formulas which are definable in LTL and CTL but which need the existential path quantifier in CTL.)

We now look at the extension ECTL*. Recall the composition theorem for a fixed path. Note that this gives us inductively a set of component formulas which have

to hold at each component. Consider, e.g., the formula $(rUs) \wedge (tUv)$ on the fixed path. Then, we get for the subformula rUs the condition "components with rUc_1, respectively rUc_2, until a component with rUs" and the analogous condition for tUv. (As above, the fixed path decides whether, in the current component, we take (rUc_1 and tUc_1) or (rUc_2 and tUc_2).) If we now want to express conditions for the state formula $E((rUs) \wedge (tUv))$, we cannot simply write an "E" in front of every component formula, because we have to ensure that the conditions rUc_1 and tUc_1, respectively the other two, are checked *on the same path fragment*. Thus, we have to use more complex component formulas which also consider combinations of the known component formulas – here, we would need at least the additional formula $E(rUc_1 \wedge tUc_1)$. This poses two problems: First, it is not clear how the composition technique works over these combinations of formulas and second, if we can find a composition theorem for this case, the size of the decomposition may become larger.

It may be a challenging task to overcome these problems for the logics ACTL* and ECTL*. However, we may be able to find constructions for other interesting fragments of CTL* as, e.g., CTL+X^{-1}, or a direct construction for formulas of the form $AFG\varphi$ and $AGF\varphi$.

7 Conclusion and Outlook

In this thesis, we discussed the composition technique in the field of model-checking. It allows to deduce the truth of a logic formula in a product or sum of components from information about the truth values of formulas in these components. We analysed two main questions:

1. For which combinations of logics and types of structures, respectively their product or sum, is the composition technique applicable?
2. How much information about the components do we need?

For the first question, the classical results by Feferman and Vaught, respectively Shelah, give us, on the one hand, a composition theorem for products and FO logic and, on the other hand, for (disjoint ordered) sums of ordering and MSO logic. In the field of model-checking, FO logic is too weak to express many interesting properties like, e.g., reachability. Thus, our focus for the first question lied on products. We extended the known results by Rabinovich, respectively Wöhrle and Thomas, to show a composition theorem for finitely synchronized products of transition systems and FO logic with reachability by modulo counting paths (FO(Reg1R)). We also extended their results by capturing not only a finite number but also an infinite number of components. Using a proof schema of Rabinovich, we further showed that various other extensions of the logic (especially regular reachability) lead to a failure of the composition technique – in most cases already for the special cases of asynchronous or direct products of simple transition systems (basically a copy of the natural numbers with some predicates). An overview of these results was shown in Table 4.1.

For the second question, the bad news was that, in general, the non-elementary size of the decomposition is unavoidable [GJL12]. However, this left (and still leaves) possible improvements for special logics and special structures, in particular, for structures with a bounded out-degree of outgoing transitions as the proof in [GJL12] relies on the unbounded degree. We discussed two such cases: LTL over a disjoint ordered sum of words and CTL over a disjoint ordered sum of (marked) trees. In both cases, we found a composition theorem with only[1] exponential

[1] "Only" in comparison to the non-elementary complexity in general.

size of the decomposition. We exploited the structure of LTL, respectively CTL: each subformula "looks only in the future", i.e., it is interpreted in the current component or/and later components. This allowed us to get a construction based on a simultaneous induction over both the subformulas of the given formula and the parts of the sum starting at a component.

Outlook and Further Research

Concerning the composition technique for the products, we have seen a comprehensive overview of the frontiers of the applicability of the composition technique. Furthermore, in the cases where the composition technique is applicable, the non-elementary size of the decomposition limits its applicability. A possible way to overcome both situations would be to adapt the idea we used for sums: to look at special logics and special structures instead of arbitrary transition systems. However, here, the situation becomes more complicated as it is not clear how to deduce the truth of a formula from truth values of formulas in smaller "subproducts" as it was done in the case of disjoint ordered sums. Moreover, for the cases where the composition theorem fails, it fails already in the case of very simple transition systems. All together, it might be a quite challenging task to find further results on products.

For the composition technique for disjoint ordered sums, on the one hand, we could try to further extend the results to interesting fragments of CTL*. In particular, it would be interesting to capture the past quantifier (X^{-1}) and see if CTL+X^{-1} would be still sufficient in the index tree. (Here, we might need to restrict the marked trees to have only one node per c_i symbol.) Other past quantifiers might be a bigger challenge due to the "looking forward" structure of the proof. On the other hand, we might also look at other index structures, e.g., ring structures and directed acyclic graphs instead of trees.

Index

Symbols & Notation

$\lhd$	subformula relation, page 20
$\sqsubseteq$	prefix relation (in a tree model), page 13
$\mathcal{A}$	structure, page 11
$\langle \alpha_1, \ldots, \alpha_k; \beta \rangle$	interface information tuple, page 31
$\beta_{\text{Partition}}(I_1, \ldots, I_n; I)$	MSO formula: $I_1, \ldots, I_k$ form partition of I, page 21
$\text{Card}_{j,k}(X)$	CMSO formula expressing X has $j \pmod k$ elements, page 22
$\text{cf}(\delta)$	(set of) component formulas for δ, page 67
$\text{CL}(\delta)$	closure of the formula δ, page 20
$\text{dom}(\mathcal{A})$	domain of the structure $\mathcal{A}$, page 11
Ind	index structure, page 12
$\mathcal{K}$	transition system, page 16
$L(\pi)$	labeling sequence of the predicates of a path π, page 17
$l(\pi)$	labeling sequence of the transitions of π, page 17
$\text{md}(\varphi)$	modal depth of CTL or LTL formula φ, page 24
$[n]$	the set $\{1, \ldots, n\}$, page 11
$\mathbb{N}$	natural numbers (without 0), page 11
$\mathbb{N}_0$	natural numbers with 0, page 11
$\text{Path}_{l,k}(x, y)$	relation expressing there exists path from x to y divisible by k with remainder l, page 21
π	path (fragment), page 17
$\pi[i, j]$	path fragment from i to j, page 17

Bibliography

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008. (pages 1, 2, 17, 24, 25, 26, 27, 29, 88)

[Boj08] Mikołaj Bojańczyk. The common fragment of ACTL and LTL. In Roberto M. Amadio, editor, *Proceedings of the Theory and Practice of Software, 11th International Conference on Foundations of Software Science and Computational Structures, FOSSACS'08/ETAPS'08*, volume 4962 of *Lecture Notes in Computer Science*, pages 172–185, Berlin, Heidelberg, 2008. Springer-Verlag. (page 109)

[Büc62] J. R. Büchi. On a decision method in restricted second-order arithmetic. In Ernest Nagel, Patrick Suppes, and Alfred Tarski, editors, *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science (LMPS'60)*, pages 1–11. Stanford University Press, 1962. (page 78)

[BvBW07] P. Blackburn, J. van Benthem, and F. Wolter. *Handbook of Modal Logic*, volume 3. Elsevier, 2007. (page 22)

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata: Techniques and Applications*. 2007. electronic book, release October, 12th 2007. (pages 8, 14)

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal Logic. In Dexter Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981. (page 1)

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8:244–263, 1986. (page 26)

[CGP99] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999. (page 1)

[CK90] Chen Chung Chang and H. J. Keisler. *Model Theory*. North Holland, Amsterdam, 1990.

[CS01] Edmund M. Clarke and Bernd-Holger Schlingloff. Model checking. In Alan Robinson and Andrei Voronkov, editors, *Handbook of automated reasoning*, pages 1635–1790. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, 2001. (page 1)

[DGKS07] Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt. Model Theory Makes Formulas Large. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *ICALP'07: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 913–924. Springer, 2007. (pages 5, 43, 44)

[EF05] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer Monographs in Mathematics. Springer, November 2005. (pages 20, 35)

[EFT96] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Einführung in die mathematische Logik*. Spektrum Akademischer Verlag, Heidelberg, Darmstadt, fourth edition, 1996. (page 20)

[Fel08] Ingo Felscher. The Compositional Method and Regular Reachability. In Vesa Halava and Igor Potapov, editors, *Proceedings of the 2nd Workshop on Reachability Problems, Liverpool, UK, September 15–17,2008*, volume 223 of *Electronic Notes in Theoretical Computer Science*, pages 103–117. Elsevier Science Publishers, 2008. (pages 5, 8)

[Fel12] Ingo Felscher. LTL-Model-Checking via Model Composition. In Alain Finkel, Jerome Leroux, and Igor Potapov, editors, *Reachability Problems, 6th International Workshop, RP 2012, Bordeaux, France, September 17-19, 2012. Proceedings*, volume 7550 of *Lecture Notes of Computer Science*, pages 42–53. Springer, 2012. (page 65)

[FT09] Ingo Felscher and Wolfgang Thomas. Compositionality and Reachability with Conditions on Path Lengths. *International Journal of Foundations of Computer Science*, 20(5):851–868, May 2009. (pages 5, 8, 31, 64)

[FT11] Ingo Felscher and Wolfgang Thomas. On Compositional Failure Detection in Structured Transition Systems. Technical Report AIB-2011-12, RWTH Aachen University, August 2011.

[FV59] S. Feferman and R. Vaught. The first-order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959. (pages 2, 3, 7, 64)

[GJL12] Stefan Göller, Jean Christoph Jung, and Markus Lohrey. The Complexity of Decomposing Modal and First-Order Theories. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, LICS 2012, pages 325–334, Washington, DC, USA, 2012. IEEE Computer Society. (pages 5, 43, 44, 64, 111)

[GR08] Tobias Ganzow and Sasha Rubin. Order-Invariant MSO is Stronger than Counting MSO in the Finite. In Susanne Albers and Pascal Weil, editors, *25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*, volume 1 of *Leibniz International Proceedings in Computer Science*, pages 313–324. Schloss Dagstuhl, Leibniz-Zentrum für Informatik, Germany, 2008. (page 22)

[GS79] Yuri Gurevich and Saharon Shelah. Modest Theory of Short Chains II. *The Journal of Symbolic Logic*, 44(4):491–502, 1979. (page 3)

[GS85] Yuri Gurevich and Saharon Shelah. The Decision Problem for Branching Time Logic. *The Journal of Symbolic Logic*, 50(3):668–681, September 1985. (page 3)

[GS98] Dov M. Gabbay and Valentin B. Shehtman. Products of modal logics, part 1. *Logic Journal of IGPL*, 6(1):73–146, 1998. (page 3)

[Gur85] Yuri Gurevich. Monadic Second-Order Theories. In J. Barwise and S. Feferman, editors, *Model-Theoretic Logics*, Perspectives in Mathematical Logic, chapter XIII, pages 479–506. Springer-Verlag, 1985. (page 3)

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory*. Series in Computer Science. Addison–Wesley, Boston, MA, USA, 1978. (page 22)

[HKT02] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2002. (pages 22, 23)

[Hod97] Wilfrid Hodges. *Model theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1997. (pages 3, 7, 11, 19, 35, 37, 41)

[Kam68] Johan Anthony Willem Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, Computer Science Department, University of California at Los Angeles, USA, 1968. (page 65)

[Lib04] Leonid Libkin. *Elements of Finite Model Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 2004.

[LS95] François Laroussinie and Philippe Schnoebelen. A Hierarchy of Temporal Logics with Past. *Theoretical Computer Science*, 148(2):303–324, September 1995. (page 31)

[Mak04] Johann A. Makowsky. Algorithmic uses of the Feferman-Vaught Theorem. *Annals of Pure and Applied Logic*, 126(1-3):159–213, 2004. (pages 3, 43)

[Mer01] Stephan Merz. Model Checking: A Tutorial Overview. In Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark Dermot Ryan, editors, *Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes (MOVEP '00)*, volume 2067 of *Lecture Notes in Computer Science*, pages 3–38, London, UK, 2001. Springer-Verlag. (page 1)

[Mos52] Andrzej Mostowski. On Direct Products of Theories. *The Journal of Symbolic Logic*, 17(1):1–31, 1952. (page 3)

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 46–67. IEEE Computer Society Press, 1977. (page 23)

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *5th International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351, London, UK, 1982. Springer. (page 1)

[Rab04] Alexander Rabinovich. Selection and Uniformization in Generalized Product. *Logic Journal of IGPL*, 12(2):125–134, 2004.

[Rab07] Alexander Rabinovich. On compositionality and its limitations. *ACM Transactions on Computational Logic*, 8(1), January 2007. (pages 4, 5, 7, 8, 31, 43, 46, 58, 64)

[She75] Saharon Shelah. The Monadic Theory of Order. *The Annals of Mathematics*, 102(3):379–419, 1975. (pages 2, 3, 7, 65)

[Tho97a] Wolfgang Thomas. Ehrenfeucht Games, the Composition Method, and the Monadic Theory of Ordinal Words. In Jan Mycielski, Grzegorz Rozenberg, and Arto Salomaa, editors, *Structures in Logic and Computer Science, A Selection of Essays in Honor of A. Ehrenfeucht*, volume 1261 of *Lecture Notes in Computer Science*, pages 118–143. Springer–Verlag, 1997. (pages 3, 7, 21, 38, 41, 43, 65)

[Tho97b] Wolfgang Thomas. Languages, automata and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages, vol. 3 Beyond words*, pages 389–455. Springer–Verlag, New York, NY, USA, 1997. (page 78)

[WT07] Stefan Wöhrle and Wolfgang Thomas. Model Checking Synchronized Products of Infinite Transition Systems. *Logical Methods in Computer Science*, 3(4), 2007. (pages 4, 5, 7, 45, 46, 64)