LTL-Model-Checking via Model Composition

Ingo Felscher

RWTH Aachen University, 52074 Aachen, Germany,
felscher@automata.rwth-aachen.de,
http://automata.rwth-aachen.de/~felscher/

Abstract. We develop a composition technique for linear time logic (LTL) over ordered disjoint sums of words. This technique allows to reduce the validity of an LTL formula in the sum to LTL formulas over the components. It is known that for first order logic (FO) and even its three variable fragment FO^3 the number of formulas generated for the components is at least non-elementary in the size of the input formula. We show that for LTL – expressively equivalent to FO logic – over an ordered disjoint sum of words the number of formulas for the components can be kept at an at most exponential growth in the size of the input formula.

Keywords: model-checking, linear time logic, model composition

1 Introduction

From a practical point of view, the core issues in model-checking are problems of reachability (or of non-reachability, i. e., safety). Also in a more general context, the reachability problem is the core of model-checking: The standard solution of the model-checking problem for the linear-time temporal logic (LTL) is the transformation to a non-emptiness problem for Büchi automata (see e. g. the monograph [1]). This non-emptiness problem is the question whether a final state q in a given Büchi automaton can be reached and from q the state q is again reachable.

The basis of the solution of the model-checking problem is a composition result on infinite words. Acceptance of an infinite word by a Büchi automaton means that it consists of a sequence $w_0w_1\dots$ of finite segments such that w_0 leads the automaton to a final state q whereas all other w_i lead the automaton from q back to q. Composition of $w_0, w_1, \dots$ to an infinite word is easy in the following sense: If we know which state transformations each w_i defines in the given automaton $\mathcal{A}$, then we know whether the automaton accepts the infinite word $w_0w_1\dots$. Here the state transformations of a finite word w is given by the set of all state pairs (p, q) such that $\mathcal{A}$ can proceed from p to q via w, and whether this is possible while passing a final state. If $\mathcal{A}$ has n states then clearly the number of possible state transformations is of order 2^{n^2}.

The present paper studies this problem of composition in the framework of logic, without resorting to automata. We consider LTL, which is known to be expressively equivalent to first-order logic over ordered word models. For first-order logic (and also for MSO-logic) there is a well-known composition theory: One considers the formulas up to some given quantifier rank n, defines the n-type of a word w as the set of sentences up to quantifier rank n that are true in w, and obtains the following composition theorem: The n-types of w_1 and w_2 effectively determine the n-type of $w_1 w_2$, and the n-type of w determines effectively the n-type of the ω-word $www\dots$. An essential drawback in relation to automata theory is the fact that the number of n-types has a non-elementary growth in n.

In our main theorem below we show that this drawback can be avoided for the (expressively equivalent) logic LTL. We develop a version of LTL-n-types for word models that allow composition as mentioned above where the number of types only grows exponentially in n. Basically, this closes the complexity gap between the automata theoretic method and the logic-based approach for composition.

We give a short overview of the history of composition over products and unions: In 1959, Feferman and Vaught [5] developed a technique, which allows to deduce the truth of a formula in a product of structures from formulas in the components and information about the index structure of the product. Shelah extended these results and succeeding work of Fraïssé [8] and Ehrenfeucht [4] to a composition method for MSO logic over orderings. He showed that the monadic second order theory of a sum of orderings – the components (in the simplest case finite words) – can be derived from MSO information of these components and their index structure. Related results are [2,10,11,12,14] and in the field of model-checking [6,7,13,15]. For the cases where the composition theorem is applicable it allows to reduce the model checking problem in a product or sum to model checking problems over the components. This would be an improvement for systems which consist of "simple" components as the difficulty results from building the actual product or sum and performing model checking over this large structure. However, as mentioned above, one common drawback of this method is the complexity: the number of formulas generated by the method is non-elementary in the size of the given formula. In [3] it has been shown that this is also a lower bound in general for FO logic and a disjoint sum of structures with unbounded branching degree. As we consider words with the less-than relation we also have unbounded branching degree. The result in [3] has recently been strengthened: Also for first order logic with three variables (FO^3 logic) the size of the decomposition is non-elementary [9]. There it was also shown that the size becomes double exponential for FO^2 logic. Note that in the present paper we deal with *ordered* disjoint sums and that over words, LTL is as expressive as FO($<$) logic.

The paper is structured as follows: in the next section we repeat some standard definitions for logics and ordered disjoint sums of structures to fix notation. Further, for the composition theorem we define interface information tuples which consist of two parts: a set of LTL formulas and an MSO formula. The LTL formulas are interpreted in the components of a disjoint sum and the

MSO formula states conditions on the index structure, namely, in which of the components the LTL formulas have to hold.

In Section 3 we show the composition theorem for LTL logic. For this, we first introduce the possible formula types which may hold in the components. The set of these formulas is an extension of the set of subformulas of the given formula. The idea of the proof of the main theorem is that we inductively construct interface information tuples for the subformulas of the given formula. In these interface information tuples we use MSO formulas which state a condition for the first component, (local) conditions between neighbor components and a global condition which amounts to a Büchi condition in logic. These MSO formulas are simple in the sense that they only contain second order quantifiers of the form $\exists X \subseteq Y$. As a side-result, we provide a new, logic-based transformation of LTL-formulas into Büchi automata. (Here we refer to the composition of a model from single elements ("letters").) Note that the results presented here are work in progress. In particular, we work on replacing the simple MSO formulas by FO formulas.

We explain a simplified version of the main idea for $\varphi U \psi$, if φ and ψ are atomic propositions p and q: For pUq we start with either pUq or Gp in the first component. The local condition is that either pUq holds at the current component, or Gp holds at the current component and pUq is satisfied later, i. e. pUq or Gp holds at the next component. As global condition we have that Gp should not hold in all components. For the general formula $\varphi U \psi$ we would also have to guarantee that the local and global conditions for φ and ψ are satisfied and that $\varphi U \psi$ may occur repeated e. g. inside a formula $(\varphi U \psi)Uq$.

In Section 4 we conclude by giving a summary and an outlook.

2 Technical Preliminaries

In this section we first introduce structures and sums of them, logics, and then present the "interface information tuples" to speak about the formulas which hold in the components of a sum. We use the abbreviation $[n]$ for the set $\{1, \dots, n\}$.

2.1 Structures

In this paper we discuss (non-empty) linear orderings extended by predicates $P_1, \dots, P_y$. To simplify notation we identify each element of the structure by the predicates which hold at this element, so we restrict ourselves to words over the Boolean alphabet $\mathbb{B}^y$ for some $y \in \mathbb{N}$. A finite word $w = b_1 \dots b_m$ can be formally represented by a *word model* $\underline{w} = (\mathrm{dom}(w), \mathrm{Suc}, <, \min, \max, P_1, \dots, P_y)$ with domain $\mathrm{dom}(w) = [m]$, successor relation Suc, less-than relation $<$, minimal and maximal positions $\min, \max$ and unary Boolean predicates $P_1, \dots, P_y$. Infinite words α can be represented by word models with $\mathrm{dom}(\alpha) = \mathbb{N}$. For further simplification, whenever we speak of a word or write w/α we usually mean the word model. We write $\underline{w} \models \varphi$, if the word model $\underline{w}$ satisfies the formula φ.

We consider (ordered) disjoint sums over words by concatenating the words in a fixed sequence. Formally, for word models $\underline{w_1}, \dots, \underline{w_n}$ with $\underline{w_i} = (\mathrm{dom}(w_i), <_i, P_1, \dots, P_y)$ where $\mathrm{dom}(w_i)$ is finite for $i < n$, we define the *ordered disjoint sum over* $\underline{w_i}, i \in [n]$ as $\underline{w} = (\mathrm{dom}(w), <, \min, \max, P_1, \dots, P_y)$ where $\mathrm{dom}(w) = \{1, \dots, d_1, d_1 + 1, \dots, \sum_{i \in [n]} d_i\}$ for $d_i = |\mathrm{dom}(w_i)|$, $<, \min, \max$, and P_i are the defined in the natural way. The words $w_1, \dots, w_n$ are called the *components of* w. The *(infinite) ordered disjoint sum over* $w_i, i \in \mathbb{N}$ where each w_i is finite is defined analogously.

We consider not only $[n]$ or $\mathbb{N}$ for the set of indices but any linearly ordered set. In general, an index structure is a structure $Ind = (I, \sigma_I)$ with a countable set I which is used for the indices and a signature σ_I. An index ordering is an index structure $(I, <)$ with a linear order $<$.

2.2 Logics

We introduce the syntax of *linear time logic* (LTL), *first-order logic* (FO) and *monadic second order logic* (MSO) to fix notation. Let φ, ψ be LTL formulas and P a predicate, then $p, \neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, F\varphi, G\varphi, \varphi U \psi, \varphi R \psi$ are also LTL formulas. For a (finite or infinite) word model $\underline{w}$, let $\underline{w}[i]$ denote the word model starting from the i-th position. We have $\underline{w}[i] \models p$ iff the predicate P holds at the i-th position of $\underline{w}$. Negation, conjunction and disjunction are treated as usual. For the Next-operator we have $\underline{w}[i] \models X\varphi$ iff $\underline{w}[i+1] \models \varphi$. For a finite word $w = a_0 \dots a_n$ we define $\underline{w}[n+1] \models \varphi \Leftrightarrow \varphi = \mathsf{ff}$. The semantics of the Until-/Release-operators are: $\underline{w}[i] \models \varphi U \psi :\Leftrightarrow \exists j \geq i : \underline{w}[j] \models \psi \wedge \forall k, i \leq k < j : \underline{w}[k] \models \varphi$ and $\underline{w}[i] \models \psi R \varphi :\Leftrightarrow \forall k \geq i : \underline{w}[k] \models \varphi \vee \exists j \geq i : \underline{w}[j] \models \psi \wedge \forall k, i \leq k \leq j : \underline{w}[k] \models \varphi$. Finally and Globally are used as abbreviations: $F\varphi = \mathsf{tt} U \varphi$ and $G\varphi = \mathsf{ff} R \varphi$. To be able to easily check for the last letter of a finite word, we introduce the abbreviation $\mathsf{last} := X\mathsf{ff}$.

An LTL formula is in negation normal form if negation symbols only occur at the level of atomic formulas. It is well-known that any given LTL formula can be converted into an equivalent formula in negation normal form and that the size of this formula is at most exponential in the size of the given formula. The modal depth $\mathrm{md}(\varphi)$ of an LTL formula φ is defined as follows: $\mathrm{md}(p) = 0$, $\mathrm{md}(\neg\varphi) = \mathrm{md}(X\varphi) = \mathrm{md}(\varphi)$, $\mathrm{md}(\varphi \vee \psi) = \mathrm{md}(\varphi \wedge \psi) = \max(\mathrm{md}(\varphi), \mathrm{md}(\psi))$, $\mathrm{md}(F\varphi) = \mathrm{md}(G\varphi) = \mathrm{md}(\varphi) + 1$, $\mathrm{md}(\varphi U \psi) = \mathrm{md}(\varphi R \psi) = \max(\mathrm{md}(\varphi), \mathrm{md}(\psi)) + 1$.

The syntax of MSO logic is defined as follows: for predicates P, first-order variables $x, y, \dots$ and second-order variables $X, Y, \dots$ the atomic formulas are $x = y, x < y, P(x), Y(x)$. An atomic formula or its negation is a *literal.* Given MSO formulas φ, ψ with free variables $x, y, \dots, X, Y, \dots$ we get the MSO formulas $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \exists x\varphi, \exists X\varphi, \forall x\varphi, \forall X\varphi$. FO logic is defined as MSO logic, but without the second-order variables and quantifiers.

To denote that a formula δ is a subformula of γ, we write $\delta \preceq \gamma$. Let $cl(\varphi)$ be the set of all subformulas of γ, called the *closure of* φ.

2.3 Interface Information

The aim of this paper is to reduce the validity of a formula in an ordered disjoint sum to information about the validity of formulas in the components. We now introduce a method to express this information. An *interface information tuple for a formula* δ is defined as the tuple $\langle \delta_1, \dots, \delta_k; \beta(X_1, \dots, X_k)\rangle$ with formulas δ_i and MSO formula β. In literature [7,13], this interface information tuple is also called *MSO profile* or *determining sequence*. The formulas δ_i are interpreted in the components and the MSO formula in the index structure. The set X_j "contains" the indices of the components in which the formula δ_j holds. Via the interface information we can describe conditions over these component indices.

We consider a small example for LTL: $\langle Fp; \exists i X_1(i)\rangle$. The MSO formula describes that there has to be (at least) one component in the set X_1, i. e. that Fp has to hold for this component. Note that this condition is equivalent to "Fp holds in the disjoint sum".

To simplify notation later, for an interface information tuple $\langle \delta_1, \dots, \delta_k;$ $\beta_\delta(X_1, \dots, X_k)\rangle$ we also allow to use the formulas $\delta_1, \dots, \delta_k$ as indices for the sets $X_1, \dots, X_k$ instead of the index number, i. e. we write X_φ to denote the set X_j where $\delta_j = \varphi$. Thus, in the example above we would write $\exists i X_{Fp}(i)$. We define the size of an interface information tuple as the number k of formulas.

For the proof of Theorem 1 we distinguish between the possibilities that a subformula δ of the given formula γ holds at one or at more states of the current component. Note that the formula γ can state that δ has to hold at exactly one state of the current component (consider e. g. γ itself as subformula or $\gamma = F\delta$) or that δ holds at more states (e. g. if $\gamma = \delta U q$ or $\gamma = G\delta$) or γ can allow both, e. g. for $\gamma = GF\delta$. We write δ *occurs repeated (in the current component)* if we refer to the possibility that δ holds at more than one states and we write δ *occurs not repeated* if we refer to exactly one state.

3 Composition Theorem for LTL

In this section we present the main theorem: we show that for an ordered disjoint sum, every LTL formula can be described by an interface information tuple, i. e. it can be decomposed into a set of LTL formulas and an MSO formula that describes which of the LTL formulas have to hold in which components. If we consider single letters as components, this can be seen as an alternative approach to LTL model-checking via Büchi automata which avoids the conversion of the formula to a Büchi automata by directly describing the decomposition in MSO logic. As in the standard approach we analyze the structure of the LTL formulas by local and global compatibility conditions. Note that the global compatibility condition amounts to a Büchi condition expressed in MSO logic.

For the theorem, we first define the set of possible formulas for the components – the *extended closure* of the given formula. To show the theorem we inductively construct interface information tuples by giving initial, local and global conditions for all types of subformulas. The induction has two parameters: the subformulas and the number of components of the sum. We show the construction with a detailed explanation.

The most interesting case of the theorem is that the index ordering is $\mathbb{N}$ and all components are finite. However, the theorem also holds for finite index orderings. In this case the last component may also be an ω-word.

We begin with the definition of the extended closure. Remember the simplified decomposition of the formula pUq from the introduction: if Gp holds at the current component either pUq or Gp has to hold at the next component. For a given formula δ we call the set of formulas which may hold at the next component the *set of component formulas* $\mathrm{cf}(\delta)$. It is defined as follows:

- $\mathrm{cf}(\varphi U \psi) = \{\bar{\psi}, \bar{\varphi} U \bar{\psi}, G\bar{\varphi}, (\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}' \mid \bar{\varphi}, \bar{\varphi}' \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$
- $\mathrm{cf}(\psi R \varphi) = \{\bar{\varphi} U(\bar{\varphi} \wedge \bar{\psi}), G\bar{\varphi}, (\bar{\varphi} U(\bar{\varphi} \wedge \bar{\psi})) \vee G\bar{\varphi}' \mid \bar{\varphi}, \bar{\varphi}' \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$
- $\mathrm{cf}(\varphi \vee \psi) = \{\bar{\varphi}, \bar{\psi}, \bar{\varphi} \vee \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$
- $\mathrm{cf}(\varphi \wedge \psi) = \{\bar{\varphi} \wedge \bar{\psi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi), \bar{\psi} \in \mathrm{cf}(\psi)\}$
- $\mathrm{cf}(X\varphi) = \{\neg\mathsf{last} \wedge X\bar{\varphi}, \neg\mathsf{last} \rightarrow X\bar{\varphi}, \mathsf{last} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$
- $\mathrm{cf}(G\varphi) = \{G\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$
- $\mathrm{cf}(F\varphi) = \{F\bar{\varphi} \mid \bar{\varphi} \in \mathrm{cf}(\varphi)\}$

The *extended closure* $\mathrm{ecl}(\delta)$ is defined analogously: in every case we extend the set of formulas for δ by the previous formulas $\bar{\varphi}$, respectively $\bar{\psi}$. We use the set $\mathrm{ecl}(\delta)$ as the set of LTL formulas in the interface information tuples. Formally, $\mathrm{ecl}(\delta)$ is defined as follows:

- $\mathrm{ecl}(\varphi U \psi) = \{\bar{\psi}, \bar{\varphi} U \bar{\psi}, G\bar{\varphi}, (\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}' \mid \bar{\varphi}, \bar{\varphi}' \in \mathrm{ecl}(\varphi), \bar{\psi} \in \mathrm{ecl}(\psi)\} \cup \mathrm{ecl}(\varphi)$
- $\mathrm{ecl}(\psi R \varphi) = \{\bar{\varphi} U(\bar{\varphi} \wedge \bar{\psi}), G\bar{\varphi}, (\bar{\varphi} U(\bar{\varphi} \wedge \bar{\psi})) \vee G\bar{\varphi}' \mid \bar{\varphi}, \bar{\varphi}' \in \mathrm{ecl}(\varphi), \bar{\psi} \in \mathrm{ecl}(\psi)\} \cup \mathrm{ecl}(\varphi) \cup \mathrm{ecl}(\psi)$
- $\mathrm{ecl}(\varphi \vee \psi) = \{\bar{\varphi}, \bar{\psi}, \bar{\varphi} \vee \bar{\psi} \mid \bar{\varphi} \in \mathrm{ecl}(\varphi), \bar{\psi} \in \mathrm{ecl}(\psi)\}$
- $\mathrm{ecl}(\varphi \wedge \psi) = \{\bar{\varphi} \wedge \bar{\psi} \mid \bar{\varphi} \in \mathrm{ecl}(\varphi), \bar{\psi} \in \mathrm{ecl}(\psi)\} \cup \mathrm{ecl}(\varphi) \cup \mathrm{ecl}(\psi)$
- $\mathrm{ecl}(X\varphi) = \{\neg\mathsf{last} \wedge X\bar{\varphi}, \neg\mathsf{last} \rightarrow X\bar{\varphi}, \mathsf{last} \mid \bar{\varphi} \in \mathrm{ecl}(\varphi)\} \cup \mathrm{ecl}(\varphi)$
- $\mathrm{ecl}(G\varphi) = \{G\bar{\varphi} \mid \bar{\varphi} \in \mathrm{ecl}(\varphi)\} \cup \mathrm{ecl}(\varphi)$
- $\mathrm{ecl}(F\varphi) = \{F\bar{\varphi} \mid \bar{\varphi} \in \mathrm{ecl}(\varphi)\} \cup \mathrm{ecl}(\varphi)$

Note that the modal depth of the formulas in $\mathrm{cf}(\delta)$ and $\mathrm{ecl}(\delta)$ is the same as the modal depth for δ.

We call an MSO formula $\beta(Z_1, \ldots, Z_m)$ *simple* if it has the following form: $\exists X_1 \subseteq Z_1 \ldots \exists X_m \subseteq Z_m\, \beta'(X_1, \ldots, X_m)$ where $\beta'(X_1, \ldots, X_m)$ uses only FO quantifiers.

Theorem 1. *For a given an index ordering $(I, <)$ with index set I the following statement holds: For every LTL formula γ of modal depth r in negation normal form we can compute an interface information tuple $\langle \gamma_1, \ldots, \gamma_m; \beta(Z_1, \ldots, Z_m)\rangle$ with LTL formulas γ_j of modal depth r – interpreted in the components – and a simple MSO formula $\beta(Z_1, \ldots, Z_m)$ – interpreted in the index ordering such that for every disjoint sum $\underline{w}$ of components $\underline{w_i}$ ($i \in I$):*

$$\underline{w} \models \gamma \Leftrightarrow (I, <) \models \beta[I_1, \ldots, I_m]$$

where $I_k = \{i \in I \mid \underline{w_i} \models \gamma_k\}$ for $k \in [m]$.
The size of the decomposition is at most exponential in $|\mathrm{cl}(\gamma)|$.

In the proof of Theorem 1 we inductively construct interface information tuples $\langle \delta_1, \dots, \delta_k; \beta_\delta(X_1, \dots, X_k)\rangle$ for the subformulas δ of the given formula γ. As mentioned above the formulas $\delta_1, \dots, \delta_k$ form the set $\mathrm{ecl}(\delta)$ which contains all possible types of formulas which may hold in a component. The MSO formula β_δ defines in which component these formulas have to hold. It ranges over the variables $X_1, \dots, X_k$ where X_i "contains" the components where δ_i holds.

The MSO formula β_δ is simple and has the following format: it contains initial, local and global compatibility conditions via formulas β^0_δ, β^L_δ and β^G_δ. The formulas β^0_δ and β^L_δ can be seen as the induction start and step of an induction over the component indices. The initial condition states that one of the formulas of $\mathrm{cf}(\delta)$ has to hold at the first component. The formula β^L_δ expresses conditions which have to hold between neighbor components. (Consider e. g. $\delta = pUq$, then β^L_δ states: if Gp holds at the current component then q, Gp or pUq (or $Gp \vee (pUq)$) has to hold at the next component.) The formula β^G_δ ensures that the second part of each Until-subformula eventually holds via a Büchi condition expressed in MSO logic, e. g. for pUq in the disjoint sum we have to disallow Gp (and $(pUq) \vee Gp$) in all components.

The MSO formula β_δ expresses that for all formulas $\bar\delta$ in $\mathrm{ecl}(\delta)$ there is a subset of components where $\bar\delta$ has to hold and for these subsets the formulas $\beta^0_\delta, \beta^L_\delta$ and β^G_δ are fulfilled. Formally, β_δ has the form

$$\beta_\delta := \underset{\bar\delta \in \mathrm{ecl}(\delta)}{\exists} Y_{\bar\delta} \subseteq X_{\bar\delta}[\beta^0_\delta \wedge \beta^L_\delta \wedge \beta^G_\delta] \text{ with } \beta^L_\delta := \forall i(\bigwedge_{\bar\delta \in \mathrm{ecl}(\delta)} (Y_{\bar\delta}(i) \to \beta_{\delta,\bar\delta}(i)))$$

where we use the abbreviation $\exists_{\bar\delta \in \mathrm{ecl}(\delta)} Y_{\bar\delta} \subseteq X_{\bar\delta} := \exists Y_1 \subseteq X_1 \dots \exists Y_k \subseteq X_k$ for $\mathrm{ecl}(\delta) = \{\delta_1, \dots, \delta_k\}$.

The local compatibility condition β^L_δ lists for all components and all $\bar\delta \in \mathrm{ecl}(\delta)$ conditions $\beta_{\delta,\bar\delta}(i)$ which have to hold between neighbor components. These $\beta_{\delta,\bar\delta}(i)$ are defined depending on the type of the formula δ. We now define the formulas $\beta^0_\delta, \beta^G_\delta$ and $\beta_{\delta,\bar\delta}$ for every subformula δ and every $\bar\delta \in \mathrm{ecl}(\delta)$.

Induction start:
We start with predicates, i. e. $\delta = p$. Then we have $\mathrm{ecl}(\delta) = \{p\}$ and as formula β^0_δ that the first component has to be in the set Y_p, i. e. $\beta^0_\delta = Y_p(1)$. For predicates we have an empty global condition and empty local conditions, i. e. $\beta^G_\delta = \mathtt{tt}$ and $\beta_{\delta,p}(i) = \mathtt{tt}$. The case of negated predicates $\neg p$ is analogous.

Induction hypothesis:

Let $\underline{w}^{\geq h}$ denote the ordered disjoint sum starting with the h-th component. Further, let $I^{\geq h} := \{i \in I \mid i \geq h\}$. The induction hypothesis is divided into two parts: On one hand we assume that the result already holds for all disjoint sums $\underline{w}^{\geq i}$ starting from any component i, any state in this component and all *real* subformulas φ of δ. On the other hand we assume that the result holds for all disjoint sums $\underline{w}^{\geq i}$ starting from a component $i \geq h+1$, any state in this component and (also) δ itself.

Induction step:

Given the current component h and the current state u (or current states u_m) and the formula δ we apply the induction hypothesis for either the direct subformulas of δ and states from u (or all u_m) onwards in the current component or for δ itself (or the direct subformulas of δ) and the first state of the $(h+1)$-th component.

We begin with $\delta = \varphi U \psi$: By induction hypothesis we have given β_φ and β_ψ and thus, the local conditions $\beta_{\varphi,\bar{\varphi}}(i)$, $\beta_{\psi,\bar{\psi}}(i)$ (for $\bar{\varphi} \in \mathrm{ecl}(\varphi)$, $\bar{\psi} \in \mathrm{ecl}(\psi)$) and the global conditions $\beta^G_\varphi, \beta^G_\psi$.

We first consider that δ occurs not repeated. Obviously, the formula $\varphi U \psi$ holds at the current state u in the disjoint sum (starting from component h) if either ψ holds directly at u or if φ holds from u onwards up to a state v where ψ holds. The state v may be in the same component or in a later component. Note that the last case in particular means that φ holds on all states of the current component (from u onwards). These three cases are captured by the formulas $\bar{\psi}$,$\bar{\varphi} U \bar{\psi}$ respectively $G\bar{\varphi}$ for the current component and can be seen on the left side of Figure 1. We allow these cases (and a currently redundant case) in the formula for the induction start: $\beta^0_\delta = \bigvee_{\hat{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\hat{\psi} \in \mathrm{cf}(\psi)} (Y_{\hat{\psi}}(1) \vee Y_{\hat{\varphi} U \hat{\psi}}(1) \vee Y_{G\hat{\varphi}}(1) \vee Y_{(\hat{\varphi} U \psi) \vee G\hat{\varphi}}(1))$.

Now we define the local conditions for these formulas: first, we assure that the local conditions for $\bar{\varphi}$ respectively for $\bar{\psi}$ have to be fulfilled only (!) if $\bar{\varphi}$ respectively $\bar{\psi}$ occur[1] in $\bar{\delta}$. For the case $G\bar{\varphi}$ (and the case $(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}$) we further have to assure that sometime later $\varphi U \psi$ holds in the disjoint sum by guaranteeing that one of the four cases of $\mathrm{cf}(\delta)$ holds at the next component.

- $\beta_{\delta,\bar{\psi}}(i) = \beta_{\psi,\bar{\psi}}(i)$
- $\beta_{\delta,\bar{\varphi} U \bar{\psi}}(i) = \beta_{\varphi,\bar{\varphi}}(i) \wedge \beta_{\psi,\bar{\psi}}(i)$
- $\beta_{\delta,G\bar{\varphi}}(i) = \beta_{\varphi,\bar{\varphi}}(i) \wedge$
  $\bigvee_{\hat{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\hat{\psi} \in \mathrm{cf}(\psi)} (Y_{\hat{\psi}}(i+1) \vee Y_{\hat{\varphi} U \hat{\psi}}(i+1) \vee Y_{G\hat{\varphi}}(i+1) \vee Y_{(\hat{\varphi} U \psi) \vee G\hat{\varphi}}(i+1))$
- $\beta_{\delta,(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}}(i) = \beta_{\varphi,\bar{\varphi}}(i) \wedge \beta_{\psi,\bar{\psi}}(i) \wedge$
  $\bigvee_{\hat{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\hat{\psi} \in \mathrm{cf}(\psi)} (Y_{\hat{\psi}}(i+1) \vee Y_{\hat{\varphi} U \hat{\psi}}(i+1) \vee Y_{G\hat{\varphi}}(i+1) \vee Y_{(\hat{\varphi} U \psi) \vee G\hat{\varphi}}(i+1))$

We now consider that δ occurs repeated. This leads to the following possibilities: (1) at all of these states ψ holds, (2) at all of these states $\varphi U \psi$ (but not necessarily at all ψ) holds, (3) at all of these states $G\varphi$ holds and $\varphi U \psi$ is satisfied later or (4) there are some states where $\varphi U \psi$ and some where $G\varphi$ holds (and $\varphi U \psi$ is satisfied later). These cases are captured by the four possibilities in $\mathrm{cf}(\delta)$. They are shown at the right side of Figure 1. Note that we distinguish between $G\bar{\varphi}$ and $(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}$, because in the last case, we have to enforce also the local compatibility conditions for $\bar{\psi}$ and not only those for $\bar{\varphi}$.

As already mentioned in the introduction we have to add global compatibility conditions to prevent $G\bar{\varphi}$ or $(\bar{\varphi} U \bar{\psi}) \vee G\bar{\varphi}$ on all components. Apart from this, we have to add the global compatibility conditions for the subformulas φ, ψ. This

[1] This is the reason why we have to distinguish between the cases $\bar{\psi}$ and $\bar{\varphi} U \bar{\psi}$.

is (both) done by $\beta_\delta^G = \bigwedge_{\hat\varphi\in \mathrm{cf}(\varphi)} \bigwedge_{\hat\psi\in \mathrm{cf}(\psi)} \neg\exists k \forall i \geq k (Y_{G\hat\varphi}(i) \vee Y_{(\hat\varphi U \hat\psi)\vee G\hat\psi}(i)) \wedge \beta_\varphi^G \wedge \beta_\psi^G$

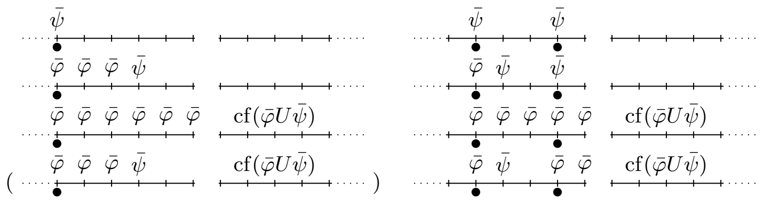
$\bar\psi$
$\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\psi$
$\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\varphi$
cf$(\bar\varphi U \bar\psi)$
$\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\psi$
cf$(\bar\varphi U \bar\psi)$
$\bar\psi$ $\bar\psi$
$\bar\varphi$ $\bar\psi$ $\bar\psi$
$\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\varphi$ $\bar\varphi$
cf$(\bar\varphi U \bar\psi)$
$\bar\varphi$ $\bar\psi$ $\bar\varphi$ $\bar\varphi$
cf$(\bar\varphi U \bar\psi)$

Fig. 1. Cases for $\delta = \varphi U \psi$ occurring not repeated / repeated

Before we proceed with the other cases, let us first consider an example which uses only Until-operators.

Example 1. We consider the examples pUq and $(rUs)Uq$. For $\delta = pUq$ we get $\mathrm{cf}(\delta) = \{q, pUq, Gp, (pUq) \vee Gp\}$ and $\mathrm{ecl}(\delta) = \mathrm{cf}(\delta) \cup \{p\}$. The formula β^0_{pUq} is $Y_q(1) \vee Y_{pUq}(1) \vee Y_{Gp}(1) \vee Y_{(pUq)\vee Gp}(1)$, i. e. either q, pUq or Gp (or $(pUq) \vee Gp$) holds at the first component. If q or pUq holds at any component, no more conditions have to be satisfied. Thus, $\beta_{\delta,q} = \mathtt{tt}$ and $\beta_{\delta,pUq} = \mathtt{tt}$. Otherwise, one of the formulas $q, pUq, Gp, (pUq) \vee Gp$ has to hold at the next component. Thus, $\beta_{\delta,Gp} = \beta_{\delta,(pUq)\vee Gp} = (Y_q(i+1) \vee Y_{pUq}(i+1) \vee Y_{Gp}(i+1) \vee Y_{(pUq)\vee Gp}(i+1))$. For this example, $(pUq) \vee Gp$ would not be necessary as δ occurs not repeated. Further, the distinction between q and pUq is also superfluous, because there are no further local conditions for p, i. e. $\beta_{p,p}(i) = \mathtt{tt}$. However, if we consider the second example $\delta = \varphi U q$ with $\varphi = rUs$ the distinction between q and $\bar\varphi U q$ is necessary: we have $\beta_{\delta,q}(i) = \mathtt{tt}$ but e. g. $\beta_{\delta,GrUq}(i) = \beta_{\varphi,Gr}(i) = Y_s(i+1) \vee Y_{rUs}(i+1) \vee Y_{Gr}(i+1) \vee Y_{(rUs)\vee Gr}(i+1)$ as in this case rUs has still to be satisfied.

We continue with $\delta = \varphi \vee \psi$: Given the initial, local and global compatibility conditions for φ and ψ, we define the conditions for δ. The formula β^0_δ allows that φ or ψ holds at the first component: $\beta^0_\delta = \bigvee_{\hat\varphi\in\mathrm{cf}(\varphi)} Y_{\hat\varphi}(1) \vee \bigvee_{\hat\psi\in\mathrm{cf}(\psi)} Y_{\hat\psi}(1) \vee \bigvee_{\hat\varphi\in\mathrm{cf}(\varphi)} \bigvee_{\hat\psi\in\mathrm{cf}(\psi)} Y_{\hat\varphi\vee\hat\psi}(1)$. Note that as δ might occur repeated we also have $\bar\varphi \vee \bar\psi \in \mathrm{cf}(\delta)$ to force both the local compatibility conditions for φ and ψ if on some of the states φ and on some ψ holds. We get

- $\beta_{\delta,\bar\varphi}(i) = \beta_{\varphi,\bar\varphi}(i)$ and $\beta_{\delta,\bar\psi}(i) = \beta_{\psi,\bar\psi}(i)$
- $\beta_{\delta,\bar\varphi\vee\bar\psi}(i) = \beta_{\varphi,\bar\varphi}(i) \wedge \beta_{\psi,\bar\psi}(i)$
- $\beta^G_\delta = \beta^G_\varphi \wedge \beta^G_\psi$

Now, we consider $\delta = X\varphi$: We can have three possible formulas for the current component: (A) $\neg\mathsf{last} \wedge X\bar\varphi$, (B) $\neg\mathsf{last} \rightarrow X\bar\varphi$ and (C) last (for $\bar\varphi \in \mathrm{cf}(\varphi)$).

If δ occurs not repeated there are two reasons why the formula $X\varphi$ holds: either (A) φ holds at the next state of the current component and the current state is not the last one, or (B) the current state is the last one and φ holds from the next component onwards. The condition (A) is expressed by $\neg\mathsf{last} \wedge X\bar{\varphi}$ and the condition (B) by last and forcing that φ holds at the next component via the local compatibility condition $\beta_{\delta,\mathsf{last}}$. These conditions are shown on the left side of Figure 2. (Condition (C) is not needed for this case.)

If $\delta = X\varphi$ holds at more than one states of the current component, then (C) either the last state is within these states or (A) not. The condition (C) is captured by $X\bar{\varphi}$ for all of these states without the last one ($\neg\mathsf{last} \rightarrow X\bar{\varphi}$) and by forcing that φ holds at the next component as above. The condition (A) is handled as above. Note that we can add (B) as disjunctive element in this case as last is false for more than one state. These conditions are shown on the right side of Figure 2.

As initial condition we get $\beta^0_\delta = \bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)}(Y_{\neg\mathsf{last}\wedge X\hat{\varphi}}(1) \vee Y_{\mathsf{last}}(1) \vee Y_{\neg\mathsf{last}\rightarrow X\hat{\varphi}}(1))$. The global compatibility condition is $\beta^G_\delta = \beta^G_\varphi$ and the local ones are:

(A) $\beta_{\delta,\neg\mathsf{last}\wedge X\bar{\varphi}}(i) = \beta_{\varphi,\bar{\varphi}}(i)$
(B) $\beta_{\delta,\mathsf{last}}(i) = \bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)} Y_{\hat{\varphi}}(i+1)$
(C) $\beta_{\delta,\neg\mathsf{last}\rightarrow X\bar{\varphi}}(i) = \beta_{\varphi,\bar{\varphi}}(i) \wedge \bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)} Y_{\hat{\varphi}}(i+1)$

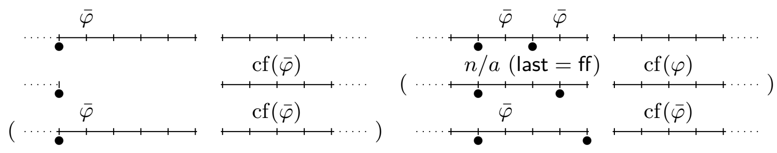
$\bar{\varphi}$
$\mathrm{cf}(\bar{\varphi})$
$\bar{\varphi}$
$\mathrm{cf}(\bar{\varphi})$
$\bar{\varphi}$ $\bar{\varphi}$
n/a ($\mathsf{last} = \mathsf{ff}$)
$\mathrm{cf}(\varphi)$
$\bar{\varphi}$
$\mathrm{cf}(\bar{\varphi})$

Fig. 2. Cases for $\delta = X\varphi$ occurring not repeated / repeated

The cases $\delta = \varphi \wedge \psi$ and $\delta = \psi R\varphi$ are analogous to the cases $\varphi \vee \psi$ and $\varphi U\psi$, so we only give the formal descriptions. We omit $F\varphi$ and $G\varphi$ as they can be derived via $F\varphi = \mathsf{tt}U\varphi$ and $G\varphi = \mathsf{ff}R\psi$.

- $\delta = \varphi \wedge \psi$:
  - $\beta^0_\delta = \bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)} \bigvee_{\hat{\psi}\in\mathrm{cf}(\psi)}(Y_{\hat{\varphi}}(1) \wedge Y_{\hat{\psi}}(1))$
  - $\beta_{\delta,\bar{\varphi}}(i) = \beta_{\varphi,\bar{\varphi}}(i)$ and $\beta_{\delta,\bar{\psi}}(i) = \beta_{\psi,\bar{\psi}}(i)$
  - $\beta_{\delta,\bar{\varphi}\wedge\bar{\psi}}(i) = \beta_{\varphi,\bar{\varphi}}(i) \wedge \beta_{\psi,\bar{\psi}}(i)$
  - $\beta^G_\delta = \beta^G_\varphi \wedge \beta^G_\psi$
- $\delta = \psi R\varphi$:
  - $\beta^0_\delta = \bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)} \bigvee_{\hat{\psi}\in\mathrm{cf}(\psi)}(Y_{\hat{\varphi}U(\hat{\varphi}\wedge\hat{\psi})}(1) \vee Y_{G\hat{\varphi}}(1) \vee Y_{(\hat{\varphi}U(\hat{\varphi}\wedge\hat{\psi}))\vee G\hat{\varphi}}(1))$
  - $\beta_{\delta,\bar{\varphi}U(\bar{\varphi}\wedge\bar{\psi})}(i) = \beta_{\bar{\varphi}}(i) \wedge \beta_{\bar{\psi}}(i)$
  - $\beta_{\delta,G\bar{\varphi}}(i) = \beta_{\bar{\varphi}}(i) \wedge$
    $\bigvee_{\hat{\varphi}\in\mathrm{cf}(\varphi)} \bigvee_{\hat{\psi}\in\mathrm{cf}(\psi)}(Y_{\hat{\varphi}U(\hat{\varphi}\wedge\hat{\psi})}(i+1) \vee Y_{G\hat{\varphi}}(i+1) \vee Y_{(\hat{\varphi}U(\hat{\varphi}\wedge\hat{\psi}))\vee G\hat{\varphi}}(i+1))$

- $\beta_{(\bar{\varphi} U(\bar{\varphi} \wedge \bar{\psi})) \vee G\bar{\varphi}} = \beta_{\bar{\varphi}}(i)) \wedge \beta_{\bar{\psi}}(i) \wedge$
  $\bigvee_{\hat{\varphi} \in \mathrm{cf}(\varphi)} \bigvee_{\hat{\psi} \in \mathrm{cf}(\psi)} (Y_{\hat{\varphi} U(\hat{\varphi} \wedge \hat{\psi})}(i+1) \vee Y_{G\hat{\varphi}}(i+1) \vee Y_{(\hat{\varphi} U(\hat{\varphi} \wedge \hat{\psi})) \vee G\hat{\varphi}}(i+1))$
- $\beta^G_\delta = \beta^G_\varphi \wedge \beta^G_\psi$

To estimate the complexity of the construction, we count the number of formulas for the components, calculate their size and the size of the MSO formula from the interface information tuple.

From the definition of $\mathrm{ecl}(\delta)$ we get for each step $|\mathrm{ecl}(\delta)| \leq c \cdot |\mathrm{ecl}(\varphi)|^2 \cdot |\mathrm{ecl}(\psi)|$ for a $c \in \mathbb{N}$ (and $|\mathrm{ecl}(\psi)| = 0$ if δ contains no ψ). Thus, there are exponentially many formulas for the components – $|\mathrm{ecl}(\delta)| \in O(2^{|\mathrm{cl}(\delta)|})$. Further, for all $\bar{\delta} \in \mathrm{ecl}(\delta)$, $|\bar{\delta}| \leq 3 \cdot |\bar{\varphi}| + |\bar{\psi}|$, so each formula is at most exponential in the size of the input formula. For the parts of the MSO formula we get a length of at most $|\mathrm{ecl}(\delta)|$ for $\exists_{\bar{\delta} \in \mathrm{ecl}(\delta)} Y_{\bar{\delta}} \subseteq X_{\bar{\delta}}$ and $|\mathrm{cf}(\delta)|$ ($\leq |\mathrm{ecl}(\delta)|$) for both β^0_δ and β^G_δ. For each of the $|\mathrm{ecl}(\delta)|$ many formulas $\beta_{\delta,\bar{\delta}}$ we also have a length of at most $\leq |\mathrm{ecl}(\delta)|$. Thus, in total the size of β_δ is at most $O(2^{2^{|\mathrm{cl}(\delta)|}})$.

4 Summary and Outlook

In this paper we have shown that the validity of an LTL formula in an ordered disjoint sum can be deduced from information about the components and the index ordering. We have built interface information tuples – LTL formulas for the components and a simple MSO formula which states in which components these formulas have to hold. We proved that – in contrast to FO logic over disjoint sums – here, the size of the decomposition is at most exponential in the size of the input formula.

Current work deals with generalizing this result to computational time logic (CTL) over sums and products of transition systems. Here, it is also of interest to generalize the ordered index structure to a tree.

Acknowledgments

Thanks are due to Wolfgang Thomas for fruitful discussions about the subject of this paper and to anonymous referees whose remarks have led to a clearer presentation.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
2. Chang, C.C., Keisler, H.J.: Model Theory. North Holland, Amsterdam (1990)
3. Dawar, A., Grohe, M., Kreutzer, S., Schweikardt, N.: Model Theory Makes Formulas Large. In: ICALP'07: 34th Int. Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 4596, pp. 913–924. Springer (2007)

4. Ehrenfeucht: An application of games to the completeness problem for formalized theories. Fundamenta mathematicae 49, 129–141 (1961)
5. Feferman, S., Vaught, R.: The first-order properties of products of algebraic systems. Fundamenta Mathematicae 47, 57–103 (1959)
6. Felscher, I.: The Compositional Method and Regular Reachability. Electronic Notes in Theoretical Computer Science 223, 103–117 (December 2008)
7. Felscher, I., Thomas, W.: Compositionality and Reachability with Conditions on Path Lengths. Int. Journal of Foundations of Computer Science 20(5), 851–868 (May 2009)
8. Fraïssé: Sur quelques classifications des systèmes de relations. Publications Scientifiques de l'université d'Alger 1(A), 35–182 (1954)
9. Göller, S., Jung, J.C., Lohrey, M.: The complexity of decomposing modal and first-order theories. Lecture Notes of Computer Science to appear (2012)
10. Hodges, W.: Model Theory, Encyclopedia of Mathematics and its Applications, vol. 42. Cambridge University Press, Cambridge (1993)
11. Makowsky, J.A.: Algorithmic Uses of the Feferman-Vaught Theorem. Annals of Pure and Applied Logic 126(1–3), 159–213 (2004)
12. Mostowski, A.: On Direct Products of Theories. The Journal of Symbolic Logic 17(1), 1–31 (1952)
13. Rabinovich, A.: On Compositionality and its Limitations. ACM Transactions on Computational Logic 8(1) (January 2007)
14. Thomas, W.: Ehrenfeucht Games, the Composition Method, and the Monadic Theory of Ordinal Words. In: Mycielski, J., Rozenberg, G., Salomaa, A. (eds.) Structures in Logic and Computer Science, A Selection of Essays in Honor of A. Ehrenfeucht. LNCS, vol. 1261, pp. 118–143. Springer–Verlag (1997)
15. Wöhrle, S., Thomas, W.: Model Checking Synchronized Products of Infinite Transition Systems. In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science. pp. 2–11. LNCS, IEEE Computer Society, Washington, DC, USA (2004)