Electronic Communications of the EASST
Volume 47 (2012)

Proceedings of the 11th International Workshop on Graph Transformation and Visual Modeling Techniques (GTVMT 2012)

Towards Alternating Automata for Graph Languages

H.J. Sander Bruggink, Mathias Hülsbusch and Barbara König

14 pages

Guest Editors: Andrew Fish, Leen Lambers
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Towards Alternating Automata for Graph Languages

H.J. Sander Bruggink, Mathias Hülsbusch and Barbara König *

University of Duisburg-Essen, Germany
sander.bruggink@uni-due.de, mathias.huelsbusch@uni-due.de, barbara_koenig@uni-due.de

Abstract: In this paper we introduce alternating automata for languages of arrows of an arbitrary category, and as an instantiation thereof alternating automata for graph languages. We study some of their closure properties and compare them, with respect to expressiveness, to other methods for describing graph languages. We show, by providing several examples, that many graph properties (of graphs of bounded path width) can be naturally expressed as alternating automata.

Keywords: alternating automata, graph automata

1 Introduction

Alternating variants of Turing machines, finite automata and pushdown automata were introduced in [7]. They are a generalization of the non-deterministic variants of the automata in which each state has either a universal or an existential quantifier associated with it. A universal state accepts a word when all outgoing transitions are accepting, while an existential state accepts when one or more outgoing transitions are accepting. In this paper we study alternating graph automata, by first defining a kind of alternating finite automaton which operates on arrows of an arbitrary category, and then instantiating it to the category of (cospans of) graphs.

The main motivation of the work is the following. In [5] automaton functors were introduced as an automaton model for recognizable graph languages. Automaton functors have nice closure and decidability properties, which were used, among others, to automatically verify invariants of graph transformation systems [2]. However, the main disadvantages of automaton functors are:

(a) They tend to be quite large (and tend to grow exponentially when extended to graphs of larger path width).
(b) Proving that a construct actually is an automaton functor is often non-trivial.
(c) They inherit certain disadvantages from finite automata. Constructing an automaton which accepts the complement of the language of a given automaton functor, for example, requires determinization, resulting in an exponential blowup of an already large state space.

The alternating automata that we present in this paper emerged as a way to tackle all three of these problems, at the cost of losing decidability of some decision problems, such as deciding whether the language of an automaton is empty:

(a) They make automata smaller and easier to construct.
(b) They do not need to satisfy the functor property.
(c) Constructing an alternating automaton for the complement of a language of an alternating automaton, as well as for the union and intersection of two languages, is possible in linear time.

* This work was supported by the DFG-project GaReV.

This paper is a first survey of alternating graph automata and their properties and expressive power.

The paper is organized as follows: In Section 2 we recapitulate some notions from category theory and graph theory. In Section 3 we define alternating automata which accept arrows of an arbitrary category. Then, in Section 4, we instantiate this definition to the category of cospans of graphs, obtaining alternating graph automata, and study their properties. Specifically, we give examples in Subsection 4.2, study their membership problem in Subsection 4.3 and compare their expressive power to other mechanism for describing graph languages in Section 5. Finally, in Section 6 we conclude and give pointers for further research.

2 Preliminaries

Orders and relations. A *quasi-order* (also called *preorder*) is a reflexive, transitive relation; a *strict order* is an irreflexive, transitive relation. A relation $<$ on A is *well-founded* if it does not have infinite descending chains, that is, if there do not exists $a_1, a_2, a_3 \ldots \in A$ such that $a_1 > a_2 > a_3 > \cdots$.

Categories. We presuppose a basic knowledge of category theory. For arrows $f\colon A \to B$ and $g\colon B \to C$, the composition of f and g is denoted $(f;g)\colon A \to C$. The category **Rel** has sets as objects and relations as arrows. Its subcategory **Set** has only the total, functional relations (functions) as arrows.

Let $\mathbf{C}$ be a category in which all pushouts exist. A *concrete cospan* α *in* $\mathbf{C}$ is a pair $\langle \ell_\alpha, r_\alpha \rangle$ of $\mathbf{C}$-arrows such that $J \xrightarrow{\ell_\alpha} G \xleftarrow{r_\alpha} K$. In such a cospan, J will be called the left or *inner interface*, while K will be called the right or *outer interface* and G the *center object*.

Concrete cospans are isomorphic if their middle objects are (such that the isomorphism commutes with the component morphisms of the cospan). A cospan is an isomorphism class of concrete cospans. In the following we will confuse cospans and concrete representatives thereof.

Cospans are the arrows of so-called cospan categories. That is, for a category $\mathbf{C}$ with pushouts, the cospan category of $\mathbf{C}$, denoted $\mathrm{Cospan}(\mathbf{C})$, has the same objects as $\mathbf{C}$. The isomorphism class of a cospan $\alpha\colon J \xrightarrow{\ell_\alpha} G \xleftarrow{r_\alpha} K$ in $\mathbf{C}$ is an arrow from J to K in $\mathrm{Cospan}(\mathbf{C})$. Composition of two cospans $\alpha = \langle \ell_\alpha, r_\alpha \rangle$ and $\beta = \langle \ell_\beta, r_\beta \rangle$ is computed by taking the pushout of the arrows r_α and ℓ_β; this is well-defined because taking a pushout is unique up to isomorphism.

In cospan categories, we will use lowercase Greek letters to refer to cospans (arrows in the cospan category), while we will use lowercase roman letters to refer to arrows in the base category.

Graphs. Let Σ be a set of possible labels. Each label m of Σ is associated with an arity $ar(m)$. A *hypergraph* over Σ (in the following also simply called *graph*) is a structure $G = \langle V, E, lab, att \rangle$, where V is a finite set of nodes, E is a finite set of edges, $lab\colon E \to \Sigma$ assigns a label to each edge and $att\colon E \to V^*$ maps each edge to a finite sequence of nodes attached to it, such that $|att(e)| = ar(lab(e))$, for all e. A *discrete graph* is a graph without edges; the discrete graph with node set $\{1, \ldots, k\}$ is denoted by D_k. A graph morphism is a structure preserving map between two graphs. The category of graphs and graph morphisms is denoted by **Graph**.

Graphically, nodes are represented by black circles. In examples, we will only use binary edges. That is, $ar(m) = 2$ for all $m \in \Sigma$. Such edges will be denoted by an arrow with label besides it, as usual. In examples, where the signature contains only a single label, the label will not be shown.

A cospan $\alpha\colon J \xrightarrow{\ell_\alpha} G \xleftarrow{r_\alpha} K$ in **Graph** can be viewed as a graph (G) with two interfaces (J and K). Informally said, only elements of G which are in the image of one of the interfaces can be "touched", in the sense that they can be connected to or fused with other elements. By $[G]$ we denote the trivial cospan $\varnothing \to G \leftarrow \varnothing$, the graph G with two empty interfaces.

Cospans of graphs are intimately connected with the double pushout approach to graph rewriting [11]. A rewriting rule p can be defined as a pair of cospans $\lambda : \varnothing \to L \leftarrow I$ and $\rho : \varnothing \to R \leftarrow I$ with the same outer interface. A graph G rewrites to H by applying rule p if and only if $[G] = \lambda ; \kappa$ and $[H] = \rho ; \kappa$ for some cospan $\kappa\colon I \to K \leftarrow \varnothing$ (where K is an arbitrary graph). This approach to graph transformation is easily seen to be equivalent to the double pushout approach.

A *path decomposition* of a graph G is a sequence $X_1, \ldots, X_n$ of sets of nodes of G (called *bags*), such that

- each node v of G is contained in at least one bag X_i;
- for each edge e of G, there is a bag X_i which contains all nodes adjacent to e; and
- for each node v of G, the bags which contains v form a (connected) subsequence of $X_1, \ldots, X_n$.

The width of a path decomposition is $|X| - 1$, where X is the largest bag of the decomposition. The *path width* of a graph is the width of its smallest path decomposition. Path width is tightly connected to cospan decompositions [1]: if α is a cospan and $\varphi_1, \ldots, \varphi_n$ are cospans with discrete interfaces such that $\varphi_1 ; \cdots ; \varphi_n = \alpha$, the path width of the center graph of α is bounded by the maximum of the path widths of the center graphs of the φ_i.

3 Alternating Automata on Arrows in a Category

In this section we define alternating automata on an arbitrary category, and derive some general results about them. In the next section we instantiate the definition on the category Cospan(**Graph**) to obtain alternating graph automata.

Define the following quasi-order on arrows of a category $\mathbf{C}$: $a \leq b$ if $b = f ; a$ for some arrow f. Note that, for $a \leq b$ to hold, a and b do not need to have the same domain (but they do need to have the same codomain).

Definition 1 Let $\mathbf{C}$ be a category.

(i) Let $\sqsubset$ be a (partial) strict order on arrows of $\mathbf{C}$. The relation $\sqsubset$ is called *progressing*, if it is a subrelation of $\leq$ (that is, if $a \sqsubset b$ implies $a \leq b$) and for each infinite descending sequence $a_1 \geq a_2 \geq a_3 \cdots$ there are only finitely many indices i such that $a_i \sqsupset a_{i+1}$ holds.[1]

(ii) Let $\sqsubset$ be an progressing relation. A $\mathbf{C}$-arrow $f\colon A \to B$ is *$\sqsubset$-productive*, if $a \sqsubset f ; a$ for all arrows $a\colon B \to C$ (where C is arbitrary).

[1] Note, that this condition is stronger than just requiring $\sqsubset$ to be a well-founded subrelation of $\leq$. Let, for example, $\sqsubset = \{\langle 2x-1, 2x\rangle \mid x \in \mathbb{Z}\}$. Then $\sqsubset$ is a well-founded subrelation of $\leq$, but $-1, -2, -3, \ldots$ is an infinite descending sequence with infinitely many indices where $a_i > a_{i+1}$.

Definition 2 Let $\mathbf{C}$ be a category and $\sqsubset$ a progressing strict order. An *alternating* $(\mathbf{C},\sqsubset)$-*automaton* is a structure $\mathcal{A} = \langle U, Q, inf, quant, S, \delta \rangle$, where

- U is a finite set of $\mathbf{C}$-objects, called the *interfaces*,
- Q is a finite set of *states*,
- $inf\colon Q \to U$ is a function which assigns an interface to each state,
- $quant\colon Q \to \{\forall, \exists\}$ assigns a *quantifier* to each state,
- $S \subseteq Q$ is the set of *initial states*, and
- $\delta \subseteq Q \times Arr(\mathbf{C}) \times Q$ is the *transition relation*, where $Arr(\mathbf{C})$ denotes the arrows of $\mathbf{C}$ and we require that $inf(q) = dom(a)$ and $inf(q') = cod(a)$ if $\langle q, a, q' \rangle \in \delta$.
- Let a cycle be a finite sequence $q_0, a_0, \ldots, a_{n-1}, q_n$, where $q_i \in Q$ for each $0 \leq i \leq n$, $q_0 = q_n$ and $\langle q_i, a_i, q_{i+1} \rangle \in \delta$ for $0 \leq i < n$. For each cycle $q_0, a_0, \ldots, a_{n-1}, q_n$ in the automaton it must hold that at least one of the a_i is $\sqsubset$-productive.

Alternating automata do not have final states: universally quantified states without successors play the role of accepting states, while, conversely, existentially quantified states without successors play the role of rejecting states.

The last condition of the definition ensures that we can use induction to define the language accepted by an alternating automaton (see Definition 3), and prove correctness of the constructions on alternating automata (see Theorem 1). Without the condition, an automaton could get caught in an endless loop, in which no part of the arrow is effectively "read in". Now we can define, given an alternating automaton $\mathcal{A} = \langle U, Q, inf, quant, S, \delta \rangle$, a well-founded order on pairs consisting of an arrow a and a state q as follows: $\langle a, q \rangle < \langle a', q' \rangle$ if there is a $\mathbf{C}$-arrow f such that $a' = f;a$ and $\langle q, f, q' \rangle \in \delta$. This ordering is well-founded: If an infinite descending sequence $\langle a_1, q_1 \rangle > \langle a_2, q_2 \rangle > \cdots$ would exist, one state q must occur infinitely many times. Since, by definition, $a_1 \geq a_2 \geq \cdots$, there must, by the last condition of Definition 2, be infinitely many indices i where $a_i \sqsupset a_{i+1}$, which cannot happen.

Definition 3 Let $\mathcal{A} = \langle U, Q, inf, quant, s, \delta \rangle$ be an alternating $(\mathbf{C},\sqsubset)$-automaton.

(i) A state $q \in Q$ accepts a $\mathbf{C}$-arrow $c\colon inf(q) \to K$ (where K is a $\mathbf{C}$-object), written $\mathcal{A}, q \models c$, if:

- $quant(q) = \exists$ and there exist $\mathbf{C}$-arrows $f\colon inf(q) \to J$ and $c'\colon J \to K$ and a state $q' \in Q$ such that $c = f;c'$, $\langle q, f, q' \rangle \in \delta$ and $\mathcal{A}, q' \models c'$.
- $quant(q) = \forall$ and for all $\mathbf{C}$-arrows $f\colon inf(q) \to J$ and $c'\colon J \to K$ and states $q' \in Q$ such that $c = f;c'$ and $\langle q, f, q' \rangle \in \delta$, it holds that $\mathcal{A}, q' \models c'$.

(ii) The *language* of $\mathcal{A}$, written $\mathrm{L}(\mathcal{A})$, is defined as

$$\mathrm{L}(\mathcal{A}) = \bigcup_{s \in S} \{ c \mid \mathcal{A}, s \models c \}.$$

(iii) A language L is called an *alternating* $(\mathbf{C},\sqsubset)$*-language* if there exists an alternating $(\mathbf{C},\sqsubset)$-automaton $\mathcal{A}$ such that $L = \mathrm{L}(\mathcal{A})$.

The conditions from [3], which can be seen as a generalization of nested application conditions [10] from graphs to arbitrary categories, are a special case of alternating automata, namely the alternating automata without loops.

For constructions on automata it is often convenient to assume that an automaton has exactly one initial state for each of its interfaces. We can impose this condition without restricting the expressive power, as is shown in the following proposition.

Proposition 1 *For each alternating* $(\mathbf{C},\sqsubset)$*-automaton* $\mathcal{A} = \langle U, Q, inf, quant, S, \delta\rangle$*, there exists an alternating* $(\mathbf{C},\sqsubset)$*-automaton* $\mathcal{A}' = \langle U', Q', inf', quant', S', \delta'\rangle$ *such that* $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{A}')$ *and* $|\{q \in S' \mid inf'(q) = K\}| = 1$ *for each* $\mathbf{C}$*-object* $K \in U$*.*

Proof. We build $\mathcal{A}'$, by modifying $\mathcal{A}$ as follows: The old start states are no longer start states. For each interface K we add a new existentially quantified start state s_K with id-labeled transitions to the old start states. This new automaton clearly accepts the same language and is of the desired form. □

Theorem 1 *Let* $\mathbf{C}$ *be a category and* $\sqsubset$ *a progressing relation. The class of alternating languages is closed under union, intersection and complement. In fact, for alternating* $(\mathbf{C},\sqsubset)$*-automata* $\mathcal{A}$ *and* $\mathcal{B}$*, alternating* $(\mathbf{C},\sqsubset)$*-automata that accept* $\mathrm{L}(\mathcal{A}) \cup \mathrm{L}(\mathcal{B})$*,* $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B})$ *and* $\mathbf{C} \setminus \mathrm{L}(\mathcal{A})$ *can be efficiently constructed.*

Proof. Let $\mathcal{A} = \langle U_\mathcal{A}, Q_\mathcal{A}, inf_\mathcal{A}, quant_\mathcal{A}, S_\mathcal{A}, \delta_\mathcal{A}\rangle$ and $\mathcal{B} = \langle U_\mathcal{B}, Q_\mathcal{B}, inf_\mathcal{B}, quant_\mathcal{B}, S_\mathcal{B}, \delta_\mathcal{B}\rangle$. Without loss of generality, we assume that $Q_\mathcal{A} \cap Q_\mathcal{B} = \varnothing$. In all cases, we construct a new automaton $\mathcal{C} = \langle Q_\mathcal{C}, inf_\mathcal{C}, quant_\mathcal{C}, S_\mathcal{C}, \delta_\mathcal{C}\rangle$.

Union. The disjoint union of the alternating automata $\mathcal{A}$ and $\mathcal{B}$ accepts the union of $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

Intersection. Assume, without loss of generality, that $\mathcal{A}$ and $\mathcal{B}$ are of the form proposed in Proposition 1. We construct the alternating automaton $\mathcal{C}$ by taking the disjoint union of $\mathcal{A}$ and $\mathcal{B}$ and adding a new (universal) start state s_K for each interface object K and id_K-labeled transitions from s_K to the start states of $\mathcal{A}$ and $\mathcal{B}$ with the same interface. Formally, we take

$$S_\mathcal{C} := \{s_K \mid K \in U_\mathcal{A} \cap U_\mathcal{B}\} \quad \text{and} \quad Q_\mathcal{C} := Q_\mathcal{A} \cup Q_\mathcal{B} \cup S_\mathcal{C}$$

where the states of $S_\mathcal{C}$ are assumed to be new, that is $S_\mathcal{C} \cap (Q_\mathcal{A} \cup Q_\mathcal{B}) = \varnothing$. We define

$$inf_\mathcal{C}(q) := \begin{cases} inf_\mathcal{A}(q) & \text{if } q \in Q_\mathcal{A} \\ inf_\mathcal{B}(q) & \text{if } q \in Q_\mathcal{B} \\ K & \text{if } q = s_K \in S_\mathcal{C} \end{cases} \quad \text{and} \quad quant_\mathcal{C}(q) := \begin{cases} quant_\mathcal{A}(q) & \text{if } q \in Q_\mathcal{A} \\ quant_\mathcal{B}(q) & \text{if } q \in Q_\mathcal{B} \\ \forall & \text{if } q \in S_\mathcal{C}. \end{cases}$$

Finally, the transition relation $\delta_{\mathcal{C}}$ is defined as

$$\begin{aligned}\delta_{\mathcal{C}} := \delta_{\mathcal{A}} \cup \delta_{\mathcal{B}} \cup \\ \{\langle s_K, id_K, s\rangle \mid s \in S_{\mathcal{A}} \wedge inf_{\mathcal{A}}(s) = K\} \cup \\ \{\langle s_K, id_K, s\rangle \mid s \in S_{\mathcal{B}} \wedge inf_{\mathcal{B}}(s) = K\}\end{aligned}$$

The new automaton trivially satisfies the cycle condition, because it contains no cycles which were not also in either $\mathcal{A}$ or $\mathcal{B}$. Also, it is now easily seen that $\mathrm{L}(\mathcal{C}) = \mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B})$.

Complement. Assume, without loss of generality, that $\mathcal{A}$ and $\mathcal{B}$ are of the form proposed in Proposition 1. We build an automaton which accepts the complement of $\mathrm{L}(\mathcal{A})$ by making all existential states universal and vice versa. That is, $Q_{\mathcal{C}} := Q_{\mathcal{A}}$, $inf_{\mathcal{C}} := inf_{\mathcal{A}}$, $S_{\mathcal{C}} := S_{\mathcal{A}}$, $\delta_{\mathcal{C}} := \delta_{\mathcal{A}}$. Furthermore,

$$quant_{\mathcal{C}}(q) = \begin{cases} \forall & \text{if } quant_{\mathcal{A}}(q) = \exists \\ \exists & \text{if } quant_{\mathcal{A}}(q) = \forall. \end{cases}$$

Now we show, by induction, that for any **C**-arrow c and state $q \in Q_{\mathcal{A}}$ it holds, that $\mathcal{A}, q \models c$ if and only if $\mathcal{C}, q \not\models c$. The required result follows then from the fact that this holds in particular for the start states, and that the start states are unique for each interface object.

($\Rightarrow$): Assume $\mathcal{A}, q \models c$. We proceed by induction on q and c. There are two cases to consider.

If $quant_{\mathcal{A}}(q) = \exists$, then, by definition, there are a state q' and arrows f, c' such that $c = f;c'$, $\langle q, f, q'\rangle \in \delta_{\mathcal{A}}$ and $\mathcal{A}, q' \models c'$. By the induction hypothesis $\mathcal{C}, q' \not\models c'$. However, this means that is not the case that it holds for all states q' and arrows f, c' such that $c = f;c'$ and $\langle q, f, q'\rangle \in \delta_{\mathcal{A}}$ that $\mathcal{C}, q' \models c'$, and thus $\mathcal{C}, q \not\models c$.

If $quant_{\mathcal{A}}(q) = \forall$, then, by definition, for all states q' and all arrows f, c' such that $c = f;c'$ and $\langle q, f, q'\rangle \in \delta_{\mathcal{A}}$ it holds that $\mathcal{A}, q' \models c'$. By the induction hypothesis, $\mathcal{C}, q' \not\models c'$ for all q', f, c' satisfying the conditions. But this means, that there are no q', f, c' satisfying the conditions such that $\mathcal{C}, q' \models c'$, and therefore $\mathcal{C}, q \not\models c$.

($\Leftarrow$): Since the construction is symmetric, this case follows from ($\Rightarrow$). □

4 Alternating Graph Automata

In this section we instantiate alternating automata from Definition 2 to the category of cospans of graphs (see Section 2). We will restrict our attention to cospans which have discrete interfaces (that is the interfaces of the cospans consist of nodes only) and injective right morphisms.

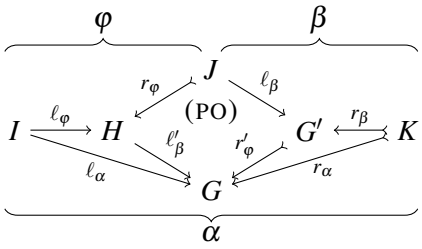
4.1 Definition of Alternating Graph Automata

In order to instantiate the definition, we need to define a progressing (see Definition 1) relation. In the rest of the paper, we use the following one: Call a cospan $\varphi\colon J \xrightarrow{\ell_\varphi} G \xleftarrow{r_\varphi} K$ *cancellable* if r_φ is surjective (since we restrict to discrete interfaces, this means that G does not contain edges either). We now define $\alpha \prec \beta$ if $\beta = \varphi;\alpha$ for some non-cancellable cospan φ.

Proposition 2 *The order $\prec$ on cospans of graphs is progressing on cospans with injective right morphisms.*

Proof. By definition, $\prec$ is a sub-order of $\leq$. It remains to show that for each infinite descending sequence $\gamma_1 \geq \gamma_2 \geq \cdots$ there are only finitely many indices i where $\gamma_i \succ \gamma_{i+1}$.

Let α and β be cospans such that $\beta \prec \alpha$, that is there is a cospan φ such that $\alpha = \varphi;\beta$. The situation is depicted below:



Because the r_φ is injective and the middle square of the diagram is a pushout, r'_φ is injective. Thus, the number of nodes in G' is less or equal than the number of nodes in G, and the number of edges in G' is less or equal than the number of edges in G. However, if φ is non-cancellable, then r_φ is not surjective, and thus, because the middle square is a pushout, r'_φ is not surjective. This means that either the number of edges or the number of nodes strictly decreases. This shows that $\prec$ is progressing. □

Note that, by construction and Definition 1, the non-cancellable cospans are exactly the $\prec$-productive ones.

Definition 4

(i) An *alternating graph automaton* is an alternating $(\mathrm{Cospan}(\mathbf{Graph}), \prec)$-automaton which has only discrete graphs as interfaces, and for each transition $\langle q, \varphi, q' \rangle$, where $\varphi = J -\ell_\varphi\rightarrow G \leftarrow r_\varphi - K$, it holds that r_φ is injective.

(ii) The *graph language* of an alternating graph automaton $\mathcal{A}$ is defined as:

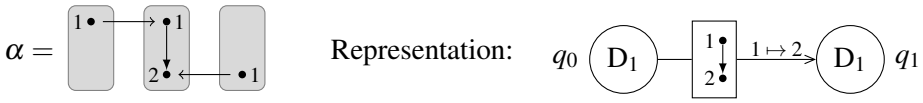
$$\mathrm{G}(\mathcal{A}) = \{H \mid [H] \in \mathrm{L}(\mathcal{A})\}$$

Note that, for an alternating graph automaton $\mathcal{A}$, there is a difference between $\mathrm{L}(\mathcal{A})$ and $\mathrm{G}(\mathcal{A})$: the first contains cospans while the second contains graphs.

Paths through alternating graph automata correspond to cospan decompositions of the graph. Although alternating graph languages are not bounded by path width (for example, the language of all graphs is accepted by an alternating automaton consisting of a single universally quantified initial state without successors), the fact that each automaton has only finitely many interfaces implies that the part that the automaton actually "looked at" in each state has a bounded path width. How the interface size of an alternating graph automaton relates exactly to a path width bound of the accepted language, remains a topic for further research.

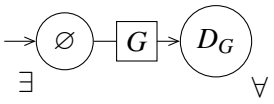
4.2 Examples

Below, we give some examples of automata which accept various graph languages. We represent the automata graphically. A state is depicted as a circle with its interface written inside it and its quantifier written beside it. As interfaces we always take a discrete graph with node set $\{1,\ldots,n\}$, called D_n. The transitions are given as arrows, labeled with cospans. The cospans are given by drawing its central graph; the two graph morphisms are implied by the node labels. If (and only if) it is not the case that the nth interface node maps to the nth central graph node, an explicit mapping is given. That is, a transition from state q_0 to q_1 labeled by the cospan α given below, will be represented as follows:



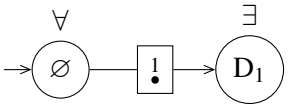
Note that the $1 \mapsto 2$ label means, that node 1 of the interface of q_1 is mapped to node 2 of the center graph of the cospan; this is the direction of the right morphism of the cospan, *not* the direction of the transition in the automaton.

Example 1 (Subgraph automaton) *Let G be a graph (over an arbitrary signature). The alternating graph automaton $\mathcal{SG}_G$ which accepts all cospans $\varnothing \to H \leftarrow K$ such that H contains G as a subgraph is:*



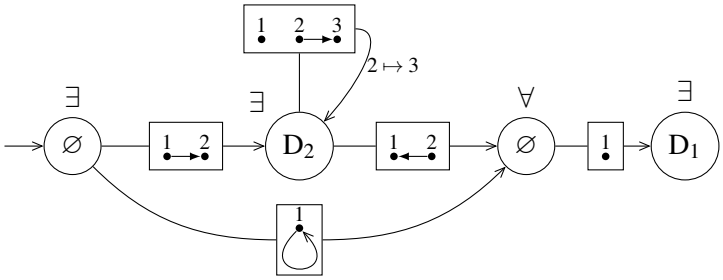
where D_G is the discrete graph with the same node set as G.

Example 2 (Empty graph) *Assume the signature contains no* 0*-ary labels. The following alternating graph automaton $\mathcal{E}$ accepts only the empty graph:*



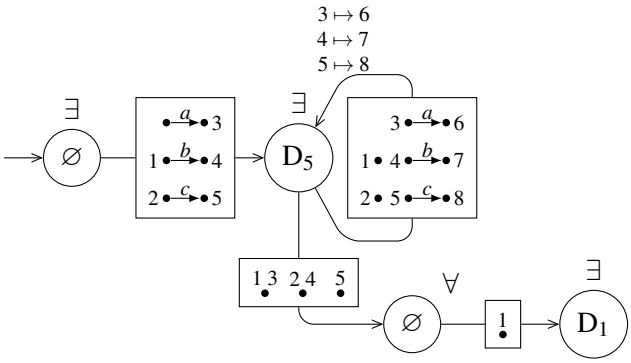
The automaton works as follows: if the target graph (with empty interfaces) can be decomposed into a cospan which has the effect of adding a node and another cospan, then the automaton moves to an existentially quantified state without outgoing transitions, which means that the graph will not be accepted. If not, the acceptance condition for the universally quantified initial state is trivially satisfied.

Example 3 (Circle graphs) *Let Σ contain a single binary edge label. The following alternating automaton accepts* circle graphs*, that is graphs whose edges form a single circle.*

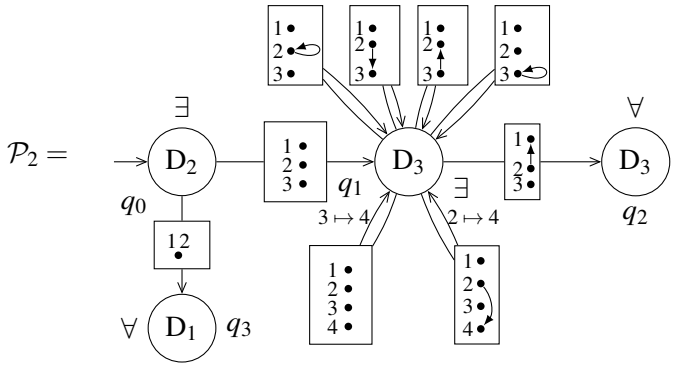


The automaton works as follows. In the right we see the automaton from Example 2 as a subautomaton. It is there to make sure that in the end the entire graph is processed. In the initial state, there are two cases. Either there is a loop, in which the case the automaton immediately checks whether the entire graph was processed, or there is an edge between two different nodes which needs to be completed to a full circle: in each traversal of the loop the circle is extended by one edge, until an edge is found to the starting node, in which case the circle is completed.

Example 4 ($a^n b^n c^n$) *To show that alternating automata can accept non-recognizable graph languages, we give an alternating automaton which accepts string graphs of the form $a^n b^n c^n$, where $n \geq 1$ (cf. Example I.3.6 of [9]).*



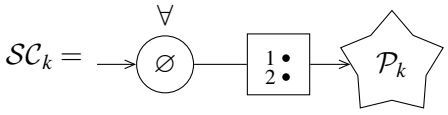
Example 5 (Reachability) *Let G be a graph of path width smaller than k, and let $f\colon \mathrm{D}_2 \to G$ be a graph morphism which picks two nodes of G. The alternating graph automaton $\mathcal{P}_k$ accepts the cospan $\mathrm{D}_2 \xrightarrow{f} G \xleftarrow{} \varnothing$ if and only if there is a path from $f(2)$ to $f(1)$ in G. We present the automaton for $k = 2$:*



The automaton works as follows. First, in state q_0, if the two nodes of the interface are the same, the automaton may move directly to the accepting state q_3. Otherwise, an auxiliary node is taken from the graph non-deterministically, and the automaton moves to q_1. There, the path is searched. At all times, node 1 of the interface points to the target of the path, node 2 is the current end of the path, and node 3 is an auxiliary node. This auxiliary node is needed to read in parts of the graph which are not part of the path.

The four loops above state q_1 read in edges which do not lie on the path. The bottom left loop replaces the auxiliary node in the interface by a new node. The bottom right loop extends the path by a single edge. Finally, if the path can be finished the automaton moves to the accepting state q_2.

The path automaton can be easily extended to larger k by including more auxiliary nodes (and adding more transitions). It can be used as a component in other alternating automata; for example, the following automaton expresses that a graph is strongly connected (works only for graph up to path width k):



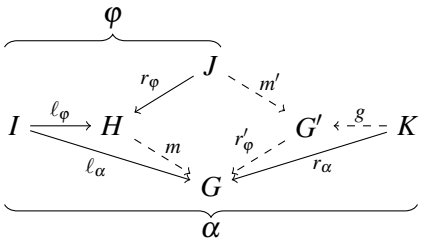
4.3 Membership Problem

The membership problem is defined as follows: given an alternating graph automaton $\mathcal{A}$ and a cospan $\alpha\colon J \to G \leftarrow K$, does it hold that $\alpha \in \mathrm{L}(\mathcal{A})$? In this section we present some insights into a naive algorithm to decide the membership problem for alternating graph automata.

An important step in solving the membership problem is computing cospan complements, that is, given cospans $\varphi\colon I \xrightarrow{\ell_\varphi} G \xleftarrow{r_\varphi} J$ and $\alpha\colon I \xrightarrow{\ell_\alpha} H \xleftarrow{r_\alpha} K$, finding all cospans $\alpha'\colon J \to H' \leftarrow K$ such that $\varphi\,;\alpha' = \alpha$.

First of all, it is clear that the membership problem is decidable if computing cospan complements is possible. Because of the condition that all loops in the automaton contain at least one non-cancellable cospan, it is not possible that the automaton gets caught in one loop indefinitely: at some point the cospan which has been built by composing the cospans on the transitions previously in the path cannot be extended anymore to the sought-after cospan. Thus, we can traverse the automaton recursively, and test the acceptance condition.

Finally, computing cospan complements is possible because computing pushout complements is. Given cospans φ and α as above, we first find all matches $m\colon H \to G$ such that $\ell_\alpha = \ell_\varphi\,;m$. For each such match, we subsequently find all pushout complements G' of r_φ and m (see the diagram to the right). Finally, for each pushout complement we check if there exists a g such that $g\,;r'_\varphi = r_\alpha$, and add the cospan $\alpha'\colon J \xrightarrow{m'} G' \xleftarrow{g} K$ to the set of cospan complements if it exists. Note that, since we restrict to cospans with injective right morphisms, there is at most one cospan complement per match m, because there is at most one pushout complement per match m [8].



5 Comparison

In this subsection we compare alternating graph automata with automaton functors and hyperedge replacement systems with respect to expressive power. First we compare alternating automata with hyperedge replacement grammars.

Definition 5 Assume the label set Σ is partitioned into a set N of *non-terminals* and a set T of *terminals*. A handle of a non-terminal $A \in N$ is a cospan $\gamma_A : \varnothing \xrightarrow{\ell} E_A \xleftarrow{r} J$, where J is a discrete graph, $E_A = \langle V, E, att, lab \rangle$ is a graph where $|V| = ar(A)$, $E = \{e\}$, $lab(e) = A$, the elements of $att(e)$ are pairwise distinct, and the morphism r is injective and surjective on nodes.

A *hyperedge replacement grammar* (HRG) is a set of rules in which the left-hand side is a handle, together with a start symbol $S \in N$. A HRG is *linear*, if all rules contain at most one non-terminal edge in the right-hand side.

The language generated by an HRG $\mathcal{G}$ is defined as the reachable graphs containing only terminals: $\mathrm{L}(\mathcal{G}) = \{G = \langle V, E, att, lab \rangle \mid E_S \Rightarrow^* G \text{ and for all } e \in E : lab(e) \in T\}$.[2]

To show that alternating graph automata are more expressive than linear HRGs, we need the following auxiliary lemma:

Lemma 1 *For fixed J and K, the number of cancellable cospans $\varphi : J \xrightarrow{\ell_\varphi} G \xleftarrow{r_\varphi} K$ is finite.*

Proof. Since r_φ must be surjective, there are finitely many possible G (up to isomorphism). The number of graph morphisms from one finite graph (J) into another (G) is finite, so for each of the finitely many possible G there are only finitely many possible ℓ_φ. □

Theorem 2 *The class of languages generated by linear HRGs is a strict subclass of the class of languages accepted by alternating graph automata.*

Proof. The fact that the two language classes are not equal follows from the observation that the language of all graphs is accepted by an alternating graph automaton (take the automaton consisting of a single state, which is initial and universally quantified, and an empty transition relation), but not generated by an HRG.

It remains to show that each language generated by an HRG is also accepted by an alternating graph automaton. For a graph G and hyperedge e of G, let $d(G,e) : \mathrm{D}_{|att(e)|} \to G$ be a morphism whose image consists of the nodes adjacent to e. Furthermore, let $d'(G,e)$ be the corresponding morphism from $\mathrm{D}_{|att(e)|}$ to G', where G' is G without e.

Let $p = L \to R$ be a linear hyperedge replacement rule. Let e_l be the non-terminal hyperedge from L and let e_r be the non-terminal hyperedge from R, if there is one. (Since p is assumed to be linear, there is at most one.) Let R' be R without e_r. The rule p induces a morphism $\ell : \mathrm{D}_{|att(e_l)|} \to R'$. The rule p can now be translated into the cospan $\varphi_p = \langle \ell, r \rangle$, where r is $d'(R, e_r)$ if R contains a non-terminal, or the unique morphism $\varnothing \to R'$ otherwise.

For a linear HRG with non-terminals V, rule set R and initial non-terminal s, we can now construct an alternating pseudo-automaton $\mathcal{A}$ with state set $S = V \cup \{\varnothing\}$. (We call it pseudo-

[2] See [9] for a different (but equivalent with respect to expressive power) definition of HRGs.

automaton, because it may contain cycles without productive cospans.) The pseudo-automaton has a transition from state x to state y, whenever there is a rule p with x in the left-hand side and y in the right-hand side. Each such transition is labeled with the cospan φ_p. For rules with no non-terminal on the right-hand side, the transitions have the state $\varnothing$ as target. The initial state is s. In addition, we add the automaton from example 2 and fuse its initial state with the state $\varnothing$. All states but $\varnothing$ are labeled with the $\exists$ quantifier. Now each derivation of a graph in $\mathcal{P}$ corresponds to a path $\mathcal{A}$ from s to $\varnothing$ and vice versa.

Call a path through $\mathcal{A}$ cancellable if it contains only transitions labeled with cancellable cospans. From the pseudo-automaton $\mathcal{A}$ we build an alternating automaton $\mathcal{B}$ which satisfies the condition that each loop must contain at least one productive cospan. The automaton $\mathcal{B}$ contains the same states as $\mathcal{A}$. Furthermore, for each transition $v \xrightarrow{\varphi} v'$ of $\mathcal{A}$ and each cancellable path $\vec{x} = x_1 \xrightarrow{\psi_1} x_2 \cdots x_{n-1} \xrightarrow{\psi_{n-1}} x_n$ from an arbitrary state x_1 to v (that is, $x_n = v$), a transition labeled with $\psi_1 ; \cdots ; \psi_{n-1} ; \varphi$ from x_1 to w is added to $\mathcal{B}$. Note that there are possibly infinitely many paths (because of loops), however Lemma 1 ensures that only finitely many transitions will be added to $\mathcal{B}$. Finally, for each cancellable path $\vec{x} = x_1 \xrightarrow{\psi_1} x_2 \cdots x_{n-1} \xrightarrow{\psi_{n-1}} x_n$ from some state x_1 to $\varnothing$, a transition labeled with $\psi_1 ; \cdots ; \psi_{n-1}$ from x_1 to $\varnothing$ is added to $\mathcal{B}$. By this construction, the only transitions labeled with cancellable cospans lead to $\varnothing$, which lies on no cycle. Also, for each path in $\mathcal{A}$ from s to $\varnothing$ there is an equivalent path in $\mathcal{B}$ and vice versa. □

Non-linear HRGs are strictly stronger than linear ones. Still, they are not more expressive than alternating graph automata, because they cannot generate the language of all graphs. Because of the following conjecture, it is probably also not the case that alternating graph automata are stronger than HRGs.

Conjecture 1. *There does not exist an alternating graph automaton which accepts the language of binary trees.*

The intuition behind the conjecture is, that the part of a graph an alternating graph automaton looks at has a bounded path width. To recognize a tree, one must process the entire graph, and trees have an unbounded path width. We need to formalize these ideas in order to prove the conjecture, which is future work.

Now we compare alternating automata with automaton functors, which were introduced in [5].

Definition 6 (Automaton functor) A graph automaton functor is a structure $\mathcal{A} = \langle \mathcal{A}_0, I, F \rangle$, where

- $\mathcal{A}_0 \colon \mathrm{Cospan}(\mathbf{Graph}) \to \mathbf{Rel}$ is a functor which maps every (discrete) graph J to a finite set $\mathcal{A}_0(J)$ (the *state set of* J) and every cospan $\alpha \colon J \to G \leftarrow K$ to a relation $\mathcal{A}_0(\alpha) \subseteq \mathcal{A}_0(G) \times \mathcal{A}_0(H)$ (the *transition function* of c),
- $I \subseteq \mathcal{A}_0(\varnothing)$ is the set of initial states and
- $F \subseteq \mathcal{A}_0(\varnothing)$ is the set of final states.

A cospan $\alpha \colon \varnothing \to G \leftarrow \varnothing$ is accepted by $\mathcal{A}$, if $\langle q, q' \rangle \in \mathcal{A}_0(\alpha)$ for some $q \in I$ and $q' \in F$. The language accepted by $\mathcal{A}$, denoted by $\mathrm{L}(\mathcal{A})$, contains exactly the cospans accepted by $\mathcal{A}$. The

graph language accepted by $\mathcal{A}$ is defined as

$$\mathrm{G}(\mathcal{A}) = \{G \mid [G] \in \mathrm{L}(\mathcal{A})\}.$$

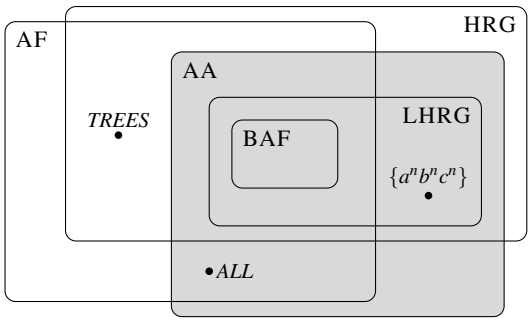
A *bounded automaton functor* is an automaton functor on the subcategory of Cospan(**Graph**) which contain only the (discrete) graphs of size $\leq k$ as object, where $k \in \mathbb{N}$ is the bound of the automaton.

Corollary 1 *The class of languages accepted by bounded automaton functors is a strict subclass of the class of languages accepted by alternating graph automata.*

Proof. In [4] it was shown, that the class of languages accepted by bounded automaton functor is a strict subset of the class of languages generated by linear HRGs. The proposition follows by combining this result with Theorem 2. □

General automaton functors, the language class of which are the recognizable graph languages, are not strictly more expressive than alternating graph automata. In particular, automaton functors cannot "count", so there exists no automaton functor which accepts string graphs of the form $a^n b^n c^n$. There exists an alternating graph automaton which accepts this language, however (see Example 4). Conversely, Conjecture 1 implies that alternating graph automata are not strictly more expressive than general automaton functors, either, because binary trees can be expressed in monadic second-order logic, and therefore form a recognizable graph language.

To sum up the results of this section, the expressive power of alternating graph automata (denoted by AA) is compared to that of linear and general HRGs (denoted by LHRG and HRG, respectively) and bounded and general automaton functors (denoted by BAF and AF, respectively) are represented in the Venn diagram on the right; additionally the position of three languages within the diagram is shown. Note that the fact that trees are not accepted by an alternating graph automaton is only a conjecture.



6 Conclusion and Further Research

We defined alternating automata that operate on arrows of an arbitrary category, and as a special case obtained alternating graph automata. We have shown by example that alternating graph automata can elegantly express various graph properties, and have compared their expressive power to other mechanisms for describing graph languages. It turns out that they are strictly more expressive than bounded automaton functors and linear hyperedge replacement grammars, while we conjecture that they are incomparable with respect language class inclusion with general hyperedge replacement grammars and recognizable graph languages.

A first direction for further research, is to generalize the results to larger classes of cospans (for example, cospans with non-injective right morphisms or non-discrete interfaces). Also, we would

like a clearer understanding of the exact relationship between path width and graphs which are accepted by alternating graph automata, and techniques for showing that an alternating graph automaton which accepts a certain language does not exist.

Second, on a more applied level, we need to study more decision problems. Since alternating automata contain conditions as a special case, which are more expressive than first-order logic [3], the emptiness problem (deciding whether or not the language of an alternating automaton is empty) is not decidable. It is an open problem if the emptiness problem is also undecidable for graphs of bounded path width. Note that, because complementation can be done in linear time, the emptiness and universality problems are equivalent with respect of computational complexity.

Third, as mentioned in Section 3, alternating automata can be seen as the conditions of [3] with loops, which have the same expressive power as the first-order fragment of the logic on subobjects [6]. It is interesting to see if there exists a logic which describes the same languages as alternating graph automata, for example a first-order logic with a fixed-point operator.

Bibliography

[1] C. Blume, H. J. S. Bruggink, M. Friedrich, and B. König. Treewidth, pathwidth and cospan decompositions. In *Proceedings of GT-VMT 2011*, 2011.

[2] C. Blume, H. J. S. Bruggink, and B. König. Recognizable graph languages for checking invariants. In *Proc. of GT-VMT '10*, Electronic Communications of the EASST, 2010.

[3] H. J. S. Bruggink, R. Chauderlier, M. Hülsbusch, and B. König. Conditional reactive systems. In *Proceedings of FSTTCS 2011*, 2011.

[4] H. J. S. Bruggink and M. Hülsbusch. Decidability and expressiveness of finitely representable recognizable graph languages. In *Proceedings of GT-VMT 2011*, 2011.

[5] H. J. S. Bruggink and B. König. On the recognizability of arrow and graph languages. In *Proceedings of ICGT '08*, 2008.

[6] H. J. S. Bruggink and B. König. A logic on subobjects and recognizability. In *Proceedings of IFIP-TCS '10*, volume 323 of *IFIP-AICT*, pages 197–212. Springer, 2010.

[7] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 21(1), 1981.

[8] H. Ehrig. Introduction to the algebraic theory of graph grammars. In *Proceedigns of the 1st International Workshop on Graph Grammars and Their Applications to Computer Science and Biology*, 1979.

[9] A. Habel. *Hyperedge Replacement: Grammars and Languages*. Springer, 1992.

[10] A. Habel and K.-H. Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Stuctures in Computer Science*, 19:245–296, 2009.

[11] V. Sassone and P. Sobociński. Reactive systems over cospans. In *Proceedings of LICS 2005*, 2005.