Math. Struct. in Comp. Science (2018), *vol.* 28, *pp.* 1290–1332. © Cambridge University Press 2018
doi:10.1017/S096012951800018X

Recognizable languages of arrows and cospans

H. J. SANDER BRUGGINK and BARBARA KÖNIG

Universität Duisburg-Essen, Duisburg, Germany
Email: barbara_koenig@uni-due.de

Received 1 March 2012; revised 1 November 2017

In this article, we generalize Courcelle's recognizable graph languages and results on monadic second-order logic to more general structures.
First, we give a category-theoretical characterization of recognizability. A recognizable subset of arrows in a category is defined via a functor into the category of relations on finite sets. This can be seen as a straightforward generalization of finite automata. We show that our notion corresponds to recognizable graph languages if we apply the theory to the category of cospans of graphs.
In the second part of the paper, we introduce a simple logic that allows to quantify over the subobjects of a categorical object. Again, we show that, for the category of graphs, this logic is equally expressive as monadic second-order graph logic (MSOGL). Furthermore, we show that in the more general setting of hereditary pushout categories, a class of categories closely related to adhesive categories, we can recover Courcelle's result that every MSOGL-expressible property is recognizable. This is done by giving an inductive translation of formulas of our logic into automaton functors.

1. Introduction

Regular languages have been studied extensively in computer science and have a large number of applications, such as model checking (Bouajjani et al. 2000) and termination analysis (Geser et al. 2004). The notions of regularity and finite automata can be straightforwardly generalized to trees and tree automata, opening the possibility to define regular tree languages and exploit the convenient closure properties that these languages enjoy. In recent years, the success of regular languages has sparked interest in obtaining a similar notion for other classes of objects, in particular graphs. Courcelle (1990) and Courcelle and Engelfriet (2012) focused on the notion of recognizability – which is equivalent to regularity in the case of word languages – in an algebra of graphs with interfaces. It turns out that recognizable graph languages in this setting can be characterized by locally finite congruences. Bozapalidis and Kalampakas (2006) explored a similar characterization based on magmoids.

In a seminal result, Courcelle showed that every graph language definable in monadic second-order graph logic (MSOGL) is recognizable. This means that one can obtain finite automata to accept path decompositions of graphs and tree automata to accept tree decompositions of paths. This has interesting complexity-theoretic consequences in that the membership problem for any MSOGL-expressible language can be decided in linear time when restricted to graphs of bounded treewidth. Some NP-complete problems (such as graph colorability) are definable in MSOGL.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

In this paper, we will show that the results on recognizable graph languages can be generalized to a categorical setting where we shift our attention from graphs to objects in categories which are closely related to adhesive categories (Lack and Sobociński 2005). This implies that the theory holds for all kinds of graph-like structures, such as directed graphs, hypergraphs, graphs with higher order edges, graphs with scopes, bigraphs, etc. Furthermore, we believe that the greater generality leads to elegant constructions, such as the construction of automata from monadic second-order formulas.

This paper contains the following contributions:

Comparison of two notions of recognizability and introduction of automaton functors. We compare the algebraic notion of recognizability by Courcelle with a categorical notion of recognizability, strongly related to composition-representative subsets introduced by Griffing (2003).

We show that if we use the notion of Griffing and work in the category of cospans of graphs, we recover exactly the notion of recognizability proposed by Courcelle. Although both approaches rely on the same basic ideas, the proof is non-trivial. Furthermore, it gives us a close relation with the double-pushout (DPO) approach which can be characterized as a reactive system over cospans (Sassone and Sobociński 2005).

In addition, we extend the notion of Griffing with a so-called automaton functor that – when instantiated to the word case – specializes to the notion of deterministic or non-deterministic finite automaton. We show that standard constructions on finite automata such as determinization or minimization can also performed on automaton functors.

Introduction of a logic on subobjects and comparison to MSOGL. We develop a logic, similar to MSOGL, which describes properties of objects in a category. We show that in the category of graphs our logic has the same expressive power as MSOGL. Hence, we generalize MSOGL to arbitrary categories.

Recognizability of languages defined in the logic of subobjects. We recover, in a general setting, the well-known result by Courcelle (Bouderon and Courcelle 1987) that every graph language definable in monadic second-order logic is recognizable. More specifically, we prove, for hereditary pushout categories (HPCs) with some mild additional conditions, that each language characterized by a formula in the logic on subobjects is recognizable. We do this by giving an inductive construction (on the structure of the formula). We believe that this construction is quite elegant compared to other known constructions, such as Courcelle (1990), Flum et al. (2002) and Courcelle and Engelfriet (2012). It allows to construct correct automata directly from the formulas, without defining them and proving their correctness manually, which can easily become quite tedious.

In summary, we are giving a more abstract view on the results concerning recognizability where we show that these results are not specific to graphs, but can be seen in a more general categorical setting. This follows a trend in graph rewriting, initiated – among other papers – by Lack and Sobociński (2005), which shows that many results on graphs can be generalized. So far, this has mainly been used in the context of graph transformation (Ehrig et al. 2006) and the derivation of behavioural equivalences (Ehrig and König 2006; Sassone and Sobociński 2005), but here we show that the setting of adhesive categories (here used in a slight variant), which requires that certain gluing and intersection operations on objects are well-behaved, is a suitable framework to recover

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

results on recognizability and logics, which also rely on composition and decomposition techniques for graphs.

The paper is based on two articles: 'On the Recognizability of Arrow and Graph Languages,' which appeared in the proceedings of ICGT '08 (Bruggink and König 2008) and 'A Logic on Subobjects and Recognizability,' which appeared in the proceedings of IFIP-TCS '10 (Bruggink and König 2010).

We require some basic knowledge of category theory (pushouts and pullbacks).

The rest of the paper is structured as follows: In Section 2, we give some category-theoretical preliminaries and introduce the category of graphs and graph morphisms. Then, in Section 3, we consider recognizable languages for the general case of arrow languages, which specialize to recognizable graph languages in Section 4. There, we also make an extensive comparison to the notion of recognizability by Courcelle. Subsequently, we introduce our logic on subobjects in Section 5 and compare it to MSOGL in Section 6. Finally, we prove a generalization of Courcelle's result that every language definable in monadic second-order logic is recognizable in Section 7 and we conclude (Section 8).

2. Preliminaries

We briefly review the concepts from category theory, graphs, and graph morphisms that will play a role in this paper. For more detailed introductions, we refer to Mac Lane (1971) and Ehrig et al. (2006).

First, we introduce some notational conventions. When A is a set, A^* denotes the set of finite sequences of elements of A. If $f : A \to B$ is a function, then $f^* : A^* \to B^*$ is the natural extension of f to sequences, that is $f^*(a_1 \dots a_n) = f(a_1) \dots f(a_n)$. By $\mathbb{N}_k$, we denote the prefix of $\mathbb{N}$ of length k, that is $\mathbb{N}_k = \{0, 1, \dots, k-1\}$.

Let $f : A \to B$ be a function and let a, b be two elements, which are not necessarily contained in A or B. Then, $f[a \mapsto b] : A \cup \{a\} \to B \cup \{b\}$ denotes the function defined as follows:

$$f[a \mapsto b](a') = \begin{cases} b & \text{if } a' = a \\ f(a') & \text{otherwise.} \end{cases}$$

2.1. *Category theory*

We presuppose basic knowledge of category theory. Given a category $\mathbf{C}$, the objects of $\mathbf{C}$ are denoted by $Obj(\mathbf{C})$ and the arrows of $\mathbf{C}$ by $Arr(\mathbf{C})$. The identity arrow of an object G will be denoted by id_G (or just by id if G is clear from the context). For a morphism $f : A \to B$, we denote by $dom(f) = A$ the domain and by $cod(f) = B$ the codomain of f. When f and g are (composable) morphisms, we write $f ; g$ for the morphism f postcomposed with g, that is, $f ; g = g \circ f$.

A *monomorphism*, or *mono* for short, is a morphism $f : B \to C$ such that for all morphisms $g_1, g_2 : A \to B$, it holds that $g_1 ; f = g_2 ; f$ implies $g_1 = g_2$. In many categories, among others the category **Graph** presented in Section 2.2, monos correspond to injective morphisms.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Let $f : A \rightarrowtail T$ and $g : B \rightarrowtail T$ be monos with the same codomain T. We write $f \sqsubseteq g$ if there exists an arrow $h : A \to B$ (which is necessarily unique) such that $h ; g = f$. The subobject lattice $Sub(T)$ of an object T is formed by isomorphism classes of monos with codomain T, where $\sqsubseteq$ forms the inclusion order. (In practice, we will take representatives of the isomorphism classes.)

The category **Rel** is the category which has sets as objects, relations as arrows and relation composition as composition operator. The category **Set** is the subcategory in which all arrows are in fact total functions. We mention here that a functor $\mathcal{A}$ from some category **C** into **Set** is easily extended to a functor from **C** to **Rel** by postcomposing it with the embedding functor.

Let **C** be a category. A concrete cospan is a pair of **C**-arrows $\langle c_L, c_R \rangle$ with the same codomain: $c : J \xrightarrow{c_L} G \xleftarrow{c_R} K$. Here, J and K are called the inner and outer interfaces of c, respectively, and G is called the centre object. We define the equivalence relation $\sim$ on concrete cospans as follows: $c \sim d$ if there exists an isomorphism k from G to H, such that $c_L ; k = d_L$ and $c_R ; k = d_R$. We define a cospan to be a $\sim$-equivalence class of concrete cospans. Note that $\sim$-equivalent cospans have isomorphic centre objects, but identical inner and outer interfaces. Cospans will be represented by representatives of the equivalence class, and the inner and outer interface of the cospan are the inner and outer interface of the concrete cospans it consists of.

If **C** is a category in which all pushouts exist, we can form the category $Cospan(\mathbf{C})$ as follows. The objects are the objects of **C** and the arrows from J to K are the cospans with inner interface J and outer interface K. The identity for an object G is given by the (equivalence class of) the concrete cospan consisting of twice the identity arrow of G: $\langle \mathrm{id}_G, \mathrm{id}_G \rangle : G \to G \leftarrow G$. Let $c : J \to G \leftarrow M$ and $d : M \to H \leftarrow K$ be (representatives of) cospans (where the codomain of c equals the domain of d): The composition of c and d is defined by the commuting diagram

$$\begin{array}{ccccccc} & & & & M & & & & \\ & & & {\scriptstyle c_R}\swarrow & & \searrow{\scriptstyle d_L} & & & \\ J & \xrightarrow{c_L} & G & & (\mathrm{PO}) & & H & \xleftarrow{d_R} & K \\ & & & {\scriptstyle f}\searrow & & \swarrow{\scriptstyle g} & & & \\ & & & & M' & & & & \end{array}$$

where the middle diamond is a pushout, and the resulting composed cospan is defined as

$$(c ; d) : J \xrightarrow{c_L ; f} M' \xleftarrow{d_R ; g} K.$$

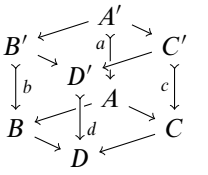
Note that taking the pushout is unique up to isomorphism, so because cospans are $\sim$-equivalence classes, this is well-defined.

In the last part of the paper, we will restrict our attention to the so-called HPCs introduced by Heindel (2009), which are related to the well-known adhesive categories (Lack and Sobociński 2005). Most adhesive categories are also HPC, including all topoi. In particular, the categories **Set** of sets and **Graph** of hypergraphs (see Section 2.2) are HPC. A category **C** is an HPC if

i. **C** has pushouts along monos;
ii. **C** has all pullbacks;

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

iii. given a cube diagram as shown on the right, where a, b and c are monos, the bottom face is a pushout and the back faces are pullbacks, we have that the top face is a pushout if and only if the front faces are pullbacks and d is a mono.



Differently from adhesive categories, the vertical arrows in the cube must be mono, instead of one of the arrows in the lower square.

2.2. *Graphs and graph morphisms*

Let a set Σ of (ranked) labels be given. Each label $\ell \in \Sigma$ is supplied with an arity $ar(\ell) \in \mathbb{N}$. A *hypergraph* G (in the following also called a *graph*) is a structure $\langle V_G, E_G, att_G, lab_G \rangle$, where V_G is the set of nodes of G, E_G is the set of edges of G, $att_G : E_G \to V_G^*$ is the *attachment function* and $lab_G : E_G \to \Sigma$ is the *labelling function*, such that $|att_G(e)| = ar(lab_G(e))$ for all $e \in E_G$. The empty graph is denoted by $\emptyset$. A *graph morphism* f from G to H is a pair $\langle f_V, f_E \rangle$ of functions $f_V : V_G \to V_H$ and $f_E : E_G \to E_H$ such that the following hold:

$$f_E ; lab_H = lab_G \quad \text{and} \quad f_E ; att_H = att_G ; f_V^*,$$

where f_V^* is the natural extension of f_V to sequences.

The category **Graph** is the category which has as objects finite (hyper)graphs, and as arrows graph morphisms.

We define the following 'special' graphs and morphisms: A graph is discrete if it consists only of nodes. The graph D_k is the discrete graph with node set $\mathbb{N}_k = \{0, \ldots, k-1\}$. Furthermore, for each $A \in \Sigma$, we define the graph E_A, consisting of a single A-labelled edge and adjacent (pairwise unequal) nodes, and the morphisms $e_A^i : D_1 \to E_A$, mapping the single node in D_1 to the node connected to the ith port of the single edge $e \in E_A$ (that is, to the ith node of $att_{E_A}(e)$). Furthermore, we define $Epi(A)$ to be the set of epimorphisms with domain E_A, up to isomorphism (in the case of **Graph**, this set is finite).

In particular, we will work with the category $Cospan(\mathbf{Graph})$. Composing cospans involves constructing a pushout. Concretely, taking a pushout of graph morphisms $f : U \to G$, $g : U \to H$ in **Graph** means to take the disjoint union of G and H and to factor through the smallest equivalence $\equiv$ (on nodes and edges) that satisfies $f(x) \equiv g(x)$ for all items (i.e., for all nodes and edges) x of U.

Cospans of graphs are intimately connected with the DPO approach to graph rewriting (see Sassone and Sobociński (2005), similar ideas have been formulated by Gadducci and Heckel (1997)). DPO rewriting rules are spans of graphs of the form

$$p : L \xleftarrow{\varphi_L} I \xrightarrow{\varphi_R} R.$$

We consider the cospans $\ell : \emptyset \to L \xleftarrow{\varphi_L} I$ and $r : \emptyset \to R \xleftarrow{\varphi_R} I$ as left- and right-hand sides. Now, for a graph G, let $[G] : \emptyset \to G \leftarrow \emptyset$ be the cospan consisting of G with empty source and target. Then DPO rewriting can also be defined as follows: the graph G rewrites to H by applying rule p if and only if $[G] = \ell ; c$ and $[H] = r ; c$ for some cospan c.

For the examples, we will usually consider unlabelled, directed (multi)graphs, i.e., we take $\Sigma = \{\star\}$, with $ar(\star) = 2$. Hence, $E = E_\star$ is the graph consisting of a single (directed)

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

edge connecting two nodes, and the morphisms $src = e^0_\star$ and $tgt = e^1_\star$ map the single node of D_1 to the source and the target of the edge in E, respectively.

3. Recognizable languages of arrows

In this section, we define automaton functors, an automaton model for recognizable languages, and study some of their properties.

3.1. *Automaton functors*

We consider a fixed category $\mathbf{C}$. In order to be able to talk about sets, we will require that $\mathbf{C}$ is *locally small*, that is, for every two objects, the class of arrows between them is a set, called *homset*. Furthermore, in the following, a subset of a homset will be called an *(arrow) language*.

Definition 3.1 (Automaton functor). Let $\mathbf{C}$ be a category. An *automaton functor for* $\mathbf{C}$ is a structure $\mathcal{A} = \langle \mathcal{A}_0, I_{X \in Obj(\mathbf{C})}, F_{X \in Obj(\mathbf{C})} \rangle$, where $\mathcal{A}_0 : \mathbf{C} \to \mathbf{Rel}$ is a functor, which maps each object $X \in Obj(\mathbf{C})$ to a set $\mathcal{A}_0(X)$ (called the *state set of* X) and each arrow $f \in Arr(\mathbf{C})$ to a relation $\mathcal{A}_0(f)$ (called the *transition relation* of f), and I and F are families of *initial* and *final states*, indexed by objects of $\mathbf{C}$, such that $I_X \subseteq \mathcal{A}_0(X)$ and $F_X \subseteq \mathcal{A}_0(X)$ for all $X \in \mathcal{A}_0(X)$.

An automaton functor $\mathcal{A}$ is called *finite* if $\mathcal{A}_0(X)$ is finite for all $X \in Obj(\mathbf{C})$. It is called *deterministic* if every relation $\mathcal{A}_0(f)$ is a function and I_X contains exactly one element for each $X \in Obj(\mathbf{C})$.

Let J, K be two objects of $\mathbf{C}$. The (J,K)-*language* of $\mathcal{A}$, denoted $L_{J,K}(\mathcal{A})$ is defined as follows:

$f : J \to K$ is contained in $L_{J,K}(\mathcal{A})$ if and only if there exist $s \in I_J$ and $t \in F_K$ such that $\langle s, t \rangle \in \mathcal{A}_0(f)$.

A language $L_{J,K}$ of arrows from J to K is *recognizable*, if there exists a finite automaton functor $\mathcal{A}$ which accepts it.

In the following, we will usually write $\mathcal{A}(X)$ and $\mathcal{A}(f)$ for $\mathcal{A}_0(X)$ and $\mathcal{A}_0(f)$.

The intuition of the definition is to have a mapping into a finite domain that respects compositionality and identities, that is, which is a functor. The functor properties (especially the preservation of composition) ensure that a cospan can be processed piece by piece and that decomposing the arrow in different ways does not affect acceptance in any way. This is a variation from the case of words where there is essentially only one way to decompose a word into atomic components.

Often, we will restrict our attention to finite automaton functors. Then, automaton functors accept exactly the recognizable languages à la Courcelle (see Theorem 4.14). In a sense, $\mathcal{A}(f)$ gives the transition relation of the finite automaton, only that here we do not have a single set of states, but a set of states for every object. This means that we have infinitely many sets (which corresponds to the infinitely many sorts in the case of Courcelle (1990)).

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Example 3.2 (Finite automata). Let $\mathcal{F}$ be a non-deterministic finite automaton over the alphabet Σ, with (finite) state set Q, start states $I \subseteq Q$, final states $F \subseteq Q$ and transition relation $\delta : Q \times \Sigma \to \wp(Q)$. We consider the free monoid of Σ, that is, the category with one single object X and words over Σ as arrows from X to X. A finite automaton $\mathcal{F}$ corresponds to an automaton functor for this category: We construct an automaton functor $\mathcal{A} = \langle \mathcal{A}_0, I^{\mathcal{A}}, F^{\mathcal{A}} \rangle$ which recognizes a language L if and only if L is accepted by $\mathcal{F}$ as follows:

— $\mathcal{A}(X) = Q$.
— For $w \in \Sigma^*$, $\mathcal{A}(w) = \{\langle s, t \rangle \mid t \in \hat{\delta}(s, w)\}$, where $\hat{\delta}$ is the extension of δ from letters to words.
— $I_X^{\mathcal{A}} = I$ and $F_X^{\mathcal{A}} = F$.

Example 3.3 (Tree automata). Also, tree automata can be seen as a special case of an automaton functor: Take as the category a Lawvere theory where objects are natural numbers and arrows from m to n are n-tuples of terms with m holes. Arrow composition is the usual term substitution.

As an example take two unary function symbols f, g, a binary function symbol h and a constant a. Now, the 3-tuple $\langle f(x_1), h(x_1, x_2), g(f(a)) \rangle$ has two holes, i.e., two variables x_1, x_2. Hence, it can be regarded as an arrow from 2 to 3. Then trees can be represented as arrows from 0 to 1, since they are single terms (or 1-tuples of terms) without variables.

If Q is the set of states of the tree automaton $\mathcal{T}$, then $\mathcal{A}$ assigns to m the product set Q^m. Furthermore, the relation associated to a term is induced by the transition function of the tree automaton $\mathcal{T}$, the initial state of the automaton functor is the single element of Q^0 and final states are one-element tuples consisting of a final state of $\mathcal{T}$.

Note that this specific instantiation has similarly been considered by Griffing (2003).

Two automaton functors are equivalent if they recognize the same language. Sometimes, we are mainly interested in languages of arrows which start at a fixed object, for instance, the initial object; therefore, we parametrize the notion of equivalence with a source object.

Definition 3.4 (Equivalence). Let $\mathcal{A}$ and $\mathcal{B}$ be automaton functors. $\mathcal{A}$ and $\mathcal{B}$ are said to be *J-equivalent* if they recognize the same languages $L_{J,K}$ of arrows from J to K, for arbitrary K. They are *equivalent* if they are J-equivalent for all $\mathbf{C}$-objects J.

Symmetrically, it would also be possible to work with the notion of K-equivalence in the following.

We will now give a characterization of recognizable language in terms of congruence classes, similar to Myhill–Nerode equivalences in the case of regular string languages.

Definition 3.5 (Congruence). Let $\mathbf{C}$ be a category. Let a family of relations

$$\equiv = \{\equiv_{J,K} \mid J, K \text{ are objects of } \mathbf{C}\}$$

be given, where the components $\equiv_{J,K}$ are equivalence relations on $\mathbf{C}$-arrows from J to K. We call $\equiv$ a (right) congruence if the following holds for all arrows $a, a' : J \to K$, $b : K \to M$:

If $a \equiv_{J,K} a'$, then $(a\,;b) \equiv_{J,M} (a'\,;b)$.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

A congruence $\equiv$ is *locally finite* if each $\equiv_{J,K} \in \equiv$ is an equivalence relation of finite index (i.e., it has finitely many equivalence classes).

Let $a : J \to K$ be an arrow. In the following, we will write $[\![a]\!]_{J,K}$, or simply $[\![a]\!]$ if J and K are clear from the context, to denote the congruence class of which a is a member. We will also usually write $a \equiv b$ instead of $a \equiv_{J,K} b$.

Proposition 3.6 (Myhill–Nerode characterization). Let $\mathbf{C}$ be a category, J, K $\mathbf{C}$-objects and $L_{J,K}$ a set of $\mathbf{C}$-arrows from J to K. The language $L_{J,K}$ is recognizable in $\mathbf{C}$, if and only if there exists a locally finite congruence $\equiv$ such that $L_{J,K}$ is the union of finitely many equivalence classes of $\equiv_{J,K}$.

Proof. ($\Rightarrow$): Let $\mathcal{A} = \langle \mathcal{A}_0, I, F \rangle$ be the automaton functor which recognizes the language $L_{J,K}$. We construct $\equiv = \{\equiv_{M,N} \mid M, N \in \mathbf{C}\}$ as follows. Let $a, a' : M \to N$ be $\mathbf{C}$-arrows. We define $a \equiv_{M,N} a'$ if and only if $\mathcal{A}(a) = \mathcal{A}(a')$. Since there are only finitely many relations between two finite sets, it is clear that each $\equiv_{M,N}$ is an equivalence relation of finite index. The fact that $\equiv$ is a congruence follows from the fact that $\mathcal{A}$ is a functor, and thus respects composition. Now, we have that

$$L_{J,K} = \bigcup \{[\![a]\!] \mid a : J \to K \text{ and } \mathcal{A}(a)(I_J) \cap F_K \neq \emptyset\}.$$

($\Leftarrow$): This direction of the proof only works for sets of arrows starting from the fixed object J. Let a congruence $\equiv = \{\equiv_{M,N} \mid M, N \in \mathbf{C}\}$ and the language $L_{J,K}$ be given. We define the deterministic automaton functor $\mathcal{A} : \mathbf{C} \to \mathbf{Set}$ as follows:

— For an object H, we define $\mathcal{A}(H) = \{[\![a]\!] \mid a : J \to H\}$.
— For an arrow $b : H \to M$ and $[\![a]\!] \in \mathcal{A}(H)$, we define $\mathcal{A}(b)([\![a]\!]) = [\![a\,;b]\!]$.

We show that $\mathcal{A}$ is a functor, that is, that it respects composition and identities:

— *Composition.* Let $b : H \to M$ and $c : M \to N$ be $\mathbf{C}$-arrows.

$$\begin{aligned}(\mathcal{A}(b)\,;\mathcal{A}(c))\left([\![a]\!]_{J,H}\right) &= \mathcal{A}(c)\left(\mathcal{A}(b)\left([\![a]\!]_{J,H}\right)\right) = \mathcal{A}(c)\left([\![a\,;b]\!]_{J,M}\right)\\ &= [\![a\,;b\,;c]\!]_{J,N} = \mathcal{A}(b\,;c)\left([\![a]\!]_{J,H}\right)\end{aligned}$$

— *Identities.* Let $a : J \to H$ be a $\mathbf{C}$-arrow. Then,

$$\mathcal{A}(\mathrm{id}_H)([\![a]\!]) = [\![a\,;\mathrm{id}_H]\!] = [\![a]\!] = \mathrm{id}_{\mathcal{A}(H)}([\![a]\!]).$$

Finally, we define I_J and F_K as follows:

$$\begin{aligned}I_J &= \{\mathrm{id}_J\},\\ F_K &= \{[\![a]\!] \mid a \in L_{J,K}\},\end{aligned}$$

as required. □

Note that the proof of Proposition 3.6 only works if we fix a specific start object. Dually, it would also have been possible to fix a target object and to concentrate on left congruences. But for our examples and for the comparison to the work of Courcelle fixing a start object seems to be more natural.

The paper by Griffing (2003) does not introduce the notion of an automaton functor, but it shows that composition-representative subsets (which are our recognizable languages)

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

can be characterized via locally finite congruences. In addition, the paper gives another – equivalent – characterization via a functor from **C** into a category with finite homsets. The recognizable languages are then the preimages of subsets of a finite homset.

3.2. *Determinization and closure properties*

One of the advantages of our characterization over characterizations utilizing finite homsets or congruences, is that we can talk explicitly about determinization and minimization. In this section, we consider these two notions as well as closure properties. These results, different from the results in Section 4, are not particularly deep or surprising; usually, they can be shown quite straightforwardly. We show them here for completeness and in order to illustrate that our notion of recognizability is reasonable.

Proposition 3.7 (Determinization). For every automaton functor, there exists an equivalent deterministic automaton functor.

Proof. (Sketch.) The construction is similar to the case of finite automata: We replace every set of states by its powerset. □

Proposition 3.8 (Closure under boolean operators). Suppose we have two recognizable languages of arrows, $L^1_{J,K}$ and $L^2_{J,K}$. Then, also $L^1_{J,K} \cap L^2_{J,K}$, $L^1_{J,K} \cup L^2_{J,K}$ and $(\overline{L^1_{J,K}})$ (the complement of $L^1_{J,K}$) are recognizable.

Proof. (Sketch.) Again, the construction resembles the case of finite automata: We work with deterministic automaton functors, take the cross product of the state sets and define the final states appropriately. □

Note that the results of this section depend on the specific nature of the categories **Rel** and **Set**. It is future research to find a more abstract characterization which still allows the techniques and constructions of this section, for instance, where the codomain of the automaton functor is an arbitrary category with finite homsets instead of **Rel**.

4. Automaton functors for recognizable graph languages

In this section, we apply the theory of the previous section to recognizing languages of graphs. First, we show how graph languages can be recognized by considering the category of cospans of graphs. Then we briefly introduce Courcelle's algebraic notion of recognizable graph languages, and finally we show that we can recognize the same graph languages as Courcelle.

4.1. *Recognizing languages of cospans of graphs*

In the following, the category under consideration will be *Cospan*(**Graph**), that is, the category of cospans of graphs, or put differently, the category of graphs with inner and outer interfaces. If we want to talk only about graphs without interfaces, we can restrict ourselves to languages of cospans with empty interfaces, that is, cospans where both the source and target are the empty graph.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Definition 4.1. A set L of graphs is recognizable whenever

$$L_{\emptyset,\emptyset} = \{[G] : \emptyset \to G \leftarrow \emptyset \mid G \in L\}$$

is a recognizable language in *Cospan*(**Graph**).

Below, we give an example for a recognizable graph language. It is not surprising that it is recognizable since it is definable in MSOGL and all such languages are recognizable in the sense of Courcelle (1990; 1997). However, the example provides some intuition into the notion of recognizability.

Example 4.2 (k-colourability). Let G be a graph. A k-colouring of G is a function $f : V_G \to \mathbb{N}_k$ such that for all $e \in E_G$ and for all $v_1, v_2 \in att_G(e)$ it holds that $f(v_1) \neq f(v_2)$ if $v_1 \neq v_2$. We show that the language of all k-colourable graphs is recognizable, by considering the following automaton functor $\mathcal{A} = \langle \mathcal{A}_0, I, F \rangle$ for *Cospan*(**Graph**):

— Every graph J is mapped to $\mathcal{A}(J)$, the set of all k-colourings of J:

$$\mathcal{A}(J) = \{f : V_J \to \mathbb{N}_k \mid f \text{ is a } k\text{-coloring of } J\}.$$

— For a cospan $c : J \to G \leftarrow K$, the relation $\mathcal{A}(c)$ relates two colourings f_J, f_K, whenever there exists a colouring f for G such that $f(c_{\mathrm{L}}(v)) = f_J(v)$ for every node $v \in V_J$ and $f(c_{\mathrm{R}}(v)) = f_K(v)$ for every node $v \in V_K$.

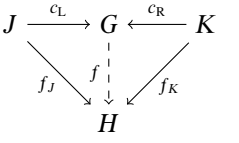
Specifically, we have that $\mathcal{A}(\emptyset) = \{\emptyset\}$ where $\emptyset$ is the empty colouring. Then in order to accept all k-colourable graphs with empty interfaces we define the initial and final states as $I_\emptyset = F_\emptyset = \{\emptyset\}$: A graph $\emptyset \to G \leftarrow \emptyset$ is accepted whenever the two empty mappings are related.

The automaton functor works as follows. The states record the colours of the nodes in the interfaces. For each cospan which is read, it is checked that the new parts of the graph are compatible with the colouring already present. For example, a state in which nodes v_1 and v_2 have the same colour will not be related to any successor state if the post-composing the cospan has the effect of connecting v_1 and v_2.

Note that it is well known that k-colourability of graphs is an NP-complete property. Intuitively, this manifests itself in the fact that interfaces may grow unboundedly, leading to an exponential explosion of the size of the state sets. However, if we restrict ourselves to graphs of bounded treewidth, there are efficient algorithms for k-colourability (see also the related discussion in the conclusion).

Example 4.3 (Occurrence graph language). Let H be a fixed graph. We consider the language L_H of all graphs G for which a morphism $f : G \to H$ exists. The language L_H is recognizable.

The functor $\mathcal{A}$ associates to every graph J the set of all morphisms $J \to H$. For a cospan $c : J \to G \leftarrow K$, it relates a morphism $f_J : J \to H$ to a morphism $f_K : K \to H$ whenever there exists a morphism $f : G \to H$ such that $c_{\mathrm{L}} ; f = f_J$ and $c_{\mathrm{R}} ; f = f_K$. All states are initial and final.



The above notion of graph language is a weaker notion than recognizability. It has been considered before; see, for instance, Kuncak and Rinard (2003) and Bruggink (2007).

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

4.2. *Robustness*

We now show the robustness of recognizability by restricting ourselves to injective, edge-injective and discrete interfaces. Intuitively, we show that recognizability is robust in the sense that certain variations yield the same notion. These results will also be important for the comparison with Courcelle's notion of recognizability (see Section 4.4).

We use proof principles already explored in Courcelle and Weil (2005) where robustness proofs are based on the characterization of recognizable languages in terms of locally finite congruences (see Definition 3.5). The results and proofs all follow the same pattern: Let $\mathbf{D}$ be a subcategory of $\mathbf{C}$ and let J, K be two objects of $\mathbf{D}$. We want to show that every language of arrows from J to K is recognizable in $\mathbf{C}$ if and only if it is recognizable in $\mathbf{D}$. The direction from left to right is obvious, we simply restrict the congruence or the automaton functor accordingly. The real challenge is the direction from right to left. In this case, take a congruence $\equiv_{\mathrm{R}}$ on $\mathbf{D}$ and construct a congruence $\equiv'_{\mathrm{R}}$ on $\mathbf{C}$ that is locally finite and refines $\equiv_{\mathrm{R}}$ when restricted to $\mathbf{D}$. Since $\equiv'_{\mathrm{R}}$ refines $\equiv_{\mathrm{R}}$ any union of equivalence classes can still be represented as a union of (possibly more) equivalence classes, and hence recognizability is preserved.

We now show the convenient fact that restricting our attention to cospans with injective interfaces does not limit the descriptive power of the formalism. Hence, let $\mathbf{G} = \mathit{Cospan}(\mathbf{Graph})$ be a cospan category and let $\mathbf{G}_{\mathrm{inj}}$ be its subcategory that consists only of the cospans with injective interface morphisms.

Proposition 4.4. Let a class $L_{J,K}$ of graphs with injective interfaces J, K be called *injectively recognizable* whenever $L_{J,K}$ is recognizable in $\mathbf{G}_{\mathrm{inj}}$. Then, $L_{J,K}$ is recognizable in $\mathbf{G}$ if and only if it is injectively recognizable.

Proof. For both directions of the proof, we use the characterization of recognizability via congruences (Proposition 3.6).

($\Rightarrow$): Trivial, because $\mathbf{G}_{\mathrm{inj}}$ is a subcategory of $\mathbf{G}$, and thus we can use the congruence for $\mathbf{G}$ also in the case of $\mathbf{G}_{\mathrm{inj}}$.

($\Leftarrow$): Let $\equiv_{\mathrm{R}}$ be a congruence for $L_{J,K}$ in the sense of Definition 3.5 for the category $\mathbf{G}_{\mathrm{inj}}$. We now define an equivalence $\equiv'_{\mathrm{R}}$ on $\mathbf{G}$ and show that it is a congruence of finite index.

For this, we first need the notion of *merger pairs*. For a cospan $c : J \to c_{\mathrm{L}} \leftarrow c_{\mathrm{R}} - K$, the set $M(c)$ of merger pairs consists of all pairs of cospans $\langle m_J, m_K \rangle$ of the form

$$m_J : J' \xrightarrow{\mathrm{id}} J' \xleftarrow{m_J^{\mathrm{R}}} J, \quad m_K : K \xrightarrow{m_K^{\mathrm{L}}} K' \xleftarrow{\mathrm{id}} K',$$

such that m_J^{R} and m_K^{L} are surjective and $m_J \,;c\,; m_K$ is an injective cospan. In a sense, the merger pairs induce an equivalence on the interfaces which relates at least the interface items which have the same image under c_{L} or c_{R}. Furthermore, since we only deal with finite graphs, there are only finitely many 'merger cospans' up to isomorphism.

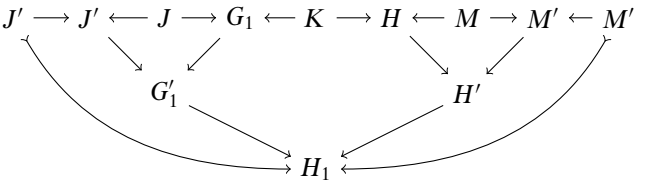
We now define $\equiv'_{\mathrm{R}}$ as follows: For two cospans $c_1 : J \to G_1 \leftarrow K$ and $c_2 : J \to G_2 \leftarrow K$, it holds that $c_1 \equiv'_{\mathrm{R}} c_2$ whenever $M(c_1) = M(c_2)$ and for all $\langle m_J, m_K \rangle \in M(c_1)$ we have $m_J \,;c_1\,; m_K \equiv_{\mathrm{R}} m_J \,;c_2\,; m_k$. Note that for an injective cospan c, the pairs consisting of the

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

identity cospans on J and K are among the merger pairs and hence $\equiv'_{\mathrm{R}}$ is a refinement of $\equiv_{\mathrm{R}}$. Furthermore, it can be shown that $\equiv'_{\mathrm{R}}$ is locally finite whenever $\equiv_{\mathrm{R}}$ is.

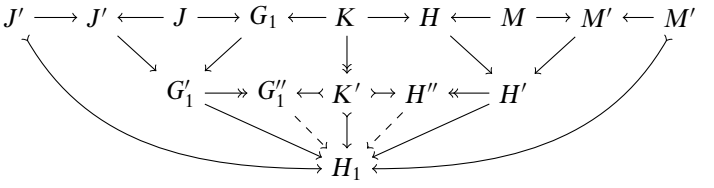
We now have to show that $\equiv'_{\mathrm{R}}$ is a congruence. For this, take two cospans $c_1 \equiv'_{\mathrm{R}} c_2$ with $c_i : J \to G_i \leftarrow K$, where $i \in \{1,2\}$, and another cospan $d : K \to H \leftarrow M$. We have to prove that $c_1 ; d \equiv'_{\mathrm{R}} c_2 ; d$.

Now take a merger pair of $c_1 ; d$, i.e., choose $\langle m_J, m_M \rangle \in M(c_1 ; d)$. We will show that there are cospans m_K, m'_K such that $\langle m_J, m_K \rangle \in M(c_1)$, $\langle m'_K, m_M \rangle \in M(d)$ and $m_J ; c_1 ; d ; m_M = m_J ; c_1 ; m_K ; m'_K ; d ; m_M$. For this, consider the diagram below which represents the composition $(m_J ; c_1) ; (d ; m_M)$ via pushouts where the order of composition is indicated by the brackets.



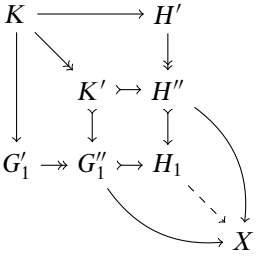
Note that the two 'triangles' on the left and right are pushouts and that the middle 'pentagon' is also a pushout. In the following, we will denote injective morphisms by $\rightarrowtail$ and surjective morphisms by $\twoheadrightarrow$.

Now consider the morphism $K \to H_1$ and take its factorization into a surjective morphism $K \twoheadrightarrow K'$, followed by an injective morphism $K' \rightarrowtail H_1$. In the category of graphs, this factorization always exists and is unique up to isomorphism. In addition, take the pushouts of $K \to G'_1$, $K \twoheadrightarrow K'$ and of $K \to H'$, $K \twoheadrightarrow K'$, obtaining graphs G''_1 and H'' and corresponding mediating morphisms. Now the situation is as follows:



Note also that the morphisms $J' \to J' \to G'_1 \to G''_1$ and $M' \to M' \to H' \to H''$ must be injective, since they split the injective morphisms $J' \to H_1$, $M' \to H_1$.

We will now show that the square K', G''_1, H'', H_1 commutes and that it is a pushout. We are in the situation depicted on the right, where the outer square is a pushout. We have $K \twoheadrightarrow K' \rightarrowtail G''_1 \rightarrowtail H_1 = K \to G'_1 \twoheadrightarrow G''_1 \rightarrowtail H_1 = K \to H' \twoheadrightarrow H'' \rightarrowtail H_1 = K \twoheadrightarrow K' \rightarrowtail H'' \rightarrowtail H_1$. And since $K \twoheadrightarrow K'$ is surjective, we can conclude that $G'_1 \twoheadrightarrow G''_1 \rightarrowtail H_1 = G'_1 \rightarrowtail H'' \rightarrowtail H_1$. Now assume a commuting square K', G''_1, H'', X. Since the outer square is a pushout, we obtain a mediating morphism $H_1 \to K$. The triangles G''_1, H_1, X and H'', H_1, X commute since they commute if we precompose them with the morphisms $G'_1 \twoheadrightarrow G''_1$ and $H' \twoheadrightarrow H''$, respectively.

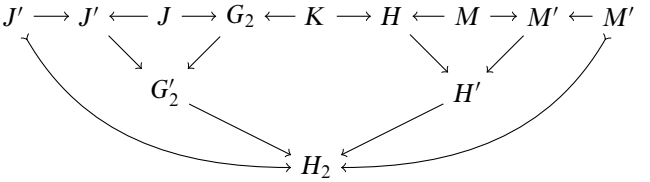


https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Furthermore, the morphism $H_1 \to X$ must be unique since any other mediating morphism would also be a mediating morphism for the outer pushout.

Hence, we get $m_K : K \twoheadrightarrow K' \leftarrow\!\!\text{id}\!\!- K'$ and $m'_K : K' -\!\!\text{id}\!\!\rightarrow K' \twoheadrightarrow K$ and from the diagram above it follows that $m_J ; c_1 ; d ; m_M = m_J ; c_1 ; m_K ; m'_K ; d ; m_M$. Since both $m_J ; c_1 ; m_K$ and $m'_K ; d ; m_M$ are injective cospans and $c_1 \equiv'_{\mathrm{R}} c_2$, we get $m_J ; c_1 ; m_K ; m'_K ; d ; m_M \equiv_{\mathrm{R}} m_J ; c_2 ; m_K ; m'_K ; d ; m_M$.

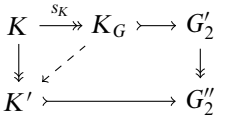
It is now left to show that $m_J ; c_2 ; m_K ; m'_K ; d ; m_M = m_J ; c_2 ; d ; m_M$ and that this is an injective cospan. We first consider the diagram for $(m_J ; c_2) ; (d ; m_M)$.



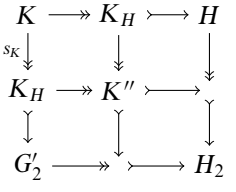
Next, we consider the factorization of $K \to G'_2$ into a surjective morphism followed by an injective morphism, i.e.,

$$K \to G'_2 = K \overset{s_K}{\twoheadrightarrow} K_G \rightarrowtail G'_2$$

We construct the pushout of $s_K : K \twoheadrightarrow K_G$ and $K \to G'_2$ as shown below in two steps. First, we take the pushout of s_K with itself, which gives us K_G and the identity morphisms since s_K is surjective. Then we take the pushout of $K_G \rightarrowtail G'_2$ and the identity on K_G, resulting in G'_2. Now consider the cospan $\hat{m} : K -s_K\!\rightarrow K_G \leftarrow\!\!\text{id}\!\!- K_G$. Because of the considerations above, we can conclude that $\langle m_J, \hat{m} \rangle \in M(c_2) = M(c_1)$. Furthermore, we can show that $\hat{m}$ is uniquely characterized to be the most 'general' partner for m_J. Consider for instance another cospan $m_K : K \to K' \leftarrow K'$ with $\langle m_J, m_K \rangle \in M(c_2)$. Because of the diagonalization property of factorizations, we obtain a unique morphism $K_G \to K'$ which makes the diagram commute (see diagram above on right). This property uniquely characterizes s_K (up to isomorphism).



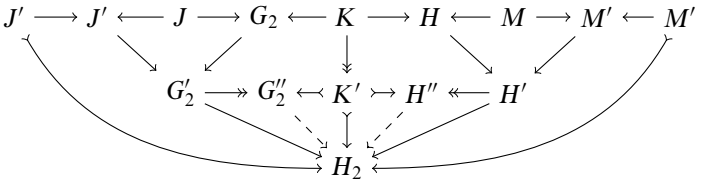
The same characterization holds for the factorization of $K \to G'_1$ (since the merger pairs are identical), and we can conclude that $K \to G'_1$ factorizes into $K \overset{s_K}{\twoheadrightarrow} K_G \rightarrowtail G'_1$. Now consider also the factorization of $K \to H$ into $K \twoheadrightarrow K_H \rightarrowtail H$ and split the pushout of $K \to G'_2$ and $K \to H$ as shown below:



Hence, taking the pushout of $K \twoheadrightarrow K_G$ and $K \twoheadrightarrow K_H$ gives us the unique factorization of $K \to H_2$. The same is true in the case of H_1 where the factorization is $K \to H_2 = K \twoheadrightarrow K' \rightarrowtail H_1$. Hence, K' and K'' must be isomorphic.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Furthermore, in the diagram for $(m_J ; c_2) ; (d ; m_M)$ above, we can split the middle pushout in a way identical to the splitting of $(m_J ; c_1) ; (d ; m_M)$ and obtain a pushout consisting of K', G_2'', H'', H_2 (see below).



This implies that $m_J ; c_1 ; d ; m_M = m_J ; c_1 ; m_K ; m'_K ; d ; m_M$. Furthermore, $J' \rightarrowtail H_2$ and $M' \rightarrowtail H_2$ must be injective: $J \rightarrowtail G_2''$ is injective (since it is the first leg of the injective cospan $m_J ; c_1 ; m_K$) and if we compose with the injective morphism $G_2'' \rightarrowtail H_2$, we obtain $J' \rightarrowtail H_2$. Second, $M' \rightarrowtail H''$ is injective (since it is the second leg of the injective cospan $m'_K ; d ; m_M$) and if we compose with the injective morphism $H'' \rightarrowtail H_2$, we obtain $M' \rightarrowtail H_2$. Hence, we have $M(c_1 ; d) = M(c_2 ; d)$ and this concludes the proof. □

We now restrict our attention to the subcategory $\mathbf{G}_{\text{einj}}$ of cospans with interface morphisms which are injective on edges. The result is not that interesting in its own right, but it is a necessary auxiliary step for Proposition 4.6.

Corollary 4.5. Let a class $L_{J,K}$ of graphs with edge-injective interfaces J, K be called *edge-injectively recognizable* whenever $L_{J,K}$ is recognizable in $\mathbf{G}_{\text{einj}}$. Then $L_{J,K}$ is recognizable in $\mathbf{G}$ if and only if it is edge-injectively recognizable.

Proof. We adapt the proof of Proposition 4.4 by extending the notion of merger pairs. For $c : J \rightarrow G \leftarrow K$, let $M_e(c)$ be the set of merger pairs $\langle m_J, m_K \rangle$ such that $m_J ; c ; m_K$ is edge-injective. In addition, let $M_l(c)$ be the set of pairs of cospans $\langle m_J, m_K \rangle$ such that $m_J ; c ; m_K$ is edge-injective on the left leg and injective on the right†. Clearly, $M_l(c) \subseteq M_e(c)$.

Given a congruence $\equiv_{\mathrm{R}}$ that characterizes $L_{J,K}$, we now define $\equiv'_{\mathrm{R}}$ as follows: For two cospans $c_i : J \rightarrow G_i \leftarrow K$, where $i \in \{1, 2\}$, it holds that $c_1 \equiv'_{\mathrm{R}} c_2$ whenever $M_l(c_1) = M_l(c_2)$, $M_e(c_1) = M_e(c_2)$ and for all $(m_J, m_K) \in M_e(c_1)$, we have $m_J ; c_1 ; m_K \equiv_{\mathrm{R}} m_J ; c_2 ; m_k$. Clearly, $\equiv'_{\mathrm{R}}$ is a refinement of $\equiv_{\mathrm{R}}$ and it is of finite index.

The rest of the proof carries through as before with some slight adaptations: Note that we obtain that $J' \rightarrow H_1$, $M' \rightarrow H_1$ are edge-injective (instead of injective as before). Again, we factorize $K \rightarrow H_1$ into an injective and surjective morphism, i.e., $K \rightarrow H_1 = K \twoheadrightarrow K' \rightarrowtail H_1$. Since we have $M_l(c_1) = M_l(c_2)$, we obtain a factorization $K \twoheadrightarrow K' \rightarrowtail H_2$ for $K \rightarrow H_2$ with an identical morphism $K \twoheadrightarrow K'$.

Hence, the rest of the proof works as before, we only have to show that $M_l(c_1 ; d) = M_l(c_2 ; d)$ (and the same for M_e). Take for instance, $\langle m_K, m_M \rangle \in M_l(c_1 ; d)$. In this case, $m_K ; c_1 ; d ; m_M$ is edge-injective on the left and injective on the right which means $J' \rightarrow H_1$ is edge-injective and $M' \rightarrowtail H_1$ is injective. This implies that $J' \rightarrow G_1''$ is edge-injective

† This asymmetry is caused by the fact that we need a congruence with respect to composition on the right.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

and $M' \rightarrowtail H''$ is injective (since they split (edge-)injective morphisms). Since $K' \rightarrowtail G_1''$ and $K' \rightarrowtail H''$ are injective, this means that $\langle m_J, m_K \rangle \in M_l(c_1)$ and $\langle m_K', m_M \rangle \in M(d)$. Hence, also $\langle m_J, m_K \rangle \in M_l(c_2)$ and we have that $J' \to G_2''$ is edge-injective. So $J' \to H_2$ is edge-injective, since we postcompose $J' \to G_2''$ with the injective morphism $G_2'' \rightarrowtail H_2$. And finally $M' \rightarrowtail H_2$ is injective since we post-compose $M' \rightarrowtail H''$ with the injective morphism $H'' \rightarrowtail H_1$. □

Similar to the restriction to injective interfaces, we now show the fact that restricting our attention to cospans with discrete interfaces does not limit the descriptive power of the formalism. This allows us to restrict our attention to discrete interfaces in the following. Let $\mathbf{G} = \mathit{Cospan}(\mathbf{Graph})$ be a cospan category and let $\mathbf{G}_{\text{dis}}$ be its subcategory that consist only of the cospans with discrete interfaces, i.e., with interface graphs that do not contain edges.

Proposition 4.6. Let a class $L_{J,K}$ of graphs with discrete interfaces J, K be called *discretely recognizable* whenever $L_{J,K}$ is recognizable in $\mathbf{G}_{\text{dis}}$. Then $L_{J,K}$ is recognizable in $\mathbf{G}$ if and only if it is discretely recognizable.

Proof. For both directions of the proof, we use the characterization of regularity via congruences (Proposition 3.6).

($\Rightarrow$): Trivial, because $\mathbf{G}_{\text{dis}}$ is a subcategory of $\mathbf{G}$, and thus we can use the congruence for $\mathbf{G}$ also in the case of $\mathbf{G}_{\text{dis}}$.

($\Leftarrow$): We use the result that whenever a language is edge-injectively recognizable, then it is recognizable (Corollary 4.5). Hence, it is sufficient to show that whenever a language is discretely recognizable, then it is edge-injectively recognizable. So let $\equiv_{\text{R}}$ be a congruence on discrete cospans (as in Definition 3.5). We need the following operator *discr*: let $c : J \rightarrow_{c_{\text{L}}} G \leftarrow_{c_{\text{R}}} K$ be a cospan which is injective on edges. Now take as J', K' be the discrete graphs underlying J, K (which are obtained by removing all edges). Furthermore, remove from G all edges that are in the image of c_{L}. Now we get a new cospan $c' = \mathit{discr}(c) : J' \to G' \leftarrow K'$. Observe that we have $\mathit{discr}(c_1 ; c_2) = \mathit{discr}(c_1) ; \mathit{discr}(c_2)$, whenever c_1, c_2 are both edge-injective and $c_1 ; c_2$ is defined. This does not hold whenever c_1, c_2 are not injective on edges since on the right-hand side of the equation fewer edges might be merged.

Now define a new equivalence $\equiv_{\text{R}}'$, on edge-injective cospans, with $c_1 \equiv_{\text{R}}' c_2$ if and only if $\mathit{discr}(c_1) \equiv_{\text{R}} \mathit{discr}(c_2)$. We now show that $\equiv_{\text{R}}'$ is a congruence: Let $c_1 \equiv_{\text{R}}' c_2$. Then we have that

$$\mathit{discr}(c_1 ; d) = \mathit{discr}(c_1) ; \mathit{discr}(d) \equiv_{\text{R}} \mathit{discr}(c_2) ; \mathit{discr}(d) = \mathit{discr}(c_2 ; d),$$

which implies that $c_1 ; d \equiv_{\text{R}}' c_2 ; d$. □

In a sense, the result above is mirrored in the fact that graph rewriting does not lose any expressive power if we restrict to discrete interfaces.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

4.3. *Courcelle's algebra of graphs*

We will now give a short introduction to Courcelle's algebraic notion of recognizable graph languages (Courcelle 1990, 1997). Courcelle's notion of recognizable graph language is widely accepted as a notion of regularity for graphs. Also, Courcelle showed that if a language is definable in monadic second-order logic then it is recognizable. In Habel et al. (1993), Courcelle's notion has been found to be analogous to the notions 'finite graph property' and 'compatible graph property,' which were developed in different contexts. For instance, compatible properties arise in connection with hyperedge replacement grammars.

In Courcelle (1997), the relevant graph algebra is called HR-algebra (and in addition another algebra, called VR-algebra is investigated). However, here in this paper, we use the algebra introduced in Courcelle (1990), since it is closer to our notion of cospan composition and hence yields simpler proofs.

Note that there are two major differences between cospan composition and the algebra of graphs introduced below: First, the graph algebra considers only discrete interfaces, and we have already shown how to bridge this gap via Proposition 4.6. Second, cospans have two interfaces, whereas graphs in the algebra have only one.

First, we give some preliminary definitions: Let P, Q be arbitrary sets. For functions $f : \mathbb{N}_n \to P$ and $g : \mathbb{N}_m \to Q$, we define the function $f \odot g : \mathbb{N}_{n+m} \to P \cup Q$ as follows:

$$(f \odot g)(i) = \begin{cases} f(i) & \text{if } i < n \\ g(i-n) & \text{otherwise.} \end{cases}$$

An n-ary hypergraph is a pair $\mathbb{G} = \langle \mathrm{base}_{\mathbb{G}}, \zeta_{\mathbb{G}} \rangle$ consisting of a hypergraph $\mathrm{base}_{\mathbb{G}}$ and a mapping $\zeta_{\mathbb{G}} : \mathbb{N}_n \to V$, where V is the node set of $\mathrm{base}_{\mathbb{G}}$. The function $\zeta_{\mathbb{G}}$ is called the *interface* of the graph, and its range the *external nodes*. Basically, an n-ary hypergraph corresponds to a graph with an empty internal and discrete external interface.

Bouderon and Courcelle (1987) define the following atomic operations on n-ary graphs:

Redefinition of external nodes. Let an n-ary hypergraph $\mathbb{G}$ be given, and let $\sigma : \mathbb{N}_m \to \mathbb{N}_n$ be a function. The redefinition of $\mathbb{G}$ under σ is

$$\mathsf{redef}_{\sigma}(\mathbb{G}) = \langle \mathrm{base}_{\mathbb{G}}, (\sigma \,; \zeta_{\mathbb{G}}) \rangle.$$

Note that this means that $\mathsf{redef}_{\sigma}(\mathbb{G})$ is an m-ary graph.

Fusion of external nodes. Let $\mathbb{G}$ be an n-ary graph, and θ an equivalence relation on $\mathbb{N}_n$. The fusion of $\mathbb{G}$ over θ, denoted $\mathsf{fuse}_{\theta}(\mathbb{G})$, is obtained by fusing the nodes of $\mathbb{G}$ according to θ. The result is again an n-ary graph.

Disjoint union. Let $\mathbb{G}$ be an n-ary graph and $\mathbb{H}$ an m-ary graph. The disjoint union of $\mathbb{G}$ and $\mathbb{H}$ is defined as

$$\mathbb{G} \oplus \mathbb{H} = \langle \mathrm{base}_{\mathbb{G}} \oplus \mathrm{base}_{\mathbb{H}}, \zeta_{\mathbb{G}} \odot \zeta_{\mathbb{H}} \rangle$$

(we assume here that the node sets of $\mathrm{base}_{\mathbb{G}}$ and $\mathrm{base}_{\mathbb{H}}$ are disjoint and that $\oplus$ denotes the disjoint union on base graphs.)

We will now define recognizability in the sense of Courcelle via congruences. There is an alternative, but equivalent, definition of recognizable subsets as preimages of algebra homomorphisms.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Definition 4.7. Let $\equiv_C$ be an equivalence on n-ary hypergraphs that relates only hypergraphs with the same arity. It is called *locally finite* if for each n there are only finitely many equivalence classes. It is called a *congruence* if the following conditions hold:

— If $\mathbb{G} \equiv_C \mathbb{H}$, then $\mathsf{redef}_\sigma(\mathbb{G}) \equiv_C \mathsf{redef}_\sigma(\mathbb{H})$.
— If $\mathbb{G} \equiv_C \mathbb{H}$, then $\mathsf{fuse}_\theta(\mathbb{G}) \equiv_C \mathsf{fuse}_\theta(\mathbb{H})$.
— If $\mathbb{G}_1 \equiv_C \mathbb{H}_1$ and $\mathbb{G}_2 \equiv_C \mathbb{H}_2$, then $\mathbb{G}_1 \oplus \mathbb{G}_2 \equiv_C \mathbb{H}_1 \oplus \mathbb{H}_2$.

A set L of n-ary graphs is called *Courcelle recognizable* if it is the union of finitely many equivalence classes of a locally finite congruence.

We will in the following show that the notion of recognizability of Courcelle coincides with our notion, hence the notion of 'Courcelle recognizability' is redundant. However, we will keep it for the moment in order to properly distinguish the two notions of recognizability.

4.4. *Equivalence of the two notions of recognizability*

In this subsection, we show that our notion of recognizable graph language is equivalent to Courcelle's. In Courcelle's notion, there is only one (discrete) interface. The role of cospan composition is played by the operators defined above. Note that in our notion of recognizability, we have no operation of disjoint union, hence we are considering path decompositions instead of tree decompositions as in the case of Courcelle. Hence, the two notions of recognizability look subtly different.

Encouraged by the result of Proposition 4.6, we restrict our attention to cospans with discrete interfaces. Remember that the discrete graph with node set $\mathbb{N}_n$ is represented by D_n, we make use of the fact that each discrete graph with n nodes is isomorphic to D_n.

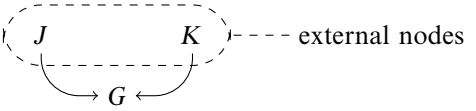
To formalize the equivalence between both notions of recognizability, we must first associate (sets of) cospans of graphs with n-ary graphs.

Definition 4.8. For each discrete graph D with n nodes, we fix in advance an isomorphism $\mathrm{di}_D : D_n \to D$ such that $\mathrm{di}_{D_1 \oplus D_2} = \mathrm{di}_{D_1} \odot \mathrm{di}_{D_2}$.

We define the function bend which maps a cospan $c : J \to_{c_L} G \leftarrow_{c_R} K$, where J, K are discrete interfaces with n, m nodes, respectively, to an $(n+m)$-ary graph as follows:

$$\mathrm{bend}(c) = \langle G, (\mathrm{di}_J ; c_L) \odot (\mathrm{di}_K ; c_R) \rangle.$$

The name bend is inspired by the fact that the function basically 'bends' a cospan so that its inner and outer interface are together, and then interprets the resulting figure as a $(m+n)$-ary hypergraph, as illustrated below:



Lemma 4.9. Let $c : \emptyset \to G \leftarrow J$, $d : \emptyset \to H \leftarrow J$, $c' : \emptyset \to G' \leftarrow K$ and $d' : \emptyset \to H' \leftarrow K$ be cospan, and $\equiv_R$ a congruence on cospans. If $\mathrm{bend}(c) = \mathrm{bend}(c')$ and $\mathrm{bend}(d) = \mathrm{bend}(d')$, then it holds that $c \equiv_R d$ if and only if $c' \equiv_R d'$.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press



Fig. 1. Simulating Courcelle's graph operation by cospans. On the left: example of cospan simulating redefinition of external nodes; on the right: example of cospan simulating fusion of external nodes.

Proof. Follows from the fact that if $\mathrm{bend}(c) = \mathrm{bend}(c')$ and $\mathrm{bend}(d) = \mathrm{bend}(d')$, an isomorphism $f : K \to J$ exists such that $c_{\mathrm{R}} = f\,; c_{\mathrm{R}}'$. This allows us to construct the cospan $e : J \xrightarrow{\mathrm{id}_J} J \xleftarrow{f} K$ such that $c\,; e = c'$ and $d\,; e = d'$, which, because $\equiv_{\mathrm{R}}$ is a congruence by assumption, proves the left-to-right direction of the proof. The other direction works symmetrically. □

Definition 4.10. Let J be a discrete graph with n nodes. We define the following cospans with source J (see Figure 1):

i. Let $s : K \to J$ be a morphism from a discrete graph K to J. Then we define

$$\mathit{redef}_s : J \xrightarrow{\mathrm{id}_J} J \xleftarrow{s} K.$$

ii. Let D be a discrete graph and $t : J \to D$ a morphism.

$$\mathit{fuse}_t : J \xrightarrow{t} D \xleftarrow{t} J.$$

In the following, we use the monoidal operation $\oplus$ which is the disjoint sum on cospans. Formally, it is obtained by taking the coproducts of the middle graph and of the inner and outer interfaces. The new interface morphisms are then obtained as mediating morphisms.

Lemma 4.11. Let the n-node discrete graph J and cospan $c : \varnothing \to G \leftarrow J$ be given.

i. Let $\sigma : \mathbb{N}_m \to \mathbb{N}_n$ be a function. For each m-node discrete graph K, there exists an arrow $s : K \to J$ such that

$$\mathrm{bend}(c\,; \mathit{redef}_s) = \mathsf{redef}_\sigma(\mathrm{bend}(c)).$$

ii. Let θ be an equivalence relation on $\mathbb{N}_n$. There exist a discrete graph D and arrow $t : J \to D$ such that

$$\mathrm{bend}(c\,; \mathit{fuse}_t) = \mathsf{fuse}_\theta(\mathrm{bend}(c)).$$

Proof.

i. We consider σ as a graph morphism from D_m to D_n and take $s = \mathrm{di}_K^{-1}\,; \sigma\,; \mathrm{di}_J$. The composed cospan $c\,; \mathit{redef}_s$ now looks like

$$(c\,; \mathit{redef}_s) : \varnothing \xrightarrow{c_{\mathrm{L}}} G \xleftarrow{s\,; c_{\mathrm{R}}} K.$$

Then we do (because J and K are discrete, we confuse graph morphisms and functions on nodes)

$$\begin{aligned}\mathrm{bend}(c\,; \mathit{redef}_s) &= \langle G, \mathrm{di}_K\,; s\,; c_{\mathrm{R}}\rangle = \langle G, \mathrm{di}_K\,; \mathrm{di}_K^{-1}\,; \sigma\,; \mathrm{di}_J\,; c_{\mathrm{R}}\rangle = \langle G, \sigma\,; \mathrm{di}_J\,; c_{\mathrm{R}}\rangle \\ &= \mathsf{redef}_\sigma(\langle G, \mathrm{di}_J\,; c_{\mathrm{R}}\rangle) = \mathsf{redef}_\sigma(\mathrm{bend}(c)).\end{aligned}$$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

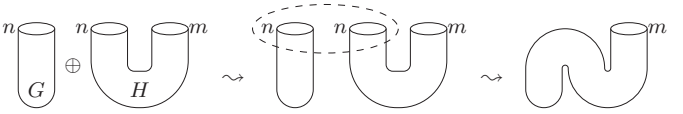


Fig. 2. Simulating cospan composition with Courcelle's operations. First, we construct the disjoint union of the graphs In the second step, the indicated external nodes are fused and then removed from the external nodes by a redefinition.

ii. Let θ_{map} be the function which maps each element of $\mathbb{N}_n$ to its respective θ-equivalence class, let D be the discrete graph with node set $\{[\![k]\!]_\theta \mid k \in \mathbb{N}_n\}$ and define the arrow t as $t = \text{di}_J^{-1}\,;\theta_{\text{map}}$. Because cospan composition is define by a pushout, the cospan $(c\,;\mathit{fuse}_t)$ is the same as c but with the nodes fused according to θ. □

Lemma 4.12. For cospans $c : \varnothing \to G \leftarrow J$ and $d : \varnothing \to H \leftarrow M$, it holds that $\text{bend}(c \oplus d) = \text{bend}(c) \oplus \text{bend}(d)$.

Proof. From the following equation:

$$\begin{aligned}\text{bend}(c \oplus d) &= \langle G \oplus H, \text{di}_{J\oplus K}\,; c_{\text{R}} \oplus d_{\text{R}}\rangle = \langle G \oplus H, (\text{di}_J\,; c_{\text{R}}) \odot (\text{di}_K\,; d_{\text{R}})\rangle \\ &= \langle G, \text{di}_J\,; c_{\text{R}}\rangle \oplus \langle H, \text{di}_K\,; d_{\text{R}}\rangle = \text{bend}(c) \oplus \text{bend}(d).\end{aligned}$$ □

Lemma 4.13. Let cospans $c : I \to G \leftarrow J$ and $d : J \to H \leftarrow K$ be given, where I has m nodes, J has n nodes and K has k nodes. Then,

$$\text{bend}(c\,; d) = \mathsf{redef}_\sigma(\mathsf{fuse}_\theta(\text{bend}(c) \oplus \text{bend}(d))),$$

where $\sigma : \mathbb{N}_{m+k} \to \mathbb{N}_{m+2n+k}$ is defined as

$$\sigma(x) = \begin{cases} x + 2n & \text{if } x \geqslant m \\ x & \text{otherwise,}\end{cases}$$

and θ is the smallest equivalence relation on $\mathbb{N}_{m+2n+k}$, which, for $m \leqslant x < m + n$, relates x to $x + n$ (see Figure 2).

Proof. We have that

$$\mathbb{M}_0 = \text{bend}(c) \oplus \text{bend}(d) = \langle G \oplus H, (\text{di}_I\,; c_{\text{L}}) \odot (\text{di}_J\,; c_{\text{R}}) \odot (\text{di}_J\,; d_{\text{L}}) \odot (\text{di}_K\,; d_{\text{R}})\rangle.$$

By taking the composition of $c : I \xrightarrow{c_{\text{L}}} G \xleftarrow{c_{\text{R}}} J$ and $d : J \xrightarrow{d_{\text{L}}} H \xleftarrow{d_{\text{R}}} K$, we obtain $f : G \to M, g : H \to M$ as the pushout of $c_{\text{R}}, d_{\text{L}}$ and hence the cospan $c\,; d : I \xrightarrow{c_{\text{L}}\,; f} M \xleftarrow{d_{\text{R}}\,; g} K$. Because of the pushout, we have that

$$\mathbb{M}_0' = \mathsf{fuse}_\theta(\mathbb{M}_0) = \langle M, (\text{di}_I\,; c_{\text{L}}\,; f) \odot (\text{di}_J\,; c_{\text{R}}\,; f) \odot (\text{di}_J\,; d_{\text{L}}\,; g) \odot (\text{di}_K\,; d_{\text{R}}\,; g)\rangle.$$

Finally,

$$\mathbb{M} = \mathsf{redef}_\sigma(\mathbb{M}_0') = \langle M, (\text{di}_I\,; c_{\text{L}}\,; f) \odot (\text{di}_K\,; d_{\text{R}}\,; g)\rangle.$$

On the other hand, a simple application of the definition reveals that

$$\text{bend}(c\,; d) = \langle M, (\text{di}_I\,; c_{\text{L}}\,; f) \odot (\text{di}_K\,; d_{\text{R}}\,; g)\rangle,$$

as required. □

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

We will now show the main theorem of this section: Languages recognized by automaton functors coincide with Courcelle-recognizable languages.

Theorem 4.14. Let J be a discrete graph. A set of graphs L is the $(\varnothing, J)$-language of some automaton functor $\mathcal{A}$ if and only if bend(L) is Courcelle recognizable.

Proof. ($\Rightarrow$) By Proposition 3.6, there exists a locally finite congruence

$$\equiv_{\mathrm{R}} = \{R_{J,K} \mid K, M \text{ are discrete graphs}\}$$

such that L is the union of some $R_{\varnothing,J}$-equivalence classes. We define

$$\mathrm{bend}(c) \equiv_{\mathrm{C}} \mathrm{bend}(d) \text{ if } c \equiv_{\mathrm{R}} d.$$

This is well defined because of Lemma 4.9 and the fact that bend is surjective. We now show that $\equiv_{\mathrm{C}}$ is a congruence in Courcelle's sense.

— *Redefinition and fusion.* Let $\mathbb{G} = \langle G, \zeta_{\mathbb{G}} \rangle$, $\mathbb{H} = \langle H, \zeta_{\mathbb{H}} \rangle$ be n-ary graphs such that $\mathbb{G} \equiv_{\mathrm{C}} \mathbb{H}$, and consider the cospans

$$c : \varnothing \xrightarrow{c_{\mathrm{L}}} G \xleftarrow{c_{\mathrm{R}}} J \qquad d : \varnothing \xrightarrow{d_{\mathrm{L}}} H \xleftarrow{d_{\mathrm{R}}} J$$

such that $\mathrm{bend}(c) = \mathbb{G}$ and $\mathrm{bend}(d) = \mathbb{H}$. By definition, it holds that $c \equiv_{\mathrm{R}} d$. By Lemma 4.11 (i), $\mathsf{redef}_\sigma(\mathbb{G}) = \mathrm{bend}(c\,; \mathit{redef}_s)$ and $\mathsf{redef}_\sigma(\mathbb{H}) = \mathrm{bend}(d\,; \mathit{redef}_s)$ (for suitably chosen s). Because $\equiv_{\mathrm{R}}$ is a congruence, and $c \equiv_{\mathrm{R}} d$, it holds, by definition, that $\mathsf{redef}_\sigma(\mathbb{G}) \equiv_{\mathrm{C}} \mathsf{redef}_\sigma(\mathbb{H})$, as required. In the case of fusion, we arrive at the result analogously, by using Lemma 4.11 (ii).

— *Disjoint union.* Let $\mathbb{G}_1, \mathbb{H}_1$ be n-ary graphs, and $\mathbb{G}_2, \mathbb{H}_2$ m-ary graphs, such that $\mathbb{G}_i \equiv_{\mathrm{C}} \mathbb{H}_i$, for $i \in \{1, 2\}$. Consider cospans c_1, c_2, d_1, d_2 such that $\mathrm{bend}(c_i) = \mathbb{G}_i$ and $\mathrm{bend}(d_i) = \mathbb{H}_i$. By definition, it holds that $c_i \equiv_{\mathrm{R}} d_i$. By Lemma 4.12, it holds that $\mathbb{G}_1 \oplus \mathbb{G}_2 = \mathrm{bend}(c_1 \oplus c_2)$ and $\mathbb{H}_1 \oplus \mathbb{H}_2 = \mathrm{bend}(d_1 \oplus d_2)$. Since it holds that

$$\begin{aligned} c_1 \oplus c_2 = c_1 \,;(\mathrm{id}_{J_1} \oplus c_2) &\equiv_{\mathrm{R}} d_1 \,;(\mathrm{id}_{J_1} \oplus c_2) \\ &= c_2 \,;(d_1 \oplus \mathrm{id}_{J_2}) \equiv_{\mathrm{R}} d_2 \,;(d_1 \oplus \mathrm{id}_{J_2}) = d_1 \oplus d_2, \end{aligned}$$

by definition it must be the case that $\mathbb{G}_1 \oplus \mathbb{G}_2 \equiv_{\mathrm{C}} \mathbb{H}_1 \oplus \mathbb{H}_2$.

($\Leftarrow$) Let the congruence $\equiv_{\mathrm{C}}$ in the sense of Courcelle be given. We define the relation $\equiv_{\mathrm{R}}$ as follows:

$$c \equiv_{\mathrm{R}} d \text{ if } \mathrm{bend}(c) \equiv_{\mathrm{C}} \mathrm{bend}(d).$$

We show that $\equiv_{\mathrm{R}}$ is a congruence. Let the following cospans be given:

$$\begin{aligned} c &: I \to G \leftarrow J & \mathrm{bend}(c) &= \mathbb{G} \\ d &: I \to H \leftarrow J & \mathrm{bend}(d) &= \mathbb{H} \\ p &: J \to M \leftarrow K & \mathrm{bend}(p) &= \mathbb{M}, \end{aligned}$$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

where $c \equiv_R d$. We need to show that $(c\,;p) \equiv_R (d\,;p)$. By definition, it holds that $\mathbb{G} \equiv_C \mathbb{H}$, and since $\equiv_C$ is a congruence, it holds that

$$\mathsf{redef}_\sigma(\mathsf{fuse}_\theta(\mathbb{G} \oplus \mathbb{M})) \equiv_C \mathsf{redef}_\sigma(\mathsf{fuse}_\theta(\mathbb{H} \oplus \mathbb{M})),$$

where σ, θ are as given in Lemma 4.13. By the same lemma, it holds that

$$\begin{aligned} \mathrm{bend}(c\,;p) &= \mathsf{redef}_\sigma(\mathsf{fuse}_\theta(\mathbb{G} \oplus \mathbb{M})) \\ \mathrm{bend}(d\,;p) &= \mathsf{redef}_\sigma(\mathsf{fuse}_\theta(\mathbb{H} \oplus \mathbb{M})), \end{aligned}$$

and therefore, by definition, $(c\,;p) \equiv_R (d\,;p)$, as required. □

Note that one of the reasons why the proof works is the fact that the category of cospans is compact-closed, which means that certain 'bending' laws, similar to the ones used in the proof above, hold.

5. A logic on subobjects

We will now describe a logic on subobjects that generalizes MSOGL in a categorical setting. The logic will be used to describe properties of objects of $\mathbf{C}$ by checking inclusion of subobjects and by quantifying over subobjects (either of fixed or arbitrary shape). In the comparison with monadic second-order logic, it will later become clear that variables standing for subobjects of fixed shape correspond to first-order variables, whereas variables standing for subojects of arbitrary shape, correspond to second-order variables.

We now introduce the syntax and semantics of the logic for a fixed category $\mathbf{C}$.

5.1. *Syntax*

Let *Var* be a countably infinite set of variables. A variable typing is a partial map $\tau : \mathit{Var} \rightharpoonup \mathit{Obj}(\mathbf{C}) \cup \{\Omega\}$. There are following two kinds of variables:

— *First-order variables* of sort T (where T is an arbitrary object of $\mathbf{C}$), representing subobjects of fixed structure.
— *Second-order variables* of sort Ω, representing arbitrary subobjects.

Unless otherwise indicated, first-order variables are denoted by lowercase letters (x, y, z) and second-order variables by capitals (X, Y, Z).

Subobject *expressions* are generated by the grammar $e := X \mid f \fatsemi x$, where X is a variable of sort Ω, x is a variable of sort T and $f : T' \rightarrowtail T$ is a mono. We then say that the expression $f \fatsemi x$ is of sort T'. Intuitively, f restricts the graph denoted by x to a subgraph T'. Variables of sort Ω cannot be precomposed with monos.

The set $\mathit{Form}(\tau)$ of formulas typed by τ is specified by the following grammar:

$$\begin{aligned} \mathit{Form}(\tau) := {} & e_1 \sqsubseteq e_2 \mid \mathit{Form}(\tau) \wedge \mathit{Form}(\tau) \mid \neg \mathit{Form}(\tau) \mid \\ & (\exists X : \Omega)\, \mathit{Form}(\tau[X \mapsto \Omega]) \mid (\exists x : T)\, \mathit{Form}(\tau[x \mapsto T]). \end{aligned}$$

Note that in a formula of the form $e_1 \sqsubseteq e_2$, the two expressions can have arbitrary sorts, but the variables in e_1, e_2 have to be typed according to τ. Furthermore, we define the

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

following usual abbreviations:

$$\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \bot \equiv \neg(x = x) \qquad (\forall X : \Omega)\,\varphi \equiv \neg(\exists X : \Omega)\,\neg\varphi$$
$$x = y \equiv x \sqsubseteq y \wedge y \sqsubseteq x \qquad \varphi \to \psi \equiv \neg\varphi \vee \psi \qquad (\forall x : T)\,\varphi \equiv \neg(\exists x : T)\,\neg\varphi$$

The set of free variables of a formula φ, denoted $FV(\varphi)$, is defined as usual.

Note that we do not define syntax for defining the morphism f and the object T in expressions and formulas of the form $f \fatsemi x$ and $(\exists \varphi : T)$. The exact syntax needed for this depends on the category, and falls outside the scope of this paper.

5.2. *Semantics*

Let φ be a formula typed by variable typing τ and B a $\mathbf{C}$-object. A B-valuation η for φ is a function which assigns

— to each $x \in FV(\varphi)$, with $\tau(x) = T$ (where $T \neq \Omega$), a mono $v_x : T \rightarrowtail B$; and
— to each $X \in FV(\varphi)$, with $\tau(X) = \Omega$, a mono $v_X : V \rightarrowtail B$ (where V is arbitrary).

Now we can define the semantics of formulas of the logic. Let φ be a subobject formula typed by τ and let B be a $\mathbf{C}$-object. We say that $B, \eta \models \varphi$, for some B-valuation η, whenever

— $B, \eta \models X \sqsubseteq Y$ if $\eta(X) \sqsubseteq \eta(Y)$;
— $B, \eta \models f \fatsemi x \sqsubseteq g \fatsemi y$ if $f ; \eta(x) \sqsubseteq g ; \eta(y)$;
— $B, \eta \models f \fatsemi x \sqsubseteq Y$ if $f ; \eta(x) \sqsubseteq \eta(Y)$;
— $B, \eta \models X \sqsubseteq g \fatsemi y$ if $\eta(X) \sqsubseteq g ; \eta(y)$;
— $B, \eta \models \varphi_1 \wedge \varphi_2$ if $B, \eta \models \varphi_1$ and $B, \eta \models \varphi_2$;
— $B, \eta \models \neg\varphi$ if $B, \eta \not\models \varphi$;
— $B, \eta \models (\exists x : T)\,\varphi$ if there is a mono $v : T \rightarrowtail B$ such that $B, \eta[x \mapsto v] \models \varphi$;
— $B, \eta \models (\exists X : \Omega)\,\varphi$ if there is a mono $v : V \rightarrowtail B$ such that $B, \eta[X \mapsto v] \models \varphi$.

Furthermore, for a closed formula φ, we write $B \models \varphi$ whenever $B, \eta \models \varphi$ for the empty valuation η. In the definition above, the relation $\sqsubseteq$ on the right-hand side stands for subobject inclusion as defined in Section 2.1. Note that the definition works for any category. However, the results of Section 7 (translation of formulas into automaton functors) will only be valid for hereditary pushout categories (which satisfy some additional conditions).

Example 5.1 (Logic on subobjects in arbitrary categories). Let $\mathbf{C}$ be an arbitrary category. We can define the following formulas:

— The join of two expressions:

$$(e = e_1 \sqcup e_2) := e_1 \sqsubseteq e \wedge e_2 \sqsubseteq e \wedge (\forall X : \Omega)\,\big((e_1 \sqsubseteq X \wedge e_2 \sqsubseteq X) \to e \sqsubseteq X\big).$$

— Irreducible objects:

$$irr(e) := (\forall X : \Omega)\,(\forall Y : \Omega)\,(e = X \sqcup Y \to (e = X \vee e = Y)).$$

The formula expresses that a subobject e is irreducible, i.e., that it can only be expressed as the join of two subobjects in a trivial way. In the case of graphs, this means that e is either a single node or a single edge (with potentially fused nodes).

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Example 5.2 (Logic on subobjects for graphs). In this example, we use the category of unlabelled directed multigraphs. Recall from Section 2.2 that D_1 is the graph which has one node and no edges, E is the graph which has two nodes connected by a single edge and $src, tgt : D_1 \to E$ map the node of D_1 to the source and target node of the single edge in E, respectively.

— There exist two edges which have the same target node:

$$(\exists x : E)\,(\exists y : E)\,(tgt \fatsemi x = tgt \fatsemi y).$$

— The subgraph X is closed under reachability, that is, if v is a node such that $v \in X$ and w is a node reachable from x, then it must hold that $w \in X$:

$$RC(X : \Omega) := (\forall y : E)\,(src \fatsemi y \sqsubseteq X \to tgt \fatsemi y \sqsubseteq X).$$

— There exists a path from node x to node y. This is formalized as follows: For all subobjects X which are closed under reachability and contain x, it must hold that X contains y as well. Formally,

$$Path(x, y : D_1) := (\forall Z : \Omega)\,\big((\mathrm{id} \fatsemi x \sqsubseteq Z \wedge RC(Z)) \to \mathrm{id} \fatsemi y \sqsubseteq Z\big).$$

6. Comparison to monadic second-order graph logic

Consider the category **Graph** as presented on page 1294. We show that for this category the logic on subobjects has the same expressive power as MSOGL (Courcelle 1990, 1997). We do this by defining translations from monadic second-order logic to the logic on subobjects and vice versa, and proving that a graph satisfies a formula if and only if it satisfies the translation of the formula.

6.1. *Monadic second-order graph logic*

First, we define MSOGL, mainly in order to fix notation and terminology. This logic is one of the most important specification logics for graphs. To be more precise, we are here using MSO2, which allows quantification over both nodes and edges, different from MSO1, where only quantification over nodes is allowed.

MSOGL is a sorted second-order logic with four kinds of variables: First-order node variables (range over nodes), first-order edge variables (range over edges), second-order node variables (range over sets of nodes) and second-order edge variables (range over sets of edges). As a notational convention, first-order variables will be denoted by lowercase letters (x, y, z) and second-order variables by capitals (X, Y, Z). The syntax of MSOGL is given by the following grammar:

$$\varphi := \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid (\exists X : V)\,\varphi \mid (\exists X : E)\,\varphi \mid (\exists x : v)\,\varphi \mid (\exists x : e)\,\varphi \mid$$
$$x = y \mid x \in X \mid edge_A(x, y_1, \ldots, y_{ar(A)}),$$

where typing must be respected, that is, in formulas of the form $x = y$, both variables have the same type and in formulas of the form $x \in X$ it holds that x is a first-order node (edge) variable and X a second-order node (edge) variable. Formulas of the form

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

$edge_A(x, y_1, \ldots, y_{ar(A)})$ denote that the edge x has label A and is adjacent to the nodes $y_1, \ldots, y_n$.

A graph $G = \langle V_G, E_G, att_G, lab_G \rangle$ satisfies a formula φ, written $G \models \varphi$, if there exists a valuation θ, mapping first-order variables to nodes and edges of G and second-order variables to sets of nodes and sets of edges of G, such that $G, \theta \models_{\mathrm{M}} \varphi$, where

— $G, \theta \models_{\mathrm{M}} x = y$ if $\theta(x) = \theta(y)$,
— $G, \theta \models_{\mathrm{M}} x \in X$ if $\theta(x) \in \theta(X)$,
— $G, \theta \models_{\mathrm{M}} edge_A(x, y_1, \ldots, y_n)$ if $lab_G(\theta(x)) = A$ and $att_G(\theta(x)) = \theta(y_1) \ldots \theta(y_n)$,
— $G, \theta \models_{\mathrm{M}} \varphi_1 \wedge \varphi_2$ if $G, \theta \models_{\mathrm{M}} \varphi_1$ and $G, \theta \models_{\mathrm{M}} \varphi_2$,
— $G, \theta \models_{\mathrm{M}} \neg\varphi$ if $G, \theta \not\models_{\mathrm{M}} \varphi$,
— $G, \theta \models_{\mathrm{M}} (\exists x : v)\, \varphi$ if there is a $v' \in V_G$ such that $G, \theta[x \mapsto v'] \models_{\mathrm{M}} \varphi$,
— $G, \theta \models_{\mathrm{M}} (\exists x : e)\, \varphi$ if there is a $e' \in E_G$ such that $G, \theta[x \mapsto e'] \models_{\mathrm{M}} \varphi$,
— $G, \theta \models_{\mathrm{M}} (\exists X : V)\, \varphi$ if there is a $V \subseteq V_G$ such that $G, \theta[X \mapsto V] \models_{\mathrm{M}} \varphi$,
— $G, \theta \models_{\mathrm{M}} (\exists X : E)\, \varphi$ if there is a $E \subseteq E_G$ such that $G, \theta[X \mapsto E] \models_{\mathrm{M}} \varphi$,

In Courcelle (1990), an extension to monadic second-order logic is presented which also considers cardinality constraints of the form $card_{n,p}(X)$, expressing that the set represented by X contains n elements modulo p, which are omitted here for simplicity. It is currently not entirely clear to us how to integrate a similar predicate into the logic on subobjects in a natural way.

6.2. *From monadic second-order logic to the logic on subobjects*

We define a translation $[\![\cdot]\!]_{\mathrm{S}}$ from formulas of MSOGL to formulas of the logic on subobjects. For this, we define an edge typing as a map ζ from first-order edge variables (labelled with A) to epimorphisms (with domain E_A, where E_A is defined in Section 2.2). We define the translation function $[\![\varphi]\!]_{\mathrm{S}} = [\![\varphi]\!]^{\emptyset}_{\mathrm{S}}$, where, for an edge typing ζ, $[\![\varphi]\!]^{\zeta}_{\mathrm{S}}$ is inductively defined as follows:

$$
\begin{aligned}
[\![\neg\varphi]\!]^{\zeta}_{\mathrm{S}} &:= \neg[\![\varphi]\!]^{\zeta}_{\mathrm{S}} & [\![x = y]\!]^{\zeta}_{\mathrm{S}} &:= x = y \\
[\![\varphi \wedge \psi]\!]^{\zeta}_{\mathrm{S}} &:= [\![\varphi]\!]^{\zeta}_{\mathrm{S}} \wedge [\![\psi]\!]^{\zeta}_{\mathrm{S}} & [\![x \in X]\!]^{\zeta}_{\mathrm{S}} &:= x \sqsubseteq X \\
[\![(\exists X : V)\, \varphi]\!]^{\zeta}_{\mathrm{S}} &:= (\exists X : \Omega)\, [\![\varphi]\!]^{\zeta}_{\mathrm{S}} & [\![(\exists x : v)\, \varphi]\!]^{\zeta}_{\mathrm{S}} &:= (\exists x : D_1)\, [\![\varphi]\!]^{\zeta}_{\mathrm{S}} \\
[\![(\exists X : E)\, \varphi]\!]^{\zeta}_{\mathrm{S}} &:= (\exists X : \Omega)\, [\![\varphi]\!]^{\zeta}_{\mathrm{S}} & [\![(\exists x : e)\, \varphi]\!]^{\zeta}_{\mathrm{S}} &:= \bigvee_{A \in \Sigma} \bigvee_{f \in Epi(A)} (\exists x : cod(f))\, [\![\varphi]\!]^{\zeta[x \mapsto f]}_{\mathrm{S}}
\end{aligned}
$$

$$
[\![edge_A(x, y_1, \ldots, y_n)]\!]^{\zeta}_{\mathrm{S}} := \begin{cases} \bigwedge_{1 \leqslant i \leqslant ar(A)} (e^i_A \,;\, \zeta(x)) \fatsemi x = y_i & \text{if } dom(\zeta(x)) = E_A \\ \bot & \text{otherwise.} \end{cases}
$$

A key difference between MSOGL and the logic on subobjects is that second-order variables of the first are evaluated to sets of only nodes or of only edges, whereas second-order variables in the second are evaluated to arbitrary subobjects potentially consisting of nodes and edges. The following technical lemma establishes that the parts which are 'of the wrong type' can be ignored, which is needed in the correctness proof of the translation of MSOGL formulas to formulas of the logic on subobjects.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Lemma 6.1. Let φ be a formula of MSOGL, let G be a graph and η a G-valuation for $[\![\varphi]\!]_{\mathrm{S}}^{\zeta}$.

i. Suppose X is a second-order node variable which occurs free in φ (and therefore is in the domain of η), and let η' be constructed from η by restricting $\eta(X)$, which is a subgraph of G, to its nodes. Then, $G, \eta \models [\![\varphi]\!]_{\mathrm{S}}^{\zeta}$ if and only if $G, \eta' \models [\![\varphi]\!]_{\mathrm{S}}^{\zeta}$.
ii. Suppose X is second-order edge variable which occurs free in φ, and let η' be constructed from η by restricting $\eta(X)$ only to its edges and the nodes connected to them. Then, $G, \eta \models [\![\varphi]\!]_{\mathrm{S}}^{\zeta}$ if and only if $G, \eta' \models [\![\varphi]\!]_{\mathrm{S}}^{\zeta}$.

Proof. By induction on the structure of φ, using in the base case, the observation that $\varphi \equiv (x = y)$ can only hold if x and y are both first-order node or both first-order edge variables, and $\varphi \equiv (x \in X)$ can only hold if x is a first-order node variable and X a second-order node variable, or x is a first-order edge variable and X a second-order edge variable. □

Proposition 6.2. Let G be a graph, and φ a closed formula of monadic second-order logic. Then $G \models_{\mathrm{M}} \varphi$ if and only if $G \models [\![\varphi]\!]_{\mathrm{S}}$.

Proof. For each valuation θ and each first-order edge variable x in the domain of θ, let G_x^{θ} be the graph G restricted to the edge $\theta(x)$ and the nodes connected to it. Now define $f_x^{\theta} : E_A \to G_x^{\theta}$, where $A = lab_G(\theta(x))$, as the morphism which maps the single edge in its domain to $\theta(x)$, and the nodes accordingly. By construction, f_x^{θ} is an epimorphism. Define the *edge typing* et_{θ} as follows: $et_{\theta}(x) = f_x^{\theta}$, for each first-order edge variable x in the domain of θ.

In order to complete the proof, we also define for each MSOL-valuation θ a valuation $[\![\theta]\!]_{\mathrm{S}}$ for the logic on subobjects as follows:

— For each first-order node variable $y : v$ in the domain of θ, $[\![\theta]\!]_{\mathrm{S}}(y)$ is the morphism $v_y : D_1 \to G$, which maps the node in D_1 to $\theta(y)$.
— For each first-order edge variable $x : e$ in the domain of θ, $[\![\theta]\!]_{\mathrm{S}}(x)$ equals the embedding $v_x : G_x^{\theta} \to G$.
— For each second-order node variable $Y : V$ in the domain of θ, $[\![\theta]\!]_{\mathrm{S}}(Y)$ equals an (arbitrary) injective morphism $v_Y : D_k \to G$, where $k = |\theta(Y)|$, such that $\{v \mid \exists v' \in D_k, v_Y(v') = v\} = \theta(Y)$.
— For each second-order edge variable $X : E$ in the domain of θ, $[\![\theta]\!]_{\mathrm{S}}(X)$ equals an injective morphism $v_X : H \to G$, where H is a graph without isolated nodes, such that $\{e \mid \exists e' \in H, v_X(e') = e\} = \theta(X)$.

Now, we prove by induction the following stronger proposition: $G, \theta \models_{\mathrm{M}} \varphi$ if and only if $G, [\![\theta]\!]_{\mathrm{S}} \models [\![\varphi]\!]_{\mathrm{S}}^{et_{\theta}}$. For convenience we assume, without loss of generality, that all quantifiers bind different variables.

— *Base case.* If $\varphi \equiv (x = y)$ or $\varphi \equiv (x \in X)$, then the result follows trivially.
Let $\varphi \equiv edge_A(x, y_1, \ldots, y_n)$.
($\Rightarrow$): Suppose $G, \theta \models_{\mathrm{M}} \varphi$. By definition, it holds that $lab_G(\theta(x)) = A$ and $att_G(\theta(x)) = \theta(y_1) \ldots \theta(y_n)$. Take $1 \leqslant i \leqslant n$. By construction, it holds that $e_A^i \,;et_{\theta}(x) \,;[\![\theta]\!]_{\mathrm{S}}(x) =$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

$[\![\theta]\!]_S(y_i)$. Therefore, $G, [\![\theta]\!]_S \models e^i_A ; et_\theta(x) ; x = y_i$. Since this holds for all appropriate i, $G, [\![\theta]\!]_S \models [\![\varphi]\!]^{et_\theta}_S$.

($\Leftarrow$): Suppose $G, [\![\theta]\!]_S \models [\![\varphi]\!]^{et_\theta}_S$. By definition, it holds for all $1 \leqslant i \leqslant n$ that $e^i_A ; et_\theta(x) ; [\![\theta]\!]_S(x) = [\![\theta]\!]_S(y_i)$. By construction of et_θ, this means that $att_G(\theta(x)) = \theta(y_1) \dots \theta(y_n)$. The arity and node label are correct by construction of $[\![\varphi]\!]^{et_\theta}_S$. Therefore, $G, \theta \models_M \varphi$.

— In the case of boolean operators, the result follows trivially from the induction hypothesis.

— Suppose $\varphi \equiv (\exists x : v)\, \varphi'$.

($\Rightarrow$): Assume $G, \theta \models_M (\exists x : v)\, \varphi'$. By definition, there exists a $v \in V_G$ such that $G, \theta[x \mapsto v] \models \varphi'$. Let $\theta' := \theta[x \mapsto v]$. By the induction hypothesis, $G, [\![\theta']\!]_S \models [\![\varphi']\!]^{et_{\theta'}}_S$. By definition, $G, \eta \models (\exists x : D_1)\, [\![\varphi']\!]^{et_{\theta'}}_S$, for some η such that $\eta[x \mapsto w] = [\![\theta']\!]_S$. By the convenience assumption mentioned above, $\eta = [\![\theta]\!]_S$. Since et_θ only uses edge variables, it holds that $et_\theta = et_{\theta'}$. Therefore, $G, [\![\theta]\!]_S \models (\exists x : D_1)\, [\![\varphi']\!]^{et_\theta}_S \equiv [\![(\exists x : v)\, \varphi']\!]^{et_\theta}_S$.

($\Leftarrow$): Assume $G, [\![\theta]\!]_S \models [\![(\exists x : v)\, \varphi']\!]^{et_\theta}_S \equiv (\exists x : D_1)\, [\![\varphi']\!]^{et_\theta}_S$. By definition, there is a $v_x : V \rightarrowtail G$ such that $G, [\![\theta]\!]_S[x \mapsto v_x] \models [\![\varphi']\!]^{et_\theta}_S$. Let w be the single node in D_1 and $v = v_x(w)$. By construction, $[\![\theta]\!]_S[x \mapsto v_x] = [\![\theta[x \mapsto v]]\!]_S$. Since $et_\theta = et_{\theta'}$, we obtain $G, [\![\theta[x \mapsto v]]\!]_S \models [\![\varphi']\!]^{et_{\theta[x \mapsto v]}}_S$, and thus by the induction hypothesis that $G, \theta[x \mapsto v] \models_M \varphi'$. From this, it follows by definition that $G, \theta \models_M (\exists x : v)\, \varphi'$.

— Let $\varphi \equiv (\exists x : e)\, \varphi'$.

($\Rightarrow$): Suppose $G, \theta \models_M (\exists x : e)\, \varphi'$. By definition, there is an $e \in E_G$ such that $G, \theta[x \mapsto e] \models_M \varphi'$. Let $\theta' := \theta[x \mapsto e]$. By the induction hypothesis, $G, [\![\theta']\!]_S \models [\![\varphi']\!]^{et_{\theta'}}_S$. By definition, it holds that $G, \eta \models (\exists x : G^{\theta'}_x)\, [\![\varphi']\!]^{et_{\theta'}}_S$, for some η such that $\eta[x \mapsto w] = [\![\theta']\!]_S$. By construction, $G^{\theta'}_x \in \{cod(f) \mid f \in Epi(A)\}$, where A is the label of e, and therefore

$$G, \eta \models \bigvee_{A \in \Sigma} \bigvee_{f \in Epi(A)} (\exists x : cod(f))\, [\![\varphi']\!]^{et_\theta[x \mapsto f]}_S.$$

Since, by the convenience assumption, it holds that $\eta = [\![\theta]\!]_S$, it holds that $G, [\![\theta]\!]_S \models [\![(\exists x : e)\, \varphi']\!]^{et_\theta}_S$.

($\Leftarrow$): Assume $G, [\![\theta]\!]_S \models [\![(\exists x : e)\, \varphi']\!]^{et_\theta}_S$. By definition, it holds that

$$G, [\![\theta]\!]_S \models \bigvee_{A \in \Sigma} \bigvee_{f \in Epi(A)} (\exists x : cod(f))\, [\![\varphi']\!]^{et_\theta[x \mapsto f]}_S.$$

Choose $A \in \Sigma$ and $f \in Epi(A)$ such that $G, [\![\theta]\!]_S \models (\exists x : cod(f))\, [\![\varphi']\!]^{et_\theta[x \mapsto f]}_S$. By definition, there is a $v_x : T \rightarrowtail G$ such that $G, [\![\theta]\!]_S[x \mapsto v_x] \models [\![\varphi']\!]^{et_\theta[x \mapsto f]}_S$. Since the type of x in $[\![\varphi']\!]^{et_\theta[x \mapsto f]}_S$ is given by $f, f ; v_x$ is well defined. Let d be the unique edge in E_A, and take $e = (f ; v_x)(d)$. Let $\theta' := \theta[x \mapsto e]$. By construction, it holds (up to isomorphism) that $[\![\varphi']\!]^{et_\theta[x \mapsto f]}_S = [\![\varphi']\!]^{et_{\theta'}}_S$. Also by construction, it holds that $[\![\theta]\!]_S[x \mapsto v_x] = \theta'$. Therefore, $G, [\![\theta']\!]_S \models [\![\varphi']\!]^{et_{\theta'}}_S$. By the induction hypothesis, $G, \theta' \models_M \varphi'$, and therefore, by definition, $G, \theta \models_M (\exists x : e)\, \varphi'$.

— Let $\varphi \equiv (\exists X : V)\, \varphi'$.

($\Rightarrow$): Suppose $G, \theta \models_M (\exists X : V)\, \varphi'$. By definition, there is a $V \subseteq V_G$ such that $G, \theta' \models_M \varphi'$, where $\theta' := \theta[X \mapsto V]$. By the induction hypothesis, $G, [\![\theta']\!]_S \models [\![\varphi']\!]^{et_{\theta'}}_S$. Now it holds, by definition, that $G, \eta \models (\exists X : \Omega)\, [\![\varphi']\!]^{et_{\theta'}}_S$ for some η such that $\eta[X \mapsto V] =$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

$[\![\theta']\!]_S$. Since by the convenience assumption, $\eta = [\![G, \theta]\!]_S$ and $[\![\varphi']\!]_S^{et_{\theta'}} = [\![\varphi']\!]_S^{et_\theta}$, it holds that $[\![G, \theta]\!]_S \models (\exists X : \Omega)\, [\![\varphi']\!]_S^{et_\theta} = [\![(\exists X : V)\, \varphi']\!]_S^{et_\theta}$.
($\Leftarrow$): Suppose $G, [\![\theta]\!]_S \models [\![(\exists X : V)\, \varphi']\!]_S^{et_\theta}$. By definition, there is a v_X such that $G, [\![\theta]\!]_S[X \mapsto v_X] \models [\![\varphi']\!]_S^{et_\theta}$. By Lemma 6.1 (i), we may assume that the domain of v_X does not contain edges. Let V be the nodes in the image of v_X. Since X is not an edge variable, $et_\theta = et_{\theta[X \mapsto V]}$, and it follows from the induction hypothesis that $G, \theta[X \mapsto V] \models_M \varphi'$, and therefore it holds by definition, that $G, \theta \models_M (\exists X : V)\, \varphi'$.

— Let $\varphi \equiv (\exists X : E)\, \varphi'$.
($\Rightarrow$): Similar to the left-to-right direction in the previous case.
($\Leftarrow$): Suppose $G, [\![\theta]\!]_S \models [\![(\exists X : E)\, \varphi']\!]_S^{et_\theta}$. By definition, this means that there is a $v_X : V \rightarrowtail G$ such that $G, [\![\theta]\!]_S[X \mapsto v_X] \models [\![\varphi']\!]_S^{et_\eta}$. By Lemma 6.1 (ii), we may assume that the domain of v_X does not contain isolated nodes. Let H be the image of v_X (without loss of generality, we may assume that H is subgraph of G, i.e., $V_H \subseteq V_G$ and $E_H \subseteq E_G$). Since X is not an edge variable, $et_\theta = et_{\theta[X \mapsto E_H]}$, and it follows from the induction hypothesis, that $G, \theta[X \mapsto V] \models_M \varphi'$, and therefore it holds by definition, that $G, \theta \models_M (\exists X : V)\, \varphi'$. □

6.3. *From the logic on subobjects to monadic second-order logic*

We define a translation $[\![\cdot]\!]_M$ from formulas of our logic to formulas of MSOGL. The main difference between the two logics is that in our logic, we can quantify over arbitrary subobjects, while in MSOGL, we can only quantify over nodes, edges, sets of nodes and sets of edges. Thus, a single quantification in the logic on subobjects will in general correspond to more than one quantification in MSOGL. In order to make sure that the multiple variables in an MSOGL-formula evaluate to a possible subobject, we need to express the following two 'consistency properties' as MSOGL-formulas:

— Let T be a graph and $f : \mathcal{X} \to (V_T \cup E_T)$ a bijection between first-order variables and nodes and edges of T. Then,

$$\begin{aligned} struct_f(T) &:= \bigwedge_{e \in E_T} edge_{lab_T(e)}(f^{-1}(e), (f^{-1})^*(att_T(e))) \\ &\quad \wedge pneq(f^{-1}(V_T)) \wedge pneq(f^{-1}(E_T)) \\ pneq(\{x_1, \ldots, x_n\}) &:= \bigwedge_{1 \leqslant i \leqslant n} \bigwedge_{1 \leqslant j \leqslant n, j \neq i} \neg(x_i = x_j). \end{aligned}$$

The formula $struct_f(T)$ expresses that the nodes and edges assigned to the variables in the codomain of f build a graph isomorphic to T.

— Let X_V be a second-order node variable and X_E a second-order edge variable. Then we define

$$\begin{aligned} cons(X_E, X_V) &:= \bigwedge_{A \in \Sigma} cons_A(X_E, X_V) \\ cons_A(X_E, X_V) &:= (\forall x : e)\, \big(x \in X_E \to \\ &\qquad (\forall y_1, \ldots, y_{ar(A)} : v)\, \big(edge_A(x, y_1, \ldots, y_{ar(A)}) \to \\ &\qquad\quad y_1 \in X_V \wedge \cdots \wedge y_{ar(A)} \in X_V\big)\big). \end{aligned}$$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

The formula $cons(X_E, X_V)$ expresses that all nodes adjacent to an edge in X_E must be in X_V, i.e., the two sets are consistent.

A *variable mapping* ξ is a function which maps

— each free first-order variable x of type T in the formula over subobjects to a bijection from a subset of first-order node and edge variables in the MSOGL-formula to the nodes and edges of T, and

— each free second-order variable X of the source formula to a pair of a second-order node and a second-order edge variable in the target formula.

Now, we define $[\![\varphi]\!]_{\mathrm{M}} = [\![\varphi]\!]^{\emptyset}_{\mathrm{M}}$, where $[\![\varphi]\!]^{\xi}_{\mathrm{M}}$, parameterized by a variable mapping ξ, is inductively defined as follows:

$$
\begin{aligned}
[\![\neg\varphi]\!]^{\xi}_{\mathrm{M}} &:= \neg[\![\varphi]\!]^{\xi}_{\mathrm{M}} \\
[\![\varphi \wedge \psi]\!]^{\xi}_{\mathrm{M}} &:= [\![\varphi]\!]^{\xi}_{\mathrm{M}} \wedge [\![\psi]\!]^{\xi}_{\mathrm{M}} \\
[\![(\exists X : \Omega)\, \varphi]\!]^{\xi}_{\mathrm{M}} &:= (\exists X_E : E)\,(\exists X_V : V)\, cons(X_E, X_V) \wedge [\![\varphi]\!]^{\xi[X \mapsto \langle X_E, X_V\rangle]}_{\mathrm{M}} \\
&\quad \text{where } X_E \text{ and } X_V \text{ are fresh variables.} \\
[\![(\exists x : T)\, \varphi]\!]^{\xi}_{\mathrm{M}} &:= (\exists x_1, \ldots, x_m : e)\,(\exists y_1, \ldots, y_n : v)\, struct_{\xi'(x)}(T) \wedge [\![\varphi]\!]^{\xi'}_{\mathrm{M}} \\
&\quad \text{where } T \text{ is a graph with } m \text{ edges } (e_1, \ldots, e_m) \text{ and } n \text{ nodes} \\
&\quad (v_1, \ldots, v_n), \text{ the } x_i \text{ and } y_i \text{ are fresh variables, and} \\
&\qquad \xi' := \xi[x \mapsto \{\langle x_1, e_1\rangle, \ldots, \langle x_m, e_m\rangle, \langle y_1, v_1\rangle, \ldots, \langle y_n, v_n\rangle\}]. \\
[\![X \sqsubseteq Y]\!]^{\xi}_{\mathrm{M}} &:= (\forall x : e)\left(x \in X_E \to x \in Y_E\right) \wedge (\forall x : v)\left(x \in X_V \to x \in Y_V\right) \\
&\quad \text{where } (X_E, X_V) := \xi(X) \text{ and } (Y_E, Y_V) := \xi(Y). \\
[\![f \fatsemi x \sqsubseteq X]\!]^{\xi}_{\mathrm{M}} &:= x_1 \in X_E \wedge \cdots \wedge x_m \in X_E \wedge y_1 \in X_V \wedge \cdots \wedge y_n \in X_V \\
&\quad \text{where } (X_E, X_V) := \xi(X), \\
&\qquad \{x_1, \ldots, x_m\} := \{u \mid \exists e : \langle u, f(e)\rangle \in \xi(x)\} \text{ and} \\
&\qquad \{y_1, \ldots, y_n\} := \{w \mid \exists v : \langle w, f(v)\rangle \in \xi(x)\}. \\
[\![X \sqsubseteq f \fatsemi x]\!]^{\xi}_{\mathrm{M}} &:= (\forall x' : e)\left(x' \in X_E \to (x' = x_1 \vee \cdots \vee x' = x_m)\right) \wedge \\
&\quad (\forall y' : v)\left(y' \in X_V \to (y' = y_1 \vee \cdots \vee y' = y_n)\right) \\
&\quad \text{where } (X_E, X_V) := \xi(X), \\
&\qquad \{x_1, \ldots, x_m\} := \{u \mid \exists e : \langle u, f(e)\rangle \in \xi(x)\} \text{ and} \\
&\qquad \{y_1, \ldots, y_n\} := \{w \mid \exists v : \langle w, f(v)\rangle \in \xi(x)\}. \\
[\![f \fatsemi x \sqsubseteq g \fatsemi x']\!]^{\xi}_{\mathrm{M}} &:= (x_1 = x'_1 \vee \cdots \vee x_1 = x'_p) \wedge \cdots \wedge (x_m = x'_1 \vee \cdots \vee x_m = x'_p) \wedge \\
&\quad (y_1 = y'_1 \vee \cdots \vee y_1 = y'_q) \wedge \cdots \wedge (y_n = y'_1 \vee \cdots \vee y_n = y'_q) \\
&\quad \text{where } \{x_1, \ldots, x_m\} := \{u \mid \exists e : \langle u, f(e)\rangle \in \xi(x)\}, \\
&\qquad \{y_1, \ldots, y_n\} := \{w \mid \exists v : \langle w, f(v)\rangle \in \xi(x)\}, \\
&\qquad \{x'_1, \ldots, x'_p\} := \{u \mid \exists e : \langle u, f(e)\rangle \in \xi(x')\} \text{ and} \\
&\qquad \{y'_1, \ldots, y'_q\} := \{w \mid \exists v : \langle w, f(v)\rangle \in \xi(x')\}.
\end{aligned}
$$

Proposition 6.3. Let G be a graph, and φ a closed formula of the logic on subobjects. Then, $G \models \varphi$ if and only if $G \models_{\mathrm{M}} [\![\varphi]\!]_{\mathrm{M}}$.

Proof. We define the translation $[\![\eta]\!]^{\xi}_{\mathrm{M}}$, which translates a valuation of the logic on subobjects to a MSOGL-valuation, with a variable mapping as parameter, as follows:

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

— For each first-order variable x in the domain of η, and each variable x' in the domain of $\xi(x)$, $[\![\eta]\!]^{\xi}_{\mathrm{M}}(x') := \xi(x)(x')$.
— For each second-order variable X in the domain of η, $[\![\eta]\!]^{\xi}_{\mathrm{M}}(X_V) := V$ and $[\![\eta]\!]^{\xi}_{\mathrm{M}}(X_Y) := E$, where $\xi(X) = \langle X_V, X_E \rangle$, V is the set of nodes in the image of $\eta(X)$ and E is the set of edges in the image of $\eta(X)$.

We now prove, by induction on the structure of φ, the following stronger property: Let G be a graph, η a valuation and ξ a variable mapping, then $G, \eta \models \varphi$ if and only if $G, [\![\eta]\!]^{\xi}_{\mathrm{M}} \models_{\mathrm{M}} [\![\varphi]\!]^{\xi}_{\mathrm{M}}$. For convenience, we assume here, without loss of generality, that all monos in the codomain of η are in fact embeddings, that is, $\eta(x)(p) = p$ for all nodes and edges p. Furthermore, we assume for convenience that all quantifiers bind different variables.

— In the base case, the result follows easily. In the ($\Rightarrow$)-direction, it is easy to show that the translated formula holds, while in the ($\Leftarrow$)-direction, the inclusion morphism can be easily constructed.
— For boolean operations, the result follows easily from the induction hypothesis.
— Let $\varphi \equiv (\exists X : \Omega)\, \varphi'$. That means that

$$[\![\varphi]\!]^{\xi}_{\mathrm{M}} \equiv (\exists X_E : E)\, (\exists X_V : V)\, cons(X_E, X_V) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}},$$

where $\xi' = \xi[X \mapsto \langle X_E, X_V \rangle]$.
($\Rightarrow$): Assume $G, \eta \models \varphi$. By definition, there is a $v_X : V \rightarrowtail G$ such that $G, \eta[X \mapsto v_X] \models \varphi'$, where $\eta' = \eta[\mapsto v_X]$. By the induction hypothesis, it holds that $G, [\![\eta']\!]^{\xi'}_{\mathrm{M}} \models_{\mathrm{M}} [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$. By construction, all nodes in $[\![\eta']\!]^{\xi'}_{\mathrm{M}}(X_V)$ are connected to an edge in $[\![\eta']\!]^{\xi'}_{\mathrm{M}}(X_E)$, and thus $G, [\![\eta']\!]^{\xi'}_{\mathrm{M}} \models_{\mathrm{M}} cons(X_E, X_V) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$. Therefore, it holds by definition that $G, [\![\eta]\!]^{\xi}_{\mathrm{M}} \models_{\mathrm{M}} [\![\varphi]\!]^{\xi}_{\mathrm{M}}$, as required.
($\Leftarrow$): Let $\theta = [\![\eta]\!]^{\xi}_{\mathrm{M}}$ and assume that $G, \theta \models_{\mathrm{M}} [\![\varphi]\!]^{\xi}_{\mathrm{M}}$. By definition, there are $E \subseteq E_G$ and $V \subseteq V_G$ such that $G, \theta' \models_{\mathrm{M}} cons(X_E, X_V) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$, where $\theta' = \theta[X_E \mapsto E, X_V \mapsto V]$. Let H be the subgraph of G consisting of the nodes in X_V and the edges in X_E, and let $h : H \rightarrowtail G$ be the embedding of H into G. Because $G, \theta' \models_{\mathrm{M}} cons(X_E, X_V)$, H and h are well defined. By construction, $[\![\eta[X \mapsto h]]\!]^{\xi'}_{\mathrm{M}} = \theta'$ and therefore it follows from the induction hypothesis that $G, \eta[X \mapsto h] \models \varphi'$, and thus that $G, \eta \models \varphi$, as required.
— Let $\varphi \equiv (\exists x : T)\, \varphi'$. This means that

$$[\![\varphi]\!]^{\xi}_{\mathrm{M}} \equiv (\exists x_1, \ldots, x_n : e)\, (\exists y_1, \ldots, y_m : v)\, struct_{\xi'(x)}(T) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}},$$

where $\xi'(y) = \xi(y)$ for $y \neq x$, and $\xi(x)$ maps the variables x_i, y_j (for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$) to the respective edges and nodes of T.
($\Rightarrow$): Assume $G, \eta \models \varphi$. By definition, there is a $v : T \rightarrowtail G$, such that $G, \eta' \models \varphi'$, where $\eta' = \eta[x \mapsto v]$. By the induction hypothesis, it holds that $G, [\![\eta']\!]^{\xi'}_{\mathrm{M}} \models_{\mathrm{M}} [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$. By construction, the image under $[\![\eta']\!]^{\xi'}_{\mathrm{M}}$ of $x_1, \ldots, x_n, y_1, \ldots, y_m$ is isomorphic to T, and therefore $G, [\![\eta']\!]^{\xi'}_{\mathrm{M}} \models_{\mathrm{M}} struct_{\xi'(x)}(T) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$. By construction, $[\![\eta']\!]^{\xi'}_{\mathrm{M}}$ differs from $[\![\eta]\!]^{\xi}_{\mathrm{M}}$ only in that the latter assigns nodes and edges to $x_1, \ldots, x_n, y_1, \ldots, y_m$. Therefore $G, [\![\eta]\!]^{\xi}_{\mathrm{M}} \models_{\mathrm{M}} [\![\varphi]\!]^{\xi}_{\mathrm{M}}$, as required.
($\Leftarrow$): Let $\theta := [\![\eta]\!]^{\xi}_{\mathrm{M}}$ and assume that $G, \theta \models_{\mathrm{M}} [\![\varphi]\!]^{\xi}_{\mathrm{M}}$. By definition, there are $e_1, \ldots, e_n$ and $v_1, \ldots, v_m$ such that $G, \theta' \models_{\mathrm{M}} struct_{\xi'(x)}(T) \wedge [\![\varphi']\!]^{\xi'}_{\mathrm{M}}$, where $\theta' = \theta[x_1 \mapsto e_1, \ldots, x_n \mapsto$

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

$e_n, y_1 \mapsto v_1, \ldots, y_m \mapsto v_m]$. Let H be the subgraph of G such that $V_H = \{v_1, \ldots, v_m\}$ and $E_H = \{e_1, \ldots, e_n\}$, and let $h : H \rightarrowtail G$ be the embedding of H in G. Because $G, \theta' \models_{\mathrm{M}} \mathit{struct}_{\xi'(x)}(T)$, H and h are well defined, and in particular, H is isomorphic to T. Let $t : T \to H$ be the isomorphism between T and H, and let $h' = t\,;h$. By construction, $\theta' = [\![\eta[x \mapsto h']]\!]_{\mathrm{M}}^{\xi'}$. It follows from the induction hypothesis that $G, \eta[x \mapsto h'] \models \varphi'$, and thus by definition that $G, \eta \models \varphi$. □

7. Logic and recognizability

In this section, we prove a generalization of Courcelle's result which says that every language definable in MSOGL is recognizable (Courcelle 1990, 1997): We show that every language of objects of a hereditary pushout category definable by our logic is recognizable in the sense described in Bruggink and König (2008), by giving an encoding of a formula into an automaton functor.

In order to achieve this result, we need two *finiteness requirements* that ensure finiteness of the sets of states: We first require that the objects in our category are finite in the sense that they contain only finitely many subobjects. Second, we require that the category enjoys the property that each composable pair of arrows has finitely many pushout complements (up to isomorphism). The category of finite graphs and many other categories satisfy both conditions.

7.1. *Objects with valuations*

In order to be able to use structural induction on the logical formula, we need objects which keep track of the free variables. To this end, we introduce a category of objects with valuations.

We fix a hereditary pushout category $\mathbf{C}$ with a strict initial object 0[‡].

Definition 7.1 (Object with valuation). Let τ be a variable typing. An *object with valuation* (short: *V-object*) of type τ is a triple $D = \langle \mathrm{B}^D, \eta^D, \sigma^D \rangle$, where

— B^D (the *base object*) is a $\mathbf{C}$-object;
— η^D maps each variable x for which $\tau(x)$ is defined to a mono $\eta^D(x) : V_x^D \rightarrowtail \mathrm{B}^D$ (the *valuation morphism*); and
— σ^D maps each variable x for which $\tau(x)$ is defined and $\tau(x) \neq \Omega$ to an arrow $\sigma^D(x) : V_x^D \to \tau(x)$ (the *typing morphism*).

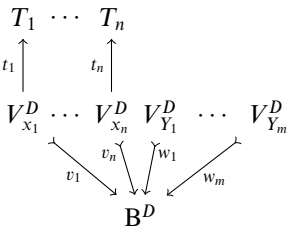
We say that D is *well typed* whenever all arrows $\sigma^D(x)$ are identities.

We will sometimes call the subobjects $\eta^D(x)$ of B^D *vertical interfaces* in order to distinguish them from the left and right objects of a cospan (the *horizontal interfaces*).

[‡] We require an initial object since we want to convert an arbitrary object A into the corresponding cospan $0 \to A \leftarrow 0$. Then the automaton functor recognizes such cospans by 'starting' and 'ending' with the initial object.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Let τ be defined only on the variables $x_1, \ldots, x_n$ and $Y_1, \ldots, Y_m$, with $\tau(x_i) = T_i$ and $\tau(Y_j) = \Omega$ (for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$). Then a V-object of type τ is a diagram as shown on the right (where $t_i = \sigma^D(x_i)$, $v_i = \eta^D(x_i)$ and $w_j = \eta^D(Y_j)$). Note that if D is well-typed (all typing morphisms are identities), η^D corresponds exactly to a valuation (see Section 5). For the translation, we need to allow typing morphisms which are not identities, however, because the part of an object decomposition we are currently looking at might not contain the subobject we are looking for in its entirety.



We introduce the following operators for diagram extension and restriction. Let $D = \langle \mathrm{B}^D, \eta^D, \sigma^D \rangle$ be a V-object of type τ. By $drop_x(D)$, we denote the V-object of type $\tau|_{Var \setminus \{x\}}$ from which the morphisms and objects associated to x have been dropped. Furthermore, let $v : V \rightarrowtail \mathrm{B}^D$ and $t : V \rightarrowtail T$ be given. Then, $add^v_y(D) = \langle \mathrm{B}^D, \eta^D[y \mapsto v], \sigma^D \rangle$ denotes the V-object of type $\tau[y \mapsto \Omega]$, which is obtained by adding the arrow v (indexed by y) to D. Similarly, $add^{v,t}_y(D) = \langle \mathrm{B}^D, \eta^D[y \mapsto v], \sigma^D[y \mapsto t] \rangle$ denotes the V-object of type $\tau[y \mapsto T]$ which is obtained by adding the arrows v and t (indexed by y).

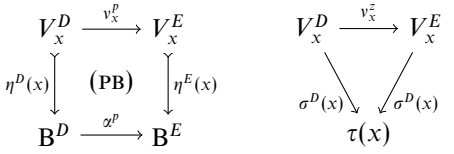
We consider the following category of V-objects.

Definition 7.2 (Category of objects with valuations). Let τ be a variable typing. We define the category $\mathbf{V}^{\mathbf{C}}_{\tau}$ (or simply $\mathbf{V}_{\tau}$) as follows.

The objects of $\mathbf{V}^{\mathbf{C}}_{\tau}$ are the objects with valuation of type τ (see Definition 7.1). An arrow $p : D \to E$ of $\mathbf{V}^{\mathbf{C}}_{\tau}$ consists of $\mathbf{C}$-arrows

$$\alpha^p : \mathrm{B}^D \to \mathrm{B}^E \quad \text{and} \quad v^p_x : \eta^D(x) \to \eta^E(x)$$

(for each $x \in Var$ such that $\tau(x)$ is defined) such that the square below is a pullback and the triangle commutes (whenever $\tau(x) \neq \Omega$):



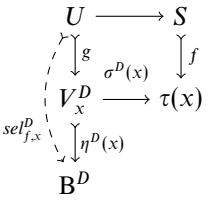
The two diagrams above specify the following: the valuation morphisms of D can be obtained from the valuation morphisms of E by taking a pullback (if α^p is mono, this is simply some form of restriction). Furthermore, the typing morphisms must be consistent.

We will consider (semi-abstract) cospans over $\mathbf{V}^{\mathbf{C}}_{\tau}$, i.e., we will work with the category $Cospan(\mathbf{V}^{\mathbf{C}}_{\tau})$. Note that due to property (iii) on page 1294 pushouts in $\mathbf{V}^{\mathbf{C}}_{\tau}$ – and hence cospan composition in $Cospan(\mathbf{V}^{\mathbf{C}}_{\tau})$ – can be computed componentwise by taking pushouts of all the component morphisms and obtaining the morphisms of the resulting diagram as mediating morphisms. In the initial object 0, all objects (apart from the $\tau(x)$) are the (strict) initial object of $\mathbf{C}$. We extend the operation $drop_x$ to $\mathbf{V}^{\mathbf{C}}_{\tau}$-cospans in the straightforward way.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

We will use a so-called *selection morphism* that identifies a subobject of the base object, depending on a variable x and (in the first-order case) on a given morphism f.

Given a $\mathbf{V}_\tau$-object $D = \langle \mathrm{B}^D, \eta^D, \sigma^D \rangle$ and a variable x such that $\tau(x)$ is defined, we define the selection morphism $sel^D_x := \eta^D(x)$ whenever $\tau(x) = \Omega$.
In the case that $\tau(x) \neq \Omega$, we parametrize by a morphism $f : S \to \tau(x)$) as follows. We take the pullback of $\sigma^D(x)$ and f, and obtain an object U and arrows $U \to S$ and $g : U \to V^D_x$ (see the diagram on the right). Now we take $sel^D_{f,x} := g\,;v^D_x$.



We extend the definition of selection morphism to expressions of subobject logic in the obvious way, i.e., $sel^D_{f \fatsemi x} := sel^D_{f,x}$.

Lemma 7.3. Let $p : D \to E$ be a $\mathbf{V}_\tau$-arrow, and e an expression whose variable occurs in the domain of τ. Furthermore, let U^D, U^E be the $\mathbf{C}$-objects such that $sel^D_e : U^D \rightarrowtail \mathrm{B}^D$ and $sel^E_e : U^E \rightarrowtail \mathrm{B}^E$. Then there exists an arrow $\mu : U^D \to U^E$ such that

$$\begin{array}{ccc} U^D & \xrightarrow{\mu} & U^E \\ {\scriptstyle sel^D_e}\downarrow & & \downarrow{\scriptstyle sel^E_e} \\ \mathrm{B}^D & \xrightarrow{\alpha^p} & \mathrm{B}^E \end{array}$$

is a pullback.

Proof. If $e = X$, where $\tau(X) = \Omega$, the result follows by definition. Let $e = f \fatsemi x$, where $\tau(x) \neq \Omega$. We construct the following commuting diagram:

$$\begin{array}{ccccc} U^D & \xrightarrow{\mu} & U^E & \longrightarrow & S \\ {\scriptstyle g_1}\downarrow & & {\scriptstyle g_2}\downarrow & & \downarrow{\scriptstyle f} \\ V^D_x & \longrightarrow & V^E_x & \underset{t^E_x}{\rightrightarrows} & \tau(x) \\ {\scriptstyle v^D_x}\downarrow & & {\scriptstyle v^E_x}\downarrow & & \\ \mathrm{B}^D & \underset{\alpha^p}{\longrightarrow} & \mathrm{B}^E & & \end{array}$$

where U^D is constructed by taking a pullback and the bottom-most and right-most squares are pullbacks by definition. Now, the desired result follows simply by the pullback theorem (recall, that $sel^D_{x,f} = g^D\,;v^D_x$ and $sel^D_{x,f} = g^E\,;v^E_x$). □

Lemma 7.4. Let $p : D \to E$ be a $\mathbf{V}_\tau$-arrow, and e_1, e_2 expressions in which only variables in the domain of τ occur. Then it holds that

$$sel^E_{e_1} \sqsubseteq sel^E_{e_2} \Longrightarrow sel^D_{e_1} \sqsubseteq sel^D_{e_2}.$$

Proof. The proof is straightforward from the definitions. □

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

7.2. *From logic to automaton functors*

We will now present an inductive encoding that takes a formula $\varphi \in \mathit{Form}(\tau)$ and translates it into an automaton functor $\mathcal{A}^{\varphi}_{\tau}$ (or simply $\mathcal{A}$) for the category $\mathit{Cospan}(\mathbf{V}^{\mathbf{C}}_{\tau})$. The definition is divided into four subcases: *atomic formulas*, *boolean operations*, *first-order quantification* and *second-order quantification*.

Atomic formulas: Suppose $\varphi \equiv e_1 \sqsubseteq e_2$. Since the variables contained in e_1, e_2 are free and hence the corresponding vertical interfaces are present, it is enough to just check inclusion of those interfaces. This information is recorded in just one state $\star$ and we produce the empty relation whenever inclusion fails.

Formally, we define the automaton functor $\mathcal{A}^{\varphi}_{\tau}$ as follows. To each object D, $\mathcal{A}^{\varphi}_{\tau}$ assigns the one-element set $\{\star\}$ if $\mathit{sel}^{D}_{e_1} \sqsubseteq \mathit{sel}^{D}_{e_2}$ (where $\star$ is the initial as well as the final state), and the empty set otherwise.

For a cospan $c : D \dashv\!\rightarrow E \leftarrow\!\vdash F$, we define $\mathcal{A}^{\varphi}_{\tau}(c)$ to be the identity relation on $\{\star\}$ whenever $\mathit{sel}^{E}_{e_1} \sqsubseteq \mathit{sel}^{E}_{e_2}$ (and hence also $\mathit{sel}^{D}_{e_1} \sqsubseteq \mathit{sel}^{D}_{e_2}$, $\mathit{sel}^{F}_{e_1} \sqsubseteq \mathit{sel}^{F}_{e_2}$ due to Lemma 7.4) and the empty relation otherwise.

Boolean operations: Suppose $\varphi = \varphi_1 \wedge \varphi_2$ or $\varphi = \neg\varphi'$. In this case, compute the automaton functors for the subformulas and apply the standard techniques for intersection or complement to the automaton functors (due to the results of Section 3.2 these operations can be performed on all automaton functors, independent of the category). In the latter case (negation/complement), it is necessary to first construct the equivalent deterministic automaton functor.

Second-order quantification: Since the second-order quantification case is much simpler than the first-order case, we present it first. Suppose $\varphi \equiv (\exists X : \Omega)\, \varphi'$. Let $\mathcal{A}' := \mathcal{A}^{\varphi'}_{\tau'}$, where $\tau' = \tau[X \mapsto \Omega]$, be the automaton functor constructed for φ'.

The domain of the automaton functor for the subformula φ' is a category of V-objects with an additional vertical interface. The automaton functor for φ works in the following way: non-deterministically, a satisfying assignment for X is chosen and then it behaves like the automaton functor for φ'.

The automaton functor $\mathcal{A}$ for φ is formally defined as follows:

$$\mathcal{A}(D) = \bigcup_{v \in \mathit{Sub}(\mathrm{B}^D)} \mathcal{A}'(\mathit{add}^{v}_{X}(D)) \times \{v\},$$

where $v \in \mathit{Sub}(\mathrm{B}^D)$ means that v ranges over representatives of isomorphism classes of monos into B^D. To a cospan $c : D \dashv\!\rightarrow E \leftarrow\!\vdash F$, we assign the following relation $\mathcal{A}(c) \subseteq \mathcal{A}(D) \times \mathcal{A}(F)$: $\langle q, v^D \rangle$ is in relation with $\langle q', v^F \rangle$ whenever there exists a cospan

$$c' : \mathit{add}^{v^D}_{X}(D) \to E' \leftarrow \mathit{add}^{v^F}_{X}(F)$$

of type τ' such that $\mathit{drop}_X(c') = c$ and $\langle q, q' \rangle \in \mathcal{A}'(c')$.

Finally, a state $\langle q, v \rangle$ is initial (final) if and only if q is initial (final).

First-order quantification: Suppose $\varphi \equiv (\exists x : T)\, \varphi'$. Let $\mathcal{A}' := \mathcal{A}^{\varphi'}_{\tau'}$, where $\tau' = \tau[x \mapsto T]$, be the automaton functor constructed for φ'.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

As in the case of second-order quantification, we non-deterministically choose a satisfying assignment for x. This time, however, we have to track how much of the sought after subobject (T) has already been recognized (see explanation below).

We define the new automaton functor $\mathcal{A}^{\varphi}_{\tau}$ as follows:

$$\mathcal{A}^{\varphi}_{\tau}(D) = \bigcup_{\substack{(v : V \to \mathrm{B}^D) \in Sub(\mathrm{B}^D),\\ (t : V \to T)}} \mathcal{A}'(add_x^{v,t}(D)) \times \{\langle v, t_1, t_2\rangle \mid t = t_1 ; t_2, \text{POC exists}\},$$

where, by 'POC exists,' we mean that there exists arrows s_1, s_2 such that $t_1 ; t_2 = s_1 ; s_2$ and the four arrows form a pushout. In the cases of v and the decomposition $t_1 ; t_2$, we actually take representatives of the respective isomorphism classes.

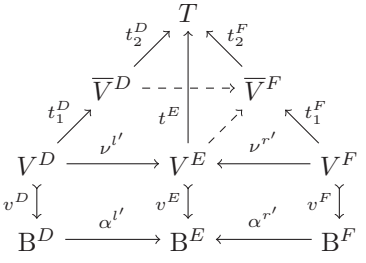
To a cospan $c : D -l\to E \leftarrow r- F$, we assign the following relation $\mathcal{A}^{\varphi}_{\tau}(c) \subseteq \mathcal{A}^{\varphi}_{\tau}(D) \times \mathcal{A}^{\varphi}_{\tau}(F)$: $\langle q, v^D, t_1^D, t_2^D\rangle$ is in relation with $\langle q', v^F, t_1^F, t_2^F\rangle$ whenever c can be extended to a cospan

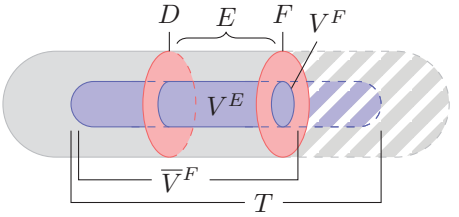
$$c' : add_x^{v^D,(t_1^D ; t_2^D)}(D) \xrightarrow{l'} add_x^{v^E,t^E}(E) \xleftarrow{r'} add_x^{v^F,(t_1^F ; t_2^F)}(F),$$

satisfying the following conditions:

— $\langle q, q'\rangle \in \mathcal{A}'(c')$.

— We have the commuting diagram below (on the left), where $\overline{V}^F$ is the pushout object of t_1^D and $v^{l'}$.





A state $\langle q, v, t_1, t_2\rangle$ is initial if and only if q is initial and t_1 is an identity. It is final if and only if q is final and the arrow t_2 is an identity.

The intuition behind the diagram above is the following: We are tracking an object of type T, while we are reading the 'complete' object step by step. The vertical interface V^F (with mono v^F) denotes the part of the outer interface that corresponds to this object. On the other hand, $\overline{V}^F$ (with mapping t_2^F into the type T) tells which part of the object we have already seen. Finally, t_1^F relates the part which is currently in the (outer) interface to the part we have already seen (see image above, on the right). Similar explanations can be given for v^D, t_1^D and t_2^D. Finally, $\overline{V}^F$ is obtained from $\overline{V}^D$ by gluing those (new) parts of the tracked object that have been seen in the current cospan, namely V^E. (Additional intuition is provided by the example in Section 7.3).

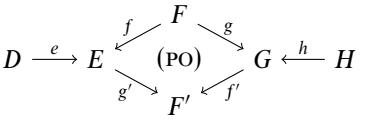
Proposition 7.5. $\mathcal{A}^{\varphi}_{\tau}$, constructed for a formula φ and a typing τ as described above, is a functor, that is, it preserves identities and composition.

Proof. It is fairly straightforward to show that identities are preserved by the functor. Hence, we will only prove that composition is preserved by induction on the structure of

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

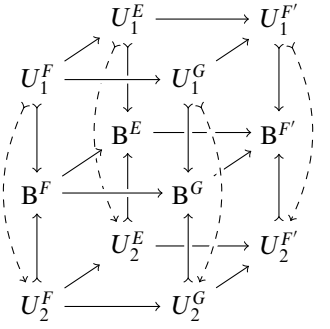
φ. The case of first-order quantification is similar, but more complex, than the case of second-order quantification. Hence, we will only prove the former and omit the latter.

We have the following composition of $\mathbf{V}_\tau$-cospans c_1 and c_2, where the middle square is a pushout:



We need to show that $\mathcal{A}(c_1 ; c_2) = \mathcal{A}(c_1) ; \mathcal{A}(c_2)$.

— *Base case.* Suppose $\varphi \equiv e_1 \sqsubseteq e_2$. We prove the stronger property that, given the composition diagram above, it holds that $\mathcal{C}(f) ; \mathcal{C}(g)^{-1} = \mathcal{C}(g')^{-1} ; \mathcal{C}(f')$. The relevant part of the diagram looks as follows:



where the straight vertical arrows are the respective selection morphisms of the expressions. There are following four cases to consider:

- If $sel^E_{e_1} \not\sqsubseteq sel^E_{e_2}$ and $sel^G_{e_1} \not\sqsubseteq sel^G_{e_2}$, then we know by Lemma 7.4 that $sel^{F'}_{e_1} \not\sqsubseteq sel^{F'}_{e_2}$. Both if $sel^F_{e_1} \sqsubseteq sel^F_{e_2}$ and $sel^F_{e_1} \not\sqsubseteq sel^F_{e_2}$, it holds that $\mathcal{C}(f) = \mathcal{C}(\mathrm{g}) = \mathcal{C}(f') = \mathcal{C}(g') = \emptyset$, and the property holds trivially.
- Suppose that $sel^E_{e_1} \sqsubseteq sel^E_{e_2}$ but $sel^G_{e_1} \not\sqsubseteq sel^G_{e_2}$. By the first assumption and Lemma 7.4, we know that $sel^F_{e_1} \sqsubseteq sel^F_{e_2}$. By the second assumption and the lemma, we know that $sel^{F'}_{e_1} \not\sqsubseteq sel^{F'}_{e_2}$. That means that $\mathcal{C}(f) = \mathrm{id}$ and $\mathcal{C}(\mathrm{g}) = \mathcal{C}(f') = \mathcal{C}(g') = \emptyset$. Therefore, $\mathcal{C}(f) ; \mathcal{C}(g)^{-1} = \mathcal{C}(g')^{-1} ; \mathcal{C}(f')$, as required.
- The case that $sel^E_{e_1} \not\sqsubseteq sel^E_{e_2}$ and $sel^G_{e_1} \sqsubseteq sel^G_{e_2}$ is handled symmetrically.
- Finally, suppose that $sel^E_{e_1} \sqsubseteq sel^E_{e_2}$ and $sel^G_{e_1} \sqsubseteq sel^G_{e_2}$. By Lemma 7.4, it must hold that $sel^F_{e_1} \sqsubseteq sel^F_{e_2}$. Since the top horizontal face is a pushout by construction and the cube property, there is a mediating morphism from $V^{F'}_Y$ to $U^{F'}_1$, and thus $sel^{F'}_{e_1} \sqsubseteq sel^{F'}_{e_2}$. This means that $\mathcal{C}(f) = \mathcal{C}(\mathrm{g}) = \mathcal{C}(f') = \mathcal{C}(g') = \mathrm{id}$, and therefore the desired property holds also in this case.

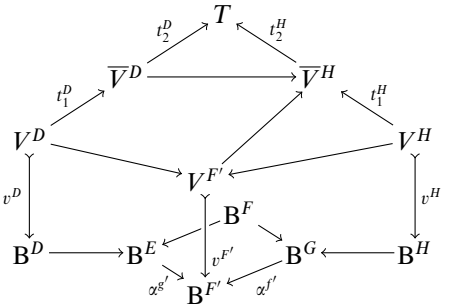
— Suppose $\varphi = (\exists x : T)\, \varphi'$. Let $\mathcal{A}$ and $\mathcal{A}'$ be the automaton functors constructed for φ and φ', respectively. By the induction hypothesis, $\mathcal{A}'$ preserves composition. We need to prove that $\mathcal{A}$, constructed for φ, also preserves composition.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

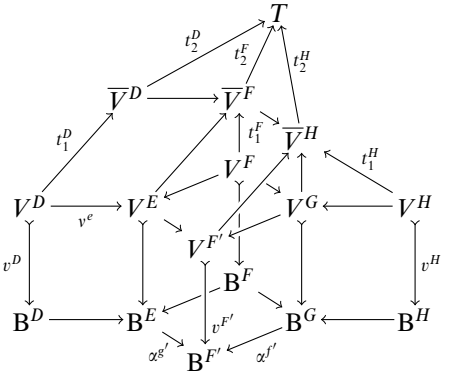
($\Rightarrow$): Assume $\langle\langle q^D, v^D, t_1^D, t_2^D\rangle, \langle q^H, v^H, t_1^H, t_2^H\rangle\rangle \in \mathcal{A}(c)$. By definition, there is a cospan

$$c' : add_x^{v^D,t^D}(D) \xrightarrow{l'} add_x^{v^{F'},t^{F'}}(F') \xleftarrow{r'} add_x^{v^H,t^H}(H),$$

where $t^D = t_1^D ; t_2^D$, $t^H = t_1^H ; t_2^H$, $\langle q^D, q^H\rangle \in \mathcal{A}(c')$ and the following diagram can be constructed:



We construct V^E, V^F and V^G by taking pullbacks, and take a pushout of the arrows t_1^D and v^e (see diagram), obtaining an object $\overline{V}^F$, which lies 'on' the arrow $\overline{V}^D \to \overline{V}^H$ by the pushout theorem.



Note that $V^D, \overline{V}^D, V^E, \overline{V}^F$ is a pushout by construction, and $V^F, \overline{V}^F, V^G, \overline{V}^H$ is a pushout because it is the composition of the pushout squares V^F, V^E, V^G, $V^{F'}$ and V^E, $\overline{V}^F$, $V^{F'}$, $\overline{V}^H$.

We proceed analogously as the previous case: We have obtained a diagram depicting the composition of two cospans c_1 and c_2, such that $c = c_1 ; c_2$, that fulfill the conditions of the definition. Now, we can use the induction hypothesis to get a state q^F such that $\langle\langle q^D, v^D, t_1^D, t_2^D\rangle, \langle q^F, v^F, t_1^F, t_2^F\rangle\rangle \in \mathcal{A}(c_1)$ and $\langle\langle q^F, v^F, t_1^F, t_2^F\rangle, \langle q^H, v^H, t_1^H, t_2^H\rangle\rangle \in \mathcal{A}(c_2)$, as required.

($\Leftarrow$): Suppose that

$$\langle\langle q^D, v^D, t_1^D, t_2^D\rangle, \langle q^F, v^F, t_1^F, t_2^F\rangle\rangle \in \mathcal{A}(c_1) \quad \text{and}$$
$$\langle\langle q^F, v^F, t_1^F, t_2^F\rangle, \langle q^H, v^H, t_1^H, t_2^H\rangle\rangle \in \mathcal{A}(c_2)$$

for some state $\langle q^F, v^F, t_1^F, t_2^F\rangle$. By definition, $\langle q^D, q^F\rangle \in \mathcal{A}(c_1')$ and $\langle q^F, q^H\rangle \in \mathcal{A}(c_2')$, and therefore, by the induction hypothesis, $\langle q^D, q^H\rangle \in \mathcal{A}(c')$. The situation we have is

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

depicted in the second figure of the left-to-right direction. We simply obtain that the square $V^D, V^{F'}, \overline{V}^H, \overline{V}^D$ is a pushout by applying the pushout theorem twice.

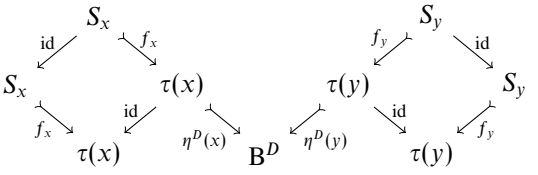
— Suppose $\varphi = (\exists X : \Omega)\, \varphi'$.
The second-order case is fairly similar to the first-order case and simpler in structure. Hence, we omit it. □

The next theorem shows that a well-typed V-object satisfies a logical formula if and only if it is accepted by the corresponding automaton functor.

Theorem 7.6. Let φ be a formula, let τ be a typing and let $D = \langle \mathrm{B}^D, \eta^D, \sigma^D \rangle$ be a well-typed V-object of type τ. Then, $\mathrm{B}^D, \eta^D \models \varphi$ if and only if D is contained in the $(0,0)$-language of $\mathcal{A}^\varphi_\tau$, that is, if there exists an initial state i_0 and a final state f_0 with $\langle i_0, f_0 \rangle \in \mathcal{A}^\varphi_\tau(c)$, where c is the unique cospan of the form $0 \to D \leftarrow 0$.

Proof. The desired result is shown by induction on the structure of φ. Note that, by the assumption that D is well-typed, σ^D maps only to identities.

— Let $\varphi \equiv e_1 \sqsubseteq e_2$. As an example of this case, we treat $\varphi \equiv f_x \fatsemi x \sqsubseteq f_y \fatsemi y$. All other such formulas can be treated similarly.
It holds that $D, \eta^D(x) \models f_x \fatsemi x \sqsubseteq f_y \fatsemi y$ if and only if $f_x ; \eta^D(x) \sqsubseteq f_y ; \eta^D(y)$. We construct the following diagram, and observe that it consists of two pullbacks by construction:



Hence, the above inclusion holds if and only if $\mathcal{A}^\varphi_\tau(c)$ relates $\star$ and $\star$, where c is the cospan $0 \to D \leftarrow 0$. And this means exactly that D is contained in the $(0,0)$-language of $\mathcal{A}^\varphi_\tau$ (since $\star$ is the initial as well as the final state).

— In the case of boolean operators, we employ the induction hypothesis and rely on the corresponding constructions for automaton functors.

— Let $\varphi = (\exists X : \Omega)\, \varphi'$. Let $\mathcal{A}$ be the automaton functor constructed for φ and $\mathcal{A}'$ the one for φ'.
Now assume that $\mathrm{B}^D, \eta^D \models (\exists X : \Omega)\, \varphi'$, which is equivalent to the fact that

$$\mathrm{B}^D, \eta^D[x \mapsto v] \models \varphi' \text{ for some } v : V \to \mathrm{B}^D.$$

Since $add^v_X(D) = \langle \mathrm{B}^D, \eta^D[x \mapsto v], \sigma^D \rangle$, it follows from the induction hypothesis that this holds if and only if $add^v_X(D)$ is contained in the $(0,0)$-language of $\mathcal{A}'$, that it, $\mathcal{A}'(c')$ relates an initial state q and a final state q', where c' is the cospan $0 \to add^v_X(D) \leftarrow 0$. Let c be the cospan $0 \to D \leftarrow 0$. The fact that c can be extended to c' such that q and q' are related by $\mathcal{A}'(c')$ is (by definition) equivalent to the fact that $\langle q, v_0 \rangle$ and $\langle q', v_0 \rangle$ are related by $\mathcal{A}(c)$, where c is the cospan $0 \to D \leftarrow 0$ and $v_0 : 0 \to 0$. Furthermore, $\langle q, v_0 \rangle$ must be initial and $\langle q', v_0 \rangle$ final. This means that D is contained in the $(0,0)$-language of $\mathcal{A}(c)$.

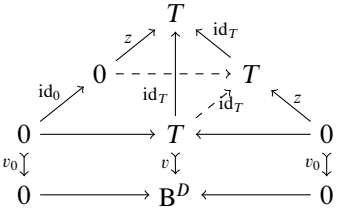
https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

— Let $\varphi = (\exists x : T)\, \varphi'$. Let $\mathcal{A}$ be the automaton functor constructed for φ and $\mathcal{A}'$ the one for φ'.

Now assume that $\mathrm{B}^D, \eta^D \models (\exists x : T)\, \varphi'$, which is equivalent to the fact that

$$\mathrm{B}^D, \eta^D[x \mapsto v] \models \varphi' \text{ for some } v : T \to \mathrm{B}^D.$$

Since $add_x^{v,\mathrm{id}_T}(D) = \langle \mathrm{B}^D, \eta^D[x \mapsto v], \sigma^D[x \mapsto \mathrm{id}_T] \rangle$, the induction hypothesis implies that this holds if and only if $add_x^{v,\mathrm{id}_T}(D)$ is contained in the $(0,0)$-language of $\mathcal{A}'$, that is, $\mathcal{A}'(c')$ relates an initial state q and a final state q', where c' is the cospan $0 \to add_x^{v,\mathrm{id}_T}(D) \leftarrow 0$. Hence, we have the following situation where T is the pushout object of $0 \to 0$ and $0 \to V$, and furthermore, $z : 0 \to T$ and $v_0 : 0 \to 0$:



The two bottom squares in the above diagram are pullbacks because of the v_0-arrows. This means that the initial state $\langle q, v_0, \mathrm{id}_0, z \rangle$ and the final state $\langle q', v_0, z, \mathrm{id}_T \rangle$ are related by $\mathcal{A}(c)$ where c is the cospan $0 \to D \leftarrow 0$. This in turn means that D is contained in the $(0,0)$-language of $\mathcal{A}(c)$. □

Finiteness. We will briefly discuss the restrictions on the finiteness of the state sets. For this, we first require that the objects in our category are finite in the sense that they contain only finitely many subobjects. In the case of graphs that would be the finite graphs and in the case of sets the finite sets.

In addition, we need another requirement: In the encoding of first-order quantification, we split a given arrow t into $t = t_1; t_2$ such that all pushout complements for this split exist. In order to guarantee finiteness, it is hence also necessary that a given arrow t only admits finitely many such splits up to isomorphism.

It is currently not clear to us how subobject finiteness is related to the condition of having finitely many pushout complement splits. Clearly, the latter property implies the former, but the other direction is unclear. In any case, the category **Graph** satisfies both requirements.

Summarizing our results, we obtain the following corollary.

Corollary 7.7. For a hereditary pushout category **C** that satisfies the finiteness properties stated above, every language of objects that is definable in the logic on subobjects is recognizable.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

7.3. *Detailed example*

As an example, we consider (in the category of cospans of unlabelled directed graphs) the translation into an automaton functor of the formula

$$\neg RC(X) \equiv (\exists y : E)\left(src \fatsemi y \sqsubseteq X \wedge \neg(tgt \fatsemi y \sqsubseteq X)\right),$$

which expresses that the subgraph X is not closed under reachability.

The automaton functor for the atomic formula $\varphi_2 := tgt \fatsemi y \sqsubseteq X$ maps each graph, in which the target of edge y does not lie in X, to the empty set, and all other graphs to the state set $\{\star\}$. It maps a cospan to $\mathrm{id}_{\{\star\}}$ if the above also holds for the middle object of the cospan. The automaton functor of the other atomic formula, $\varphi_1 := src \fatsemi y \sqsubseteq X$, is built analogously.

To calculate the negation of φ_2, we must first make its automaton functor deterministic by means of the powerset construction (see Section 3.2). Graphs are now mapped to either the state set $\{\varnothing, \{\star\}\}$ (where $\{\star\}$ is initial as well as final) or the state set $\{\varnothing\}$ (no initial or final states), and cospans are mapped accordingly. After the negation, final and non-final states are swapped.

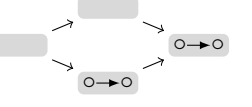
To construct the automaton functor for $\varphi_1 \wedge \neg\varphi_2$, we use a Cartesian product construction. In the rest of the example, we will restrict our attention to an object D' with $\mathcal{A}^{\varphi_1}(D') = \{\star\}$ and $\mathcal{A}^{\neg\varphi_2}(D') = \{\varnothing, \{\star\}\}$. For this object D', we have that

$$\mathcal{A}^{\varphi_1 \wedge \neg\varphi_2}(D') = \{\langle\{\star\}, \varnothing\rangle, \langle\{\star\}, \{\star\}\rangle\},$$

where $\langle\{\star\}, \{\star\}\rangle$ is the initial and $\langle\{\star\}, \varnothing\rangle$ the final state. By reaching the final state, we record that both the source node of the edge assigned to y is contained in X and the target node is not contained in X.

Finally, we build an automaton functor $\mathcal{A}$ for $\varphi \equiv (\exists y : E)(\varphi_1 \wedge \neg\varphi_2)$. Recall that $E = \circ\!\!\rightarrow\!\!\circ$. We obtain a V-object $D = drop_y(D')$. Assume it has base object $\mathrm{B}^D = D_2 = \circ \quad \circ$, a two-node discrete graph. States of the new automaton functor consist of arbitrary monos $v : V \rightarrowtail \mathrm{B}^D$, allowed decompositions t_1, t_2 of arbitrary typing morphisms $t : V \to E$, and the states of D extended with v, t_1, t_2. Let us list the possible decompositions of the typing morphism for all $2^2 = 4$ possible v into B^D:

— For $v : \varnothing \rightarrowtail \mathrm{B}^D$, there is a single typing morphism, and two possible decompositions, given by the two legs of the pushout diagram on the right. The top leg expresses the situation where no part of the sought after subobject has been encountered yet, whereas the bottom leg expresses the situation that the entire sought after subobject was already encountered.



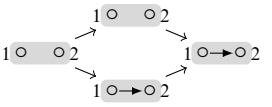
— For $v = 1 \circ \rightarrowtail 1 \circ \quad \circ$, there are two possible typing morphisms (mapping the node to the source and the target of the edge, respectively), with two decompositions each:

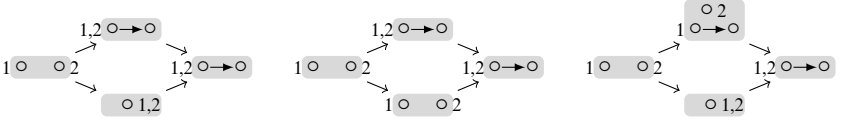


https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

— For $v = \circ 2 \rightarrowtail \circ \quad \circ 2$, four decompositions are obtained analogously[§].

— Let $v = \mathrm{id}_{B^D}$. There are four possible typing morphisms. For $t = 1\circ \quad \circ 2 \rightarrow 1\circ\!\!\rightarrow\!\!\circ 2$, there are two allowed decompositions, which can be read from the diagram on the right. For $t = 1\circ \quad \circ 2 \rightarrow 1{,}2\circ\!\!\rightarrow\!\!\circ$, the allowed decompositions can be read from the pushout diagrams below. Since two decompositions occur twice, there are four decompositions in total.





For $t = 1\circ \quad \circ 2 \rightarrow 2\circ\!\!\rightarrow\!\!\circ 1$ and $t = 1\circ \quad \circ 2 \rightarrow \circ\!\!\rightarrow\!\!\circ 1{,}2$, we symmetrically obtain five decompositions.

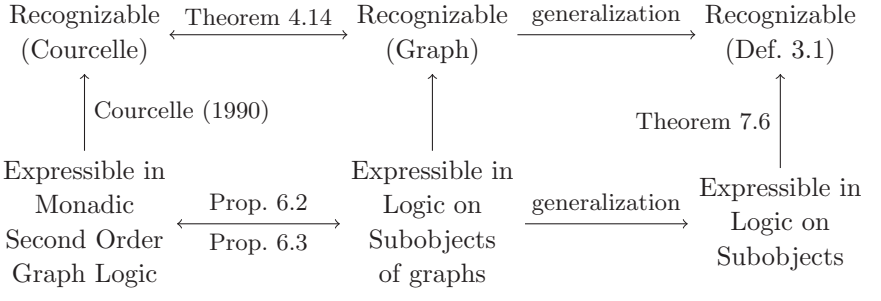
As an example, consider the decomposition $1\circ \quad \circ 2 \rightarrow 1\circ\!\!\rightarrow\!\!\circ \; \circ 2 \rightarrow 1{,}2\circ\!\!\rightarrow\!\!\circ$ (the top right in the three diagrams above). This decomposition expresses that both nodes of B^D are part of the sought after graph (as source node), but still need to be merged.

Suppose that for all possible v, t, the object D extended with v, t has the state set above (two states). The new automaton functor then has $2 \times 20 = 40$ states for the given base object. Accepting states are the ones which contain an accepting state of the automaton functor for the subterm, and in which the second part of the decomposition (t_2) is an isomorphism (that is, a subgraph isomorphic to E was found).

The automaton functor works as follows: Given a cospan $c : D \xrightarrow{c_L} E \xleftarrow{c_R} F$, it non-deterministically chooses what parts of the new information (the parts of E not in the range of c_L) belong to the sought-after type E, and then works the same as the automaton functor constructed for the subformula.

8. Conclusion

We have shown that a very general categorical notion of recognizability via automaton functors (which is related to a notion suggested in Griffing (2003)) is equivalent to a notion of recognizability for graph languages by Courcelle whenever we consider the category of cospans of graphs. The proof of this equivalence is non-trivial. Furthermore, we have introduced a logic on subobjects, shown how it is related to MSOGL and generalized the result of Courcelle saying that every MSOGL-expressible graph language is recognizable. The situation is depicted in the following diagram:



[§] The numbers beside the nodes indicate which nodes are mapped to which.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

The decomposition of graphs into cospans we have studied here is very natural in graph rewriting and it corresponds to path decompositions of graphs (for a formal elaboration of this claim, see Blume et al. (2013)). An alternative would be to use tree decompositions (also studied in the paper above). Although this should be feasible in principle in the setting of this paper, we did not choose to follow this path here since it would have added additional complexity to the encoding into automaton functors. Furthermore, in our tool RAVEN (mentioned below), we also chose to work with finite automata instead of tree automata in order to avoid the costly join operation. Note however that despite our use of path decompositions, we obtain the same notion of recognizability as Courcelle, which is based on tree decompositions.

Related work. Naturally, as stated several times in this paper, this work took great inspiration by the results on recognizability introduced by Courcelle (1990). Similar ideas are exhibited in Bozapalidis and Kalampakas (2006, 2008). The categorical generalization is similar in spirit to concepts introduced in Griffing (2003).

Cospans of graphs were also investigated, in the context of gluing graph structures, by Rosebrugh et al. (2005, 2008), but in Rosebrugh et al. (2008), their graph structures represent automata which recognize word languages rather than graph languages.

Our logic on subobjects has some reminiscences to categorical logic (Jacobs 1999; Pitts 2001). Although we are working in a categorical setting, our choice was to focus on a classical logic, quantifying over sets of subobjects, and not a categorical logic where the universe is replaced by an object. Currently, we do not see how to obtain a similar correspondence of MSOGL with a categorical logic. For instance, our predicates on subobjects can not directly be interpreted as subobjects of a product object, as would be customary for subobject logic. Although our logic falls out of scope of known categorical logics, we still believe that its intimate connection to MSOGL and recognizability makes it an interesting logic to study.

The generation of automata and/or algorithms from MSOGL-formulas has been studied in several papers (Courcelle 1990; Flum et al. 2002), but not in such an abstract setting as ours. Despite the generality of the construction, we hope that its inductive nature will be helpful for our implementation, which – similar to the contributions in (Courcelle and Durand 2012; Gottlob et al. 2010; Soguet 2008) – tries to make the appealing complexity-theoretic results (linear time for graphs of bounded path or tree width) usable in practice. In order to achieve this, we are working on a tool RAVEN (Blume et al. 2012) that implements the huge automaton functors symbolically via binary decision diagrams (Andersen 1998). The transitions of the automaton are not labelled with arbitrary, but with so-called atomic cospans. We already achieved some promising runtime results, but the implementation of the construction of automata from formulas still has to be implemented. Despite all attempts at efficiency, we believe that in practice this will work only for the nesting of very few quantifiers, which is mainly due to the state explosion caused by determinization.

We plan to use our tool for several applications. One such application provides us with a method to show that a recognizable language is invariant under DPO rewriting rules:

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Let $p: L \leftarrow I \rightarrow R$ be a DPO rule and let $\equiv_R$ be a congruence[¶] characterizing a language $\mathcal{L}$ of objects. We observe that whenever an object A is rewritten to B via p, $A \in \mathcal{L}$ and $(0 \rightarrow L \leftarrow I) \equiv_R (0 \rightarrow R \leftarrow I)$ (for an initial object 0), then we can conclude that $B \in \mathcal{L}$. This idea has been worked out in Blume et al. (2010).

For future applications, we are especially interested in termination analysis (Geser et al. 2004) and regular model checking (Bouajjani et al. 2000), which are both based on regular languages.

We would like to thank the reviewers of this paper for their valuable input. We are also grateful to Tobias Heindel for his suggestions and his help with the categorical foundations. Furthermore, we would like to thank Paolo Baldan, Andrea Corradini and Fabio Gadducci for stimulating discussions on this research topic. A special thanks goes to Christoph Blume who is working with us on related topics and is implementing the tool RAVEN. This work was supported by the DFG-project GaReV.

References

Andersen, H.R. (1998). An introduction to binary decision diagrams. Version of October 1997 with minor revisions April 1998. Lecture notes, Technical University of Denmark, unpublished manuscript.

Blume, C., Sander Bruggink, H.J. and König, B. (2010). Recognizable graph languages for checking invariants. In: *Proceedings of the of Workshop on Graph Transformation and Visual Modeling Techniques*, Electronic Communications of the EASST, vol. 29. Available at `http://journal.ub.tu-berlin.de/index.php/eceasst/article/view/408`.

Blume, C., Sander Bruggink, H.J., Engelke, D. and König, B. (2012). Efficient symbolic implementation of graph automata with applications to invariant checking. In: *Proceedings of the of International Conference on Graph Transformation*, LNCS, vol. 7562, Springer, 264–278.

Blume, C., Sander Bruggink, H.J., Friedrich, M. and König, B. (2013). Treewidth, pathwidth and cospan decompositions with applications to graph-accepting tree automata. *Journal of Visual Languages & Computing* **24** (3) 192–206. Available at `http://dx.doi.org/10.1016/j.jvlc.2012.10.002`.

Bouajjani, A., Jonsson, B., Nilsson, M. and Touili, T. (2000). Regular model checking. In: *Proceedings of Computer Aided Verification*, LNCS, vol. 1855, Springer, 403–418.

Bouderon, M. and Courcelle, B. (1987). Graph expressions and graph rewritings. *Mathematical Systems Theory* **20** (1) 81–127.

Bozapalidis, S. and Kalampakas, A. (2006). Recognizability of graph and pattern languages. *Acta Informatica* **42** (8/9) 553–581.

Bozapalidis, S. and Kalampakas, A. (2008). Graph automata. *Theoretical Computer Science* **393** (1–3) 147–165.

Bruggink, H.J.S. (2007). Towards a systematic method for proving termination of graph transformation systems (work-in-progress paper). In: *Proceedings of Graph Transformation for Concurrency and Verification.*

Bruggink, H.J.S. and König, B. (2008). On the recognizability of arrow and graph languages. In: *Proceedings of the International Conference on Graph Transformation*, LNCS, vol. 5214, Springer, 336–350.

[¶] One is here actually interested in the weaker notion of a Myhill well-quasi order instead of a congruence, but we omit the details.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press

Bruggink, H.J.S. and König, B. (2010). A logic on subobjects and recognizability. In: *Proceedings of the International Conference on FIP Theoretical Computer Science*, vol. 323, Springer, 197–212.

Courcelle, B. (1990). The monadic second-order logic of graphs I. Recognizable sets of finite graphs. *Information and Computation* **85** (1) 12–75.

Courcelle, B. (1997). The expression of graph properties and graph transformations in monadic second-order logic. In: Rozenberg, G. (ed.) *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 5. World Scientific.

Courcelle, B. and Durand, I. (2012). Automata for the verification of monadic second-order graph properties. *Journal of Applied Logic* **10** (4) 368–409.

Courcelle, B. and Engelfriet, J. (2012). *Graph Structure and Monadic Second-Order Logic, A Language-Theoretic Approach*, Cambridge University Press.

Courcelle, B. and Weil, P. (2005). The recognizability of sets of graphs is a robust property. *Theoretical Computer Science* **342** (2–3) 173–228.

Ehrig, H. and König, B. (2006). Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. *Mathematical Structures in Computer Science* **16** (6) 1133–1163.

Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. (2006). *Fundamentals of Algebraic Graph Transformation*, Springer.

Flum, J., Frick, M. and Grohe, M. (2002). Query evaluation via tree-decompositions. *Journal of the ACM* **49** (6) 716–752.

Gadducci, F. and Heckel, R. (1997). An inductive view of graph transformation. In: *12th International Workshop on Recent Trends in Algebraic Development Techniques*, LNCS, vol. 1376, Springer, 223–237.

Geser, A., Hofbauer, D. and Waldmann, J. (2004). Match-bounded string rewriting systems. *Applicable Algebra in Engineering, Communication and Computing* **15** (3–4) 149–171.

Gottlob, G., Pichler, R. and Wei, F. (2010). Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Journal of Artificial Intelligence* **174** (1) 105–132.

Griffing, G. (2003). Composition-representive subsets. *Theory and Applications of Categories* **11** (19) 420–437.

Habel, A., Kreowski, H.-J. and Lautemann, C. (1993). A comparison of compatible, finite and inductive graph properties. *Theoretical Computer Science* **110** (1) 145–168.

Heindel, T. (2009). *A category theoretical approach to the concurrent semantics of rewriting*. PhD thesis, University of Duisburg–Essen.

Jacobs, B. (1999). *Categorical Logic and Type Theory*, Studies in Logic and the Foundation of Mathematics, vol. 141, Elsevier.

Kuncak, V. and Rinard, M. (2003). Existential heap abstraction entailment is undecidable. In: *Proceedings of the of SAS*, LNCS, vol. 2694, Springer.

Lack, S. and Sobociński, P. (2005). Adhesive and quasiadhesive categories. *RAIRO – Theoretical Informatics and Applications* **39** (3) 511–545.

Mac Lane, S. (1971). *Categories for the Working Mathematician*, Springer.

Pitts, A. M. (2001). Categorical logic. In: Abramsky, S., Gabbay, D.M. and Maibaum, T.S.E. (eds.) *Handbook of Logic in Computer Science V*, Oxford University Press.

Rosebrugh, R., Sabadini, N. and Walters, R.F.C. (2005). Generic commutative separable algebras and cospans of graphs. *Theory and Applications of Categories* **15** (6) 164–177.

Rosebrugh, R., Sabadini, N. and Walters, R.F.C. (2008). Calculating colimits compositionally. In: Degano, P., De Nicola, R. and Meseguer, J. (eds.) *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, LNCS, vol. 5065, Springer, 581–592.

Sassone, V. and Sobociński, P. (2005). Reactive systems over cospans. In: *Proceedings of Logic in Computer Science*, IEEE, 311–320.

Soguet, D. (2008). *Génération automatique d'algorithmes linéaires*. PhD thesis, Université Paris-Sud.

https://doi.org/10.1017/S096012951800018X Published online by Cambridge University Press