Regular expressions for tree-width 2 graphs

Anonymous
Anonymous

Abstract

We propose a syntax of regular expressions, which describes languages of tree-width 2 graphs. We show that these languages correspond exactly to those languages of tree-width 2 graphs, definable in the counting monadic second-order logic (CMSO).

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Tree width, MSO, Regular expressions

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

1 Introduction

Regular word languages form a robust class of languages. One of the witnesses for this robustness is the variety of equivalent formalisms defining them. They can be described by finite automata, by monadic second-order (MSO) formulas, by regular expressions or by finite monoids [4, 7, 11]. Each of these formalisms has some advantages, depending on the context where it is used. For example, MSO is close to natural language, regular expressions define regular languages via their closure properties, automata have good algorithmic properties and can be used as actual algorithms to decide membership in a language, etc. Similarly, regular tree languages have equivalent formalisms, for various kinds of trees [12, 14, 10].

We will here further generalize the structures considered, by moving to graphs of bounded tree-width. Intuitively, they can be thought of as "graphs which resemble trees". In this framework, we already know that counting MSO (CMSO), an extension of MSO with counting predicates, and recognizability by algebra are equivalent [2, 3], yielding a notion that could be called "regular languages of graphs of tree-width k". Engelfriet [8] also proposes a regular expressions formalism matching this class, but by his own admission, these expressions closely mimic the behavior of CMSO. The main feature missing in Engelfriet's regular expressions is a mechanism for iteration, which is the central operator for word regular expressions: the Kleene star.

In this paper, we propose a syntax of regular expressions for languages of tree-width 2 graphs, that follow more closely the spirit of regular expressions on words, using Kleene-like iterations. This constitutes a first step towards the long term objective of obtaining such expressions for languages of graphs of tree-width k. We believe the case of tree-width 2 is already a significant step in itself. Graphs of tree-width 2 form a robust class of graphs with several interesting characterizations. One of them is the characterization via the forbidden minor K_4, the complete graph with four vertices. By the Robertson-Seymour theorem [13], it is known that for every $k \in \mathbb{N}$, the class of tree-width k graphs is characterized by a finite set of excluded minors. However, this result is not constructive, and only the forbidden minors for $k \leq 3$ are known.

Let us now give more intuition about our expressions for graphs of tree-width 2. Our Kleene-like iteration is defined in terms of least fixed points $\mu x.\ e$. However without restriction, such an operator is too powerful and takes us outside of the CMSO-definable graphs. This phenomenon actually already happens on words: with an arbitrary fixed point, we can write $\mu x.\ (axb)$, defining the non-regular language $\{a^n x b^n \mid n \in \mathbb{N}\}$. The Kleene star on words can be seen as a restriction of the least fixed point operator: only fixed points of the form $\mu x.\ ex$

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:33
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

are allowed, where x does not appear in e. Here the idea is the same, but our restriction will be more involved, and will require a fine understanding of the structure of tree-width 2 graphs.

This work was inspired by the work of Gazdag and Németh [9] on regular expressions for bisemigroups and binoids. One of the main difference with our work is that their operators are only associative, while the operations generating our graphs satisfy more properties.

The paper is structured as follows. Sec. 2 is a preliminary section where we introduce graphs of tree-width 2, the logic CMSO and recognizability by algebra, which are known to be equivalent. In Sec 3, we introduce regular expressions, explain the condition that the iteration should satisfy, and give some examples to illustrate it. At the end of this section, we state our main result, which says that this formalism is equivalent to the two introduced in the preliminary section. We introduce in Sec. 4 the logic CMSO^r, an extension of CMSO with a very restricted form of quantification over relations, and show that it is equivalent to CMSO. Based on this, we show in section 5 that regularity implies CMSO-definability. Finally, we show in section 6 that recognizability implies regularity, proving our main result.

2 Preliminaries

2.1 Tree-width 2 graphs

▶ **Definition 1** (Ranked sets). *A* ranked set *is a set where every element has an associated* arity *in* $\mathbb{N}$. *Elements of arity* 0 *are called* nullary*, of arity* 1 *are called* unary*, of arity* 2 *are called* binary *etc. An* arity-preserving function *is a function between two ranked sets which does not change arities.*

We fix in the rest of the paper an alphabet Σ, which is a ranked set whose *letters* are either unary or binary. We denote their sets Σ_1 and Σ_2 respectively.

▶ **Definition 2** (Graphs). *A graph consists of:*

- *a (not ranked) set* V *of* vertices*,*
- *a ranked set* E *of* edges *of arity* 1 *or* 2*,*
- *a function* $E \mapsto V^*$ *which maps each* n*-ary edge to its* interface*, which has length* n*,*
- *an arity-preserving* labeling $E \mapsto \Sigma$*,*
- define subgraph *a list of size* $n \in \{1,2\}$ *called its* interface*. The number* n *is called the arity of the* graph*.*

The first element of the interface of an edge is called its source*, and the last element is called its* target*. The first element of the interface of a graph is called its* input*, and the last element is called its* output[1]*. All the vertices of a graph which are not in the interface are called* inner vertices*. An* a-edge *is an edge labeled by the letter* a*.*

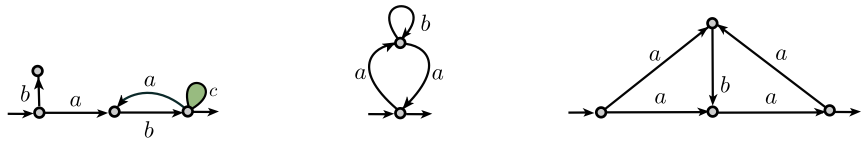
A graph is empty *if it has no edges, and if all its vertices are interface vertices.*

If v, w *are two vertices of a graph* G*, we denote by* (v, G, w) *the graph obtained from* G *by declaring* v *as its input and* w *as its output.*

▶ Remark 3. What we call here a graph is what is usually called a hypergraph (because of the unary edges) with interface.

▶ **Example 4.** We depict graphs with unlabeled ingoing and outgoing arrows to denote the input and the output, respectively. The c-edge in the leftmost graph below is a unary edge. The graph in the middle is a unary graph.

[1] For unary graphs and edges, the input equals the output.

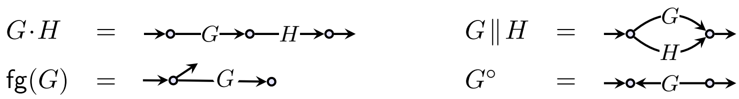


▶ **Definition 5** (tw$_2$ graphs). *Consider the* signature σ *which is a ranked set containing the binary operations* $\cdot$ *and* $\|$*, the unary operations* $^\circ$ *and* fg*, and the nullary operations* 1 *and* $\top$*. We define* tw$_2$ terms *as the terms generated by the signature* σ *and the alphabet* Σ*:*

$$t, u := \ a \ \mid \ t \cdot u \ \mid \ (t \parallel u) \ \mid \ t^\circ \ \mid \ \mathsf{fg}(t) \ \mid \ 1 \ \mid \ \top \qquad\qquad a \in \Sigma$$

We define by induction the graph $\mathcal{G}(t)$ *of a term* t*, by letting:*

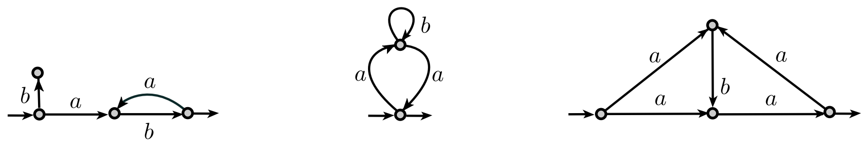
for every $a \in \Sigma_1$ *and* $b \in \Sigma_2$ *and interpreting the operations of the syntax as follows:*



In the picture above, we represent the graph G *by an arrow from its input to its output, which might be equal. For example, the graph* $\mathsf{fg}(G)$ *is obtained from* G *by relocating the output to the input. We usually write* tu *instead of* $t \cdot u$ *and give priorities to the symbols of* σ *so that* $ab \| c$ *parses to* $(a \cdot b) \| c$*. We define the set of* tw$_2$ graphs *as the graphs of the terms above. The graphs of* a *and* $a \parallel 1$*, where* $a \in \Sigma$*, are called* atomic. describe more precisely

We will sometimes identify terms with the graphs they generate. For example we may say that $(a \parallel b)$ is binary or connected to say that its graph is so.

▶ **Example 6.** Below, from left to right, two tw$_2$ graphs and a graph which is not tw$_2$.



$\mathsf{fg}(b)a(a^\circ \parallel b)$ $\qquad$ $a(b \parallel 1)a$ $\qquad$ No corresponding term

▶ Remark 7. The tw$_2$ graphs are exactly those graphs whose skeleton[2] has tree-width 2 [5].

▶ **Definition 8** (Graph languages). *Sets of graphs are called* graph languages*. A graph language is* unary *or* binary *if all its graphs have this arity.*

2.2 Counting monadic second-order logic

We introduce CMSO, the *counting monadic second-order logic*, which is used to describe graph languages.

[2] The skeleton of a graph is the graph obtained by forgetting the labels and the orientation of the edges, and by adding an edge between the input and the output.

▶ **Definition 9** (The logic CMSO). *Graphs vocabulary $\mathcal{V}$ is the ranked set which contains two binary symbols* source *and* target*, a unary symbol a for each (unary or binary) letter $a \in \Sigma$ and two unary symbols* input *and* output*.*

Let $\mathbb{X}_1$ be a countable set of first-order variables *and $\mathbb{X}_2$ a countable set of* set variables *The formulas of* CMSO *are defined as follows:*

$$\varphi, \psi := \ r(x_1, \ldots, x_n) \ \mid \ x \in X \ \mid \ x = y \ \mid \ \exists x.\varphi \ \mid \ \exists X.\varphi \ \mid \ \varphi \vee \psi \ \mid \ \neg\varphi \ \mid \ (|X| \equiv k) \ [m]$$

where r is an n-ary symbol of $\mathcal{V}$, $x_1, \ldots, x_n, x \in \mathbb{X}_1$, $X \in \mathbb{X}_2$ and $k, m \in \mathbb{N}$. Free and bound variables *are defined as usual. A* sentence *is a formula without free variables. We use the usual syntactic sugar, for example $\varphi \Rightarrow \psi$ as a shortcut for $\neg\varphi \vee \psi$.*

We define the semantics of CMSO formulas. To handle free variables, CMSO formulas are interpreted over *pointed graphs*.

▶ **Definition 10** (Semantics of CMSO). *Let G be a graph and Γ be a set of variables. An* interpretation *of Γ in G is a function mapping each first-order variable of Γ to an edge or vertex of G, and each set variable to a a set of edges and vertices of G. A* pointed graph *is a pair $\langle G, I\rangle$ where G is a graph and I is an interpretation of a set of variables Γ in G. If Γ is empty, we denote it simply as G. Let φ be a* CMSO *formula whose free variables are Γ and let $\langle G, I\rangle$ be a pointed graph such that I is an interpretation of Γ. We define the* satisfiability relation *$\langle G, I\rangle \models \varphi$ as usual, by induction on the formula φ. Here is an example of the semantics of some* CMSO *formulas:*

source(v, e) :	*the source of the edge e is the vertex v.*	input(v) :	*v is the input of G.*
target(v, e) :	*the target of the edge e is the vertex v.*	output(v) :	*v is the output of G.*
$(\lvert X\rvert = k)[m]$:	*the size of X is congruent to k modulo m.*	$a(e)$:	*e is an a-edge.*

If φ is a sentence, we define $\mathcal{L}(\varphi)$, the graph language of φ *as follows:*

$$\mathcal{L}(\varphi) = \{G \ \mid \ G \text{ is a graph and } G \models \varphi\}\,.$$

▶ **Definition 11** (CMSO definability). *A graph language is* CMSO definable *if it is the graph language of a* CMSO *sentence.*

▶ **Example 12.** The language of graphs having an a-edge from the input to the output is definable in CMSO, by the following formula for instance:

$$\varphi := \ \exists e.\ \exists i.\ \exists o.\ \mathsf{input}(i) \wedge \mathsf{output}(o) \wedge a(e) \wedge \mathsf{source}(i, e) \wedge \mathsf{target}(o, e)$$

Note that the graphs of this language may not be tw_2 graphs.

▶ **Example 13.** The set of tw_2 graphs is CMSO definable. Indeed, tw_2 graphs are those graphs which exclude K_4, the complete graph with four vertices, as minor. The set of graphs which exclude a fixed set of minors can be easily defined in CMSO [6].

The set of tw_2 graphs having an a-edge from the input to the output is definable in CMSO, by the conjunction of the formula φ of Ex. 12 and the formula defining tw_2 graphs.

We state below a *localization result*, which allows us to transform a CMSO sentence into another one which talks only about a part of the original graph.

▶ **Proposition 14.** *Let φ be a* CMSO *sentence, $x, y \in \mathbb{X}_1$ and $X \in \mathbb{X}_2$. There is a* CMSO *formula $\varphi_{|(x,X,y)}$ such that, for every graph G and interpretation $I : (x \mapsto v, X \mapsto H, y \mapsto w)$ of the variables $\{x, y, X\}$ in G, such that (v, H, w) is a subgraph of G, we have:*

$$\langle G, I\rangle \models \varphi_{|(x,X,y)} \qquad \Leftrightarrow \qquad (v, H, w) \models \varphi$$

Proof. We construct $\varphi|_{(x,X,y)}$ from φ as follows. First, we rename the variables of φ so that they become distinct from x, y and X. Then we replace every subformula $\exists z.\ \psi$ by $\exists z.\ (z \in X) \wedge \varphi$, every subformula $\exists Z.\ \psi$ by $\exists Z.\ (Z \subseteq X) \wedge \varphi$, every formula $\mathsf{input}(z)$ by $(z = x)$ and every formula $\mathsf{output}(z)$ by $(z = y)$. We show that $\varphi|_{(x,X,y)}$ has the intended semantics by induction on φ. ◀

▶ Remark 15. There is another presentation of the syntax of CMSO, where we remove first-order variables and the formulas including them, and add the following formulas:

$$X \subseteq Y \quad \text{and} \quad r(X_1 \ldots, X_n) \quad \text{where } r \text{ is an } n\text{-ary symbol of } \mathcal{V}.$$

The formula $X \subseteq Y$ is interpreted as "X is a subset of Y" and $r(X_1 \ldots, X_n)$ as "for each i, X_i is a singleton containing x_i and $r(x_1 \ldots, x_n)$". This presentation is more convenient in proofs by induction as there are less cases to analyze.

2.3 Recognizability

We can specify languages of graphs by means of σ-algebras, generalizing to graphs the notion of recognizability by monoids.

▶ **Definition 16** (σ-algebra). *A σ-algebra $\mathcal{A}$ is the collection of a set D called its* domain*, and for each n-ary operation o of σ, a function $o^{\mathcal{A}} : D^n \to D$. A homomorphism $h : \mathcal{A} \to \mathcal{B}$ between two σ-algebras $\mathcal{A}$ and $\mathcal{B}$ is a function from the domain of $\mathcal{A}$ to the domain of $\mathcal{B}$ which preserves the operations of σ.*

▶ **Definition 17** (σ-algebra of graphs). *The set of tw_2 graphs, where the operations of σ are interpreted as in Definition 5, forms a σ-algebra which we denote by $\mathcal{G}_{\mathsf{tw}_2}$.*

▶ **Definition 18** (Recognizability). *We say that a language L of tw_2 graphs is* recognizable *if there is a σ-algebra $\mathcal{A}$ with finite domain, a homomorphism $h : \mathcal{G}_{\mathsf{tw}_2} \to \mathcal{A}$ and a subset P of the domain of $\mathcal{A}$ such that $L = h^{-1}(P)$.*

▶ **Theorem 19.** *If a language of tw_2 graphs is CMSO definable, then it is recognizable.*

Proof. This result is proved in a more general framework in [1]. To adapt this to our setting, we just have to verify that our σ-algebras are also compatible with product and powerset operations. This is straightforward, as our algebras are very similar to those in [1]. ◀

2.4 Operations on graph languages

The operations of σ can be lifted from graphs to graph languages in the natural way. We say that an operation on graph languages is CMSO *compatible* if, whenever its arguments are CMSO definable, then so is its result.

▶ **Proposition 20.** *Union and the operations of σ are CMSO compatible.*

Proof. The language of $\varphi \vee \psi$ is the union of the languages of φ and ψ, for every CMSO sentences φ and ψ, this concludes the case of union.

For the operations of σ, we use the localization result. We treat the case of series composition, the other operations can be treated similarly. Let φ and ψ be two CMSO sentences. We construct the formula $\varphi \cdot \psi$ as follows. We guess two sets X and Y and an element m, which are intended to be the graph coming from φ, the graph coming from ψ and the middle vertex in between them, respectively. Then we say that X contains the input, Y

contains the output and that X and Y intersect exactly in m. Using the localization result, we say that the graph whose set of edges and vertices is X, whose input is the input of the original graph, and whose output is m satisfies φ. We say similarly that the graph whose set of edges and vertices is Y, whose input is m and whose output is the output of the original graph satisfies ψ.

$$\varphi \cdot \psi := \exists X,\ \exists Y,\ \exists m,\ \exists i,\ \exists o.\ \ \mathsf{input}(i) \wedge \mathsf{output}(o) \wedge (i \in X) \wedge (o \in Y) \wedge (X \cap Y = \{m\}) \\ \wedge\, \varphi|_{i,X,m} \wedge \psi|_{m,Y,o}$$

where $(X \cap Y = \{m\})$ is a CMSO formula saying that the intersection of X and Y is m. ◀

We define two additional operations: *substitution* and *iteration*.

▶ **Definition 21** (Substitution and iteration). *Let x be a letter, L and M be* tw_2 *graph languages and let be G a* tw_2 *graph. We define the set of graphs $G[L/x]$ by induction on G as follows:*

$$x[L/x] = L, \quad a[L/x] = a\ (a \neq x) \quad \textit{and} \quad o(G_1 \ldots, G_n)[L/x] = o(G_1[L/x], \ldots, G_n[L/x])$$

where o is an n-ary operation of σ. We define $M[L/x]$ as:

$$M[L/x] = \bigcup_{G \in M} G[L/x]$$

We define similarly the simultaneous substitution $M[\vec{L}/\vec{x}]$, where $\vec{L}$ and $\vec{x}$ are respectively a list of tw_2 *graph languages and a list of letters of the same length.*

For every $n \geq 1$, we define the language $L^{n,x}$ and the iteration $\mu x.L$ as follows:

$$L^{1,x} := L, \qquad L^{n+1,x} =: L[L^{n,x}/x] \cup L^{n,x}, \qquad \mu x.L := \bigcup_{n \geq 1} L^{n,x}.$$

▶ Remark 22. Substitution and iteration are not CMSO compatible in general. For instance, the iteration of the CMSO language $\{axb\}$, which is the set $\{a^n x b^n \mid n \in \mathbb{N}\}$, is not CMSO definable. However, under a *guard condition* that we introduce later, we recover CMSO compatibility.

We finally consider two restricted forms of iteration called *Kleene* and *parallel iteration*.

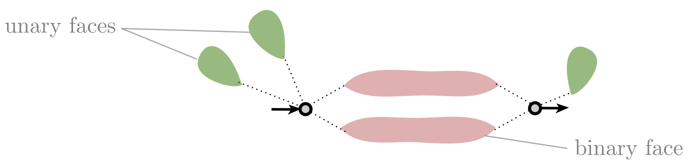
▶ **Definition 23** (Kleene and parallel iteration). *We define the* Kleene iteration *L^+ and the* parallel iteration *$L^{\|}$ of a language L as follows, where x is a letter not appearing in L:*

$$L^+ = (\mu x.\ L{\cdot}x)[L/x], \qquad\qquad L^{\|} = (\mu x.\ L \,\|\, x)[L/x].$$

2.5 Pure graphs and modules

▶ **Definition 24** (Pure graphs.). *Let G be a graph. If we remove the interface vertices of G we obtain one or several connected components which we call the* faces *of G. The* arity of a face *is the number of interface vertices of G it is incident to.*

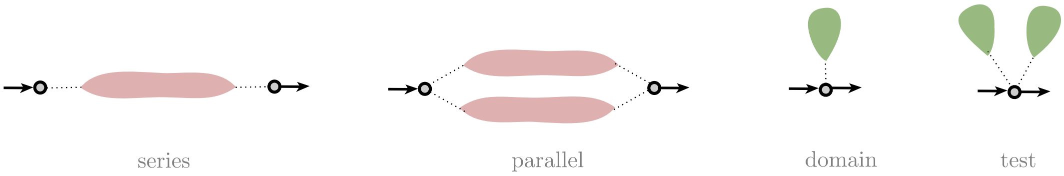
define precisely



We say that G is pure *if it has at least one face and all it faces have the same arity as itself. We say that G is* prime *if it has exactly one face, and* composite *if it has at least two faces.*

▶ Remark 25. Pure graphs are connected and non-empty. Not all graphs are pure.

▶ **Definition 26** (Type of a pure graph). *The* type *of a pure graph is a pair specifying its arity and whether it is prime or composite. We say that a graph is* series *if it is binary and prime,* parallel *if it is binary and composite,* domain *if it is unary and prime and* test *if it is unary and composite. We denote by* $\mathsf{s}, \mathsf{p}, \mathsf{d}$ *and* t *the type series, parallel, domain and test respectively. Series, parallel domain and test graphs look like this:*



A graph language is (of type) series, parallel, domain *or* test *if* ***all*** *its graphs have this type.*

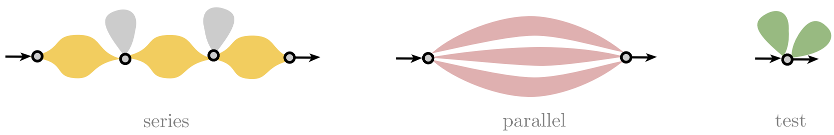
There is a canonical way to decompose pure graphs of type series, parallel and test.

▶ **Proposition 27** ([?]). *Let G be a pure graph. The graph G has the following shape:*

$$\begin{array}{rcll} G & := & P_0 \cdot U_1 \cdot P_1 \ldots U_n \cdot P_n & \text{if } G \text{ is series,} \\ G & := & S_0 \parallel \cdots \parallel S_n & \text{if } G \text{ is parallel,} \\ G & := & D_0 \parallel \cdots \parallel D_n & \text{if } G \text{ is test,} \end{array}$$

P_j being parallel or atomic, U_i unary, S_i series and D_i domain, for all $j \in [0, n], i \in [1, n]$.

Here is a picture illustrating this proposition:

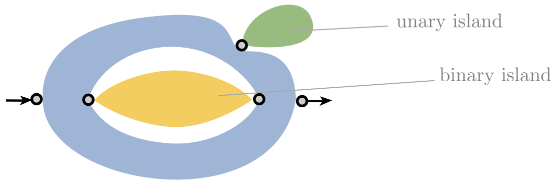


▶ **Definition 28** (Contexts). *A* context *is a graph with a unique edge, labeled by a special letter, called its* hole. *If C is a context whose hole is h and H a graph* ***with the same arity as*** *h, we define $C[H]$ as the graph obtained from the disjoint union of C and H, by removing the edge h, identifying the input of h with the input of H, the output of h with the output of H, and by letting the interface of $C[H]$ to be that of C. Let $\mathbb{S}$ be a set of* special (unary and binary) letters, *and let $n \geq 1$. An n-context is a graph such that n of its edges, called* holes, *are numbered from 1 to n, and labeled by n distinct special letters. We call 1-contexts simply* contexts.

Let C be an n-context whose holes are $h_1 \ldots, h_n$ and let $H_1, \ldots, H_n$ be graphs such that h_i and the H_i have the same arity, for all $i \in [1, n]$. We define $C[H_1, \ldots, H_n]$ as the graph obtained from the disjoint union of C and $H_1, \ldots, H_n$, by removing the holes of G, and for every $i \in [1, n]$ identifying the input of h_i with the input of H_i, the output of h_i with the output of H_i, and by letting its interface to be that of C. do I use general contexts?

▶ **Definition 29** (Islands and modules). *An* island *of a graph G is a graph H such that there is a context C satisfying $G = C[H]$. A* module *is a island which is pure. Two islands (or modules) of a graph are* parallel *if they have the same interface. Since modules are pure, we can speak of series, parallel, domain and test modules of a graph.*

The following picture illustrates a unary and binary island of a graph.



▶ Remark 30. Our notion of modules is different from the one usually used in graph theory, more precisely in the setting of *modular decompositions*.

▶ Remark 31. If I is an interface in a graph G, there is always an island of G whose interface is I, the empty graph for example. This is not the case for modules.

▶ Remark 32. The parallel composition of two islands of a graph G with the same interface is also an island of G with the same interface. Similarly, the parallel composition of two modules of a graph G with the same interface is also a module of G with the same interface. This justifies the following definition.

▶ **Definition 33** (Maximal islands and modules). *Let G be a graph and I an interface in G. The* maximal island at *I is the parallel composition of all the islands of G whose interface is I, we denote it by* $\mathsf{max\text{-}island}_G(I)$*. The* maximal module at *I is the parallel composition of all the modules of G whose interface is I, we denote it by* $\mathsf{max\text{-}module}_G(I)$*.*

▶ Remark 34. The maximal module at a given interface does not always exist.

▶ **Proposition 35.** *Being series, parallel, domain, test, an island, a module, a maximal island, a maximal module are* CMSO *definable properties.*

Proof. We propose an equivalent definition for islands which is more convenient to express in CMSO.

▶ **Lemma 36.** *Let G be a graph. A subgraph H of G is an island iff no interface vertex of G in an inner vertex of H and there is no edge from an inner vertex of H to a vertex of G outside of H.*

Proof. It is easy to see that if H is an island, then it satisfies these conditions. Suppose that H satisfies the conditions of the lemma. Let C be the graph obtained from G by removing all the edges and the inner vertices of H and by adding a edge labeled by a special letter, with the same interface as H. It is easy to see that $C[H]$ is G. ◀

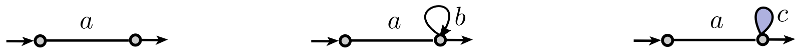
In CMSO, we can express that a graph is not a maximal module: there is a module with the same interface, which is strictly bigger. Hence we can express maximality.

Since connectivity is expressible in CMSO, we can express easily in CMSO that a subgraph is a face. We can also say if a a graph is series, by saying that it has a unique face, and this face is binary. We can proceed similarly to express that a graph is parallel, domain, test and pure. Finally, we express that a graph is a module by saying that it is an island which is pure. ◀

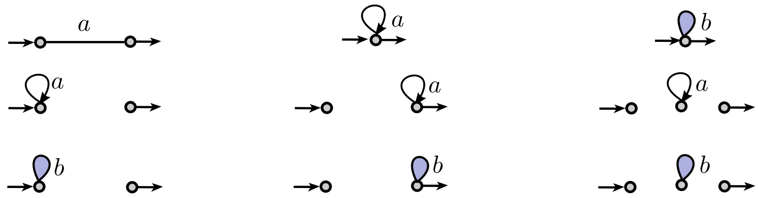
3 Regular expressions for tw_2 graphs

3.1 Regular expressions for word and multiset graphs

▶ **Definition 37** (Word and multiset alphabets). *Let Σ_{w} be the set of terms whose graphs have the following form, where $a, b \in \Sigma_2$ and $c \in \Sigma_1$:*

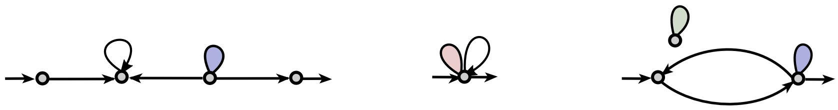


Let Σ_{m} be the set of terms whose graphs have the following form, where $a \in \Sigma_2$ and $b \in \Sigma_1$:



Word graphs *are the graphs generated from those of Σ_{w} by series composition, and* multiset graphs *are the graphs generated from those of Σ_{m} by parallel composition.*

▶ **Example 38.** Below, from left to right, a word graph and two multiset graphs.



▶ **Definition 39** (Word an multiset expressions)**.** Word expressions *are defined as follows:*

$$e, f := a \mid e \cdot f \mid e \cup f \mid e^{+} \qquad (a \in \Sigma_{\mathsf{w}})$$

Multiset pre-expressions *are defined as follows:*

$$e, f := a \mid (e \parallel f) \mid e \cup f \mid e^{\parallel} \qquad (a \in \Sigma_{\mathsf{m}})$$

Multiset expressions *are those pre-expressions, where each sub-term appearing under a parallel iteration, is built using a single element $a \in \Sigma_{\mathsf{m}}$ (all the other operations are allowed). The graph language of an expressions is defined as usual.*

▶ Remark 40. To see why the condition on multiset regular expressions is useful, consider the expression $e = (a \parallel b)$. The language of its parallel iteration is the set of multiset graphs which have the same number of a-edges and b-edges, and this is not a CMSO definable language.

3.2 Context-free expressions

▶ **Definition 41** (Context-free expressions)**.** *We define* context-free expressions *as the set of terms generated by the following syntax:*

$$\begin{aligned} e, f := \ & e_{\mathsf{w}} \mid e_{\mathsf{m}} \\ & \mid e \cdot f \mid (e \parallel f) \mid e^{\circ} \mid \mathsf{fg}(e) \mid 1 \mid \top \\ & \mid e \cup f \mid e[f/x] \mid \mu x.e \end{aligned}$$

where e_{w} and e_{m} are respectively word and multiset regular expressions. We define the language *of a context-free expression e, denoted $\mathcal{L}(e)$, by induction on e, interpreting the operations of the syntax as described in Sec. 2.4.*

Regular expressions for tw_2 *graphs* will be defined as a restriction of context-free expressions, where substitution and iteration are allowed only under a *guard condition* that we shall explain in the following.

3.3 The guard condition

▶ **Definition 42** (Guarded letters). *Let G be a graph and x a letter. We say that:*

- *x is* s-guarded in *G if x is binary and every x-labeled edge of G is parallel to a module.*
- *x is* p-guarded in *G if x is binary and no x-labeled edge of G is parallel to a module.*
- *x is* d*-guarded in G if x is unary.*
- *x is* t*-guarded in G if x is unary and no x-labeled edge of G is parallel to a module.*

Let $\tau \in \{\mathsf{s}, \mathsf{p}, \mathsf{d}, \mathsf{t}\}$ be a type and L a graph language. We say that x is τ-guarded *in L if it is τ-guarded in every graph of L.*

▶ **Definition 43** (Guard condition). *Let x be a letter, M a* tw_2 *graph language and L a pure language of type τ. The substitution $M[L/x]$ is guarded if x is τ-guarded in M. The iteration $\mu x.L$ is guarded if x is τ-guarded in L.*

We say that the iteration $\mu x.L$ is of type τ *if L is of type τ.*

▶ **Definition 44** (Regular expression). *A* regular expression *is a context-free expression where every substitution and iteration is guarded. A language of graphs is* regular *if it is the language of some regular expression.*

▶ Remark 45. When L is test and x is a unary letter, then $\mu x.L$ is always guarded.

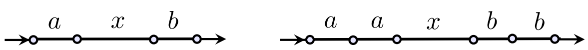
▶ **Proposition 46.** *We can decide if a context-free expression is regular.*

Proof sketch. We can annotate our regular expressions with extra information, remembering whether their language is pure, unary, binary, and in general what type of graphs it generates. This allows us to check the guard condition and propagate the annotations inductively, using the syntax tree of the expression. We start from the leaves of this tree (labeled by atomic formulas) and propagate the annotations upwards while verifying the guard condition when required, until we reach the root of the tree (labeled by the full expression). ◀

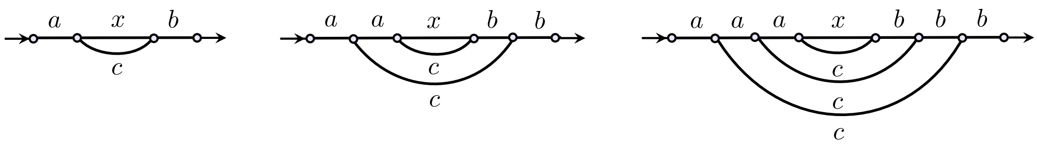
▶ Remark 47. Be aware that Prop. 46 is about deciding a syntactic property of e, namely that the iterations and substitutions are guarded. However, the problem of determining if a context-free expression *defines a* CMSO *language* is undecidable. This apparent contradiction comes from the fact that some context-free expressions, which are not guarded, define CMSO languages, as we shall see in the upcoming examples.

3.4 Examples

▶ **Example 48.** The iteration $\mu x.axb$ is not guarded. Indeed, the language of axb is series, as it contains a single series graph G. However, the letter x is not s-guarded in G, because it is not parallel to any module of G. The graph of this iteration look like this:

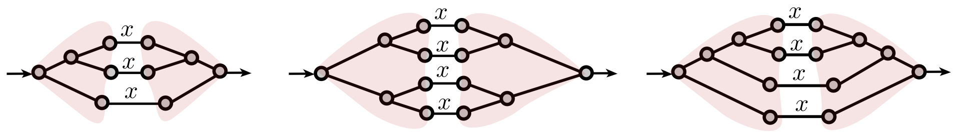


▶ **Example 49.** The iteration $\mu x.a(x \parallel c)b$ is guarded. Indeed the language of $a(x \parallel c)b$ is series, actually it contains a single graph G, depicted below left, which is series. The letter x is s-guarded in G, because it is parallel to a module, namely the c-edge. The graph of this iteration look like this:



Note the similarity between the graph language of $\mu x.axb$ and that of $\mu x.a(x \parallel c)b$: the former is obtained by forgetting the c-edges of the latter. Yet, the latter is CMSO definable, while the former is not. In the case of $\mu x.a(x \parallel c)b$, the c-edges will guide a CMSO formula to relate the a-edges and the b-edges of the same iteration depth. This is the main intuition behind the guard condition for series languages.

▶ **Example 50.** The iteration $\mu x.(axa \parallel axa)$ is guarded. Indeed, the language of $(axa \parallel axa)$ is parallel, as it contains a unique graph G (the left graph below) which is parallel. The letter x is p-guarded because all the occurrences of x are not parallel to any module of G. Note that the graphs of this expression have the following shape: they all start with a binary tree whose edges are labeled by a, end ends with the mirror image of this tree, while the corresponding leafs are connected by an x-edge. Those trees are colored in red below.



At first glance, this expression dos not seem to be CMSO definable, as it seems that we need to test whether a graph starts and ends with the same tree. We will see however that the language of this expression, as those of all regular expressions, is CMSO definable.

The guard condition is not "perfect", in the sense that some non-guarded context-free expressions might generate CMSO definable languages, as shown in the following example.

▶ **Example 51.** The context-free expression $(\mu x.axb)[1/a, 1/x, 1/b]$ is not regular because the iteration $\mu x.axb$ is not guarded. However its language, the graph of 1, is CMSO definable.

▶ Remark 52. Intuitively, the guard condition allows only those graphs where series and parallel operations alternate. This why we add the word and multiset expressions: to allow graphs where we can iterate only series or parallel operations respectively.

3.5 Main result

The main result of this paper is the following theorem:

▶ **Theorem 53.** *Let L be a language of* tw_2 *graphs. We have:*

$$L \text{ is recognizable} \quad \Leftrightarrow \quad L \text{ is CMSO definable} \quad \Leftrightarrow \quad L \text{ is regular}$$

Thanks to Thm. 19, CMSO definability implies recognizablity. We show that regularity implies CMSO definability in Sec. 5 and that recognizabilty implies regularity in Sec. 6.

4 Companion relations

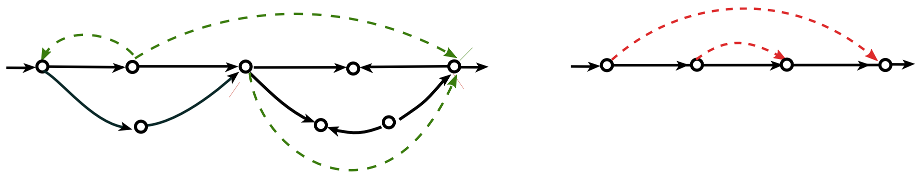
In CMSO, it is not possible to guess relations on the vertices of graphs in general. In this section, we present a very particular case of relations which can be guessed in CMSO, called *companion relations.*

▶ **Definition 54** (Paths). *A* path *p of G is a non-repeating list $(v_0, e_1, v_1, \ldots, e_n, v_n)$ where v_i is a vertex of G and e_i is an edge of G, such that the interface of e_i is either (v_{i-1}, v_i) or (v_i, v_{i-1}), for every $i \in [1, n]$. The path p is* directed *if the interface of e_i is (v_{i-1}, v_i) for every $i \in [1, n]$. The vertex v_0 is the* input *of p, v_n is its* output *and (v_0, v_n) its* interface.

▶ **Definition 55** (Companion relation). *Let G be a graph. Two paths of G are* orthogonal *if they do not share any edge, and whenever they share a vertex, it is necessarily an interface vertex of one of them.*

A relation R on the vertices of G is a companion relation *if there is a set of (pairwise) orthogonal paths P such that $(v, w) \in R$ iff (v, w) is the interface of a path $p \in P$. We say that p is a* witness *for (v, w), and that P is a* witness *for the relation R.*

▶ **Example 56.** The relation indicated by the green dotted arrows below is a companion relation. This is not the case for the one indicated by the red dotted arrows.



We introduce CMSO^r, an extension of CMSO where quantification over companion relations is possible.

▶ **Definition 57** (The logic CMSO^r). *Let $\mathbb{X}_r$ be a set of* relation variables*, whose elements are denoted $R, S, \ldots$. The formulas of CMSO^r are of the following form:*

$$\varphi := \mathsf{CMSO} \mid \exists R.\ \varphi \mid (x, y) \in R \qquad\qquad (R \in \mathbb{X}_r,\ x, y \in \mathbb{X}_1).$$

As for CMSO, we need to define the semantics of a formula over pointed graphs to handle free variables.

▶ **Definition 58** (Semantics of CMSO^r). *Let G be a graph and Γ be a set of variables. An interpretation of Γ is as usual, but here every relation variable is mapped to a binary relation on the vertices of G. We define the* satisfiability relation *$\langle G, I\rangle \models \varphi$ as usual, by induction on the formula φ. The only new cases are the quantification $\exists R$ which is interpreted as "there exists a companion relation R on the vertices of the graph", and the formulas $(x, y) \in R$ which are interpreted as "there is a pair of vertices (x, y) in R".*

4.1 The logic CMSO^r have the same expressive power as CMSO

To guess a companion relation in CMSO, we show how to encode a set of guarded paths by a collection of sets called a *footprint*.

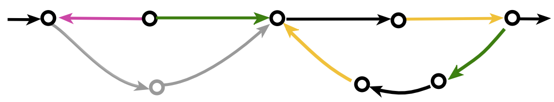
▶ **Definition 59** (Frontier edges of a path). *Let $p = (v_0, e_1, v_1, \ldots, e_n, v_n)$ be a path. If $n > 1$, we call e_1 the* opening edge *of p and e_n its* closing edge*. If $n = 0$, we call e_0 its* single edge*. Opening, closing and single edges are called the* frontier edges *of p, the other edges are called its* inner edges.

▶ **Definition 60** (Footprint). *A* footprint *in a graph G is the following collection of data: a partition of the vertices of G into* non-path *and* path *vertices, a partition of edges into* non-path *and* path *edges, a partition of path edges into* frontier *and* inner *edges, a partition of frontier edges into* opening, closing and single *edges and a partition of path edges into* direct *and* inverse *edges.*

The partition of path edges into direct and inverse ones provides them with a new orientation: they conserve their original orientation if they are direct, or get reversed (we swap the source and target) if they are inverse edges.

Let $\mathbb{F}$ be a footprint. A path p is encoded by $\mathbb{F}$ if its edges and vertices are path edges and path vertices of $\mathbb{F}$, if its inner, frontier, opening, closing and single edges are edges of the corresponding sets in $\mathbb{F}$. Moreover, p must form a directed path with the new orientation dictated by $\mathbb{F}$.

▶ **Example 61.** We represent below a footprint in the left graph of Ex. 56. Non-path edges and vertices are grey, path vertices are black, opening edges are green, closing edges are yellow, single edges are pink and all the other inner edges are black. For path edges, we display the new orientation induced by the footprint instead of the original one. The set of paths encoded by this footprint are a witness that the green relation of Ex. 56 is a companion relation.



▶ **Proposition 62.** *Let G be a graph and P a set of orthogonal paths of G. There is a footprint $\mathbb{F}$ such that P is the set of paths encoded by $\mathbb{F}$.*

Proof. We define $\mathbb{F}$ as the natural paths encoding associated to P: its path edges and path vertices are respectively the set of edges and vertices that appear in the paths of P, its frontier, inner, opening, closing and simple edges are the corresponding edges in the paths of P. Notice that it is possible reorient the edges of every path so that it becomes directed. Since the paths of P do not share any edge, we can reorient their edges consistently in order to make them directed. We define direct edges as the path edges whose orientation did not change after this operation, and inverse edges the remaining path edges.

It is clear that the paths of P are encoded by $\mathbb{F}$, let us show that they are the only ones. We start by stating the following claim which follows from the definitions.

▷ Claim 63. Let p be a path in P and v a vertex of p. If v is the source (w.r.t. the new orientation induced by $\mathbb{F}$) of an inner edge or the closing edge of p, then, by construction, v cannot be an interface vertex of p. If v is the target (w.r.t. the new orientation induced by $\mathbb{F}$) of an inner edge or the opening edge of p, then by construction, v cannot be an interface vertex of p.

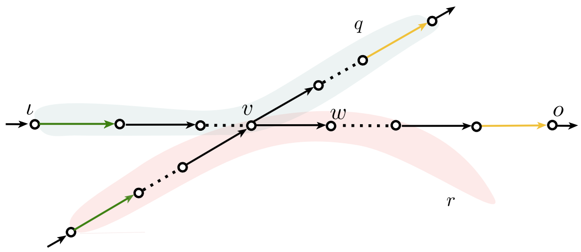
Let p be a path encoded by $\mathbb{F}$. If p has a unique edge, then it is a single edge, and by construction single edges come exclusively from paths of P with a unique edge. In this case, p is clearly a path in P.

Suppose now that p has at least two edges. The opening edge of p is, by construction, the opening edge of some path q of P. Let v be the vertex of p such that all the vertices between the input of p and v are also vertices of q and the successor of v in p, call it w, is not a vertex of q.

By Claim 63, and since v is the target of the opening edge or an inner edge of the path q, it is not an interface vertex of q.

Let e be the edge from v to w in the path p. The edge e is either an inner edge or a closing edge, and by construction, there is a path r of P containing this edge. By Claim 63, and since v is the source of an inner edge or the closing edge of r, we have that v is not an interface vertex of r.

Here is a picture illustrating this construction, where the path between ι and o is p.



We have found two paths of P, q and r, sharing a vertex v which is an interface vertex of none of them. This yields a contradiction. ◀

▶ **Theorem 64.** *If a language is* CMSO^r *definable then it is* CMSO *definable.*

Proof. Let φ be a CMSO^r formula. We transfor φ into a CMSO formula ψ as follows. We replace every quantification $\exists R.$ by a sequence of existential sets quantifications representing a footprint $\mathbb{F}$.

Saying that a subgraph (s, Z, t) is a directed path is expressible in CMSO, and checking that its different components (frontier vertices, opening and closing edges, etc) match the footprint is also easily expressible in CMSO. Hence we can define a CMSO formula encoded-path$_{\mathbb{F}}(s, Z, t)$ which says that (s, Z, t) is a path encoded by $\mathbb{F}$.

We replace every subformula of φ of the form "$(s, t) \in R$" by the following formula:

$$\exists Z.\ \text{encoded-path}_{\mathbb{F}}(s, Z, t)$$

◀

5 Regular implies CMSO definable

▶ **Theorem 65.** *If a language is regular, then it is* CMSO *definable.*

To prove Thm. 65, we proceed by induction on regular expressions. The cases of word and multiset regular expressions follow from the similar result for words and commutative words. The cases of union and the operations of the signature σ follow from Prop. 20. We are left with the cases of substitution and iteration; the rest of this section is dedicated to proving the following proposition.

▶ **Proposition 66.** *Let x be a letter and L and M be languages of* tw_2 *graphs. We have:*

$$M[L/x] \text{ is guarded and } L \text{ and } M \text{ are CMSO-definable} \quad \Rightarrow \quad M[L/x] \text{ is CMSO-definable.}$$
$$\mu x.L \text{ is guarded and } L \text{ is CMSO-definable} \quad \Rightarrow \quad \mu x.L \text{ is CMSO-definable.}$$

We handle the case of iteration, the case of substitution being similar. We show first that the iteration of a CMSO definable language, without any guard condition, is definable in an extension of CMSO where we are allowed to quantify existentially over sets of subgraphs of the input graph, which we call CMSO^d. This logic is obviously strictly more expressive then CMSO, because it amounts to quantify over sets of sets. Based on this, we show that the *guarded iteration* of a CMSO definable language is definable in CMSO^r, the extension of CMSO with companion relations defined in the previous section. This concludes the proof, the logic CMSO^r being equivalent to CMSO.

5.1 Iteration of CMSO formulas is CMSO^d definable

5.1.1 Decompositions

When a graph is in the iteration $\mu x.L$ of some language L, it is possible to structure it into a tree shaped decomposition, such that each part of this decomposition "comes from L". In the following, we define such decompositions.

▶ **Definition 67** (Independent graphs)**.** *Let G be a graph and H, K be subgraphs of G. We say that H and K are* independent *if they do not share any edge; and whenever they share a vertex, it is necessarily an interface vertex of both H and K.*

▶ Remark 68. Two independent subgraphs can share at most two vertices.

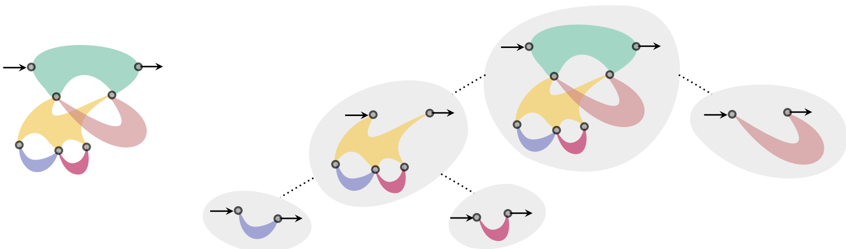
▶ **Definition 69** (Decompositions)**.** *A* decomposition *of G is a set $\mathcal{D}$ of modules of G such that $G \in \mathcal{D}$ and for every pair of graphs in $\mathcal{D}$, they are either independent, or strict module one of the other. We call the graphs of a decomposition its* components*. We call* the interfaces of $\mathcal{D}$ *the set of interfaces of its components.*

Let H and K be components of a decomposition $\mathcal{D}$. We say that H is a child *of K, if H is a module of K, and if there is no component C of $\mathcal{D}$, distinct from H and K, such that H is a module of C and C is a module of K.*

The graph G is called the head *of $\mathcal{D}$. A component of $\mathcal{D}$ is a* leave *if it does not contain another component of $\mathcal{D}$ as a module.*

▶ Remark 70. Note that the children of a component are pairwise independent.

▶ **Example 71.** Let G be the left graph below. The right picture is a decomposition of G, where the child relation is materialized by the dotted lines. The colors have no specific meaning here, but will be useful to illustrate the upcoming notion of the *components body.*



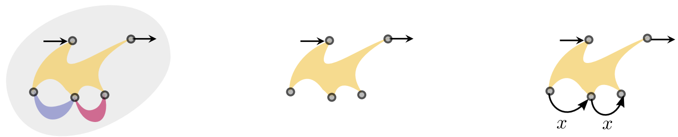
▶ **Definition 72** (Body of a component)**.** *Let G be a graph, $\mathcal{D}$ a decomposition of G and C a component of $\mathcal{D}$.*

The body *of C is the subgraph of G whose vertices are those of C minus the **inner** vertices of its children; and whose edges are those of C minus those of its children. We denote it by* $\mathsf{body}_{\mathcal{D}}(C)$*.*

The x-body of C *is the graph whose interface is the interface of C, whose vertices are the vertices of the body of C, and whose edges are the edges of the body of C plus, for each child F of C, an x-edge whose interface is the interface of F. We denote it by x-*$\mathsf{body}_{\mathcal{D}}(C)$*.*

▶ **Definition 73** (L-decompositions)**.** *Let L be a graph language. An L-decomposition of a graph G is a decomposition of G such that the x-body of each of its components is in L.*

▶ **Example 74.** Below, from left to right, a component of the decomposition of Ex. 71, its body, and its x-body. Actually, each monochromatic subgraph of G corresponds to the body of a component of this decomposition.



▶ Remark 75. The body of a component is a subgraph of G, but its x-body is not a subgraph of G in general, because of the added x-edges.

▶ **Proposition 76.** *Let L be a graph language. We have:*

$$G \in \mu x.L \qquad \Leftrightarrow \qquad \exists \mathcal{D}. \quad \mathcal{D} \textit{ is an } L\textit{-decomposition of } G.$$

The proof of this proposition is based on the following easy lemma:

▶ **Lemma 77.** *Let L be a language, $G, G_1, \dots, G_k$ be graphs and H a k-context satisfying:*

$$G = H[G_1, \dots, G_k] \qquad \textit{and} \qquad H[x, \dots, x] \in L$$

Let $\mathcal{D}_i$ be an L-decomposition for G_i, for $i \in [1, k]$, and let $\mathcal{D}$ be the following set

$$\mathcal{D} := \mathcal{D}_1 \cup \dots \cup \mathcal{D}_k \cup \{G\}\,.$$

The set $\mathcal{D}$ is an L-decomposition of G.

Proof of Prop. 76. We show by induction on $n \in \mathbb{N}$, that the property P_n is true:

$$P_n : \qquad \forall G. \quad G \in L^{n,x} \quad \Rightarrow \quad \exists \mathcal{D}. \quad \mathcal{D} \text{ is an } L\text{-decomposition of } G.$$

The base case is trivial, and the inductive case is based on Lemma 77. ◀

5.1.2 The logic CMSO^d

Let φ be a CMSO formula defining a graph language L. Using Prop 76, we can express that a graph G is in the iteration $\mu x.L$ by guessing a decomposition $\mathcal{D}$ of G, and ensuring that the x-body of each component satisfies φ. But guessing a set of subgraphs is not expressible in CMSO. This is why we introduce CMSO^d, an extension of CMSO where this is allowed.

▶ **Definition 78** (CMSO^d logic). *Let $\mathbb{X}_d$ be a set of* graph set variables, *whose elements are denoted $\mathcal{X}, \mathcal{Y} \dots$. The formulas of* CMSO^d *are of the following form:*

$$\varphi := \mathsf{CMSO} \mid \exists \mathcal{X}.\ \varphi \mid (s, Z, t) \in \mathcal{X} \qquad\qquad (\mathcal{X} \in \mathbb{X}_d, \quad Z \in \mathbb{X}_2, \quad s, t \in \mathbb{X}_1).$$

Free and bound variables are defined as usual. As for CMSO, we need to define the semantics of a formula over pointed graphs to handle free variables.

▶ **Definition 79** (Semantics of CMSO^d). *Let G be a graph and Γ be a set of variables.*

An interpretation *of Γ is a function mapping every first-order variable of Γ to an edge or vertex of G, every set variable to a set of edges and vertices of G, and every graph set variable to a set of subgraphs of G.*

We define the satisfiability relation $\langle G, I\rangle \models \varphi$ *as usual, by induction on φ. The only new cases compared to* CMSO *are the quantification $\exists \mathcal{X}$ which is interpreted as "there exists a set of subgraphs $\mathcal{X}$", and the formulas $(s, Z, t) \in \mathcal{X}$ which are interpreted as "the graph whose input is s, whose output is t and whose set of edges and vertices is Z, is an element of $\mathcal{X}$".*

▶ **Proposition 80.** *There is a* CMSO^d *formula* $\mathsf{decomp}(\mathcal{X})$, without graph set quantification, *such that for every graph* G *and every set of subgraphs* $\mathcal{D}$ *of* G*, we have:*

$$\langle G, \mathcal{X} \mapsto \mathcal{D} \rangle \models \mathsf{decomp}(\mathcal{X}) \quad \Leftrightarrow \quad \mathcal{D} \textit{ is a decomposition of } G.$$

Proof. We can express in CMSO^d that a graph is a module of another graph using Prop. 35. We can express that two graphs (s, Z, t) and (s', Z', t') are independent using the following formula:

$$(x \in Z \ \wedge \ x \in Z') \ \ \Rightarrow (x = s \ \vee \ x = s' \ \vee \ x = t \ \vee \ x = t')$$

This is how we express that $\mathcal{X}$ is a decomposition. Note that we do not need to introduce a quantification on new graph set variables. ◀

5.1.3 Iteration is expressible in CMSO^d

Given a CMSO formula φ, we construct a formula $[\![\varphi]\!]$ having $\mathcal{X}$ as unique free variable, which expresses the fact that the x-body the head of the decomposition $\mathcal{X}$ satisfies φ. To construct $[\![\varphi]\!]$, we need the following definition.

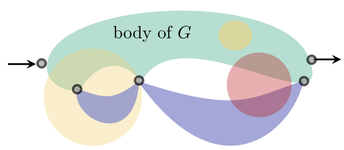
▶ **Definition 81** (Complete sets). *Let* $\mathcal{D}$ *be a decomposition of a graph* G*.*

Let H *be a set of edges and vertices of* G*. We say that* H *is* complete *if, whenever it contains an edge or an inner vertex of a child* C *of* G *(seen as a component of* $\mathcal{D}$*), then it contains all the edges and inner vertices of* C*.*

Let K *be a set of edges and vertices of the* x*-body of* G*. We denote by* $\mathsf{completion}_{\mathcal{D}}(K)$ *the set of edges and vertices of* G*, obtained from* K *by replacing every* x*-edge coming from a child* C *of* G *by the set of edges and inner vertices* C*.*

▶ Remark 82. Note that if H is complete, there is a set S such that $H = \mathsf{completion}_{\mathcal{D}}(S)$.

Here is a picture illustrating complete sets. The green part is the body of G and the purple modules are its children. The yellow sets are complete, but the pink one is not.



▶ **Proposition 83.** *There are* CMSO *formulas* $\mathsf{child}_{\mathcal{X}}(Y)$*,* $\mathsf{complete}_{\mathcal{X}}(Y)$*,* $\mathsf{body\text{-}edge}_{\mathcal{X}}(Y)$*,* $\mathsf{source}_{\mathcal{X}}(Y, Z)$*,* $\mathsf{target}_{\mathcal{X}}(Y, Z)$ *and* $\mathsf{choice}_{\mathcal{X}}(Y, Z)$ *such that, for every graph* G*, every decomposition* $\mathcal{D}$*, every subsets* H *and* K *of the edges and the vertices of* G*, we have the following, where we write, by abuse of notation,* $\mathsf{child}_{\mathcal{D}}(H)$ *instead of* $\langle G, \mathcal{X} \mapsto \mathcal{D}, Y \mapsto H \rangle \models \mathsf{child}_{\mathcal{X}}(Y)$ *etc:*

$$
\begin{array}{lcl}
\mathsf{child}_{\mathcal{D}}(s,H,t) & \Leftrightarrow & \textit{$s,t\notin H$ and $(s,H\cup\{s,t\},t)$ is a child of G w.r.t. $\mathcal{D}$.}\\
\mathsf{child}_{\mathcal{D}}(H) & \Leftrightarrow & \textit{H is the set of edges and inner vertices of a child of G w.r.t. $\mathcal{D}$.}\\
\mathsf{complete}_{\mathcal{D}}(H) & \Leftrightarrow & \textit{H is complete w.r.t. $\mathcal{D}$.}\\
\mathsf{body\text{-}edge}_{\mathcal{D}}(H) & \Leftrightarrow & \textit{H is a singleton containing an edge from $\mathsf{body}_{\mathcal{D}}(G)$.}\\
\mathsf{source}_{\mathcal{D}}(H,K) & \Leftrightarrow & \textit{there are vertices s and t of G such that $H=\{s\}$ and $\mathsf{child}_{\mathcal{D}}(s,K,t)$.}\\
\mathsf{target}_{\mathcal{D}}(H,K) & \Leftrightarrow & \textit{there are vertices s and t of G such that $H=\{t\}$ and $\mathsf{child}_{\mathcal{D}}(s,K,t)$.}\\
\mathsf{choice}_{\mathcal{D}}(H,K) & \Leftrightarrow & \textit{H is complete, K contains $\mathsf{body}_{\mathcal{D}}(G)\cap H$, and contains exactly one}\\
 & & \textit{edge of each child of G contained in H.}
\end{array}
$$

Proof. We define the formulas of the proposition as follows. The formulas between quotation marks are not primitives of CMSO but can be easily defined by CMSO formulas.

$$
\begin{aligned}
\mathsf{child}_{\mathcal{X}}(x,Y,y) \;:=\; & \neg(x\in Y)\wedge\neg(y\in Y)\;\wedge\;\exists Z.\,(\text{“}Z=Y\cup\{x,y\}\text{”}\;\wedge\;(x,Z,y)\in\mathcal{X})\\
& \wedge\big(\,\forall Z'.\forall x',\forall y'.\;(x',Z',y')\in\mathcal{X}\wedge\text{“}(x,Z,y)\text{ is a module of }(x',Z',y')\text{”}\\
& \qquad\Rightarrow\;\text{“}(x',Z',y')\text{ is the whole graph.”}\big)
\end{aligned}
$$

$$
\begin{array}{lcl}
\mathsf{child}_{\mathcal{X}}(Y) & := & \exists x.\;\exists y.\;\mathsf{child}_{\mathcal{X}}(x,Y,y)\\
\mathsf{complete}_{\mathcal{X}}(Y) & := & (\exists Z.\;\exists x.\;\mathsf{child}_{\mathcal{X}}(Z)\;\wedge\;\text{“}x\in Y\cap Z\text{”})\;\;\Rightarrow\;\;(Z\subseteq Y)\\
\mathsf{body}_{\mathcal{X}}(x) & := & \forall Z.\;\;\mathsf{child}_{\mathcal{X}}(Z)\;\Rightarrow\;\neg(x\in Z)\\
\mathsf{body\text{-}edge}_{\mathcal{X}}(Y) & := & \exists x.\;\text{“}Y=\{x\}\text{”}\;\wedge\;\mathsf{body}_{\mathcal{X}}(x)\\
\mathsf{choice}_{\mathcal{X}}(Y,Z) & := & \mathsf{complete}_{\mathcal{X}}(Y)\;\wedge\;\forall x.\;(x\in Y)\wedge\mathsf{body}_{\mathcal{X}}(x)\;\Rightarrow\;(x\in Z)\\
 & & \wedge\,\forall C.\;\mathsf{child}_{\mathcal{X}}(C)\;\wedge\;(C\subseteq Y)\Rightarrow\;\text{“}\exists!x\in(C\cap Z)\text{”}
\end{array}
$$

The formula “$\exists! x\in(C\cap Z)$” is the following:

$$\exists x.\;\text{“}(x\in C\cap Z)\text{”}\wedge\forall y.\;\text{“}(y\in C\cap Z)\text{”}\;\Rightarrow\;(y=x)$$

◀

We construct the formula $[\![\varphi]\!]$ by induction on the structure of φ. We suppose that φ is build using the syntax of CMSO where only set variables are allowed.

▶ **Definition 84.** *Let φ be a* CMSO *formula whose free variables are Γ. We define the* CMSO^d

formula $[\![\varphi]\!]$*, whose free variables are* $\Gamma \cup \{\mathcal{X}\}$*, by induction as follows:*

$$\begin{aligned}
[\![\varphi \vee \psi]\!] &= [\![\varphi]\!] \vee [\![\psi]\!] \\
[\![\neg\varphi]\!] &= \neg\,[\![\varphi]\!] \\
[\![(|Y| \equiv k)[m]]\!] &= \exists Z.\ \mathsf{choice}_{\mathcal{X}}(Y,Z) \wedge (|Z| \equiv k)[m] \\
[\![Y \subseteq Z]\!] &= Y \subseteq Z \\
[\![\mathsf{input}(Y)]\!] &= \mathsf{input}(Y) \\
[\![\mathsf{output}(Y)]\!] &= \mathsf{output}(Y) \\
[\![a(Y)]\!] &= a(Y) \qquad (a \neq x) \\
[\![x(Y)]\!] &= \mathsf{child}_{\mathcal{X}}(Y) \vee (\mathsf{body\text{-}edge}_{\mathcal{X}}(Y) \wedge x(Y)) \\
[\![\exists Y.\ \varphi]\!] &= \exists Y.\ \mathsf{complete}_{\mathcal{X}}(Y) \wedge [\![\varphi]\!] \\
[\![\mathsf{source}(Y,Z)]\!] &= (\mathsf{body\text{-}edge}_{\mathcal{X}}(Z) \wedge \mathsf{source}(Y,Z)) \vee (\mathsf{child}_{\mathcal{X}}(Z) \wedge \mathsf{source}_{\mathcal{X}}(Y,Z)) \\
[\![\mathsf{target}(Y,Z)]\!] &= (\mathsf{body\text{-}edge}_{\mathcal{X}}(Z) \wedge \mathsf{target}(Y,Z)) \vee (\mathsf{child}_{\mathcal{X}}(Z) \wedge \mathsf{target}_{\mathcal{X}}(Y,Z))
\end{aligned}$$

Transfer results are results of this form: to check that a transformation $\mathsf{f}(G)$ of a structure G satisfies a formula φ, construct a formula $\mathsf{f}^{-1}(\varphi)$ that G should satisfy. The proposition below is a transfer result, where the transformation is the x-body.

▶ **Proposition 85.** *Given a* CMSO *sentence* φ *defining, there is a* CMSO^d *formula* $[\![\varphi]\!]$ *having* $\mathcal{X}$ *as unique free variable, such that for every graph* G *and every decomposition* $\mathcal{D}$ *of* G whose components are non-empty*:*

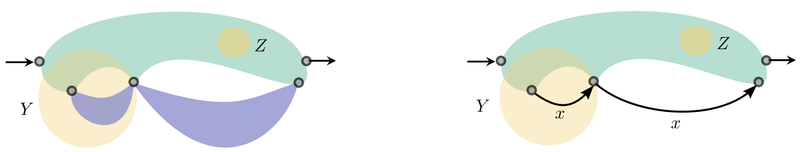
$$\langle G, \mathcal{X} \mapsto \mathcal{D}\rangle \models [\![\varphi]\!] \qquad \Leftrightarrow \qquad x\text{-}\mathsf{body}_{\mathcal{D}}(G) \models \varphi.$$

Proof. We start by the following definition.

▶ **Definition 86.** *Let* $\mathcal{D}$ *be a decomposition of* G *and* I *an interpretation of the set variables* Γ *in the* x*-body of* G*. We define* $I_{\mathcal{D}}$*, the interpretation of* $\Gamma \cup \{\mathcal{X}\}$ *in* G *as follows.*

$$\begin{aligned}
I_{\mathcal{D}} :\ & \mathcal{X} \mapsto \mathcal{D}, \\
& Y \mapsto \mathsf{completion}_{\mathcal{D}}(I(Y)), \quad Y \neq \mathcal{X}.
\end{aligned}$$

Let G be the left graph below, the purple graphs being its children for a decomposition $\mathcal{D}$. The right graph is its x-body. The right yellow circles represent the interpretation I of the variables Y and Z in the x-body. The left yellow circles represent the interpretation $I_{\mathcal{D}}$ of these variables in G.



The following lemma, which generalizes Prop. 85, can be proved by a straightforward induction, concluding the proof of this proposition.

▶ **Lemma 87.** *Let* $\mathcal{D}$ *be a decomposition of* G*,* φ *a* CMSO *formula whose variables are* Γ *and* I *an interpretation of* Γ *in the* x*-body of* G*. We have:*

$$\langle G, I_{\mathcal{D}}\rangle \models [\![\varphi]\!] \qquad \Leftrightarrow \qquad \langle x\text{-}\mathsf{body}_{\mathcal{D}}(G), I\rangle \models \varphi$$

We added the non-emptiness condition on the components of $\mathcal{D}$ to handle the case where $\varphi = (|Y| \equiv k)[m]$. ◀

The formula $[\![\varphi]\!]$ expresses the fact that the x-body of the head of a decomposition satisfies φ. Using this formula and the localization construction of Prop. 14, we construct a formula $\mu x.L$ saying that the x-body of **all** the components of a decomposition satisfy φ.

▶ **Definition 88.** *If φ is a* CMSO *formula, we let $\mu x.\varphi$ be the following* CMSO^d *formula:*

$$\mu x.\varphi := \ \exists \mathcal{X}.\ \mathsf{decomp}(\mathcal{X}) \quad \wedge \quad \forall s.\forall Z.\ \forall t.\ (s,Z,t) \in \mathcal{X} \ \Rightarrow\ [\![\varphi]\!]\,|_{(s,Z,t)}$$

The following proposition says that the language of $\mu x.\varphi$ is the iteration of that of φ.

▶ **Proposition 89.** *If φ is a* CMSO *formula defining a language of non-empty graphs, then:*

$$\mathcal{L}(\mu x.\varphi) = \mu x.\mathcal{L}(\varphi).$$

Proof. Let $[\varphi]$ be the following formula:

$$[\varphi] \ := \ \forall s.\forall Z.\ \forall t.\ \ (s,Z,t) \in \mathcal{X} \ \Rightarrow\ [\![\varphi]\!]\,|_{(s,Z,t)}$$

By Prop. 85 and Prop. 14, we can prove the following lemma:

▶ **Lemma 90.** *For every graph G and every decomposition $\mathcal{D}$ of G:*

$$\langle G, \mathcal{X} \mapsto \mathcal{D}\rangle \models [\varphi] \qquad \Leftrightarrow \qquad \forall C \in \mathcal{D},\ x\text{-}\mathsf{body}_{\mathcal{D}}(C) \models \varphi.$$

We conclude the proof by Prop. 76. ◀

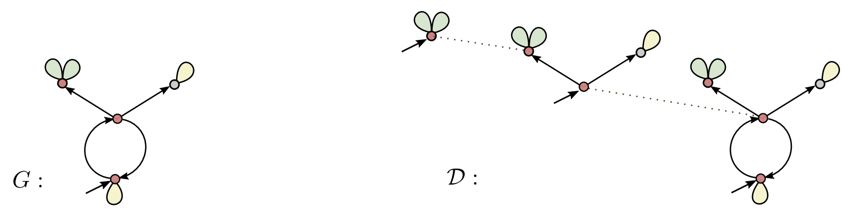
▶ **Corollary 91.** *If L is* CMSO *definable then $\mu x.L$ is* CMSO^d *definable.*

5.2 Guarded iteration of CMSO languages is CMSO^r definable

The idea here is that when the iteration $\mu x.L$ is guarded, L-decompositions can be encoded by sets of edges and vertices and by companion relations.

5.2.1 The case of test languages

Let $\mu x.L$ be a guarded iteration of type *test*, $G \in \mu x.L$ and $\mathcal{D}$ an L-decomposition of G. Suppose that G is the left graph below, and that the red vertices are the interfaces[3] of $\mathcal{D}$.



We claim that, thanks to the guard condition, this information is enough to reconstruct the whole decomposition $\mathcal{D}$. More precisely, we claim that the components of $\mathcal{D}$ are exactly the maximal modules of G, whose interfaces are the red vertices, as depicted above.

[3] Recall that test graphs are unary, hence all the components of a decomposition of G are unary.

▶ **Definition 92.** *Let G be a graph and S be a set of vertices of G. We define $\mathcal{D}_{\mathsf{t}}(S)$ as the set of maximal modules of G, whose type is test, and whose interfaces belong to S.*

▶ **Proposition 93.** *Let $\mu x.L$ be a guarded iteration of type test. We have:*

$$G \in \mu x.L \quad \Leftrightarrow \quad \exists S. \; S \textit{ is a set of vertices of } G \textit{ and } \mathcal{D}_{\mathsf{t}}(S) \textit{ is an } L\textit{-decomposition of } G.$$

Proof. ($\Rightarrow$) Follows from Prop. 76. To prove ($\Leftarrow$), we define the property P_n as follows:

$$P_n: \quad \forall G. \quad G \in L^{n,x} \;\Rightarrow\; \exists S. \; S \text{ is a set of vertices of } G \text{ and } \mathcal{D}_{\mathsf{t}}(S) \text{ is an } L\text{-decomposition of } G.$$

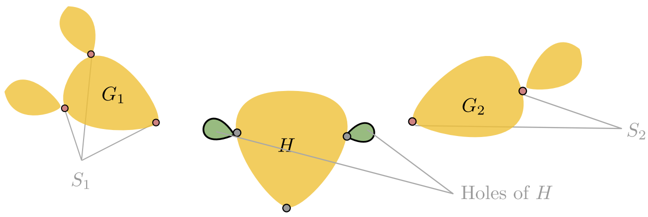
We prove, by induction on n, that P_n is valid for every $n \geq 1$, and this is enough to conclude.

When $n = 1$, take S to be the singleton containing the interface of G. We have that $\mathcal{D}_{\mathsf{t}}(S) = \{G\}$ and since $G \in L$, we have that $\mathcal{D}_{\mathsf{t}}(S)$ is an L-decomposition of G.

Let $G \in L^{n+1,x}$. By definition, there is a k-context H and graphs $G_1, \ldots, G_k$ such that:

$$G = H[G_1, \ldots, G_k], \quad H[x, \ldots, x] \in L \quad \text{and} \quad G_i \in L^{n,x}, \text{ for } i \in [1, k].$$

Thanks to the guard condition, there is no module of H parallel to a hole of H. For every $i \in [1, k]$, let S_i be the set of vertices provided by the induction hypothesis applied to the graph G_i. Here is a picture illustrating these notations:



By Lemma 77, the set of subgraphs $\mathcal{D}$ defined below is an L-decomposition of G.

$$\mathcal{D} := \mathcal{D}_{\mathsf{t}}(S_1) \cup \cdots \cup \mathcal{D}_{\mathsf{t}}(S_k) \cup \{G\}$$

To conclude we only need to find a set of vertices S of G such that $\mathcal{D}_{\mathsf{t}}(S) = \mathcal{D}$. Let $S = S_1 \cup \cdots \cup S_k \cup \{\iota\}$, where ι is the interface of G. Let us show that $\mathcal{D}_{\mathsf{t}}(S) = \mathcal{D}$. This is a consequence of the following lemma:

▶ **Lemma 94.** *Let C be a context, K a graph and I an interface in K of the same arity as the hole of C. Suppose that the hole of C is not parallel to any module. We have:*

$$\text{max-module}_{C[K]}(I) = \text{max-module}_{K}(I)$$

Proof. Let $M := \text{max-module}_K(I)$. Note that M is a module of $G[K]$, the question is its maximality. If I is not the interface of K, then M is maximal in $G[K]$ because this substitution does not add any modules to M. Suppose that I is the interface of H. In this case, we have $M = K$. Suppose by contradiction that there is a module M' of $G[K]$ strictly containing K and whose interface is I. Hence, there is a module M'' such that $M' = (K \parallel M'')$. This means that M'' is module parallel to the hole of C, which is not possible by hypothesis. ◀

◀

▶ **Theorem 95.** *Suppose that $\mu x.L$ is a guarded iteration of type test. We have:*

$$L \textit{ is } \mathsf{CMSO} \textit{ definable} \quad \Rightarrow \quad \mu x.L \textit{ is } \mathsf{CMSO} \textit{ definable}$$

Proof. Let φ be a CMSO formula whose language is L. We transform the CMSO^d formula $\mu x.\varphi$ of Def. 88, whose language is $\mu x.L$, into a CMSO formula $\mu x^g.\varphi$ of the same language. The formula $\mu x^g.\varphi$ is obtained by replacing the quantification $\exists \mathcal{X}.$ by the set quantification $\exists S.$, and by replacing every sub-formula of $\mu x.\varphi$ of the form $(s, Z, t) \in \mathcal{X}$ by this formula:

$$(s = t) \ \wedge \ s \in S \ \wedge \ \text{“}(s, Z, t) \text{ is a maximal module”}$$

The last part of this formula is expressible in CMSO thanks to Prop. 35. The language of $\mu x^g.\varphi$ is the set of graphs for which we can find an L-decomposition encoded by a set of vertices S, and this is precisely the language $\mu x.L$ thanks to Prop. 93. ◀

5.2.2 The case of domain languages

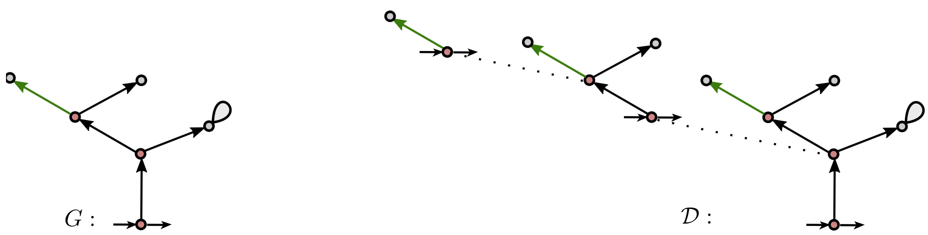
Let $\mu x.L$ be a guarded iteration of type *domain*, G a graph of $\mu x.L$ and $\mathcal{D}$ an L-decomposition of G. Contrarily to the test case, the set of vertices of G corresponding to the interfaces of $\mathcal{D}$, are not enough information to reconstruct $\mathcal{D}$. Indeed, in this case, a component of $\mathcal{D}$ whose interface is v is not necessarily the maximal module at v, but some domain module of interface v, among possibly many others. A way to determine if a domain module is in the decomposition is to check whether it contains an interface of the decomposition. This works for the components which are not the leaves of the decomposition. For the other components, we need to say explicitly which domain modules are the leaves. Since the later are pairwise independent, we can do so by coloring their inner edges and vertices.

In the following, we show that a set of vertices of a graph (representing the interfaces of a decomposition) together with a coloring of this graph (indicating which modules are leaves), is enough to recover the decomposition. This justifies the following definitions.

▶ **Definition 96** (Coloring, Active modules)**.** *A* coloring *of a graph G is a set of its edges called* leaf edges*. A module of G is* active *if it contains a leaf edge.*

▶ **Definition 97** ($\mathcal{D}_{\mathsf{d}}(S, \mathsf{col})$)**.** *Let G be a graph, S a set of vertices and* col *a coloring of G. We let $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col})$ bet the set of active modules of G of type* d*, whose interfaces belong to S.*

▶ **Example 98.** Consider the graph G below, let S be the set of red vertices and let col be the set containing the green edge. The decomposition $\mathcal{D}$ below is $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col})$.



▶ **Proposition 99.** *Let L be a domain language and $\mu x.L$ a guarded iteration. We have:*

$$G \in \mu x.L \quad \Leftrightarrow \quad \exists S, \mathsf{col}. \quad S \textit{ is a set of vertices and } \mathsf{col} \textit{ a coloring of } G \textit{ such that } \mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) \textit{ is an } L\textit{-decomposition of } G.$$

Proof. The implication ($\Leftarrow$) follows from Prop. 76. Let us prove the implication ($\Rightarrow$). Let P_n be the following property:

$$P_n : \quad \forall G.\ G \in L^{n,x} \quad \Rightarrow \quad \exists S, \mathsf{col}. \quad \begin{array}{l} S \text{ is a set of vertices and } \mathsf{col} \text{ a coloring of } G \text{ such that} \\ \mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) \text{ is an } L\text{-decomposition of } G. \end{array}$$

We prove, by induction on n, that P_n is valid for every $n \geq 1$, which is enough to conclude.

When $n = 1$, we let S be the singleton containing the interface of G and col be the coloring where all the edges of G are leaves. We have $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) = \{G\}$, and since $G \in L$, $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col})$ is an L-decomposition of G.

Let $G \in L^{n+1,x}$. There are graphs $G_1, \dots, G_k$ and a k-context H satisfying:

$$G = H[G_1, \dots, G_k], \quad H[x, \dots, x] \in L \quad \text{and} \quad G_i \in L^{n,x} \text{ for } i \in [1, k].$$

Let S_i and col_i be the set of vertices and the coloring provided by induction hypothesis applied to G_i, for $i \in [1, k]$. By Lemma 77, the set of subgraphs

$$\mathcal{D} := \mathcal{D}_{\mathsf{d}}(S_1, \mathsf{col}_1) \cup \cdots \cup \mathcal{D}_{\mathsf{d}}(S_k, \mathsf{col}_k) \cup \{G\}$$

is an L-decomposition of G. To conclude we only need to find a set of vertices S and a coloring col of G such that $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) = \mathcal{D}$. Let ι be the interface of G, we set:

$$S := S_1 \cup \cdots \cup S_k \cup \{\iota\} \qquad \text{and} \qquad \mathsf{col} := \mathsf{col}_1 \cup \cdots \cup \mathsf{col}_k$$

It is clear that $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) = \mathcal{D}$. ◀

▶ **Theorem 100.** *Suppose that $\mu x.L : \mathsf{d}$ be a guarded iteration. We have:*

$$L \textit{ is } \mathsf{CMSO} \textit{ definable} \quad \Rightarrow \quad \mu x.L \textit{ is } \mathsf{CMSO} \textit{ definable}$$

Proof. Let φ be a CMSO formula whose language is L. We transform the CMSO^d formula $\mu x.\varphi$ of Def. 88, whose language is $\mu x.L$, into a CMSO formula $\mu x^g.\varphi$ of the same language. The formula $\mu x^g.\varphi$ is obtained by replacing the quantification $\exists \mathcal{X}.$ by the set quantifications $\exists S.\ \exists \mathsf{col}.$, by saying that col is a set of edges and by replacing every sub-formula of $\mu x.\varphi$ of the form $(s, Z, t) \in \mathcal{X}$ by the following formula:

$$(s = t) \ \wedge \ s \in S \ \wedge \ \text{“}(s, Z, t) \text{ is a module of type domain”} \ \wedge \ \text{“}(s, Z, t) \text{ is active”}$$

Being a module of type domain is expressible in CMSO thanks to Prop. 35. Being active is CMSO definable by the following formula:

$$\exists x \in Z.\ x \in \mathsf{col}$$

The language of $\mu x^g.\varphi$ is the set of graphs for which there is an L-decomposition encoded by a set of vertices S and a coloring col. This is precisely the language $\mu x.L$ by Prop. 99. ◀

5.2.3 The case of parallel languages

The case of guarded iterations of type parallel is similar to the test case. Let $\mu x.L$ be a guarded iteration of type parallel, G a graph of $\mu x.L$ and $\mathcal{D}$ an L-decomposition of G. We show that the set of interfaces I of $\mathcal{D}$ is enough to recover the whole decomposition $\mathcal{D}$, because its components are the maximal modules of G whose interfaces belong to I. However, in this case, the set of interfaces I is no longer a set of vertices, but a set of pairs of vertices, that is a relation on the vertices of G. We will show that this relation is necessarily a companion relation. Using this result and the fact that CMSO and CMSO^r have the same expressive power, we prove that the iteration is CMSO definable.

▶ **Definition 101** ($\mathcal{D}_{\mathsf{p}}(R)$)**.** *Let G be a graph and R a relation on the vertices of G. We define $\mathcal{D}_{\mathsf{p}}(R)$ as the set of maximal modules of G, whose type is parallel, and whose interfaces belong to S.*

▶ **Proposition 102.** *Let $\mu x.L$ be a guarded iteration of type parallel. We have:*

$$G \in \mu x.L \quad \Leftrightarrow \quad \exists R. \quad R \textit{ is a set of vertices of } G \textit{ and} \\ \mathcal{D}_{\mathsf{p}}(R) \textit{ is an } L\textit{-decomposition of } G.$$

Proof. ($\Rightarrow$) Follows from Prop. 76. To prove ($\Leftarrow$), we let P_n be the following property:

$$\forall G. \quad G \in L^{n,x} \quad \Rightarrow \quad \exists R. \quad R \text{ is a relation on the vertices of } G \text{ and} \\ \mathcal{D}_{\mathsf{p}}(R) \text{ is an } L\text{-decomposition of } G.$$

We prove, by induction on n, that P_n is valid for every $n \geq 1$, and this is enough to conclude.

When $n = 1$, take R to be the singleton containing the interface of G. We have that $\mathcal{D}_{\mathsf{p}}(R) = \{G\}$ and since $G \in L$, we have that $\mathcal{D}_{\mathsf{p}}(R)$ is an L-decomposition of G.

Let $G \in L^{n+1,x}$. There is a k-context H and graphs $G_1, \ldots, G_k$ such that:

$$G = H[G_1, \ldots, G_k], \quad H[x, \ldots, x] \in L \quad \text{and} \quad G_i \in L^{n,x}, \text{ for } i \in [1, k].$$

Thanks to the guard condition, the holes of H have no parallel modules. For every $i \in [1, k]$, let R_i be the relation provided by the induction hypothesis applied to the graph G_i. By Lemma 77, the following set of subgraphs:

$$\mathcal{D} := \mathcal{D}_{\mathsf{p}}(R_1) \cup \cdots \cup \mathcal{D}_{\mathsf{p}}(R_k) \cup \{G\}$$

is an L-decomposition of G. To conclude we only need to find a relation R on the vertices of G such that $\mathcal{D}_{\mathsf{p}}(R) = \mathcal{D}$. If I is the interface of G, we let R to be the following relation:

$$R = R_1 \cup \cdots \cup R_k \cup \{I\}$$

The fact that $\mathcal{D}_{\mathsf{t}}(S) = \mathcal{D}$ is a consequence of lemma 94. ◀

▶ **Proposition 103.** *Let $\mu x.L$ be an iteration of type parallel and let G a graph. The interfaces of every L-decomposition of G form a companion relation.*

Proof. We prove by induction on $n \geq 1$ that the interfaces of every L-decomposition of depth n of some graph G form a companion relation, witnessed by a set of paths P, such that the interface of G is witnessed by two parallel paths of P.

When $n = 1$, the decomposition $\mathcal{D}$ is reduced to the graph G. Since G is parallel, it has two parallel paths whose interface is the interface of G. Take P to be these two paths.

Suppose that $\mathcal{D}$ is a decomposition of depth $n + 1$. Hence it is of the form:

$$\mathcal{D} = \mathcal{D}_1 \cup \ldots \mathcal{D}_k \cup \{G\}$$

where $\mathcal{D}_i$ is an L-decomposition of depth at most n, of a graph G_i, for every $i \in [1, k]$. Let P_i be the set of paths provided by the induction hypothesis for $\mathcal{D}_i$, and let p_i, q_i be the two paths witnessing the interface of G_i, for $i \in [1, k]$.

We set $H := x\text{-}\mathsf{body}_{\mathcal{D}}(G)$. Since H is parallel, it has two parallel paths p and q whose interface is the interface of H. We transform the paths p and q of H into the paths p' and q'

of G as follows. The paths p' and q' are obtained from p and q respectively by the following procedure: if e is an x-edge of H which is substituted by some G_i, then replace e by the path of p_i. Let P be the following set of paths:

$$P = (P_1 \setminus \{p_1\}) \cup \ldots (P_k \setminus \{p_k\}) \cup \{p', q'\} .$$

The set P is orthogonal and witnesses the interfaces of $\mathcal{D}$. Moreover, the interface of G is witnessed by two parallel paths of P, namely p' and q'. This concludes the proof. ◀

▶ Remark 104. We do not need the guard condition for Prop. 103.

▶ **Corollary 105.** *Let $\mu x.L$ be a guarded iteration of type parallel. We have:*

$$G \in \mu x.L \quad \Leftrightarrow \quad \exists R. \quad \begin{array}{l} R \textit{ is a companion relation on the vertices of } G \textit{ and} \\ \mathcal{D}_{\mathsf{p}}(R) \textit{ is an } L\textit{-decomposition of } G. \end{array}$$

▶ **Theorem 106.** *Suppose that $\mu x.L$ is a guarded iteration of type parallel. We have:*

$$L \textit{ is } \mathsf{CMSO} \textit{ definable} \quad \Rightarrow \quad \mu x.L \textit{ is } \mathsf{CMSO} \textit{ definable}$$

Proof. Let φ be a CMSO formula whose language is L. We will transform the CMSO^d formula $\mu x.\varphi$ of Def. 88, whose language is $\mu x.L$, into a CMSO^r formula $\mu x^r.\varphi$ of the same language. The formula $\mu x^r.\varphi$ is obtained by replacing the quantification $\exists\mathcal{X}$ by the set quantification $\exists R$, and by replacing every sub-formula of $\mu x.\varphi$ of the form $(s, Z, t) \in \mathcal{X}$ by the following formula:

$$(s, t) \in R \ \wedge \ \text{“}(s, Z, t) \text{ is a maximal module”}$$

The last part of this formula is expressible in CMSO thanks to Prop. 35. The language of $\mu x^r.\varphi$ is the set of graphs for which we can find an L-decomposition encoded by a companion relation R, and this is precisely the language $\mu x.L$ thanks to Cor. 105. Since CMSO^r and CMSO have the same expressive power, this concludes the proof. ◀

5.2.4 The case of series languages

Let $\mu x.L$ be a guarded iteration of type series, G a graph in $\mu x.L$ and $\mathcal{D}$ an L-decomposition of G whose set of interfaces is I. As for the domain case, the set I is not enough to reconstruct the decomposition $\mathcal{D}$, and we need a coloring of the graph to determine which modules are the leaves of the decomposition $\mathcal{D}$. We show also that the set of interfaces I is a companion relation, which will be enough to conclude.

▶ **Definition 107** ($\mathcal{D}_{\mathsf{s}}(R, \mathsf{col})$)**.** *Let G be a graph, R a relation on the vertices of G and* col *a coloring of G. We let $\mathcal{D}_{\mathsf{s}}(R, \mathsf{col})$ bet the set of active modules of G of type series, whose interfaces belong to R.*

▶ **Proposition 108.** *Let $\mu x.L$ be a guarded iteration of type series. We have:*

$$G \in \mu x.L \quad \Leftrightarrow \quad \exists R, \mathsf{col}. \quad \begin{array}{l} R \textit{ is a relation on the vertices of } G, \\ \mathsf{col} \textit{ is a coloring of } G \textit{ and} \\ \mathcal{D}_{\mathsf{s}}(R, \mathsf{col}) \textit{ is an } L\textit{-decomposition of } G. \end{array}$$

Proof. ($\Leftarrow$) Follows from Prop. 76. To prove ($\Rightarrow$), we let P_n be the following property:

$$\forall G. \quad G \in L^{n,x} \quad \Leftrightarrow \quad \exists R, \mathsf{col}. \quad \begin{array}{l} R \text{ is a relation on the vertices of } G, \\ \mathsf{col} \text{ is a coloring of } G \text{ and} \\ \mathcal{D}_{\mathsf{s}}(R, \mathsf{col}) \text{ is an } L\text{-decomposition of } G. \end{array}$$

We prove, by induction on n, that P_n is valid for every $n \geq 1$, which is enough to conclude.

When $n = 1$, let R be the singleton containing the interface of G and let col be the coloring where all the edges of G are leaves. We have $\mathcal{D}_{\mathsf{s}}(R, \mathsf{col}) = \{G\}$, and since $G \in L$, $\mathcal{D}_{\mathsf{s}}(R, \mathsf{col})$ is an L-decomposition of G.

Let $G \in L^{n+1,x}$. There are graphs $G_1, \ldots, G_k$ and a k-context H satisfying:

$$G = H[G_1, \ldots, G_k], \quad H[x, \ldots, x] \in L \quad \text{and} \quad G_i \in L^{n,x} \text{ for } i \in [1, k].$$

Let R_i and col_i be the relation and the coloring provided by induction hypothesis applied to G_i, for $i \in [1, k]$. By Lemma 77, the following set of subgraphs:

$$\mathcal{D} := \mathcal{D}_{\mathsf{s}}(R_1, \mathsf{col}_1) \cup \cdots \cup \mathcal{D}_{\mathsf{s}}(R_k, \mathsf{col}_k) \cup \{G\}$$

is an L-decomposition of G. To conclude we only need to find a relation R on the vertices of G and a coloring col of G such that $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) = \mathcal{D}$. Let I be the interface of G, we set:

$$R := R_1 \cup \cdots \cup R_k \cup \{I\} \qquad \text{and} \qquad \mathsf{col} := \mathsf{col}_1 \cup \cdots \cup \mathsf{col}_k.$$

The fact that $\mathcal{D}_{\mathsf{d}}(S, \mathsf{col}) = \mathcal{D}$ is a consequence of following easy lemma:

▶ **Lemma 109.** *If C is a context, K a graph of interface J and I an interface in K, then:*

$$\begin{aligned} \mathsf{series\text{-}modules}_{\,C[K]}(I) &= \mathsf{series\text{-}modules}_{\,K}(I) && \textit{if} \quad I \neq J, \\ &= \mathsf{series\text{-}modules}_{\,K}(I) \cup \mathsf{series\text{-}modules}_{\,\overline{C}}(I) && \textit{if} \quad I = J. \end{aligned}$$

Where $\overline{C}$ is the context C without its hole.

◀

▶ **Proposition 110.** *Let $\mu x.L$ be a guarded iteration of type series and let G be a graph. The interfaces of every L-decomposition of G form a companion relation.*

Proof. We start by proving the following claim:

▷ Claim 111. Let H be a pure binary graph of interface I and suppose that x is s-guarded in H. There is a path p in H of interface I, such that every x-edge of p is a parallel to an x-edge in H.

Proof. We proceed by induction on the size of H. The case where H is atomic is trivial.

If H is series, then we can write H under the following form:

$$H = U_0 \cdot P_1 \cdot U_1 \cdot P_1 \ldots U_{k-1} \cdot P_k \cdot U_k$$

where P_i is a parallel graph and U_j a unary graph for every $i \in [1, k]$ and $j \in [0, k]$. For every $i \in [1, k]$, x is s-guarded in P_i, otherwise x would not be guarded in H. For every $i \in [1, k]$, the induction hypothesis applied to P_i provides us with a path p_i. We let p be the concatenation of the paths p_i, for $i \in [1, k]$. The path p satisfies the required condition.

If H is parallel, then we can write H under the following form:

$$H = H_1 \parallel \cdots \parallel H_k$$

where for every $i \in [1, k]$, the graph H_i is either (1) a series graph which is not the graph of the letter x, or (2) the graph of the letter x.

If there is $j \in [1, k]$ satisfying (1), then the letter x is s-guarded in H_j. We can conclude by induction hypothesis. Otherwise, for all $j \in [1, k]$, H_j is the graph of the letter x. Any path from the the input to the output of H satisfies the condition of the claim. ◀

We say that a path p is *safe* if it does not contain an interface vertex of G as an inner vertex. We prove by induction on $n \geq 1$ that the interfaces of every L-decomposition of depth n of some graph G form a companion relation, witnessed by a safe set of paths P.

When $n = 1$, the decomposition $\mathcal{D}$ is reduced to the graph G. Since G is series, it has a path whose interface is the interface of G. Take P to be the set containing this path.

Suppose that $\mathcal{D}$ is a decomposition of depth $n+1$. Hence it is of the form:

$$\mathcal{D} = \mathcal{D}_1 \cup \ldots \mathcal{D}_k \cup \{G\}$$

where $\mathcal{D}_i$ is an L-decomposition of depth at most n of a graph G_i, for every $i \in [1,k]$. Let P_i be the set of paths provided by the induction hypothesis for $\mathcal{D}_i$ for $i \in [1,k]$.

We set $H = x\text{-}\mathsf{body}_{\mathcal{D}}(G)$. Since x is s-guarded in H, the Claim 111 provides us with a path p. We transform the path p of H into the path p' of G as follows. The path p' is obtained from p as follows: if e is an x-edge of H which is substituted by a graph G_i, for some $i \in [1,k]$, then replace e by the path p_i witnessing the interface of G_i. We denote by Q the set of paths p_i which participated to this procedure. Let P be the following set of paths:

$$P = P_1 \cup \cdots \cup P_k \cup \{p'\} \setminus Q.$$

The set P is safe, orthogonal and witnesses the interfaces of $\mathcal{D}$. This concludes the proof. ◀

▶ Remark 112. Contrarily to Prop. 103, we need the guard condition to prove Prop. 110.

▶ **Corollary 113.** *Let $\mu x.L$ be a guarded iteration of type parallel. We have:*

$$\begin{array}{lll} G \in \mu x.L & \Leftrightarrow \quad \exists R.\ \mathsf{col}. & R \textit{ is a companion relation on the vertices of } G, \\ & & \mathsf{col} \textit{ is a coloring of } G \textit{ and} \\ & & \mathcal{D}_{\mathsf{s}}(R, \mathsf{col}) \textit{ is an } L\textit{-decomposition of } G. \end{array}$$

▶ **Theorem 114.** *Suppose that $\mu x.L$ is a guarded iteration of type series. We have:*

$$L \textit{ is } \mathsf{CMSO} \textit{ definable} \quad \Rightarrow \quad \mu x.L \textit{ is } \mathsf{CMSO} \textit{ definable}$$

Proof. Let φ be a CMSO formula whose language is L. We will transform the CMSO^d formula $\mu x.\varphi$ of Def. 88, whose language is $\mu x.L$, into a CMSO^r formula $\mu x^r.\varphi$ of the same language. The formula $\mu x^r.\varphi$ is obtained by replacing the quantification $\exists \mathcal{X}$ by the set quantifications $\exists R.\ \exists \mathsf{col}.$, expressing that col is a set of edges and by replacing every sub-formula of $\mu x.\varphi$ of the form $(s, Z, t) \in \mathcal{X}$ by the following formula:

$$(s = t) \ \wedge \ s \in S \ \wedge \ \text{“}(s, Z, t) \text{ is a module of type series”} \ \wedge \ \text{“}(s, Z, t) \text{ is active”}$$

Being a module of type series is expressible in CMSO thanks to Prop. 35. Being active is CMSO definable by the following formula:

$$\exists x \in Z.\ x \in \mathsf{col}$$

The language of $\mu x^r.\varphi$ is the set of graphs for which we can find an L-decomposition encoded by a companion relation R and a coloring col, and this is precisely the language $\mu x.L$ thanks to Prop. 108. ◀

6 Recognizable implies regular

▶ **Theorem 115.** *If a language of* tw_2*-graphs is recognizable, then it is regular.*

6.1 Preliminaries

We define below guarded modules. Intuitively, a module is guarded if it is pure of some type τ and whenever we replace it with a letter, this letter is τ-guarded in the obtained graph.

▶ **Definition 116** (Strict modules, Guarded modules). *Let G be a graph and M a module of G. We say that M is* strict *if it does not contain all the edges and vertices of G. We say that M is* guarded *if it is pure and:*

- *M is maximal, if M is parallel or test,*
- *M is parallel to another module of G, if M is series.*

The following lemma follows from the definition of guarded modules.

▶ **Lemma 117.** *Let G be a graph, M a guarded module of G of type τ, and x a letter of the same arity as M. We denote by $G[x/M]$ the graph obtained by replacing the module M by an x-labeled edge whose interface is that of M. The letter x is τ-guarded in $G[x/M]$.*

We will not show Thm. 115 directly, but we will proceed gradually, by showing that this result holds for three sub-classes of tw_2 graphs. First for *alternating-free domain-free graphs*, for *domain-free* graphs (which we define below), then for domain graphs. We finally lift these results to the whole class of tw_2 graphs.

▶ **Definition 118** (Domain-free, alternation-free graphs). *A graph is* domain-free *if all its domain modules are atomic. A graph is* alternation-free *if it has no strict guarded module.*

In all these steps, we use the following lemma:

▶ **Lemma 119.** *Let L be a language of* tw_2*-graphs, x a letter and τ a type. If L is recognizable then the restriction of L to the graphs of type τ (resp. the graphs where x is τ-guarded, word graphs, multiset graphs, domain-free graphs, alternation-free graphs) is also recognizable.*

▶ **Lemma 120.** *In a domain-free graph, every two guarded modules are either independent or module one of the other.*

In an arbitrary graph, every two domain modules are either independent or module one of the other.

▶ **Lemma 121.** *Let G, H be domain-free graphs, x a letter and suppose that H is pure and $G[H/x]$ is a guarded substitution. We have the following equality:*

$$\mathsf{guarded\text{-}modules}(G[H/x]) \;=\; \mathsf{guarded\text{-}modules}(G)[H/x] \;\cup\; \mathsf{guarded\text{-}modules}(H) \cup \{H\}$$

▶ **Lemma 122.** *Let G, H be domain graphs, x a unary letter. We have the following equality:*

$$\mathsf{domain\text{-}modules}(G[H/x]) \;=\; \mathsf{domain\text{-}modules}(G)[H/x] \;\cup\; \mathsf{domain\text{-}modules}(H)$$

define domain-modules and guarded-modules
algebras are ranked now
domain free are basically series-parallel

6.2 Alternation-free domain-free graphs

▶ **Lemma 123.** *If G is an alternation-free domain-free* tw_2 *graph, then* $G = H[\vec{T}/\vec{x}]$ *where*

- H *is either a word or a multiset graph,*
- $\vec{T}$ *are multiset graphs of type test not containing the letters* $\vec{x}$ *and*
- *the letters of* $\vec{x}$ *are* t*-guarded in* G*.*

▶ **Proposition 124.** *If a language of alternation-free domain-free* tw_2 *graphs is recognizable, then it is regular.*

Proof. Let L be a language of non-alternating domain-free graphs, $\mathcal{A}$ an algebra of domain D, $h : \mathbb{G}_{\mathsf{tw}_2}(\Sigma) \to \mathcal{A}$ a homomorphism and $F \subseteq D$ such that $h^{-1}(F) = L$. We denote by L_v the set of graphs whose image by h is v. Note that we have the following equality:

$$L \ = \ \bigcup_{f \in F} L_f.$$

We show in the following that L_v is regular for every $v \in D$, and this is enough to conclude.

For every $v \in D$, we associate a new letter x_v, and denote this new set of letters by Γ. We extend the homomorphism h to tw_2-graphs over the alphabet $\Sigma \cup \Gamma$ by letting $h(x_v) = v$ for every $x_v \in \Gamma$.

For every $v \in D$, let T_v be the set of multiset graphs over Σ of type test whose image by h is v, and let M_v be the set of word or multiset graphs over $\Sigma \cup \Gamma$, where the letters of Γ are t-guarded. Using Lem. 123, we have:

$$L_v = M_v[T_w/x_w, \ w \in D]$$

By Lem. 119 and using the fact that recognizability implies regularity for word and multiset graphs, we conclude that L_v is regular for every $v \in D$. ◀

6.3 Domain-free graphs

▶ **Proposition 125.** *If a language of domain-free graphs is recognizable, then it is regular.*

Proof. Let L be a language of domain-free graphs, $\mathcal{A}$ an algebra of domain D, $h : \mathbb{G}_{\mathsf{tw}_2}(\Sigma) \to \mathcal{A}$ a homomorphism and $F \subseteq D$ such that $h^{-1}(F) = L$. Let us show that L_v, the set of graphs over Σ whose image (by h) is v, is regular for every $v \in D$.

We associate every $v \in D$ with two new letters x_v and y_v of the same arity as v, and let $\Gamma := \{x_v \mid v \in D\}$ and $\Delta := \{y_v \mid v \in D\}$. If $Q \subseteq D$, we denote by X_D and Y_D the subsets of Γ and Δ corresponding to these elements. We extend the homomorphism h to tw_2 graphs over the alphabet $\Sigma \cup \Gamma \cup \Delta$ by letting $h(x_v) = h(y_v) = v$ for every $x_v \in \Gamma$ and $y_v \in \Delta$.

Let $v \in D$, $Q, R \subseteq D$, $X \subseteq \Gamma$, and $Y \subseteq \Delta$. We define the set of graphs $L_v^{Q,R,X,Y}$ as follows. A graph G is in this set if and only if:

- G is a domain-free graph over the alphabet $\Sigma \cup X \cup Y$,
- the image of G is v,
- the image of the strict guarded series modules of G belong to Q,
- the image of the strict guarded parallel modules of G belong to R,
- the letters of X are s-guarded in G ,
- the letters of Y are p-guarded in G.

Let $S_v^{Q,R,X,Y}$ and $P_v^{Q,R,X,Y}$ be the restriction of $L_v^{Q,R,X,Y}$ to series and parallel graphs respectively. Let us show that these two languages are regular when $X \cap X_Q = \emptyset$ and $Y \cap Y_Q = \emptyset$. We proceed by induction on the size of $Q \cup R$. When $Q = R = \emptyset$, the graphs of these sets are alternation-free. Using Lemma 119 and Prop 124, we conclude the base case.

To handle the inductive case, we show the following equalities:

$$S_v^{Q\cup\{w\},R,X,Y} = S_v^{Q,R,X\cup\{x_w\},Y}[\mu x_w.S_w^{Q,R,X\cup\{x_w\},Y}/x_w][S_w^{Q,R,X,Y}/x_w] \qquad (\dagger)$$

$$S_v^{Q,R\cup\{w\},X,Y} = S_v^{Q,R,X,Y\cup\{y_w\}}[\mu y_w.P_w^{Q,R,X,Y\cup\{y_w\}}/y_w][P_w^{Q,R,X,Y}/y_w]$$

$$P_v^{Q\cup\{w\},R,X,Y} = P_v^{Q,R,X\cup\{x_w\},Y}[\mu x_w.S_w^{Q,R,X\cup\{x_w\},Y}/x_w][S_w^{Q,R,X,Y}/x_w]$$

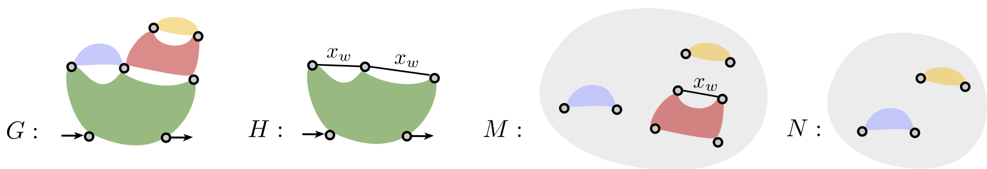
$$P_v^{Q,R\cup\{w\},X,Y} = P_v^{Q,R,X,Y\cup\{y_w\}}[\mu y_w.P_w^{Q,R,X,Y\cup\{y_w\}}/y_w][P_w^{Q,R,X,Y}/y_w]$$

Let us show the equality (†), the others are similar. The right-to-left implication follows from these inclusions which are consequence of Lem. **??**:

- $\mu x_w.S_w^{Q,R,X\cup\{x_w\},Y} \subseteq S_w^{Q\cup\{w\},R,X\cup\{x_w\},Y}$,
- $S_v^{Q,R,X\cup\{x_w\},Y}[S_w^{Q\cup\{w\},R,X\cup\{x_w\},Y}/x_w] \subseteq S_v^{Q\cup\{w\},R,X\cup\{x_w\},Y}$,
- $S_v^{Q\cup\{w\},R,X\cup\{x_w\},Y}[S_w^{Q,R,X,Y}/x_w] \subseteq S_v^{Q\cup\{w\},R,X,Y}$.

Let us show the other direction. We call w-module of a graph K a series module of K whose image by h is w. The w-normal-form of a graph K is the graph obtained from K by recursively replacing its strict w-modules by the letter x_w. This operation yields always the same graph thanks to Lem. 120. Let H be the w-normal-form of G, M be the set of w-normal-forms of its strict w-modules and N the set of w-modules of G with no strict w-modules.

The picture below illustrates these constructions. In the graph G, the colored modules are the w-modules of G.



We have that $G \subseteq H[\mu x_w.M/x_w][N/x_w]$. Note that $M \subseteq$ and $N \subseteq$. In particular the letter x_w is s-guarded in M thanks to Lem. 117. This concludes the left-to-right implication of (†).

Finally, notice that for every $v \in D$, we have:

$$L_v = L_v^{\emptyset,\emptyset,\Gamma,\Delta}[S_w^{D,D,\emptyset,\emptyset}/x_w, w \in D][S_w^{D,D,\emptyset,\emptyset}/y_w, w \in D]$$

The set $L_v^{\emptyset,\emptyset,\Gamma,\Delta}$ is a set of alternation-free and domain-free graphs, its regularity follows from Lem. 119 and Prop. 124. ◀

6.4 Domain graphs

▶ **Lemma 126.** *Let G be a domain graph whose strict domain modules, are all atomic. There is a domain-free graph H such that $G = \mathsf{fg}(H)$.*

▶ **Proposition 127.** *If a language of domain graphs is recognizable, then it is regular.*

Proof. Let L be a language of domain graphs, $\mathcal{A}$ an algebra whose domain is D, $h : \mathbb{G}_{\mathsf{tw}_2}(\Sigma) \to \mathcal{A}$ a homomorphism and $F \subseteq D$ such that $h^{-1}(F) = L$. Let us show that L_v, the set of graphs over Σ whose image is v, is regular for every $v \in D$.

We associate every $v \in D$ with a new letter x_v whose arity is the same as v and we let $\Gamma := \{x_v \mid v \in D\}$. If $Q \subseteq D$, we denote by X_D the letters of Γ corresponding to these elements. We extend the homomorphism h to tw_2-graphs over the alphabet $\Sigma \cup \Gamma$ by letting $h(x_v) = v$ for every $x_v \in \Gamma$.

Let $v \in D$, $Q \subseteq D$ and $X \subseteq \Gamma$. We define the set of graphs $L_v^{Q,X}$, by letting a graph G in this set if and only if:

- G is a domain graph over the alphabet $\Sigma \cup X$,
- the image of G is v,
- the image of the strict domain modules of G belong to Q.

Let us show that $L_v^{Q,X}$ is regular. This is enough to conclude since $L_v = L_v^{D,\emptyset}$.

We proceed by induction on the size of Q. Let $X \subseteq \Gamma$ and suppose that $Q = \emptyset$. For every $w \in D$, let M_w be the set of domain-free graphs over the alphabet $\Sigma \cup X$ whose image is w. The set M_w is the restriction of $h^{-1}(w)$ to those graphs which are domain-free. By Lem. 119, M_w is recognizable. Using Prop. 125, M_v is regular.

By Lem. 126, we have the following equation:

$$L_v^{\emptyset,X} = \bigcup_{\substack{w \in D \\ \mathsf{fg}(w)=v}} \mathsf{fg}(M_w)$$

The set $L_v^{\emptyset,X}$ is regular because M_w is regular, which concludes the base case.

To handle the inductive case, we show the following equality:

$$L_v^{Q\cup\{w\},X} = L_v^{Q,X\cup\{x_w\}}[\mu x_w.\ L_w^{Q,X\cup\{x_w\}}/x_w][L_w^{Q,X}/x_w] \qquad (\dagger)$$

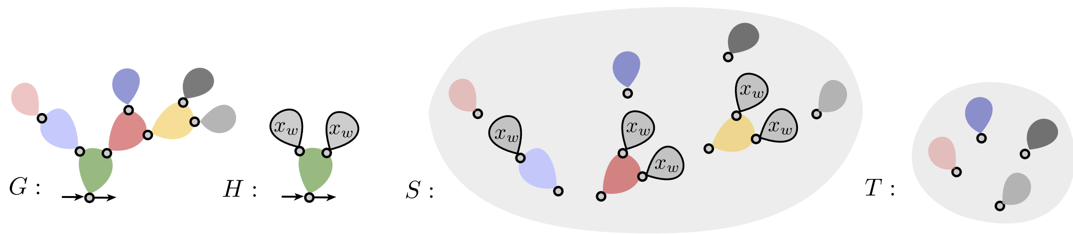
The right-to-left implication holds because of the following inclusions, which follow from Lem. ??:

- $\mu x_w.\ L_w^{Q,X\cup\{x_w\}} \subseteq L_w^{Q\cup\{w\},X\cup\{x_w\}}$,
- $L_v^{Q,X\cup\{x_w\}}[L_w^{Q\cup\{w\},X\cup\{x_w\}}/x_w] \subseteq L_v^{Q\cup\{w\},X\cup\{x_w\}}$,
- $L_v^{Q\cup\{w\},X\cup\{x_w\}}[L_w^{Q,X}/x_w] \subseteq L_v^{Q\cup\{w\},X}$.

Let us prove the other direction. Let $G \in L_v^{Q\cup\{w\},X}$ We define the graph H and the sets of graphs S and T as follows.

We call w-module of a graph K a domain module of K whose image by h is w. The w-normal-form of a graph K is the graph obtained from K by recursively replacing its strict w-modules by the letter x_w. Let H be the w-normal-form of G, S the set of w-normal-forms of its strict w-modules and T the set of w-modules of G with no strict w-modules.

The picture below illustrates these constructions. In the graph G, the inner vertices which are drawn are the interfaces of all domain modules whose image by h is w.



We have that $G \subseteq H[\mu x_w.S/x_w][T/x_w]$, which concludes the left-to-right implication.

The iteration and substitutions in (†) are guarded because the languages which we iterate and substitute and domain languages, and the variable x_w is unary. Since the languages $L_v^{Q,X\cup\{x_w\}}$, $L^{Q,X\cup\{x_w\}}$ and $L_w^{Q,X}$ are regular by induction hypothesis, we conclude that $L_v^{Q\cup\{w\},X}$ is also regular. ◀

6.5 TW$_2$ graphs

▶ **Lemma 128.** *If G is a* tw_2*-graph, then $G = H[\vec{D}/\vec{x}]$ where H is domain-free and $\vec{D}$ are domain graphs.*

Now we are ready to prove Thm. 115.

Proof of Thm. 115. Let L be a language of tw_2-graphs, $\mathcal{A}$ an algebra whose domain is D, $h : \mathbb{G}_{\mathsf{tw}_2}(\Sigma) \to \mathcal{A}$ a homomorphism and $F \subseteq D$ such that $h^{-1}(F) = L$. Let us show that L_v, the set of graphs over Σ whose image is v, is regular for every $v \in D$.

We associate every $v \in D$ with a new letter x_v whose arity is the same as v and we let $\Gamma := \{x_v \mid v \in D\}$. We extend h to tw_2-graphs over the alphabet $\Sigma \cup \Gamma$ by letting $h(x_v) = v$ for every $x_v \in \Gamma$.

For every $v \in D$, we let M_v be the set of domain-free graphs over the alphabet $\Sigma \cup \Gamma$ whose image is v, and let N_v be the set of domain graphs over the alphabet Σ whose image is v. The set M_v is the restriction of $h^{-1}(v)$ to those graphs which are domain-free, hence it is recognizable by Lem. 119. Similarly, N_v is also recognizable. Hence, by Prop. 125 and Prop. 127 they are both regular.

We have the following equation, consequence of Lem. 128:

$$L_v = M_v[N_w/x_w, w \in D]$$

Substitutions in the equation above are guarded because N_w is of type domain and x_w is unary. Since M_v and N_w are regular, the language L_v is also regular. ◀

▶ Remark 129. By analyzing this proof, we see that we never use iteration over test graphs.

7 Conclusion

We are interested in studying the complexity-theoretic properties of our expressions. For instance understanding the complexity of deciding whether an expression is guarded, and what are the costs of translations between different formalisms (expressions, CMSO, algebra). This can help us get a better grasp of what role these expressions can play, and what is the fine interplay between these different formalisms.

As stated in the introduction, this work on tree-width 2 graphs is meant to constitute a first step towards the case of tree-width k.

References

1. Mikolaj Bojanczyk. Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic. *CoRR*, abs/2008.11635, 2020.
2. Mikolaj Bojanczyk and Michal Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *LICS*, pages 407–416. ACM, 2016.
3. Mikolaj Bojanczyk and Michal Pilipczuk. Optimizing tree decompositions in MSO. In *STACS*, volume 66 of *LIPIcs*, pages 15:1–15:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
4. J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. doi:0.1002/malq.19600060105.
5. Enric Cosme-Llópez and Damien Pous. K4-free graphs as a free algebra. In *MFCS*, volume 83 of *LIPIcs*, pages 76:1–76:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
6. Bruno Courcelle and Joost Engelfriet. Graph structure and monadic second-order logic - a language-theoretic approach. In *Encyclopedia of mathematics and its applications*, 2012.
7. Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.
8. Joost Engelfriet. A regular characterization of graph languages definable monadic second-order logic. *Theor. Comput. Sci.*, 88(1):139–150, oct 1991. doi:10.1016/0304-3975(91)90078-G.
9. Zsolt Gazdag and Zoltán L. Németh. A kleene theorem for bisemigroup and binoid languages. *Int. J. Found. Comput. Sci.*, 22:427–446, 2011.
10. Ferenc Gécseg and Magnus Steinby. Tree languages. In *Handbook of Formal Languages*, 1997.
11. S.C. Kleene. Representation of events in nerve nets and finite automata. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–40, 1956. Princeton.
12. Dietrich Kuske and Ingmar Meinecke. Construction of tree automata from regular expressions. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 45(3):347–370, 2011. URL: http://www.numdam.org/articles/10.1051/ita/2011107/, doi:10.1051/ita/2011107.
13. Neil Robertson and Paul D. Seymour. Graph minors. iv. tree-width and well-quasi-ordering. *J. Comb. Theory, Ser. B*, 48:227–254, 1990.
14. James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical systems theory*, 2:57–81, 2005.