K4-free Graphs as a Free Algebra

Enric Cosme-Llópez, Damien Pous

K_4-free Graphs as a Free Algebra

Enric Cosme-Llópez Damien Pous

Univ Lyon, CNRS, ENS de Lyon, UCB Lyon 1, LIP, France
{enric.cosme-llopez,damien.pous}@ens-lyon.fr

***Abstract*—Graphs of treewidth at most two are the ones excluding the clique with four vertices (K_4) as a minor, or equivalently, the graphs whose biconnected components are series-parallel.**

We turn those graphs into a finitely presented free algebra, answering positively a question by Courcelle and Engelfriet, in the case of treewidth two.

First we propose a syntax for denoting these graphs: in addition to parallel composition and series composition, it suffices to consider the neutral elements of those operations and a unary transpose operation. Then we give a finite equational presentation and we prove it complete: two terms from the syntax are congruent if and only if they denote the same graph.

I. Introduction

The notion of treewidth is a cornerstone in (algorithmic) graph theory [13]. It measures how close a graph is from a forest, and classes of graphs of bounded treewidth often enjoy good computational properties. For instance, graph homomorphism (and thus k-colouring) becomes polynomial-time [17], so does model-checking of Monadic Second Order (MSO) formulae, and satisfiability of MSO formulae becomes decidable, even linear. (See the monograph of Courcelle and Engelfriet about monadic second order logic on graphs [11].)

Here we focus on graphs of treewidth at most two. They coincide with the *partial 2-trees*, with the K_4*-free* graphs (those that exclude the clique with four vertices (K_4) as a *minor*), and with the graphs whose biconnected components are *series-parallel* [15], [5].

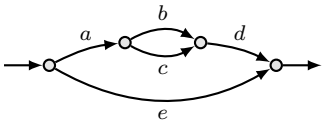
For reasons to become apparent later on, we consider the set Gph of directed graphs with edges labelled with letters $a, b, \ldots$ in some alphabet Σ, and with two distinguished vertices, called the *input* and the *output*. We represent such graphs as usual, using an unlabelled ingoing (resp. outgoing) arrow to denote the input (resp. output).

Such graphs can be composed:

- in *parallel* by putting them side by side, merging their inputs, and merging their outputs;
- in *series* by putting them one after the other and merging the output of the first one with the input of the second one

Every letter of the alphabet can be represented by the graph consisting of two vertices (the input and the output), and a single edge from the input to the output, labelled with the given letter.

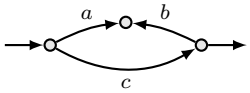
If we allow only those operations, we obtain precisely the series-parallel graphs, and it is easy to see that these form the free algebra over the signature $\langle \parallel, \cdot \rangle$ where $\parallel$ is an associative-commutative binary operation (parallel composition), and $\cdot$ is an associative binary operation (series composition). For instance, the terms $((a\cdot(b \parallel c))\cdot d) \parallel e$ and $e \parallel (a\cdot((c \parallel b)\cdot d))$ both denote the graph below, and they are equal up to associativity of $\cdot$ and commutativity of $\parallel$.



(Note that parallel composition is not idempotent: $(a\cdot b) \parallel (a\cdot b)$ and $a\cdot b$ denote distinct graphs.)

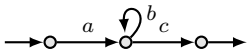
However, we cannot denote all graphs of treewidth at most two in such a way.

First the notion of treewidth does not depend on the orientation of the edges. For instance, the following graph has treewidth two, yet it is not the image of a term in the previous syntax.



To this end, we add a unary operation $\cdot^\circ$ to our signature, which we interpret in graphs as the exchange of input and output. Doing so, the terms $(a\cdot b^\circ) \parallel c$ and $(b\cdot a^\circ)^\circ \parallel c$ denote the above graph, and series-parallel graphs with converse become a free algebra when we ask that $\cdot^\circ$ is an involution that distributes over $\parallel$ and satisfies $(a\cdot b)^\circ = b^\circ\cdot a^\circ$.

Second, the treewidth of a graph does not depend on self-loops, so that the following graph actually has treewidth one (it is a tree once we remove the self-loop).



There it suffices to add a constant, 1, interpreted as the graph with a single vertex (both input and output), and no edge. Doing so, the above graph is denoted by $a\cdot(1 \parallel b)\cdot c$. While the constant 1 is clearly a neutral element for series composition $(\cdot)$, axiomatising its interactions with parallel composition is much harder, and is actually one of the key contributions of the present work. For instance, the equation $(1 \parallel a)\cdot(1 \parallel b) = 1 \parallel a \parallel b$ should belong to the theory as both sides denote the graph below:

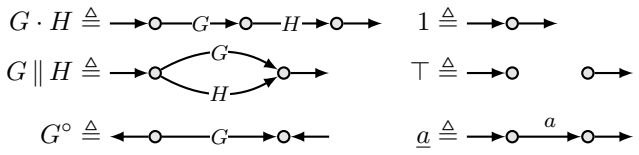
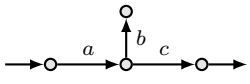


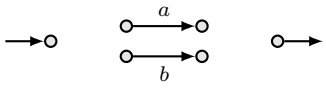
Figure 1. The 2p-algebra of graphs and the graph of a letter

Up-to this point, we have recovered the syntax of *allegories* [16], and the graphs associated to the terms are precisely the ones Freyd and Scedrov use to obtain that the theory of representable allegories is decidable, yet not finitely presentable [16, page 210].

But we still miss some graphs, like the one below:



One can also remark that we only obtain connected graphs using the above operations. Instead, treewidth allows disconnected graphs: a graph has a given treedwidth if and only if each of its connected components do. Surprisingly, it suffices to add a second constant, $\top$, interpreted as the disconnected graph with no edges and two distinct vertices (the input and the output). This allows us to obtain disconnected graphs, but also to get a term for the above connected graph, namely, $a{\cdot}((b{\cdot}\top) \parallel c)$. Again while it is clear that this constant is a neutral element for parallel composition $(\parallel)$, capturing its interactions with the other operations is non-trivial. For instance, $\top{\cdot}a{\cdot}\top{\cdot}b{\cdot}\top$ and $\top{\cdot}b{\cdot}\top{\cdot}a{\cdot}\top$ both denote the graph below, and should thus be equated.



To sum up, the set Gph of graphs forms an algebra for the signature $\langle \cdot_2, \parallel_2, \cdot^\circ_1, 1_0, \top_0 \rangle$. The various operations of this algebra are depicted in Figure 1.

Write Trm for the set of terms over the alphabet Σ and TW_2 for the set of graphs of treewidth at most two when an extra edge is added between input and output. (See Section IV for this additional condition.) One easily proves that the latter set actually forms a subalgebra of the algebra of graphs. Therefore, the function interpreting each term as a graph actually gives a function $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$.

We first prove that this function has a right-inverse: we define a function $\mathsf{t} : \mathsf{TW}_2 \to \mathsf{Trm}$ such that for all graph $G \in \mathsf{TW}_2$, $\mathsf{g}(\mathsf{t}(G))$ is isomorphic to G:

$$\mathsf{Trm} \underset{\mathsf{t}}{\overset{\mathsf{g}}{\rightleftarrows}} \mathsf{TW}_2 \qquad \mathsf{g}(\mathsf{t}(G)) \simeq G \tag{1}$$

By doing so, we get that the graphs of treewidth at most two are exactly the ones that can be expressed using the syntax.

Our key contribution then consists in giving a finite equational axiomatisation of graph isomorphism over this syntax.

$$u \parallel (v \parallel w) \equiv (u \parallel v) \parallel w \tag{A1}$$
$$u \parallel v \equiv v \parallel u \tag{A2}$$
$$u \parallel \top \equiv u \tag{A3}$$
$$u{\cdot}(v{\cdot}w) \equiv (u{\cdot}v){\cdot}w \tag{A4}$$
$$u{\cdot}1 \equiv u \tag{A5}$$
$$u^{\circ\circ} \equiv u \tag{A6}$$
$$(u \parallel v)^\circ \equiv u^\circ \parallel v^\circ \tag{A7}$$
$$(u{\cdot}v)^\circ \equiv v^\circ{\cdot}u^\circ \tag{A8}$$
$$1 \parallel 1 \equiv 1 \tag{A9}$$
$$1 \parallel u{\cdot}v \equiv 1 \parallel (u \parallel v^\circ){\cdot}\top \tag{A10}$$
$$u{\cdot}\top \equiv (1 \parallel u{\cdot}\top)\top \tag{A11}$$
$$(1 \parallel u){\cdot}v \equiv (1 \parallel u){\cdot}\top \parallel v \tag{A12}$$

Figure 2. Twelve axioms for 2p-algebras

This answers positively the question asked by Courcelle and Engelfriet in their book, for treewidth two [11, page 118].

Note that the choice of the syntax is important. Various finite syntaxes have already been proposed [9], [13], [11] to capture graphs of treewidth at most k, for a given k. However, some choices prevent finite presentations. For instance, while the *converse* operation we use could be eliminated by pushing it to the leaves, doing so would turn some of our axioms into infinite equational schemes. (See also Remark 28.)

As explained above, a few laws are rather natural like associativity of the two compositions, commutativity of $\parallel$, or the facts that 1 and $\top$ are neutral elements and that $\cdot^\circ$ is an involution. Those are the first eight laws in Figure 2. Surprisingly, the four subsequent axioms suffice to obtain a complete axiomatisation: for all terms u, v, u and v are provably equal using the axioms from Figure 2 if and only if $\mathsf{g}(u)$ and $\mathsf{g}(v)$ are isomorphic:

$$u \equiv v \qquad \Leftrightarrow \qquad \mathsf{g}(u) \simeq \mathsf{g}(v) \tag{2}$$

In other words, calling a *2p-algebra* an algebra satisfying the axioms from Figure 2, TW_2 is the free 2p-algebra.

All axioms but (A3) are independent. (See Appendix A; Axiom (A3) follows from (A9) and (A12) but we keep it for the sake of clarity.) . Correctness, i.e., the left-to-right implication from (2), is easy to establish. Indeed, it suffices to compute and compare the graphs of each equation, and to prove that valid equations are stable under graph substitution. The converse implication, completeness, is much harder. This is because there is no canonical way of extracting a term out of graph. In particular the function t we define to this end has to make choices based on the concrete representation of the input graph, so that isomorphic graphs do not always map to syntactically equal terms.

We proceed in the following way to obtain completeness. First we prove that the function $\mathtt{t}$ maps isomorphic graphs to provably equal terms:

$$G \simeq H \qquad \Rightarrow \qquad \mathtt{t}(G) \equiv \mathtt{t}(H) \tag{3}$$

Then we prove that this function is an homomorphism (up to the axioms), which allows us to deduce that for all term u, $\mathtt{t}(\mathtt{g}(u))$ is provably equal to u:

$$\mathtt{t}(\mathtt{g}(u)) \equiv u \tag{4}$$

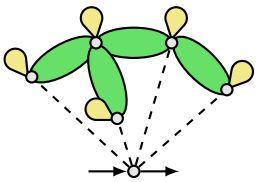
In a sense, by interpreting a term u into a graph and then reading it back, we obtain a term $\mathtt{t}(\mathtt{g}(u))$ which plays the role of a normal form even if it is not canonical (which would typically be the case in rewriting theory, or in normalisation by evaluation [4]).

Defining the function $\mathtt{t}$ satisfying (1) could be done rather easily by relying on the notion of tree decomposition. However, doing so makes it extremely difficult to obtain properties (3) and (4): the notion of tree decomposition, despite its inductive nature, does not provide enough structure. Instead, we use the fact that treewidth at most two graphs are K_4-free, and we exhibit stronger graph invariants that allow us to extract terms from graphs in a much more structured way.

For instance, when the graph is connected and when its input and output are distinct, one can compute its *checkpoints*: those vertices which all paths from the input to the output must visit. Those checkpoints are linearly ordered so that the graph necessarily has the following shape

If there is at least one checkpoint then the graph should be interpreted as a series composition. Otherwise, by the absence of K_4 as a minor, one can show that the graph necessarily is a non-trivial parallel composition.

The aforementioned step is already there in the standard result that the biconnected components of a K_4-free graph are series-parallel. More challenging is the case when the input and output coincide. In this case, we consider the checkpoints of all pairs of neighbours of the input, and we show that they form a tree which is a minor of the starting graph.



This tree is a key invariant of the isomorphism class of the graph and we show that one can extract a term for each choice of a node in this tree. This is where our function $\mathtt{t}$ has to rely on the concrete representation of the graph: although all choices of a node in the tree result in provably equal terms, they do not yield syntactically equal terms. A similar situation happens with components which are disconnected from the input and the output: we handle those recursively by taking any vertex as a new choice of input and output.

II. Related work

Except for the presence of $\top$, the algebra of graphs we work with has been proposed independently by Freyd and Scedrov [16, page 207], and by Andréka and Bredikhin [1]. They used it to characterise the equational theory of binary relations over the considered signature. Indeed for every set S, the set $\mathcal{P}(S \times S)$ of binary relations over S forms an algebra for the signature we consider in this paper: $\cdot$ is relational composition, $\|$ is set-theoretic intersection (thus it is written $\cap$ in [16], [1]), $\cdot^\circ$ is transposition (converse), 1 is the identity relation and $\top$ is the full relation. Writing $\mathrm{Rel} \vDash u \leq v$ for the containments that hold in all such algebras of relations, and $G \blacktriangleleft H$ if there exists a graph homomorphism from H to G, we have the following equivalence.

$$\mathrm{Rel} \vDash u \leq v \qquad \Leftrightarrow \qquad \mathtt{g}(u) \blacktriangleleft \mathtt{g}(v)$$

This characterisation immediately gives decidability: existence of a graph homomorphism is an NP-complete problem. Thanks to the present observation that graphs of terms have bounded treewidth, the complexity is actually polynomial [17].

Freyd and Scedrov also use this characterisation to prove that this theory is not finitely presentable [16]: every complete equational axiomatisation must contain axioms corresponding to homomorphisms equating arbitrarily many vertices at a time, and thus must be infinite. Andréka and Bredikhin go even further and show that it is not even a variety [1].

In this work we focus on isomorphism rather than on homomorphism, and this is why we do obtain a finite equational axiomatisation. Although all algebras of relations validate our axioms, these algebras cannot be free models. For instance, their parallel composition (intersection) is always idempotent.

Freyd and Scedrov remark that certain graphs cannot be the image of a term [16, page 207]; similarly, Andréka and Bredikhin use a weak form of the K_4 exclusion property [1, Lemma 7]. However, they cannot obtain a characterisation result since they do not consider $\top$, which is necessary to reach all graphs of treewidth at most two.

Our work is also really close to that of Dougherty and Gutiérrez [14], who proposed an axiomatisation of graph isomorphism for a slightly different syntax: instead of the constant $\top$, they use a unary operation $\mathrm{dom}(\cdot)$, called *domain*. This operation can be defined using $\top$: we have $\mathrm{dom}(u) = 1 \,\|\, (u \cdot \top)$; at the graphical level, it consists in relocating the ouput of a graph on its input. (Note that $\top$ cannot be defined in terms of $\mathrm{dom}(\cdot)$ and the other operations.)

Chosing this domain operation has the advantage of keeping connected graphs, and the disadvantage of being less general: disconnected graphs cannot be expressed. More importantly, the operation $\top$ being more primitive than $\mathrm{dom}(\cdot)$, we can obtain a shorter axiomatisation: while we share with [14] the nine natural axioms from Figure 2 that do not mention $\top$, the four remaining ones in this figure have to be replaced by nine axioms when using $\mathrm{dom}(\cdot)$: three about 1 and $\|$, and six about $\mathrm{dom}(\cdot)$.

To prove completeness, Dougherty and Gutiérrez compute normal forms for terms using rewriting techniques. Like in the present work, their normal forms are not canonical and some additional work is needed. In a second part of the paper, they characterise graphs of terms using a minor exclusion theorem which corresponds precisely to what we obtain in the connected case (see Remark 28).

There are however several typos or gaps in their paper which we were not able to fix.

Bauderon and Courcelle gave a syntax and a complete axiomatisation for arbitrary graphs [3]. While the overall statement is similar to ours, their syntax can hardly be related to the present one (it is infinitary, for instance), and the present results are not corollaries of their work. Although we need to present them differently for the sake of our completeness proof, the structural invariants we exhibit for treewidth at most two graphs are reminiscent of the general decomposition results of Tutte [23], which Courcelle later studied in the context of MSO [10].

III. 2p-algebra

We consider the signature $\langle \cdot_2, \|_2, \cdot^\circ_1, 1_0, \top_0 \rangle$ and we let u, v, w range over *terms* over a set Σ of variables.

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

We usually omit the $\cdot$ symbol and we assign priorities so that the term $(a \cdot (b^\circ)) \parallel c$ can be written just as $ab^\circ \parallel c$.

A *2p-algebra* is an algebra over this signature satisfying the axioms from Figure 2. We write $u \equiv v$ when two terms u and v are congruent modulo those axioms, or equivalently, when the equation holds in all 2p-algebras.

Following notations from Kleene algebra with tests (KAT) [18], we let α, β range over *tests*, those terms that are congruent to some term of the shape $1 \parallel u$. (Equivalently by axiom A9, u is a test if $u \equiv 1 \parallel u$.) The graphs of tests are precisely those whose input and output coincide.

We shall use the derived operation mentioned in Section II, *domain*, as well as its dual, *codomain*:

$$\mathrm{dom}(u) \triangleq 1 \parallel u\top \qquad \mathrm{cod}(u) \triangleq 1 \parallel \top u$$

Domain and codomain terms are tests by definition.

As is standard for involutive monoids, the first eight axioms from Figure 2 entail $1^\circ \equiv 1$, $\top^\circ \equiv \top$, and $1u \equiv u$. We will use such laws freely in the sequel. We recall the four remaining axioms below, using the above notations.

$$1 \parallel 1 \equiv 1 \tag{A9}$$
$$1 \parallel uv \equiv \mathrm{dom}(u \parallel v^\circ) \tag{A10}$$
$$u\top \equiv \mathrm{dom}(u)\top \tag{A11}$$
$$\alpha v \equiv \alpha\top \parallel v \tag{A12}$$

Thanks to converse being an involution, there is a notion of duality in 2p-algebras: one obtains a valid law when swapping the arguments of all products and exchanging domains with codomains in a valid law. (We have $\mathrm{cod}(u) \equiv \mathrm{dom}(u^\circ)$.)

Proposition 1. *The following equations hold in all 2p-algebras.*

$$\alpha^\circ \equiv \alpha \tag{5}$$
$$\alpha\beta \equiv \alpha \parallel \beta \tag{6}$$
$$\alpha(v \parallel w) \equiv \alpha v \parallel w \tag{7}$$
$$(v \parallel w)\alpha \equiv v\alpha \parallel w \tag{8}$$
$$\mathrm{dom}(uv \parallel w) \equiv \mathrm{dom}(u \parallel wv^\circ) \tag{9}$$
$$\top u^\circ \top \equiv \top u \top \tag{10}$$
$$u\top w \equiv u\top \parallel \top w \tag{11}$$
$$u\top v\top w \equiv u\top w \parallel \top v\top \tag{12}$$
$$(u \parallel \top v\top)w \equiv uw \parallel \top v\top \tag{13}$$

Proof. To prove (5) we can assume that $\alpha = 1 \parallel u$; by using axiom A10 twice, we get

$$\begin{aligned}\alpha^\circ \equiv 1 \parallel 1u^\circ &\equiv \mathrm{dom}(1 \parallel u^{\circ\circ}) \\ &\equiv \mathrm{dom}(u \parallel 1^\circ) \equiv 1 \parallel u1 \equiv \alpha\end{aligned}$$

Equation (6) follows from (7) by taking $v = 1$ and $w = \beta$. For (7), we use axiom A12 twice:

$$\alpha(v \parallel w) \equiv \alpha\top \parallel v \parallel w \equiv \alpha v \parallel w$$

Equation (8) is the dual of (7). For (9), we use axiom A10 twice:

$$\mathrm{dom}(uv \parallel w) \equiv (1 \parallel uvw^\circ) \equiv \mathrm{dom}(u \parallel wv^\circ)$$

We use (5) as well as axiom A11 and its dual for (10):

$$\top u^\circ\top \equiv \top\mathrm{dom}(u^\circ)\top \equiv \top\mathrm{cod}(u)\top \equiv \top u\top$$

Thanks to axiom A11 and its dual, it suffices to prove the laws (11) and (12) in the case where u, v and w are tests. The law $\alpha\top\beta \equiv \alpha\top \parallel \top\beta$ is the instance of axiom A12 where $v = \top\beta$. For the second one, we have

$$\begin{aligned}\alpha\top\beta\top\gamma &\equiv \alpha\top \parallel \top\beta\top\gamma && \text{(by A12)} \\ &\equiv \alpha\top \parallel \top\beta\top \parallel \top\gamma && \text{(by dual of A12)} \\ &\equiv \alpha\top\gamma \parallel \top\beta\top && \text{(by (11))}\end{aligned}$$

For the last law, we use axioms A12 and A11 twice:

$$\begin{aligned}(u \parallel \top v\top)w &\equiv (u \parallel \mathrm{dom}(\top v))w \\ &\equiv \mathrm{dom}(\top v)uw \\ &\equiv uw \parallel \mathrm{dom}(\top v)\top \\ &\equiv uw \parallel \top v\top\end{aligned}$$

□

IV. Graphs

As explained in the introduction, we consider labelled directed graphs with two designated vertices. We just call them graphs in the sequel. Note that we allow multiple edges between two vertices, as well as self-loops.

Definition 2. A *graph* is a tuple $G = \langle V, E, s, t, l, \iota, o \rangle$, where V is a finite set of *vertices*, E is a finite set of *edges*, $s, t : E \to V$ are maps indicating the *source* and *target* of each edge, $l : E \to \Sigma$ is map indicating the *label* of each edge, and

$\iota, o \in V$ are the designated vertices, respectively called *input* and *output*.

We write $G[x; y]$ for the graph G with input set to x and output set to y; we abbreviate $G[x; x]$ to $G[x]$.

Definition 3. A *(graph) homomorphism* from $G = \langle V, E, s, t, l, \iota, o\rangle$ to $G' = \langle V', E', s', t', l', \iota', o'\rangle$ is a pair $h = \langle f, g\rangle$ of functions $f : V \to V'$ and $g : E \to E'$ that respect the various components: $s' \circ g = f \circ s$, $t' \circ g = f \circ t$, $l' = g \circ l$, $\iota' = f(\iota)$, and $o' = f(o)$.

A *(graph) isomorphism* is an homomorphism whose two components are bijective functions. We write $G \simeq G'$ when there exists an isomorphism between graphs G and G'.

Proposition 4. *Graphs up to isomorphism form a 2p-algebra.*

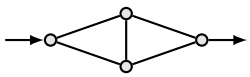
Proof. The operations of the signature on graphs have been described in the Introduction and in Figure 1. Providing an isomorphism for each law from Figure 2 is routine. □

Definition 5. Let $G = \langle V, E, s, t, l, \iota, o\rangle$ be a graph. A *tree decomposition* of G is a tree T where each node t is labelled with a set $V_t \subseteq V$ of vertices, such that:

(T1) for every vertex $x \in V$, the set of nodes t such that $x \in V_t$ is non-empty and connected in T (i.e., forms a sub-tree);
(T2) for every edge $e \in E$, there exists a node t such that $\{s(e), t(e)\} \subseteq V_t$;
(T3) there exists a node t such that $\{\iota, o\} \subseteq V_t$.

The *width* of a tree decomposition is the size of the largest set V_t minus one; the *treewidth* of a graph is the minimal width of a tree decomposition for this graph. We write TW_2 for the set of graphs of treewidth at most two.

The first two conditions in the definition of tree decomposition are standard; the third one is related to the presence of distinguished nodes: it requires them to lie together in some node of the tree. This condition ensures that the following graph is excluded from TW_2 whatever the orientation and labelling of its edges, even though it has treewidth two when forgetting about input and output.



(M_3)

Indeed, such a graph cannot be represented in the syntax we consider. (Something already observed by Freyd and Scedrov [16]—the addition of $\top$ to the syntax does not help.)

Lemma 6. *Let G, H be two graphs with an homomorphism h from G to H. Suppose we have a tree decomposition of G such that for all vertices x, y such that $h(x) = h(y)$, there is a node containing both x and y. Then this tree decomposition can be renamed by h into a tree decomposition of $h(G)$ of width smaller or equal.*

Proposition 7. *Graphs of treewidth at most two (TW_2) form a subalgebra of the algebra of graphs.*

Proof. Graphs 1 and $\top$ actually have treewidth 0 and 1, respectively. A tree decomposition for a graph G remains a tree decomposition for the graph G°.

For a series composition $G_1 \cdot G_2$, we have two graph homomorphisms h_i from G_i to $G_1 \cdot G_2$ such that $h_1(o_1) = h_2(\iota_2)$. Those homomorphisms are injective (on both vertices and edges), and $h_1(G_1) \parallel h_2(G_2) \subseteq P \triangleq \{h_1(\iota_1), h_1(o_1), h_2(o_2)\}$. Take tree decompositions of G_1 and G_2 and rename them using h_1 and h_2. By (T3), the former has a node t_1 containing $h_1(\iota_1)$ and $h_1(o_1)$, and the latter has a node t_2 containing $h_2(\iota_2)$ and $h_2(o_2)$. A tree decomposition of $G_1 \cdot G_2$ is obtained by joining t_1 and t_2 through a new node labelled P.

For a parallel composition $G_1 \parallel G_2$, we also have two graph homomorphisms h_i from G_i to $G_1 \parallel G_2$ such that $h_1(\iota_1) = h_2(\iota_2)$ and $h_1(o_1) = h_2(o_2)$. Those homomorphisms are not necessarily injective on vertices (e.g., when $\iota_1 = o_1$ and $\iota_2 \neq o_2$). Nevertheless, they identify at most the input and the output, and those are related in some node in any tree decomposition, by (T3). Thus Lemma 6 applies: we can take tree decompositions of G_1 and G_2 and rename them according to the two homomorphisms. A tree decomposition of $G_1 \parallel G_2$ is obtained by adding an edge between the nodes given by (T3) in each decomposition. □

The graphs we associate to each letter (Figure 1) also belong to this subalgebra, so that we obtain an homomorphism $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$ associating a graph of treewidth at most two to each syntactic term. When taking quotients under term congruence and graph isomorphism, this function becomes a 2p-algebra homomorphism $\mathsf{g}' : \mathsf{Trm}_{/\equiv} \to \mathsf{TW}_{2/\simeq}$. Our key result is that g' actually is an isomorphism of 2p-algebras (Corollary 41).

V. K_4-FREENESS

In this section we establish preliminary technical results about unlabelled undirected graphs with at most one edge between two vertices and without self-loops; we call those *simple graphs*. We use standard notation and terminology from graph theory [13]. In particular, we denote by xy a potential edge between two vertices x and y; an xy-path is a (possibly trivial) path whose ends are x and y; $G + xy$ is the simple graph obtained from G by adding the edge xy if x and y were not already adjacent; $G \backslash x$ is the simple graph obtained from G by removing the vertex x and the edges touching it.

Definition 8. A *minor* of a simple graph G is a simple graph obtained up to isomorphism by performing a sequence of the following operations on G: delete an edge or a vertex, contract an edge.

A cornerstone result of graph theory, Robertson and Seymour's graph minor theorem [22], states that (simple) graphs are well-quasi-ordered by the minor relation. As a consequence, the classes of graphs of bounded treewidth, which are closed under taking minors, can be characterised by finite sets of excluded minors. Two simple and standard instances

are the following ones: the graphs of treewidth at most one (the forests) are precisely those excluding the cycle with three vertices (C_3); those of treewidth at most two are those excluding the complete graph with four vertices (K_4) [15]. We eventually reprove the latter one here.


(C_3)

(K_4)

We fix a connected simple graph G in the remainder of this section.

Definition 9. The *checkpoints* between two vertices x, y are the vertices which any xy-path must visit.

$$\mathsf{CP}(x,y) \triangleq \{z \mid \text{every } xy\text{-path crosses } z\}$$

Note that for all vertices x, y, we have $\mathsf{CP}(x,x) = \{x\}$ and $\{x,y\} \subseteq \mathsf{CP}(x,y) = \mathsf{CP}(y,x)$. Two vertices x, y are *linked*, written $x \Diamond y$, when $x \neq y$ and $\mathsf{CP}(x,y) = \{x,y\}$, i.e., when there are no proper checkpoints between x and y. The *link graph* of G is the graph of linked vertices.

The graph G is a subgraph of its link graph: if xy is an edge in G then $x \Diamond y$. We also have the following properties.

Lemma 10. *Any cycle in the link graph is actually a clique.*

Lemma 11. *If xyz is a triangle in the link graph and if ι is a vertex not in G, then the graph $G + \iota x + \iota y + \iota z$ admits K_4 as a minor.*

Proof. Let π by an irredundant xy-path avoiding z. Let π' be a path not crossing π, from z to a neighbour z' of some elements from π.

- If z' has at least two neighbours in π, say x' and y' such that x' appears before y' in π. Then we can get K_4 in the enriched graph by contracting the xx' and yy' subpaths of π as well as the zz' subpath of π' into vertices, contracting the $x'y'$ subpath into an edge, and deleting irrelevant vertices.
- If z' has only one neighbour in π, say n, we can assume w.l.o.g. that $n \neq x$. Take an irredundant xz-path avoiding n. Since π and π' are disjoint, there must be a sub-path π'' of it whose ends m and z'' are respectively in π and π' but whose inner vertices (if any) avoid those sets. We have $m \neq n$ by assumption, so that K_4 can be obtained in a way similar to the previous case. □

Now fix a set U of vertices; we extend the notion of checkpoints as follows.

Definition 12. The *checkpoints* of U, $\mathsf{CP}(U)$, is the set of vertices which are checkpoints of some pair in U.

$$\mathsf{CP}(U) \triangleq \bigcup_{x,y \in U} \mathsf{CP}(x,y)$$

The *checkpoint graph* of U is the subgraph of the link graph induced by this set. We also denote this graph by $\mathsf{CP}(U)$.

Lemma 13. CP *is a closure operator on the set of vertices. In particular, for all checkpoints $x, y \in \mathsf{CP}(U)$, $\mathsf{CP}(x,y) \subseteq \mathsf{CP}(U)$.*

Lemma 14. *For every path in G between two checkpoints $x, y \in \mathsf{CP}(U)$, the sequence obtained by keeping only the elements in $\mathsf{CP}(U)$ is an xy-path in $\mathsf{CP}(U)$.*

Since G is assumed to be connected, so is $\mathsf{CP}(U)$. A key instance of a checkpoint graph is when U only contains two vertices, presumably the input and ouput of some graph: the checkpoint graph is a line in this case, and it allows us to decompose the considered graph into a sequence of series compositions.

Lemma 15. *If $U = \{x, y\}$ for some vertices x, y, then $\mathsf{CP}(U)$ is a line graph whose ends are x and y.*

Proof. For vertices z_1 and z_2 in $\mathsf{CP}(x,y)$, we say that $z_1 \preceq z_2$ if every xy-path crosses z_1 before z_2. The relation $\preceq$ is a total order on $\mathsf{CP}(x,y)$. □

The following two lemmas are helpful in Proposition 21 below, to prove that the checkpoint graph is a tree under certain circumstances.

Lemma 16. *If xy is an edge in $\mathsf{CP}(U)$, then there exists $x', y' \in U$ such that x and y belong to $\mathsf{CP}(x', y')$.*

Lemma 17. *If xyz is a triangle in $\mathsf{CP}(U)$, then there exists $x', y', z' \in U$ such that x and y (resp. x and z, y and z) belong to $\mathsf{CP}(x', y')$ (resp. $\mathsf{CP}(x', z')$, $\mathsf{CP}(y', z')$).*

As explained above we use the checkpoint graphs to decompose graphs. The following notions of intervals and bags are the basic blocks of those decompositions.

Definition 18. Let x, y be two vertices. The *strict interval* $[\![x; y]\![$ is the following set of vertices.

$$[\![x; y]\![\triangleq \{p \mid \text{there is an } xp\text{-path avoiding } y \text{ and a } py\text{-path avoiding } x\}$$

The *interval* $[\![x; y]\!]$ is obtained by adding x and y to that set. We abuse notation and write $[\![x; y]\!]$ for the subgraph of G induced by the set $[\![x; y]\!]$.

Note that while the intervals do not depend on the set U, they will mostly be used under the assumption that xy is an edge in a checkpoint graph.

Definition 19. The *bag* of a checkpoint $x \in \mathsf{CP}(U)$ is the set of vertices that need to cross x in order to reach the other checkpoints.

$$[\![x]\!]_U \triangleq \{p \mid \forall y \in \mathsf{CP}(U), \text{ any } py\text{-path crosses } x\}.$$

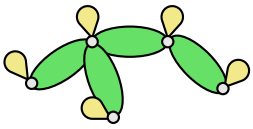
As before, we also write $[\![x]\!]_U$ for the induced subgraph of G.

Note that $[\![x]\!]_U$ depends on U and differs from $[\![x; x]\!]$ (which is always the singleton $\{x\}$).

Proposition 20. *If $\mathsf{CP}(U)$ is a tree, then the following set $\mathcal{V}$ is a partition of the vertices of G such that any edge of G appears in exactly one graph of the set $\mathcal{E}$.*

$$\mathcal{V} \triangleq \{[\![x]\!]_U \mid x \in \mathsf{CP}(U)\} \cup \{]\!]x; y[\![\mid xy \text{ edge in } \mathsf{CP}(U)\}$$
$$\mathcal{E} \triangleq \{[\![x]\!]_U \mid x \in \mathsf{CP}(U)\} \cup \{[\![x; y]\!] \mid xy \text{ edge in } \mathsf{CP}(U)\}$$

Graphically, this means G can be decomposed as in the picture below; only the vertices of $\mathsf{CP}(U)$ are depicted, the green blocks correspond to edges in $\mathsf{CP}(U)$, the yellow blocks correspond to the graphs $[\![x]\!]_U$. The leaves of $\mathsf{CP}(U)$ are elements of U; the converse does not always hold.



Proof. Let p be a vertex. Let N be the set of checkpoints $x \in \mathsf{CP}(U)$ for which there exists a px-path avoiding all checkpoints except x. N is non-empty since G is connected. One can show that N is a clique in $\mathsf{CP}(U)$ using Lemma 14, whence $|N| \leq 2$ since $\mathsf{CP}(U)$ is assumed to be a tree. If N is a singleton $\{x\}$, then $p \in [\![x]\!]_U$ and p cannot appear anywhere else in $\mathcal{V}$ by definition of N. If N is a pair $\{x, y\}$, then $p \in]\!]x; y[\![$; p cannot be in one of the $[\![z]\!]_U$ by definition of N, and if it were to belong to some other interval $]\!]x'; y'[\![$ with $x'y'$ an edge in $\mathsf{CP}(U)$ then could exhibit a cycle in $\mathsf{CP}(U)$.

Note that the above argument shows that when $p \in]\!]x; y[\![$ then p lies on an xy-path where the only checkpoints are x and y, and any path from p to a checkpoint must cross either x or y (†).

For the second part of the statement, let pq be an edge of G. If p and q belong to the same element of $\mathcal{V}$ then the edge belongs to the corresponding element in $\mathcal{E}$. Otherwise,

- either $p \in [\![x]\!]_U$ and $q \in [\![y]\!]_U$ for some distinct checkpoints x, y. Then necessarily $p = x$ and $q = y$ (by considering a py-path going through q and a qx-path going through p). The edge pq thus belongs to the graph $[\![x; y]\!]$ in $\mathcal{E}$.
- Or $p \in [\![x]\!]_U$ and $q \in]\!]y; z[\![$ for some edge yz in $\mathsf{CP}(U)$. Then there exists a qx-path going through p, so that x must be either y or z by (†). Say $x = y$. There exists a pz-path (throug q) enforcing $p = x$, so that the edge pq belongs to the graph $[\![y; z]\!]$.
- Or $p \in]\!]x; y[\![$ and $q \in]\!]x'; y'[\![$ for some distinct edges xy and $x'y'$ in $\mathsf{CP}(U)$. Assume w.l.o.g. that $x \neq x'$. By (†), y and x' cannot be equal. There is moreover an xx'-path and a yx'-path (both taking the edge pq) witnessing the fact that xx' and yx' are adjacent in $\mathsf{CP}(U)$. This contradicts $\mathsf{CP}(U)$ being a tree. □

As a consequence, when $\mathsf{CP}(U)$ is a tree, it is also a minor of G: contract all subgraphs of the form $[\![x]\!]_U$ to the vertex x and all subgraphs of the form $[\![x; y]\!]$ to the edge xy.

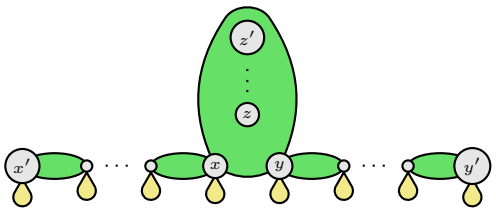
The following proposition is a key element in the developments to come. It makes it possible to extract a term out of a graph whose input and output coincide, by providing ways to chose an element where to relocate the output and resort to the easier case when input and output differ.

Proposition 21. *Assume $G = H\backslash\iota$, for some K_4-free simple graph H and some vertex ι. Further assume that U is the set of neighbours of ι in H and that this set is not empty.*

(i) $\mathsf{CP}(U)$ is a tree,
(ii) for every edge xy in $\mathsf{CP}(U)$, the graph $[\![x; y]\!] + xy$ is K_4-free,
(iii) for every vertex x in $\mathsf{CP}(U)$, the graph $H + \iota x$ is K_4-free.

(Note that G is still assumed to be connected.)

Proof. Assume by contradiction that $\mathsf{CP}(U)$ contains a cycle, and thus a triangle xyz by Lemma 10. Let x', y', z' be the elements of U given by Lemma 17. By Proposition 20 applied to the line graph $\mathsf{CP}(\{x', y'\})$, one can decompose G as follows:



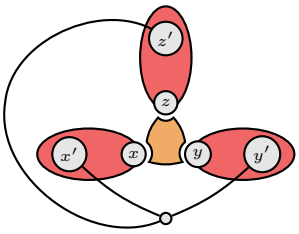
One obtains two symmetrical decompositions by using Proposition 20 on the line graphs $\mathsf{CP}(\{y', z'\})$ and $\mathsf{CP}(\{x', z'\})$. By combining those three decompositions, we get the following decomposition of G:

$$\mathcal{V} \triangleq \{X, Y, Z, T\} \qquad \mathcal{E} \triangleq \{X', Y', Z', T'\}$$

where

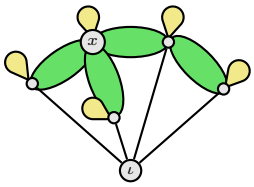
- $X \triangleq [\![x; x']\!] \cup [\![x]\!]_U \cup [\![x']\!]_U$ and X' is the corresponding induced subgraph;
- similarly for Y, Y', Z, and Z';
- $T \triangleq]\!]x; x'[\![\cup]\!]y; y'[\![\cup]\!]z; z'[\![$ and T' is the subgraph induced by $T \cup \{x, y, z\}$.

Adding back the vertex ι and the corresponding edges, the graph H looks as follows:



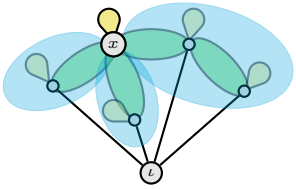
Other edges could be incident to ι, delete them and contract the blocks X, Y, and Z to obtain a minor H' of H. By Lemma 11, H' admits K_4 as a minor. This contradicts H being K_4-free.

Thus $\mathsf{CP}(U)$ is a tree (i), and by proposition 20, the graph G looks as follows:



Item (ii) follows: for any edge xy in $\mathsf{CP}(U)$, $[\![x;y]\!]+xy$ can be seen to be a minor of H; thus it cannot contain K_4 as a minor.

The last point (iii) is more delicate. If $x \in U$, then $H + \iota x = H$ is K_4-free by assumption. Otherwise, x must be an inner node of the tree $\mathsf{CP}(U)$, say with n neighbours. We can weaken the previous decomposition of the graph H to get $n+1$ components: one for $[\![x]\!]_U$ and one for each neighbour of x (the blue ones in the picture below). The latter are adjacent to ι in G while the former is not.



Consider a minoring process evolving $H + \iota x$ into K_4. The case where the edge ιx is deleted is trivial. If it is contracted, then K_4 is necessarily obtained using only one of the $n+1$ components; since $n \geq 2$, one can always contract one blue component to achieve the effect of contracting ιx. If ιx is kept to justify an edge in K_4, then name the vertices of K_4 ι, x, y, z and consider the connected components of H giving rise to y and z. Those connected components cannot contain ι nor x, and they must be adjacent to justify the yz-edge in K_4. Thus both of them are contained in one of the $n+1$ components. As previously one can contract one blue component to achieve the effect of keeping ιx. □

As a consequence of the above proposition, we have the following one, which makes it posible to decompose graphs with distinct input and output into a parallel composition when they cannot be a series composition.

Proposition 22. *Let ι, o be two distinct vertices such that $G + \iota o$ is K_4-free. We have that:*

(i) if ι and o are not adjacent in G and $\iota \Diamond o$, then the graph induced by $]\!]\iota; o[\!]$ has at least two connected components.

(ii) for every edge xy in $\mathsf{CP}(\{\iota, o\})$, the graph $[\![x;y]\!]+xy$ is K_4-free,

Proof. (i) Let $H = G + \iota o$ and $G' = H \backslash \iota$. The neighbours U of ι in H are the neighbours of ι in G, plus o. By Proposition 21, $\mathsf{CP}(U)$ (for G') is a tree. If o is a leaf in that tree then its unique neighbour must be a proper checkpoint between ι and o, contradicting the hypothesis $\mathsf{CP}(\iota, o) = \{\iota, o\}$. Otherwise its neighbours make it possible to partition $]\!]\iota; o[\!]$ into at least two connected components, thanks to Proposition 20.

(ii) By Lemma 15 and Proposition 20, $[\![x;y]\!]+xy$ can be obtained as a minor of $G + \iota o$; thus it cannot contain K_4 as a minor. □

VI. Extracting terms

Now we have enough preliminary material and we can look for a right inverse to the function $\mathsf{g} : \mathsf{Trm} \to \mathsf{TW}_2$. As explained in the Introduction, we use K_4-freeness to extract terms from graphs in a more structured way than using tree decompositions directly.

Definition 23. The *skeleton* of a graph G is the simple graph S obtained from G by forgetting input, output, labelling, edge directions, edge multiplicities, and self-loops. The *strong skeleton* of G is $S + \iota o$ if $\iota \neq o$, and S otherwise.

As an example, the strong skeleton of any instance of the graph (M_3) from Section IV is K_4. More generally, a graph belongs to TW_2 if and only if its strong skeleton has treewidth at most two in the standard sense (i.e., for simple graphs, without taking input and output into account).

Proposition 24. *The strong skeleton of every graph in TW_2 is K_4-free.*

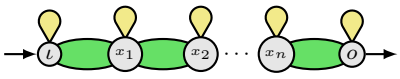
Proof. The operations used to obtain a minor do not increase the treewidth, and K_4 has treewidth three. □

Given a graph G and two vertices x, y, we write $G[\![x;y]\!]$ for the subgraph of G induced by the set $[\![x;y]\!]$ (computed in the skeleton of G), with input and output respectively set to x and y, and with self-loops on x and y removed. The strong skeleton of $G[\![x;y]\!]$ is $[\![x;y]\!]+xy$.

Similarly, given a graph G, a set U of vertices and vertex x, we write $G[\![x]\!]_U$ for the subgraph of G induced by the set $[\![x]\!]_U$ (computed in the skeleton of G), with both input and output set to x. Doing so, the skeleton and strong skeleton of $G[\![x]\!]_U$ are both $[\![x]\!]_U$. We shall omit the subscript when it is clear from the context.

Definition 25. The term $\mathsf{t}(G)$ of a graph G whose strong skeleton is K_4-free is defined by induction on the number of edges in G[1]. When G is connected there are two mains cases depending on whether the input and ouput coincide *(a)* or not *(b)*. We deal with the general case *(c)* by decomposing the graph into connected components.

a) Connected, distinct input and output: consider the line graph (Lemma 15) obtained by taking the checkpoint graph of $U = \{\iota, o\}$ in the skeleton of G. Write it as $x_0 \dots x_{n+1}$ with $\iota = x_0$ and $o = x_{n+1}$. According to Proposition 20, G looks as follows.



We set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G[\![x_0]\!])\cdot\mathsf{t}(G[\![x_0;x_1]\!])\cdot\mathsf{t}(G[\![x_1]\!])\cdot \\ \dots \cdot\mathsf{t}(G[\![x_n]\!])\cdot\mathsf{t}(G[\![x_n;x_{n+1}]\!])\cdot\mathsf{t}(G[\![x_{n+1}]\!])$$

The (strong) skeleton of each graph $G[\![x_i]\!]$ is just $[\![x_i]\!]$, which is necessarily K_4-free, as a subgraph of that of G. Proposition 22(ii) moreover ensures that so are the strong skeletons of all graphs $G[\![x_i;x_{i+1}]\!]$.

[1]More precisely, on the lexicographic product of the number of edges and the textual precedence of the three considered cases.

The above recursive calls occur on smaller graphs unless $n = 0$ and the graphs $G[\![\iota]\!]$ and $G[\![o]\!]$ are reduced to the trivial graph with one vertex and no edge (i.e., the graph 1). In such a situation,

- either ι and o are adjacent in G. Then let G' be the graph obtained by removing from G all edges between ι and o and let $u = a_1 \parallel \ldots \parallel a_i \parallel b_1^\circ \parallel \ldots b_j^\circ$ be a term corresponding to those edges. We set

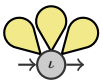
$$\mathsf{t}(G) \triangleq \mathsf{t}(G') \parallel u$$

- Or they are not, and Proposition 22(ii) applies so that we can decompose G into parallel components: $G = G_1 \parallel \ldots \parallel G_m$ with $m \geq 2$. Accordingly, we set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G_1) \parallel \ldots \parallel \mathsf{t}(G_m)$$

b) Connected, input equals output: if there are self-loops on ι, let $u = a_1 \parallel \ldots \parallel a_n$ be a term corresponding to those edges, let G' be the graph obtained by removing them, and recursively set $\mathsf{t}(G) \triangleq \mathsf{t}(G') \parallel u$.

Otherwise let H be the skeleton of G. Decompose $H \backslash \iota$ into connected components $H_1 \backslash \iota, \ldots, H_m \backslash \iota$ such that $H \simeq H_1 \cup \cdots \cup H_m$. The graph thus looks as follows.



If $m = 0$, then set $\mathsf{t}(G) = 1$. If $m > 1$, set

$$\mathsf{t}(G) \triangleq \bigcap_{i \leq m} \mathsf{t}(G_i) \ ,$$

where G_i is the subgraph of G induced by H_i. It remains to cover the case where $m = 1$. Let U be the set of neighbours of the input and compute the checkpoint graph $\mathsf{CP}(U)$ in $H \backslash \iota$. Pick an arbitrary node $x \in \mathsf{CP}(U)$. By Proposition 21(iii), the strong skeleton of $G[\iota; x]$ is K_4-free. Set

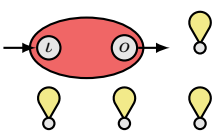
$$\mathsf{t}(G) \triangleq \mathrm{dom}(\mathsf{t}(G[\iota; x]))$$

(Remember that $\mathrm{dom}(\cdot)$ relocates the ouput to the input.)

c) General case: decompose the graph G into connected components $G_1, \ldots, G_n$. For all $i \leq n$, pick an arbitrary vertex x_i in the component G_i. There are two cases:

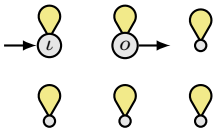
- either input and output belong to the same component, say G_j; then set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G_j) \parallel \bigcap_{i \neq j} \top \cdot \mathsf{t}(G_i[x_i]) \cdot \top$$



- or they belong to two distinct components, say ι in G_j and o in G_k, in which case we set

$$\mathsf{t}(G) \triangleq \mathsf{t}(G_j[\iota]) \cdot \top \cdot \mathsf{t}(G_k[o]) \parallel \bigcap_{i \neq j,k} \top \cdot \mathsf{t}(G_i[x_i]) \cdot \top$$



In both cases, it is easy to check that the recursive calls occur on graphs whose (strong) skeletons are subgraphs of the strong skeleton of G, and thus K_4-free.

The definition of the extraction function t ends here. Note that this function is defined on "concrete" graphs: we need to choose some vertices in cases *(b)* and *(c)*, and we can only do so by relying on the concrete identity of those vertices (e.g., to choose the smallest one, assuming they are numbers). We shall see in the following section that all those potential choices nevertheless always lead to congruent terms (Theorem 34).

We obtain the following theorem by construction.

Theorem 26. *For every graph $G \in \mathsf{TW}_2$, $\mathsf{g}(\mathsf{t}(G)) \simeq G$.*

Corollary 27. *The following are equivalent for all graph G:*
(i) G has treewidth at most two;
(ii) the strong skeleton of G is K_4-free;
(iii) G is (isomorphic to) the graph of a term.

Remark 28. When G is connected, $\mathsf{t}(G)$ does not contain occurrences of $\top$ other than those that are implicit in our uses of the domain operation (case *(b)*). Thus we obtain an alternative proof of Dougherty and Gutiérrez' characterisation [14, Section 4, Theorem 31] (their minor exclusion property is easily proved equivalent to ours—they do not mention treewidth).

Also note that we can easily avoid using 1 (but not $\mathrm{dom}(\cdot)$) when the graph does not contain self-loops and is not reduced to the trivial graph 1. When the graph does not contain self-loops and has distinct input and output, the construction can be modifed to produce terms without both 1 and $\mathrm{dom}(\cdot)$; the resulting construction becomes however less local, and we do not know how to axiomatise the 1-free reduct of 2p-algebra.

VII. Completeness of the axioms

Our goal is now to prove that the axioms of 2p-algebras are complete w.r.t. graphs: they suffice to equate any terms denoting the same graph up to isomorphism.

We first prove that t maps isomorphic graphs to congruent terms (Theorem 34 below). We need for that the following lemmas and propositions.

Lemma 29. *Let $G \in \mathsf{TW}_2$. If $\iota = o$, then $\mathsf{t}(G)$ is a test.*

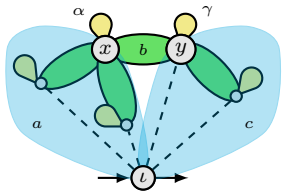
Proof. By a simple case analysis. □

Lemma 30. *For every graph $G \in \mathsf{TW}_2$, $\mathsf{t}(G^\circ) \equiv \mathsf{t}(G)^\circ$.*

Proof. Follows by a simple induction, using Lemma 29 and Equation (5). □

Proposition 31. *Let $G \in \mathsf{TW}_2$ be a graph with $\iota = o$, without self-loops on ι. Let S be its skeleton, and assume that $S\backslash\iota$ is connected. Let U be the neighbours of ι in G and consider the checkpoint graph of U in the skeleton of $S\backslash\iota$. For all checkpoints x, y, we have* $\mathrm{dom}(\mathsf{t}(G[\iota, x])) \equiv \mathrm{dom}(\mathsf{t}(G[\iota, y]))$.

Proof. The checkpoint graph is connected, so that it suffices to prove the statement when xy is an edge in the checkpoint graph. We depict the generic situation below.



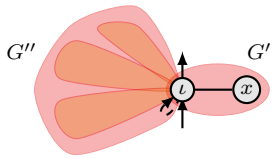
In such a case, we have $\mathsf{t}(G[\iota; y]) = \mathrm{dom}((a\alpha b \parallel c)\gamma)$ for some terms a, b, c and tests α, γ (up to associativity of $\cdot$ and associativity and commutativity of $\parallel$), and $\mathsf{t}(G[\iota; x]) \equiv \mathrm{dom}((a \parallel c\gamma b')\alpha)$ for some term b' such that $b' \equiv b^\circ$ by Lemma 30. We conclude as follows.

$$\begin{aligned}
\mathrm{dom}((a\alpha b \parallel c)\gamma) &\equiv \mathrm{dom}(a\alpha b \parallel c\gamma) && \text{(by (8))}\\
&\equiv \mathrm{dom}(a\alpha \parallel c\gamma b^\circ) && \text{(by (9))}\\
&\equiv \mathrm{dom}((a \parallel c\gamma b^\circ)\alpha) && \text{(by (8))}
\end{aligned}$$

□

Lemma 32. *Let $G \in \mathsf{TW}_2$ be a connected graph with $\iota = o$. For all neighbour x of ι, we have* $\mathsf{t}(G)\top \equiv \mathsf{t}(G[\iota, x])\top$.

Proof. The graph might have self-loops on ι and might be disconnected when removing ι. Let G' be the maximal subgraph without self-loops on ι that contains x and remains connected when removing ι. Let G'' be the subgraph of G induced by $\{\iota\} \cup (G\backslash G')$. The situation is depicted below:



Let $\alpha = \mathsf{t}(G'')$ and $u = \mathsf{t}(G'[\iota, x])$. By definition of the extraction function, we have $\mathsf{t}(G) \equiv \alpha \parallel \mathsf{t}(G')$ and $\mathsf{t}(G[\iota, x]) \equiv \alpha u$. By Proposition 31, we moreover have $\mathsf{t}(G') \equiv \mathrm{dom}(u)$. We conclude:

$$\begin{aligned}
\mathsf{t}(G)\top &\equiv (\alpha \parallel \mathrm{dom}(u))\top\\
&\equiv \alpha\mathrm{dom}(u)\top && \text{(by (6))}\\
&\equiv \alpha u\top && \text{(by axiom A11)}\\
&\equiv \mathsf{t}(G[\iota, x])\top
\end{aligned}$$

□

Proposition 33. *Let $G \in \mathsf{TW}_2$ be a connected graph. For all vertices x, y, we have* $\top\mathsf{t}(G[x])\top \equiv \top\mathsf{t}(G[y])\top$.

Proof. Since G is connected, it suffices to prove the property for every edge xy in G. For each of these, we have

$$\begin{aligned}
\top\mathsf{t}(G[x])\top &\equiv \top\mathsf{t}(G[x, y])\top && \text{(by Lemma 32)}\\
&\equiv \top\mathsf{t}(G[y, x])^\circ\top && \text{(by Lemma 30)}\\
&\equiv \top\mathsf{t}(G[y, x])\top && \text{(by (10))}\\
&\equiv \top\mathsf{t}(G[y])\top && \text{(by Lemma 32)}
\end{aligned}$$

□

Theorem 34. *Let $G, H \in \mathsf{TW}_2$ be two graphs. If $G \simeq H$ then $\mathsf{t}(G) \equiv \mathsf{t}(H)$.*

In other words, the extraction function t yields a function $\mathsf{t}' : \mathsf{TW}_{2/\simeq} \to \mathsf{Trm}_{/\equiv}$ between 2p-algebras.

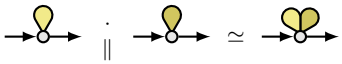
Proof. By induction following the computation of $\mathsf{t}(G)$ and $\mathsf{t}(H)$. The interesting cases are the following ones.

- Case *(b)* when $m = 1$. Modulo the axioms, the extracted terms do not depend on the chosen checkpoints, thanks to Proposition 31.
- Case *(c)*. The only difficulties occur in the connected components that contain neither the input nor the output: we arbitrarily pick a vertex which becomes both input and output. Proposition 33 shows that such a choice is inocuous modulo the axioms. □

We finally prove that t' is an homomorphism, and, in fact, an isomorphism.

Lemma 35. *Let $G, H \in \mathsf{TW}_2$ be two graphs whose input and output coincide. We have*

$$\mathsf{t}(G{\cdot}H) \equiv \mathsf{t}(G \parallel H) \equiv \mathsf{t}(G) \parallel \mathsf{t}(H) \equiv \mathsf{t}(G){\cdot}\mathsf{t}(H)$$

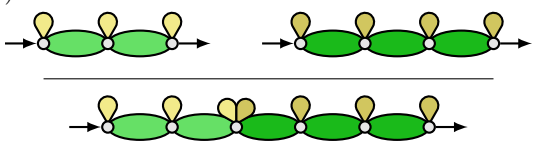


Proof. The first equation follows from Theorem 34: $G{\cdot}H$ and $G \parallel H$ are isomorphic. The second one is obtained using associativity and commutativity of $\parallel$, and axiom A9. The third one is an instance of (6) by Lemma 29. □

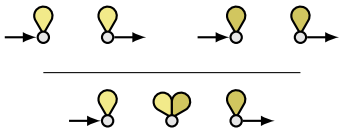
Lemma 36. *Let $G, H \in \mathsf{TW}_2$, we have* $\mathsf{t}(G{\cdot}H) \equiv \mathsf{t}(G){\cdot}\mathsf{t}(H)$.

Proof. If G (resp. H) has components disconnected from both input and output, we use (13) (resp. its dual) to extrude the corresponding terms (of shape $\top b\top$) at toplevel. Then we reason by cases.

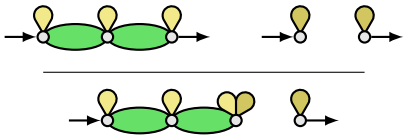
- G and H are connected. We use associativity of $\cdot$ and Lemma 35 on the last test in $\mathsf{t}(G)$ and the first test in $\mathsf{t}(H)$.



- Both G and H have exactly two connected components and their input and output are not connected. We use Lemma 35 as previously, and (12).



- H is in the latter situation, and G is connected. If the input and output of G coincide, then we just use Lemma 35. Otherwise, we proceed by induction on the size of G.

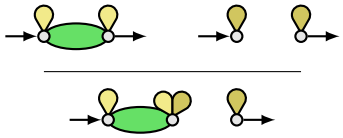


  - if there is a proper checkpoint between ι and o (in G), then decompose the graph into smaller graphs using this checkpoint: $G \simeq G_1 \cdot G_2$. We have

$$\begin{aligned}
\mathsf{t}(G \cdot H) &\equiv \mathsf{t}(G_1 \cdot G_2 \cdot H) && \text{(by Theorem 34)}\\
&\equiv \mathsf{t}(G_1)\mathsf{t}(G_2 \cdot H) && \text{(by induction on } G_1)\\
&\equiv \mathsf{t}(G_1)\mathsf{t}(G_2)\mathsf{t}(H) && \text{(by induction on } G_2)\\
&\equiv \mathsf{t}(G)\mathsf{t}(H) && \text{(by the connected case)}
\end{aligned}$$

  - if there is no checkpoint between ι and o, then let $\alpha u \beta = \mathsf{t}(G)$ and $\gamma \top \delta = \mathsf{t}(H)$. By Proposition 31, $\mathsf{t}(G \cdot H) \equiv (\alpha \parallel \mathrm{dom}(u(\beta \parallel \gamma)))\top\delta$. (Indeed, o belongs to the appropriate checkpoint graph: either ι and o are adjacent, or this is a consequence of Proposition 22 (i).) We have

$$\begin{aligned}
\mathsf{t}(G) \cdot \mathsf{t}(H) &= \alpha u \beta \gamma \top \delta\\
&\equiv \alpha \mathrm{dom}(u\beta\gamma)\top\delta && \text{(by axiom A11)}\\
&\equiv (\alpha \parallel \mathrm{dom}(u(\beta \parallel \gamma)))\top\delta && \text{(by (6))}\\
&\equiv \mathsf{t}(G \cdot H)
\end{aligned}$$



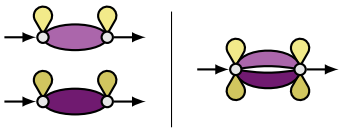
- The last case is symmetric and can be handled using Lemma 30 to get back to the previous case. □

Lemma 37. *For all graphs $G, H \in \mathsf{TW}_2$, we have $\mathsf{t}(G \parallel H) \equiv \mathsf{t}(G) \parallel \mathsf{t}(H)$.*

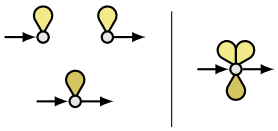
Proof. If G or H have components disconnected from both input and output, we use associativity and commutativity of $\parallel$ to reorganise them. Then we reason by cases.

- If both G and H have distinct input and output, then let $\alpha u \beta = \mathsf{t}(G)$ and $\alpha' v \beta' = \mathsf{t}(H)$. (With u and/or v possibly equal to $\top$, if G and/or H are disconnected.) We have

$$\begin{aligned}
\mathsf{t}(G) \parallel \mathsf{t}(H) &= \alpha u \beta \parallel \alpha u \beta\\
&\equiv \alpha\alpha'(u \parallel v)\beta\beta' && \text{(by (7,8) twice)}\\
&\equiv (\alpha \parallel \alpha')(u \parallel v)(\beta \parallel \beta') && \text{(by (6) twice)}\\
&\equiv \mathsf{t}(G \parallel H)\ . && \text{(by definition)}
\end{aligned}$$



- The case where one graph is disconnected and the other has equal input and output is routine, using the same laws as above.



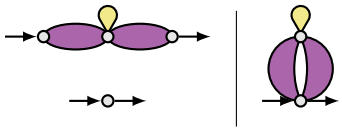
- The only remaining case is when one graph is connected, say G, and the other (H) has equal input and output. We proceed by induction on the number of vertices of G. It suffices to prove the statement when $H = 1$: indeed, we have

$$\begin{aligned}
\mathsf{t}(G \parallel H) &\equiv \mathsf{t}(G \parallel 1 \parallel H) && \text{(by Theorem 34)}\\
&\equiv \mathsf{t}(G \parallel 1) \parallel \mathsf{t}(H) && \text{(by Lemma 35)}\\
&\equiv \mathsf{t}(G) \parallel 1 \parallel \mathsf{t}(H) && \text{(by the proof below)}\\
&\equiv \mathsf{t}(G) \parallel \mathsf{t}(H) && \text{(by Lemma 29)}
\end{aligned}$$

  We can further assume that $\iota \neq o$ in G (otherwise Lemma 35 applies), and that there are no self-loops on ι and o (otherwise decompose G into $G_\iota \cdot G' \cdot G_o$ with G' satisfying the above property; we have $G \parallel 1 \simeq G' \parallel 1 \parallel G_\iota \parallel G_o$, use the fact that $\alpha u \beta \parallel 1 \equiv (u \parallel 1) \parallel \alpha \parallel \beta$).

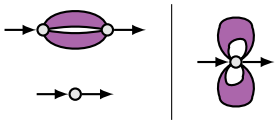
  - If there is a proper checkpoint between ι and o, write $\mathsf{t}(G) = u\beta v$ using this checkpoint. We have

$$\begin{aligned}
\mathsf{t}(G \parallel 1) &\equiv \mathrm{dom}((u \parallel v^\circ)\beta) && \text{(by Proposition 31)}\\
&\equiv \mathrm{dom}(u\beta \parallel v^\circ) && \text{(by (8))}\\
&\equiv 1 \parallel u\beta v && \text{(by axiom A10)}\\
&\equiv \mathsf{t}(G) \parallel 1
\end{aligned}$$



  - Otherwise, G is a (non-trivial) parallel composition $G_1 \parallel G_2$. We have

$$\begin{aligned}
\mathsf{t}(G \parallel 1) &\equiv \mathsf{t}(G_1 \parallel G_2 \parallel 1)\\
&\equiv \mathsf{t}(G_1) \parallel \mathsf{t}(G_2 \parallel 1) && \text{(by induction on } G_1)\\
&\equiv \mathsf{t}(G_1) \parallel \mathsf{t}(G_2) \parallel 1 && \text{(by induction on } G_2)\\
&\equiv \mathsf{t}(G) \parallel 1 && \text{(by the first case)}
\end{aligned}$$



□

Proposition 38. *The function $\mathsf{t}' : \mathsf{TW}_{2/\simeq} \to \mathsf{Trm}_{/\equiv}$ is an homomorphism of 2p-algebras.*

Proof. The cases for converse, series composition, and parallel composition are given by Lemmas 30, 36, and 37; those for constants 1 and $\top$ are straightforward. □

Theorem 39. *For every term u, we have $\mathsf{t}(\mathsf{g}(u)) \equiv u$.*

Proof. By structural induction on u, using Proposition 38 and the observation that $\mathsf{t}(\mathsf{g}(a)) \equiv a$ for every letter a. □

Corollary 40. *For all terms u and v, we have $u \equiv v$ if and only if $\mathsf{g}(u) \simeq \mathsf{g}(v)$.*

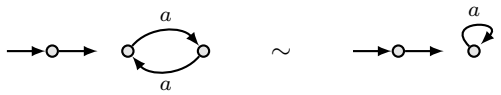
Proof. The direct implication amounts to Proposition 4. Conversely, if $\mathsf{g}(u) \simeq \mathsf{g}(v)$ then $\mathsf{t}(\mathsf{g}(u)) \equiv \mathsf{t}(\mathsf{g}(v))$ by Theorem 34, and thus $u \equiv v$ by Theorem 39. □

Corollary 41. *Graphs of treewidth at most two form the free 2p-algebra, as witnessed by the following diagram.*

$$\mathsf{Trm}_{/\equiv} \underset{\mathsf{t}'}{\overset{\mathsf{g}'}{\rightleftarrows}} \mathsf{TW}_{2/\simeq} \tag{14}$$

VIII. Future work

What is the free idempotent 2p-algebra (where parallel composition is idempotent)? One could be tempted to switch to simple directed graphs, where there is at most one edge with a given label from one vertex to another. This is however not an option: the graphs of $ab \parallel ab$ and ab are not isomorphic. One could also consider equivalences on graphs that are weaker than isomorphism. The notion of (two-way) bisimilarity [21], [20] that come to mind do not work either: such an equivalence relation on graphs certainly validates idempotency of parallel composition, but it also introduces new laws, e.g., $\top(1 \parallel aa)\top = \top(1 \parallel a)\top$, which are not even true in algebras of binary relations.



Courcelle used the algebraic theory he defined with Bauderon for arbitrary graphs [3] to propose a notion of graph recognisability [7], based on the generic framework by Mezei and Wright [19]. He proved that sets of graphs definable in MSO are recognisable. The converse does not hold in general. He later proved it for graphs of treewidth at most two [8] with a *counting* variant of MSO, conjecturing that it would be so for classes of graphs of bounded treewidth. This conjecture was proved only last year, by Bojańczyk and Pilipczuk [6].

The present work makes it possible to propose an alternative notion of recognisability for treewidth at most two, *2p-recognisability*: recognisability by a finite 2p-algebra. We conjecture that this notion coincides with recognisability. That recognisability entails 2p-recognisability is easy. The converse is harder; it amounts to proving that any finite congruence with respect to treewidth at most two graph substitutions can be refined into a finite congruence with respect to arbitrary graph substitutions. We see two ways of attacking this implication:

1) prove that 2p-recognisability entails MSO-definability, which could possibly be done along the lines of [8], by showing that our term extraction procedure is MSO-definable.
2) or use a slight generalisation of the result by Courcelle and Lagergren [12], relating recognisability to *k-recognisability* for graphs of treewidth at most k. Indeed, $2p$-recognisability is really close to 2-recognisability. Unfortunately, Courcelle and Lagergren's result is established only for unlabelled, undirected graphs, without sources, while we need labelled directed graphs with two sources.

One can easily extend our syntax to cover graphs of treewidth at most k, with k sources, for a given k (see, e.g., [9], [2]). However, we do not know how to generate finite axiomatisations in a systematic way, for every such k. Moreover, our proof strategy heavily depends on the fact that when $k = 2$, K_4 is the only excluded minor. We would need another strategy to deal with the general case since the excluded minors are not known for $k \geq 4$.

Acknowledgments

The authors are supported by the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157).

This work was also supported by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

References

[1] H. Andréka and D. A. Bredikhin. The equational theory of union-free algebras of relations. *Algebra Universalis*, 33(4):516–532, 1995.

[2] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993.

[3] M. Bauderon and B. Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20(2-3):83–127, 1987.

[4] U. Berger and H. Schwichtenberg. An inverse of the evaluation functional for typed lambda-calculus. In *Proc. LICS*, pages 203–211. IEEE, 1991.

[5] H. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.

[6] M. Bojańczyk and M. Pilipczuk. Definability equals recognizability for graphs of bounded treewidth. In *Proc. LICS*, pages 407–416. ACM, 2016.

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

[8] B. Courcelle. The monadic second-order logic of graphs V: on closing the gap between definability and recognizability. *Theoretical Computer Science*, 80(2):153–202, 1991.

[9] B. Courcelle. Graph grammars, monadic second-order logic and the theory of graph minors. In *Proc. Graph Structure Theory*, volume 147 of *Contemporary Mathematics*, pages 565–590. American Mathematical Society, 1993. Proceedings of a Joint Summer Research Conference on Graph Minors held June 22 to July 5, 1991, at the University of Washington, Seattle.

[10] B. Courcelle. The monadic second-order logic of graphs XI: Hierarchical decompositions of connected graphs. *Theoretical Computer Science*, 224(1):35–58, 1999.

[11] B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[12] B. Courcelle and J. Lagergren. Equivalent definitions of recognizability for sets of graphs of bounded tree-width. *Mathematical Structures in Computer Science*, 6(2):141–165, 1996.

[13] R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 2005.

[14] D. J. Dougherty and C. Gutiérrez. Normal forms for binary relations. *Theoretical Computer Science*, 360(1-3):228–246, 2006.

[15] R. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and Applications*, 10(2):303–318, 1965.

[16] P. Freyd and A. Scedrov. *Categories, Allegories*. North Holland. Elsevier, 1990.

[17] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1:1–1:24, 2007.

[18] D. Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.

[19] J. Mezei and J. Wright. Algebraic automata and context-free sets. *Information and Control*, 11(1–2):3 – 29, 1967.

[20] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[21] D. Park. Concurrency and automata on infinite sequences. In *Proc. Theoretical Computer Science*, pages 167–183, 1981.

[22] N. Robertson and P. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325 – 357, 2004.

[23] W. Tutte. *Graph Theory*. Addison-Wesley, Reading, MA, 1984.

Appendix A
Independence of the 2p-algebra axioms

The following finite and commutative algebras, where we set $x^\circ = x$, prove the independence of the last four axioms of 2p-algebras (Figure 2). In each case, we automatically checked that they form a model of the remaining axioms using the Coq proof assistant. See the web appendix of this work available at http://perso.ens-lyon.fr/damien.pous/tw2/. Except for (A3), the other axioms are also independent; their finite counter-models are provided on the aforementioned page.

Counter-model for axiom A9

$\cdot$	1	$\top$
1	1	$\top$
$\top$		1

$\parallel$	1	$\top$
1	$\top$	1
$\top$		$\top$

Counter-model for axiom A10

$\cdot$	1	2	3	$\top$
1	1	2	3	$\top$
2		2	2	2
3			$\top$	2
$\top$				2

$\parallel$	1	2	3	$\top$
1	1	2	1	1
2		2	2	2
3			3	3
$\top$				$\top$

(Take $a = b = 3$.)

Counter-model for axiom A11

$\cdot$	1	$\top$
1	1	$\top$
$\top$		1

$\parallel$	1	$\top$
1	1	1
$\top$		$\top$

(Take $a = \top$.)

Counter-model for axiom A12

$\cdot$	1	2	$\top$
1	1	2	$\top$
2		$\top$	2
$\top$			$\top$

$\parallel$	1	2	$\top$
1	1	2	1
2		1	2
$\top$			$\top$

(Take $a = b = 2$.)

APPENDIX B
OMMITTED PROOFS

Proof of Lemma 10. Let $x, y \in \mathsf{CP}(U)$ be two distinct vertices on the cycle, and let π_1, π_2 be the two corresponding xy-paths in $\mathsf{CP}(U)$. We have to show $x \Diamond y$. Let $p \neq x, y$ be a vertex in G. Chose $\pi \in \{\pi_1, \pi_2\}$ such that π does not cross p. Each edge zt in π can be replaced by a zt-path in G not crossing p. Doing so, we obtain an xy-path in G excluding p. Thus p cannot be a checkpoint. □

Proof of Lemma 13. Since $x \in \mathsf{CP}(x, x)$, the operator CP is extensive. It is increasing by definition. For idempotency, take vertices $z_1 \in \mathsf{CP}(x_1, y_1)$ and $z_2 \in \mathsf{CP}(x_2, y_2)$ for vertices x_1, y_1, x_2, y_2 in U. We need to prove that every vertex $t \in \mathsf{CP}(z_1, z_2)$ is a check point between elements in U. Let π_1 and π_2 be, respectively, two irredundant paths of the form x_1y_1 and x_2y_2. We distinguish four cases.

- if t neither appears in π_1 nor in π_2, then $t \in \mathsf{CP}(x_1, x_2)$. Assume this is not the case, then we can find an x_1x_2-path π_x not crossing t. The concatenation of the z_1x_1-subpath of π_1, the path π_x, and the x_2z_2-subpath of π_2 give us a z_1z_2-path avoiding t, which contradicts t being a check point in $\mathsf{CP}(z_1, z_2)$.
- If t appears in one of them, say π_1, but not in the other (π_2), then either t appears before z_1, and we prove $t \in \mathsf{CP}(y_1, y_2)$, or t appears after z_1, and we prove $t \in \mathsf{CP}(x_1, x_2)$. The proofs are done as in the first case, by exhibiting a z_1z_2-path avoiding t.
- If where t belongs to both π_1 and π_2, we distinguish four subcases according to whether t appears before or after z_1 in π_1 and before or after z_2 in π_2. □

Proof of Lemma 14. This follows from Lemma 13: we cannot miss intermediate checkpoints when we select only the elements in $\mathsf{CP}(U)$. □

Proof of Lemma 16. There exists elements x_1, x_2, y_1, y_2 in U for which $x \in \mathsf{CP}(x_1, x_2)$ and $y \in \mathsf{CP}(y_1, y_2)$. We can assume without loss of generality that $y \notin \mathsf{CP}(x_1, x_2)$ and $x \notin \mathsf{CP}(y_1, y_2)$, otherwise the statement will follow immediately. Let π_x and π_y be, respectively, two irredundant paths of the form x_1x_2 and y_1y_2 avoiding, respectively, y and x. Assume $x \notin \mathsf{CP}(x_1, y_1)$ and $x \notin \mathsf{CP}(x_2, y_2)$, then we can find paths π_1 and π_2 of the form x_1y_1 and y_2x_2 avoiding x, but then we find a contradiction, since the concatenation of the paths π_1, π_y, and π_2 give us a x_1x_2-path avoiding x. Consequently, $x \in \mathsf{CP}(x_1, y_1)$ or $x \in \mathsf{CP}(x_2, y_2)$. We have, by a similar argument, the same statement on y.

Assume, without loss of generality, that $x \in \mathsf{CP}(x_1, y_1)$ but $y \notin \mathsf{CP}(x_1, y_1)$, then necessarily, $y \in \mathsf{CP}(x_2, y_2)$. Assume that $x \notin \mathsf{CP}(x_2, y_2)$ and consider a y_2x_2-path γ_x avoiding x. We claim that $x, y \in \mathsf{CP}(x_1, y_2)$. Assume towards a contradiction the existence of a x_1y_2-path δ_x avoiding x. Then the concatenation of the paths δ_x and γ_x give us a x_1x_2-path avoiding x, and, thus, a contradiction. Similarly, if we assume the existence of a y_2x_1-path δ_y avoiding y. Then the concatenation of the paths δ_y and π_1 give us y_2x_2-path avoiding y, and, thus, a contradiction. Consequently, x and y must belong to $\mathsf{CP}(x_1, y_2)$. □

Proof of Lemma 17. Since we have an edge xy in $\mathsf{CP}(U)$, Lemma 16 give us elements x' and y' in U for which x and y are vertices in $\mathsf{CP}(x', y')$. We claim that z cannot be an element in $\mathsf{CP}(x', y')$, otherwise, we would have three checkpoints in sequence, and, since xyz is a triangle, a path from the first to the third check point avoiding the vertex in the middle. This configuration contradicts the fact that the vertex in the middle is a check point in $\mathsf{CP}(x', y')$.

Fix an irredundant $x'y'$-path π avoiding z. Assume, without loss of generality, that x appears before y in this path. There exists elements z_1 and z_2 in U for which $z \in \mathsf{CP}(z_1, z_2)$. Assume that z is neither an element in $\mathsf{CP}(x', z_1)$ nor in $\mathsf{CP}(y', z_2)$, then we can find π_1 and π_2, respectively z_1x' and $y'z_2$ paths, avoiding z. The concatenation of the paths π_1, π, and π_2 give us a z_1z_2-path avoiding z, and, thus, a contradiction. Consequently, either $z \in \mathsf{CP}(x', z_1)$ or $z \in \mathsf{CP}(y', z_2)$.

Consider the first configuration. We claim that $z' = z_1$ satisfies the conditions on the statement. Assume the existence of a $x'z$-path γ_1 avoiding x. Since zy is an edge in $\mathsf{CP}(U)$, we can find a zy-path γ_2 avoiding x. The concatenation of the paths γ_1, γ_2 and the yy'-subpath of π give us a $x'y'$ path avoiding x, which is a contradiction. Consequently, $x \in \mathsf{CP}(x', z)$. Moreover, since $z \in \mathsf{CP}(x', z')$ we conclude that $x \in \mathsf{CP}(x', z')$. By the same argument as before, the three elements x, y, and z cannot be at the same time check points of two single elements in U. Consequently, we can find an irredundant $x'z'$-path γ avoiding y. Assume the existence of a $z'y'$-path δ_y avoiding y. The concatenation of the paths γ and δ_y give us a $x'y'$-path avoiding y, which is a contradiction. Consequently, $y \in \mathsf{CP}(y', z')$. Similarly, if we assume the existence of a $y'z'$-path δ_z avoiding z, the concatenation of the paths π and δ_z gives an $x'z'$-path avoiding z, which is a contradiction. Consequently, $z \in \mathsf{CP}(y', z')$.

Finally, for the second configuration take $z' = z_2$. □