Graph Homomorphisms and Universal Algebra Course Notes

Manuel Bodirsky,
Institut für Algebra, TU Dresden,
Manuel.Bodirsky@tu-dresden.de

September 10, 2020

Disclaimer: these are course notes in draft state, and they probably contain many mistakes; please report them to *manuel.bodirsky@tu-dresden.de*.

Contents

Prerequisites. This course is designed for students of mathematics or computer science that already had an introduction to discrete structures. Almost all notions that we use in this text will be formally introduced, with the notable exception of basic concepts from complexity theory. For example, we do not formally introduce the class of polynomial-time computable functions and NP-completeness, even though these concepts are used when we discuss computational aspects of graph homomorphisms. Here we refer to an introduction to the theory of computation as for instance the book of Papadimitriou [53].

The text contains 88 exercises; the ones with a star are harder.

1 The Basics

1.1 Graphs and Digraphs

The concepts in this section are probably known to most students, and can safely be skipped; the section fixes standard terminology and conventions from graph theory and can be consulted later if needed. Almost all definitions in this section have generalisations to *relational structures*, which will be introduced in Section 4; however, we focus exclusively on graphs in this section since they allow to reach the key ideas of the underlying theory with a minimum of notation.

A *directed graph* (also *digraph*) G is a pair (V, E) of a set $V = V(G)$ of vertices and a binary relation $E = E(G)$ on V. Note that in general we allow that V is an infinite set. For some definitions and results, we require that V is finite, in which case we say that G is a *finite* digraph. However, since this course deals exclusively with finite digraphs, we will omit this most of the time. The elements (u, v) of E are called the *arcs* (or *directed edges*) of G. Note that we allow *loops*, i.e., arcs of the form (u, u); a digraph without loops is called *loopless*. If $(u, v) \in E(G)$ is an arc, and $w \in V(G)$ is a vertex such that $w = u$ or $w = v$, then we say that (u, v) and w are *incident*.

An *(undirected) graph* is a pair (V, E) of a set $V = V(G)$ of *vertices* and a set $E = E(G)$ of *edges*, each of which is an unordered pair of (not necessarily distinct) elements of V. In other words, we explicitly allow *loops*, which are edges that link a vertex with itself. Undirected graphs can be viewed as *symmetric* digraphs: a digraph $G = (V, E)$ is called *symmetric* if $(u, v) \in E$ if and only if $(v, u) \in E$. For a digraph G, we say that G' is the *undirected graph of* G if G' is the undirected graph with $V(G') = V(G)$ and where $\{u, v\} \in E(G')$ if $(u, v) \in E(G)$ or $(v, u) \in E(G)$. For an undirected graph G, we say that G' is an *orientation of* G if G' is a directed graph such that $V(G') = V(G)$ and $E(G')$ contains for each edge $\{u, v\} \in E(G)$ either the arc (u, v) or the arc (v, u), and no other arcs.

For some notions for digraphs G one can just use the corresponding notions for undirected graphs applied to the undirected graph of G; conversely, most notions for directed graphs, specialised to symmetric graphs, translate to notions for the respective undirected graphs.

1.1.1 Examples of graphs, and corresponding notation

- The *complete graph* on n vertices $\{1, \dots, n\}$, denoted by K_n. This is an undirected graph on n vertices in which every vertex is joined with any other distinct vertex (so K_n contains no loops).
- The *cyclic graph* on n vertices, denoted by C_n; this is the undirected graph with the vertex set $\{0, \dots, n-1\}$ and edge set

$$\big\{\{0,1\}, \dots, \{n-2, n-1\}, \{n-1, 0\}\big\} = \big\{\{u, v\} : |u - v| = 1 \mod n\big\}.$$

- The *directed cycle* on n vertices, denoted by $\vec{C}_n$; this is the digraph with the vertex set $\{0, \dots, n-1\}$ and the arcs $\big((0,1), \dots, (n-2, n-1), (n-1, 0)\big)$.
- The *path* with $n+1$ vertices and n edges, denoted by P_n; this is an undirected graph with the vertex set $\{0, \dots, n\}$ and edge set $\big\{\{0,1\}, \dots, \{n-1, n\}\big\}$.

- The *directed path* with $n+1$ vertices and n edges, denoted by $\vec{P}_n$; this is a digraph with the vertex set $\{0, \ldots, n\}$ and edge set $\{(0,1), \ldots, (n-1, n)\}$.
- A *tournament* is a directed loopless graph G with the property that for all distinct vertices x, y either (x, y) or (y, x) is an edge of G, but not both.
- The *transitive tournament* on $n \geq 2$ vertices, denoted by T_n; this is a directed graph with the vertex set $\{1, \ldots, n\}$ where (i, j) is an arc if and only if $i < j$.

Let G and H be graphs (we define the following notions both for directed and for undirected graphs). Then $G \uplus H$ denotes the *disjoint union of G and H*, which is the graph with vertex set $V(G) \cup V(H)$ (we assume that the two vertex sets are disjoint; if they are not, we take a copy of H on a disjoint set of vertices and form the disjoint union of G with the copy of H) and edge set $E(G) \cup E(H)$. A graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. A graph G' is an *induced subgraph* of G if $V' = V(G') \subseteq V(G)$ and $(u, v) \in E(G')$ if and only if $(u, v) \in E(G)$ for all $u, v \in V'$. We also say that G' is *induced by V' in G*, and write $G[V']$ for G'. We write $G - u$ for $G[V(G) \setminus \{u\}]$, i.e., for the subgraph of G where the vertex u and all incident arcs are removed.

We call $|V(G)| + |E(G)|$ the *size* of a graph G; note that the size of a graph G reflects the length of the representation of G as a bit-string (under a reasonable choice of graph representations). This quantity will be important when we analyse the efficiency of algorithms on graphs.

1.1.2 Paths and Cycles

We start with definitions for directed paths; the corresponding terminology is then also used for undirected graphs as explained in the beginning of this section.

A *path* P (from u_1 to u_k in G) is a sequence $(u_1, \ldots, u_k)$ of vertices of G and a sequence $(e_1, \ldots, e_{k-1})$ of edges of G such that $e_i = (u_i, u_{i+1})$ or $e_i = (u_{i+1}, u_i) \in E(G)$, for every $1 \leq i < k$. The vertex u_1 is called the *start vertex* and the vertex u_k is called the *terminal vertex* of P, and we say that P is a path *from u_1 to u_k*. Edges (u_i, u_{i+1}) are called *forward edges* and edges (u_{i+1}, u_i) are called *backward edges*. If all edges are forward edges then the path is called *directed*. If $u_1, \ldots, u_k$ are pairwise distinct then the path is called *simple*. We write $|P| := k - 1$ for the *length* of P (i.e., we count the number of edges of P). The *net length* of P is the difference between the number of forward and the number of backward edges. Hence, a path is directed if and only if its length equals its net length.

A sequence $(u_0, \ldots, u_{k-1})$ of vertices and a sequence of edges $(e_0, \ldots, e_{k-1})$ is called a *cycle* (of G) if $(u_0, \ldots, u_{k-1}, u_0)$ and $(e_0, \ldots, e_{k-1})$ form a path. If all the vertices of the cycle are pairwise distinct then the cycle is called *simple*. We write $|C| := k$ for the *length* of the cycle $C = (u_0, \ldots, u_k - 1)$. The *net length* of C is the net length of the corresponding path $(u_0, \ldots, u_{k-1}, u_0)$. The cycle C is called *directed* if the corresponding path is a directed path.

A digraph G is called *(weakly) connected* if there is a path in G from any vertex to any other vertex in G. Equivalently, G is connected if and only if it cannot be written as $H_1 \uplus H_2$ for digraphs H_1, H_2 with at least one vertex each. A *connected component* of G is a maximal (with respect to inclusion of the vertex sets) connected induced subgraph of G. A digraph G is called *strongly connected* if for all vertices $x, y \in V(G)$ there is a directed path from x to y in G. Two vertices $u, v \in V(G)$ are *at distance k* in G if the shortest path from u to v in G has length k.

Some particular notions for undirected graphs G. A *(simple) cycle* of G is a sequence $(v_1, \ldots, v_k)$ of $k \geq 3$ pairwise distinct vertices of G such that $\{v_1, v_k\} \in E(G)$ and $\{v_i, v_{i+1}\} \in E(G)$ for all $1 \leq i \leq k-1$. An undirected graph is called *acyclic* if it does not contain a cycle. A sequence $u_1, \ldots, u_k \in V(G)$ is called a *(simple) path* from u_1 to u_k in G if $\{u_i, u_{i+1}\} \in E(G)$ for all $1 \leq i < k$ and if all vertices $u_1, \ldots, u_k$ are pairwise distinct. We allow the case that $k = 1$, in which case the path consists of a single vertex and no edges. Two vertices $u, v \in G$ are *at distance* k in G if the shortest path in G from u to v has length k. We say that an undirected graph G is *connected* if for all vertices $u, v \in V(G)$ there is a path from u to v. The *connected components* of G are the maximal connected induced subgraphs of G. A *forest* is an undirected acyclic graph, a *tree* is a connected forest.

1.2 Graph Homomorphisms

Let G and H be directed graphs. A *homomorphism* from G to H is a mapping $h\colon V(G) \to V(H)$ such that $(h(u), h(v)) \in E(H)$ whenever $(u, v) \in E(G)$. If such a homomorphism exists between G and H we say that G *homomorphically maps* to H, and write $G \to H$. Two directed graphs G and H are *homomorphically equivalent* if $G \to H$ and $H \to G$.

A homomorphism from G to H is sometimes also called an *H-colouring* of G. This terminology originates from the observation that H-colourings generalise classical colourings in the sense that a graph is n-colourable if and only if it has a K_n-colouring. Graph n-colorability is not the only natural graph property that can be described in terms of homomorphisms:

- a digraph is called *balanced* (in some articles: *layered*) if it homomorphically maps to a directed path $\vec{P}_n$;
- a digraph is called *acyclic* if it homomorphically maps to a transitive tournament T_n.

The equivalence classes of finite digraphs with respect to homomorphic equivalence will be denoted by $\mathcal{D}$. Let $\leq$ be a binary relation defined on $\mathcal{D}$ as follows: we set $C_1 \leq C_2$ if there exists a digraph $H_1 \in C_1$ and a digraph $H_2 \in C_2$ such that $H_1 \to H_2$ (note that this definition does not depend on the choice of the representatives H_1 of C_1 and H_2 of C_2). If f is a homomorphism from H_1 to H_2, and g is a homomorphism from H_2 to H_3, then the composition $f \circ g$ of these functions is a homomorphism from H_1 to H_3, and therefore the relation $\leq$ is transitive. Since every graph H homomorphically maps to H, the order $\leq$ is also reflexive. Finally, $\leq$ is antisymmetric since its elements are equivalence classes of directed graphs with respect to homomorphic equivalence. Define $C_1 < C_2$ if $C_1 \leq C_2$ and $C_1 \neq C_2$. We call $(\mathcal{D}, \leq)$ the *homomorphism order* of finite digraphs.

The homomorphism order on digraphs turns out to be a *lattice* where every two elements have a supremum (also called *join*) and an infimum (also called *meet*; see Example 19). In the proof of this result, we need the notion of *direct products* of graphs. This notion of graph product[1] can be seen as a special case of the notion of direct product as it is used in model theory [45]. The class of all graphs with respect to homomorphisms forms an interesting category in the sense of category theory [42] where the product introduced above is the product in the sense of category theory, which is why this product is sometimes also called the *categorical* graph product.

Let H_1 and H_2 be two graphs. Then the *(direct-, cross-, categorical-) product* $H_1 \times H_2$ of H_1 and H_2 is the graph with vertex set $V(H_1) \times V(H_2)$; the pair $((u_1, u_2), (v_1, v_2))$ is in

[1] Warning: there are several other notions of graph products that have been studied; see e.g. [42].

$E(H_1 \times H_2)$ if $(u_1, v_1) \in E(H_1)$ and $(u_2, v_2) \in E(H_2)$. Note that the product is symmetric and associative in the sense that $H_1 \times H_2$ is isomorphic to $H_2 \times H_1$ and $H_1 \times (H_2 \times H_3)$ is isomorphic to $(H_1 \times H_2) \times H_3$, and we therefore do not specify the order of multiplication when multiplying more than two graphs. The *n-th power* H^n of a graph H is inductively defined as follows. H^1 is by definition H. If H^i is already defined, then H^{i+1} is $H^i \times H$.

Proposition 1.1. *The homomorphism order $(\mathcal{D}, \leq)$ is a* lattice*; i.e., for all $C_1, C_2 \in \mathcal{D}$*

- *there exists an element $C_1 \vee C_2 \in \mathcal{D}$, the* join *of C_1 and C_2, such that $C_1 \leq (C_1 \vee C_2)$ and $C_2 \leq (C_1 \vee C_2)$, and such that for every $U \in \mathcal{D}$ with $C_1 \leq U$ and $C_2 \leq U$ we have $C_1 \vee C_2 \leq U$.*
- *there exists an element $C_1 \wedge C_2 \in \mathcal{D}$, the* meet *of C_1 and C_2, such that $(C_1 \wedge C_2) \leq C_1$ and $(C_1 \wedge C_2) \leq C_2$, and such that for every $U \in \mathcal{D}$ with $U \leq C_1$ and $U \leq C_2$ we have $U \leq C_1 \wedge C_2$.*

Proof. Let $H_1 \in C_1$ and $H_2 \in C_2$. For the join, the equivalence class of the disjoint union $H_1 \uplus H_2$ has the desired properties. For the meet, the equivalence class of $H_1 \times H_2$ has the desired properties. □

With the seemingly simple definitions of graph homomorphisms and direct products we can already formulate very difficult combinatorial questions.

Conjecture 1 (Hedetniemi)**.** *Let G and H be finite graphs, and suppose that $G \times H \to K_n$. Then $G \to K_n$ or $H \to K_n$.*

The smallest $n \in \mathbb{N}$ such that $G \to K_n$ is also called the *chromatic number* of G, and denoted by $\chi(G)$. Clearly, $\chi(G \times H) \leq \min(\chi(G), \chi(H))$. Hedetniemi's conjecture can be rephrased as

$$\chi(G \times H) = \min(\chi(G), \chi(H)) .$$

This conjecture is easy for $n = 1$ and $n = 2$ (Exercise 4), and has been solved for $n = 3$ by El Zahar and Sauer [31]. The conjecture has been refuted in 2019 by Yaroslav Shitov [59].

Clearly, $(\mathcal{D}, \leq)$ has infinite *ascending chains*, that is, sequences $C_1, C_2, \ldots$ such that $C_i < C_{i+1}$ for all $i \in \mathbb{N}$. Take for instance $C_i := \vec{P}_i$. More interestingly, $(\mathcal{D}, \leq)$ also has infinite descending chains.

Proposition 1.2. *The lattice $(\mathcal{D}, \leq)$ contains infinite descending chains $C_1 > C_2 > \cdots$.*

Proof. For this we use the following directed graphs, called *zig-zags*, which are frequently used in the theory of graph homomorphisms. We may write an orientation of a path P as a sequence of 0's and 1's, where 0 represents a forward arc and 1 represents a backward arc. For two orientations of paths P and Q with the representation $P = p_0, \ldots, p_n \in \{0,1\}^*$ and $Q = q_0, \ldots, q_m \in \{0,1\}^*$, respectively, the *concatenation* $P \circ Q$ of P and Q is the oriented path represented by $p_0, \ldots, p_n, q_0, \ldots, q_m$. For $k \geq 1$, the *zig-zag of order k*, denoted by Z_k, is the orientation of a path represented by $11(01)^{k-1}1$. We recommend the reader to draw pictures of Z_k where forward arcs point up and backward arcs point down. Now, the equivalence classes of the graphs $Z_1, Z_2, \ldots$ form an infinite descending chain. □

Proposition 1.3. *The lattice $(\mathcal{D}, \leq)$ contains infinite* antichains*, that is, sets of pairwise incomparable elements of $\mathcal{D}$ with respect to $\leq$.*

Proof. Again, it suffices to work with orientations of paths. For $k, l \geq 1$, the k, l *multi zig-zag*, denoted by $Z_{k,l}$, is the orientation of a path represented by $1\big(1(01)^k\big)^l 1$. Our infinite antichain now consists of the equivalence classes containing the graphs $Z_{k,k}$ for $k \geq 1$. □

A *strong homomorphism* from a digraph G to a digraph H is a function from $V(G)$ to $V(H)$ such that $(f(u), f(v)) \in E(H)$ if and only if $(u, v) \in E(G)$ for all $u, v \in V(G)$. An *isomorphism* between two directed graphs G and H is an bijective strong homomorphism from G to H, i.e., $f(u) \neq f(v)$ for any two distinct vertices $u, v \in V(G)$. Note that a homomorphism $h\colon G \to H$ is an isomorphism if and only if it is bijective, and h^{-1} is a homomorphism from H to G.

Exercises.

1. How many connected components do we have in $(P_3)^3$?
2. How many weakly and strongly connected components do we have in $(\vec{C}_3)^3$?
3. Let G and H be digraphs. Prove that $G \times H$ has a directed cycle if and only if both G and H have a directed cycle.
4. Prove the Hedetniemi conjecture for $n = 1$ and $n = 2$.
5. Show that the Hedetniemi conjecture is equivalent to each of the following two statements.
   - Let n be a positive integer. If for two graphs G and H we have $G \not\to K_n$ and $H \not\to K_n$, then $G \times H \not\to K_n$.
   - Let G and H be graphs with $\chi(G) = \chi(H) = m$. Then there exists a graph K with $\chi(K) = m$ such that $K \to G$ and $K \to H$.
6. Show that Hedetniemi's conjecture is false for directed graphs.
   Hint: there are counterexamples G, H with four vertices each.
7. Show that for every $k \in \mathbb{N}$, every pair of adjacent vertices of $(K_3)^k$ has exactly one common neighbour (that is, every edge lies in a unique subgraph of $(K_3)^k$ isomorphic to K_3).
8. Show that for every $k \in \mathbb{N}$, every pair of non-adjacent vertices in $(K_3)^k$ has at least two common neighbours.
9. Show that a digraph G homomorphically maps to $\vec{P}_1$ if and only if $\vec{P}_2$ does not homomorphically map to G.
10. Construct an orientation of a tree that is not homomorphically equivalent to an orientation of a path.
11. Construct a balanced orientation of a cycle that is not homomorphically equivalent to an orientation of a path.
12. Show that for all digraphs G we have $G \to T_3$ if and only if $\vec{P}_3 \not\to G$.

13. Show that $G \to \vec{P}_n$, for some $n \geq 1$, if and only if any two paths in G that start and end in the same vertex have the same net length.

14. Show that $G \to \vec{C}_n$, for some $n \geq 1$, if and only if any two paths in G that start and end in the same vertex have the same net length modulo n.

15. Let a be an automorphism of K_n^k. Show that there are permutations $p_1, \ldots, p_k$ of $\{1, \ldots, n\}$ and a permutation q of $\{1, \ldots, k\}$ such that

$$a(x_1, \ldots, x_k) = (p_1(x_{q(1)}), \ldots, p_k(x_{q(k)})).$$

1.3 The H-colouring Problem and Variants

When does a given digraph G homomorphically map to a digraph H? For every digraph H, this question defines a computational problem, called the *H-colouring problem*. The input of this problem consists of a finite digraph G, and the question is whether there exists a homomorphism from G to H.

There are many variants of this problem. In the *precoloured H-colouring problem*, the input consists of a finite digraph G, together with a mapping f from a subset of $V(G)$ to $V(H)$. The question is whether there exists an extension of f to all of $V(G)$ which is a homomorphism from G to H. In the *list H-colouring problem*, the input consists of a finite digraph G, together with a set $S_x \subseteq V(H)$ for every vertex $x \in V(G)$. The question is whether there exists a homomorphism h from G to H such that $h(x) \in S_x$ for all $x \in V(G)$. It is clear that the H-colouring problem reduces to the precoloured H-colouring problem (it is a special case: the partial map might have an empty domain), and that the precoloured H-colouring problem reduces to the list H-colouring problem (for vertices x in the domain of f, we set $S_x := \{f(x)\}$, and for vertices x outside the domain of f, we set $S_x := V(H)$).

The *constraint satisfaction problem* is a common generalisation of all these problems, and many more. It is defined not only for digraphs H, but more generally for *relational structures* S. Relational structures are the generalisation of graphs that can have many relations of arbitrary arity instead of just one binary edge relation. The constraint satisfaction problem will be introduced formally in Section 4. If H is a digraph, then the constraint satisfaction problem for H, also denoted CSP(H), is precisely the H-colouring problem and we use the terminology interchangeably.

Note that since graphs can be seen as a special case of digraphs, H-colouring is also defined for undirected graphs H. In this case we obtain essentially the same computational problem if we only allow undirected graphs in the input; this is made precise in Exercise 18.

For every finite graph H, the H-colouring problem is obviously in NP, because for every graph G it can be verified in polynomial time whether a given mapping from $V(G)$ to $V(H)$ is a homomorphism from G to H or not. Clearly, the same holds for the precoloured and the list H-colouring problem.

We have also seen that the K_n-colouring problem is the classical n-colouring problem, which is NP-complete [36] and therefore, no polynomial-time algorithm exists for CSP(K_n) unless P=NP. However, for many graphs and digraphs H (see Exercise 19 and 20) the H-colouring problem can be solved in polynomial time. Since the 1990s, researchers have studied the question: for which digraphs H can the H-colouring problem be solved in polynomial time? It has been conjectured by Feder and Vardi [34] that H-colouring is for any finite digraph H either NP-complete or can be solved in polynomial time. This is the so-called

dichotomy conjecture. It was shown by Ladner that unless P=NP there are infinitely many complexity classes between P and NP; so the conjecture states that for H-colouring these intermediate complexities do not appear.

The dichotomy conjecture has been confirmed in 2017, independently by Bulatov [18] and by Zhuk [62], even for the larger class of constraint satisfaction problems for finite relational structures. As we will see later, if we solve the classification question for the precoloured H-colouring problem, then we solve the classification question for H-colouring problem, and vice versa. Quite surprisingly, the same is true for the complexity of constraint satisfaction problems: a complete classification into NP-complete and P for the H-colouring problem would imply a classification for the class of all CSPs, and vice versa [34].

The list H-colouring problem, on the other hand, is quickly NP-hard, and therefore less difficult to classify. And indeed, a complete classification has been obtained by Bulatov [15] already in 2003. Alternative proofs can be found in [5, 17]. Also for finite *undirected* graphs, it is known since much longer that the dichotomy conjecture holds; this course fully covers the proof of the following.

Theorem 1.4 (of [40])**.** *Let H be a finite undirected graph. If H homomorphically maps to K_2, or contains a loop, then H-colouring can be solved in polynomial time. Otherwise, H-colouring is NP-complete.*

The case that H homomorphically maps to K_2 will be the topic of Exercise 19. The entire proof of Theorem 1.4 can be found in Section 8.

Exercices.

16. Let H be a finite directed graph. Find an algorithm that decides whether there is a strong homomorphism from a given graph G to the fixed graph H. The running time of the algorithm should be polynomial in the size of G (note that we consider $|V(H)|$ to be constant).

17. Let H be a finite digraph such that $\mathrm{CSP}(H)$ can be solved in polynomial time. Find a polynomial-time algorithm that constructs for a given finite digraph G a homomorphism to H, if such a homomorphism exists.

18. Let G and H be directed graphs, and suppose that H is symmetric. Show that $f\colon V(G) \to V(H)$ is a homomorphism from G to H if and only if f is a homomorphism from the undirected graph of G to the undirected graph of H.

19. Show that for any graph H that homomorphically maps to K_2 the constraint satisfaction problem for H can be solved in polynomial time.

20. Prove that $\mathrm{CSP}(T_3)$ can be solved in polynomial time.

21. Prove that $\mathrm{CSP}(\vec{C}_3)$ can be solved in polynomial time.

22. Let $\mathcal{N}$ be the set $\{Z_1, Z_2, Z_3, \dots\}$. Show that a digraph G homomorphically maps to $\vec{P}_2$ if and only if no digraph in $\mathcal{N}$ homomorphically maps to G.

23. Suppose that $\mathrm{CSP}(G)$ and $\mathrm{CSP}(H)$, for two digraphs G and H, can be solved in polynomial time. Show that $\mathrm{CSP}(G \times H)$ and $\mathrm{CSP}(G \uplus H)$ can be solved in polynomial time as well.

24. Suppose that G and H are homomorphically incomparable, and suppose that $\mathrm{CSP}(G \uplus H)$ can be solved in polynomial time. Show that $\mathrm{CSP}(G)$ and $\mathrm{CSP}(H)$ can be solved in polynomial time as well.

25. (*) Find digraphs G and H such that $\mathrm{CSP}(G \times H)$ can be solved in polynomial time, but $\mathrm{CSP}(G)$ and $\mathrm{CSP}(H)$ are NP-hard.

26. Suppose that G and H are homomorphically incomparable. Give a polynomial-time reduction from $\mathrm{CSP}(G)$ to $\mathrm{CSP}(G) \cup \mathrm{CSP}(H)$.

1.4 Cores

An *endomorphism* of a graph H is a homomorphism from H to H. An *automorphism* of a graph H is an isomorphism from H to H. A finite graph H is called a *core* if every endomorphism of H is an automorphism. A subgraph G of H is called *a core of* H if H is homomorphically equivalent to G and G is a core.

Proposition 1.5. *Every finite graph H has a core, which is unique up to isomorphism.*

Proof. Any finite graph H has a core, since we can select an endomorphism e of H such that the image of e has smallest cardinality; the subgraph of H induced by $e(V(H))$ is a core of H. Let G_1 and G_2 be cores of H, and $f_1 \colon H \to G_1$ and $f_2 \colon H \to G_2$ be homomorphisms. Let e_1 be the restriction of f_1 to $V(G_2)$. We claim that e_1 is the desired isomorphism. Suppose for contradiction that e_1 is not injective, i.e., there are distinct x, y in $V(G_2)$ such that $e_1(x) = e_1(y)$. It follows that $f_2 \circ e_1$ cannot be injective, too. But $f_2 \circ e_1$ is an endomorphism of G_2, contradicting the assumption that G_2 is a core. Similarly, the restriction e_2 of f_2 to $V(G_1)$ is an injective homomorphism from G_1 to G_2, and it follows that $|V(G_1)| = |V(G_2)|$ and both e_1 and e_2 are bijective.

Now, since $|V(G_2)|$ is finite, $e_2 \circ e_1 \circ \cdots \circ e_2 \circ e_1 = (e_2 \circ e_1)^n = \mathrm{id}$ for large enough n. Hence, $e_2 \circ e_1 \circ \cdots \circ e_2 = (e_1)^{-1}$, so the inverse of e_1 is a homomorphism, and hence an isomorphism between G_1 and G_2. □

Since a core G of a finite digraph H is unique up to isomorphism, we call G *the* core of H. We want to mention without proof that it is NP-complete to decide whether a given digraph H is not a core [41].

Cores can be characterised in many different ways; for some of them, see Exercise 28. There are examples of infinite digraphs that do not have a core in the sense defined above; see Exercise 31. Since a digraph H and its core have the same CSP, it suffices to study $\mathrm{CSP}(H)$ for core digraphs H only. Working with cores has advantages; one of them is shown in Proposition 1.7 below. In the proof of this proposition, we need a concept that we again in later sections.

Definition 1.6. Let H be a graph and let $u, v \in V(H)$ be vertices of H. Then the graph $H/\{u, v\}$ obtained from H by *contracting* u, v is defined to be the graph with vertex set $V(H) \setminus \{u, v\} \cup \{\{u, v\}\}$ and the edge set obtained from $E(H)$ by replacing each edge in $E(H)$ of the form (x, u) or (x, v), for $x \in V(H)$, by the edge $(x, \{u, v\})$, and each edge in $E(H)$ of the form (u, x) or (v, x), for $x \in V(H)$, by the edge $(\{u, v\}, x)$.

Proposition 1.7. *Let H be a core. Then* $\mathrm{CSP}(H)$ *and precoloured* $\mathrm{CSP}(H)$ *are linear-time equivalent.*

Proof. The reduction from CSP(H) to precoloured CSP(H) is trivial, because an instance G of CSP(H) is equivalent to the instance (G, c) of precoloured CSP(H) where c is everywhere undefined.

We show the converse reduction by induction on the size of the image of the partial mapping c in instances of precoloured CSP(H). Let (G, c) be an instance of precoloured CSP(H) where c has an image of size $k \geq 1$. We show how to reduce the problem to one where the partial mapping has an image of size $k-1$. If we compose all these reductions (note that the size of the image is bounded by $|V(H)|$), we finally obtain a reduction to CSP(H).

Let $x \in V(G)$ and $u \in V(H)$ be such that $c(x) = u$. We first contract all vertices y of G such that $c(y) = u$ with x. Then we create a copy of H, and attach the copy to G by contracting $x \in V(G)$ with $u \in V(H)$. Let G' be the resulting graph, and let c' be the partial map obtained from c by restricting it such that it is undefined on x, and then extending it so that $c(v) = v$ for all $v \in V(H)$, $v \neq u$, that appear in the image of c. Note that the size of G' and the size of G only differ by a constant.

We claim that (G', c') has a solution if and only if (G, c) has a solution. If f is a homomorphism from G to H that extends c, we further extend f to the copy of H that is attached in G' by setting $f(v')$ to v if vertex v' is a copy of a vertex $v \in V(H)$. This extension of f clearly is a homomorphism from G' to H and extends c'.

Now, suppose that f' is a homomorphism from G' to H that extends c'. The restriction of f' to the vertices from the copy of H that is attached to x in G' is an endomorphism of H, and because H is a core, it is an automorphism α of H. Moreover, α fixes v for all $v \in V(H)$ in the image of c'. Let β be the inverse of α, i.e., let β be the automorphism of H such that $\beta(\alpha(v)) = v$ for all $v \in V(H)$. Let f be the mapping from $V(G)$ to $V(H)$ that maps vertices that were identified with x to $\beta(f'(x))$, and all other vertices $y \in V(G)$ to $\beta(f'(y))$. Clearly, f is a homomorphism from G to H. Moreover, f maps vertices $y \in V(G)$, $y \neq x$, where c is defined to $c(y)$, since the same is true for f' and for α. Moreover, because x in G' is identified to u in the copy of H, we have that $f(x) = \beta(f'(x)) = \beta(f'(u)) = u$, and therefore f is an extension of c. □

We have already seen in Exercise 17 that the computational problem to construct a homomorphism from G to H, for fixed H and given G, can be reduced in polynomial-time to the problem of deciding whether there exists a homomorphism from G to H. The intended solution of Exercise 17 requires in the worst-case $|V(G)|^2$ many executions of the decision procedure for CSP(H). Using the concept of cores and the precoloured CSP (and its equivalence to the CSP) we can give a faster method to construct homomorphisms.

Proposition 1.8. *If there is an algorithm that decides* CSP(H) *in time T, then there is an algorithm that constructs a homomorphism from a given graph G to H (if such a homomorphism exists) which runs in time $O(|V(G)|T)$.*

Proof. We may assume without loss of generality that H is a core (since H and its core have the same CSP). By Proposition 1.7, there is an algorithm B for precoloured CSP(H) with a running time in $O(T)$. For given G, we first apply B to (G, c) for the everywhere undefined function c to decide whether there exists a homomorphism from G to H. If no, there is nothing to show. If yes, we select some $x \in V(G)$, and extend c by defining $c(x) = u$ for some $u \in V(H)$. Then we use algorithm B to decide whether there is a homomorphism from G to H that extends c. If no, we try another vertex $u \in V(H)$. Clearly, for some u the algorithm must give the answer "yes". We proceed with the extension c where $c(x) = u$, and repeat the

procedure with another vertex x from $V(G)$. At the end, c is defined for all vertices x of G, and c is a homomorphism from G to H. Clearly, since H is fixed, algorithm B is executed at most $O(|V(G)|)$ many times. □

Exercises.

27. Show that $Z_{k,l}$ is a core for all $k, l \geq 2$.

28. Prove that for every finite directed graph G the following is equivalent:

    - G is a core.
    - Every endomorphism of G is injective.
    - Every endomorphism of G is surjective.

29. Show that the three properties in the previous exercise are no longer equivalent if G is infinite.

30. Prove that the core of a strongly connected digraph is strongly connected.

31. Show that the infinite tournament $(\mathbb{Q}; <)$ has endomorphisms that are not automorphisms. Show that every digraph that is homomorphically equivalent to $(\mathbb{Q}; <)$ also has endomorphisms that are not automorphisms.

32. Let H be a core of G. Show that there exists a *retraction* from G to H, i.e., a homomorphism e from G to H such that $e(x) = x$ for all $x \in V(H)$.

33. The set of automorphisms of a digraph G forms a group; this group is called *transitive* if for all $a, b \in V(G)$ there is an automorphism f of G such that $f(a) = b$. Show that if G has a transitive automorphism group, then the core of G also has a transitive automorphism group.

34. Show that the connected components of a core are cores that form an antichain in $(D; \leq)$; conversely, the disjoint union of an antichain of cores is a core.

35. Prove that the core of a graph with a transitive automorphism group is connected.

36. Determine the computational complexity of $\mathrm{CSP}(H)$ for

$$H := \left(\mathbb{Z}; \{(x, y) \ : \ |x - y| \in \{1, 2\}\}\right) .$$

1.5 Polymorphisms

Polymorphisms are a powerful tool for analysing the computational complexity of constraint satisfaction problems; as we will see, they are useful both for NP-hardness proofs and for proving the correctness of polynomial-time algorithms for CSPs. Polymorphisms can be seen as multi-dimensional variants of endomorphisms.

Definition 1.9. Let H be a digraph and $k \geq 1$. Then a *polymorphism of H of arity k* is a homomorphism from H^k to H.

In other words, a mapping $f\colon V(H)^k \to V(H)$ is a polymorphism of H if and only if $(f(u_1,\dots,u_k),\ f(v_1,\dots,v_k)) \in E(H)$ whenever $(u_1,v_1),\dots,(u_k,v_k)$ are arcs in $E(H)$. Note that any graph H has all *projections* as polymorphisms, i.e., all mappings $p\colon V(H)^k \to V(H)$ that satisfy for some i the equation $p(x_1,\dots,x_k) = x_i$ for all $x_1,\dots,x_k \in V(H)$.

Example 1. The operation $(x,y) \mapsto \min(x,y)$ is a polymorphism of the directed graph $\vec{T}_n = (\{1,\dots,n\};<)$. △

An operation $f\colon V(H)^k \to V(H)$ is called

- *idempotent* if $f(x,\dots,x) = x$ for all $x \in V(H)$.
- *conservative* if $f(x_1,\dots,x_k) \in \{x_1,\dots,x_k\}$ for all $x_1,\dots,x_k \in V(H)$.

Definition 1.10. A digraph H is called *projective* if every idempotent polymorphism is a projection.

The following will be shown in Section 7.

Proposition 1.11. *For all $n \geq 3$, the graph K_n is projective.*

2 The Arc-consistency Procedure

The *arc-consistency procedure* is one of the most fundamental and well-studied algorithms that are applied for CSPs. This procedure was first discovered for constraint satisfaction problems in artificial intelligence [51,52]; in the graph homomorphism literature, the algorithm is sometimes called the *consistency check algorithm*.

Let H be a finite digraph, and let G be an instance of CSP(H). The idea of the procedure is to maintain for each vertex in G a list of vertices of H, and each element in the list of x represents a candidate for an image of x under a homomorphism from G to H. The algorithm successively removes vertices from these lists; it only removes a vertex $u \in V(H)$ from the list for $x \in V(G)$, if there is no homomorphism from G to H that maps x to u. To detect vertices x,u such that u can be removed from the list for x, the algorithm uses two rules (in fact, one rule and a symmetric version of the same rule): if (x,y) is an edge in G, then

- remove u from $L(x)$ if there is no $v \in L(y)$ with $(u,v) \in E(H)$;
- remove v from $L(y)$ if there is no $u \in L(x)$ with $(u,v) \in E(H)$.

If eventually we can not remove any vertex from any list with these rules any more, the graph G together with the lists for each vertex is called *arc-consistent*. Clearly, if the algorithm removes all vertices from one of the lists, then there is no homomorphism from G to H. The pseudo-code of the entire arc-consistency procedure is displayed in Figure 1.

Note that for any finite digraph H, if AC_H rejects an instance of CSP(H), then it clearly has no solution. The converse implication does not hold in general. For instance, let H be K_2, and let G be K_3. In this case, AC_H does not remove any vertex from any list, but obviously there is no homomorphism from K_3 to K_2.

However, there are digraphs H where the AC_H is a complete decision procedure for CSP(H) in the sense that it rejects an instance G of CSP(H) if and only if G does not homomorphically map to H. In this case we say that AC *solves* CSP(H).

$\mathrm{AC}_H(G)$
Input: a finite digraph G.
Data structure: a list $L(x) \subseteq V(H)$ for each vertex $x \in V(G)$.

Set $L(x) := V(H)$ for all $x \in V(G)$.
Do
    For each $(x, y) \in E(G)$:
        Remove u from $L(x)$ if there is no $v \in L(y)$ with $(u, v) \in E(H)$.
        Remove v from $L(y)$ if there is no $u \in L(x)$ with $(u, v) \in E(H)$.
        If $L(x)$ is empty for some vertex $x \in V(G)$ then **reject**
Loop until no list changes

Figure 1: The arc-consistency procedure for $\mathrm{CSP}(H)$.

Implementation. The running time of AC_H is for any fixed digraph H polynomial in the size of G. In a naive implementation of the procedure, the inner loop of the algorithm would go over all edges of the graph, in which case the running time of the algorithm is quadratic in the size of G. In the following we describe an implementation of the arc-consistency procedure, called AC-3, which is due to Mackworth [51], and has a worst-case running time that is linear in the size of G. Several other implementations of the arc-consistency procedure have been proposed in the Artificial Intelligence literature, aiming at reducing the costs of the algorithm in terms of the number of vertices of both G and H. But here we consider the size of H to be fixed, and therefore we do not follow this line of research. With AC-3, we rather present one of the simplest implementations of the arc-consistency procedure with a linear running time.

The idea of AC-3 is to maintain a *worklist*, which contains a list of arcs (x_0, x_1) of G that might help remove a value from $L(x_0)$ or $L(x_1)$. Whenever we remove a value from a list $L(x)$, we add all arcs that are in G incident to x. Note that then any arc in G might be added at most $2|V(H)|$ many times to the worklist, which is a constant in the size of G. Hence, the while loop of the implementation is iterated for at most a linear number of times. Altogether, the running time is linear in the size of G as well.

Arc-consistency for pruning search. Suppose that H is such that AC does not solve $\mathrm{CSP}(H)$. Even in this situation the arc-consistency procedure might be useful for *pruning the search space* in exhaustive approaches to solve $\mathrm{CSP}(H)$. In such an approach we might use the arc-consistency procedure as a subroutine as follows. Initially, we run AC_H on the input instance G. If it computes an empty list, we reject. Otherwise, we select some vertex $x \in V(G)$, and set $L(x)$ to $\{u\}$ for some $u \in L(x)$. Then we proceed recursively with the resulting lists. If AC_H now detects an empty list, we backtrack, but remove u from $L(x)$. Finally, if the algorithm does not detect an empty list at the first level of the recursion, we end up with singleton lists for each vertex $x \in V(G)$, which defines a homomorphism from G to H.

```
AC-3_H(G)
Input: a finite digraph G.
Data structure: a list L(x) of vertices of H for each x ∈ V(G).
        the worklist W: a list of arcs of G.

Subroutine Revise((x_0, x_1), i)
Input: an arc (x_0, x_1) ∈ E(G), an index i ∈ {0, 1}.
    change = false
    for each u_i in L(x_i)
        If there is no u_{1-i} ∈ L(x_{1-i}) such that (u_0, u_1) ∈ E(H) then
            remove u_i from L(x_i)
            change = true
        end if
    end for
    If change = true then
        If L(x_i) = ∅ then reject
    else
        For all arcs (z_0, z_1) ∈ E(G) with z_0 = x_i or z_1 = x_i add (z_0, z_1) to W
    end if

W := E(G)
Do
    remove an arc (x_0, x_1) from W
    Revise((x_0, x_1), 0)
    Revise((x_0, x_1), 1)
while W ≠ ∅
```

Figure 2: The AC-3 implementation of the arc-consistency procedure for CSP(H).

2.1 The Power Graph

For which H does the Arc-Consistency procedure solve CSP(H)? In this section we present an elegant and effective characterization of those finite graphs H where AC solves CSP(H), found by Feder and Vardi [34].

Definition 2.1. For a digraph H, the *power graph* $P(H)$ is the digraph whose vertices are non-empty subsets of $V(H)$ and where two subsets U and V are joined by an arc if the following holds:

- for every vertex $u \in U$, there exists a vertex $v \in V$ such that $(u, v) \in E(H)$, and
- for every vertex $v \in V$, there exists a vertex $u \in U$ such that $(u, v) \in E(H)$.

The definition of the power graph resembles the arc-consistency algorithm, and indeed, we have the following lemma which describes the correspondence.

Lemma 2.2. AC_H *rejects* G *if and only if* $G \not\to P(H)$.

Proof. Suppose first that AC_H does *not* reject G. For $u \in V(G)$, let $L(u)$ be the list derived at the final stage of the algorithm. Then by definition of $E(P(H))$, the map $x \mapsto L(x)$ is a homomorphism from G to $P(H)$.

Conversely, suppose that $f\colon G \to P(H)$ is a homomorphism. We prove by induction over the execution of AC_H that for all $x \in V(G)$ the elements of $f(x)$ are never removed from $L(x)$. To see that, let $(a,b) \in E(G)$ be arbitrary. Then $((f(a),f(b)) \in E(P(H))$, and hence for every $u \in f(a)$ there exists a $v \in f(b)$ such that $(u,v) \in E(H)$. By inductive assumption, $v \in L(b)$, and hence u will not be removed from $L(a)$. This concludes the inductive step. □

Theorem 2.3. *Let H be a finite digraph. Then* AC *solves* $\mathrm{CSP}(H)$ *if and only if $P(H)$ homomorphically maps to H.*

Proof. Suppose first that AC solves $\mathrm{CSP}(H)$. Apply AC_H to $P(H)$. Since $P(H) \to P(H)$, the previous lemma shows that AC_H does not reject $P(H)$. Hence, $P(H) \to H$ by assumption.

Conversely, suppose that $P(H) \to H$. If AC_H rejects a graph G then $G \not\to H$. If AC_H does accept G, then the lemma asserts that $G \to P(H)$. Composing homomorphisms, we obtain that $G \to H$. □

Observation 2.4. Let H be a core digraph. Note that if $P(H)$ homomorphically maps to H, then there also exists a homomorphism that maps $\{x\}$ to x for all $x \in V(H)$. We claim that in this case the precoloured CSP for H can be solved by the modification of AC_H which starts with $L(x) := \{c(x)\}$ for all $x \in V(G)$ in the range of the precolouring function c, instead of $L(x) := V(H)$. This is a direct consequence of the proof of Theorem 2.3. If the modified version of AC_H solves the precoloured CSP for H, then the classical version of AC_H solves $\mathrm{CSP}(H)$. Hence, it follows that the following are equivalent:

- AC solves $\mathrm{CSP}(H)$;
- the above modification of AC_H solves the precoloured CSP for H;
- $P(H) \to H$.

Note that the condition given in Theorem 2.3 can be used to decide algorithmically whether AC solves $\mathrm{CSP}(H)$, because it suffices to test whether $P(H)$ homomorphically maps to H. Such problems about deciding properties of $\mathrm{CSP}(H)$ for given H are often called algorithmic *Meta-problems*. A naive algorithm for the above test would be to first construct $P(H)$, and then to search non-deterministically for a homomorphism from $P(H)$ to H, which puts the Meta-problem for solvability of $\mathrm{CSP}(H)$ by AC into the complexity class NExpTime (*Non-deterministic Exponential Time*). This can be improved.

Proposition 2.5. *There exists a deterministic exponential time algorithm that tests for a given finite core digraph H whether $P(H)$ homomorphically maps to H.*

Proof. We first explicitly construct $P(H)$, and then apply AC_H to $P(H)$. If AC_H rejects, then there is certainly no homomorphism from $P(H) \to H$ by the properties of AC_H, and we return 'false'. If AC_H accepts, then we cannot argue right away that $P(H)$ homomorphically maps to H, since we do not know yet whether AC_H is correct for $\mathrm{CSP}(H)$.

But here is the trick. What we do in this case is to pick an arbitrary $x \in V(P(H))$, and remove all but one value u from $L(x)$, and continue with the execution of AC_H. If AC_H then derives the empty list, we try the same with another value u' from $L(x)$. If we obtain failure for all values of $L(x)$, then clearly there is no homomorphism from $P(H)$ to H, and we return 'false'. Otherwise, if AC_H does not derive the empty list after removing all values but u from $L(x)$, we continue with another element y of $V(P(H))$, setting $L(y)$ to $\{v\}$ for some $v \in L(y)$.

We repeat this procedure until at the end we have constructed a homomorphism from $P(H)$ to H. In this case we return 'true'.

If AC_H rejects for some $x \in V(P(H))$ when $L_x = \{u\}$ for all possible $u \in V(H)$, then the adaptation of AC_H for the precoloured CSP would have given an incorrect answer for the previously selected variable (it said yes while it should have said no). By Observation 2.4, this means that $P(H)$ does *not* homomorphically map to H. Again, we return 'false'. □

The precise computational complexity to decide for a given digraph H whether $P(H) \to H$ is not known; we refer to [24] for related questions and results.

Question 1. *What is the computational complexity to decide for a given core digraph H whether $P(H) \to H$? Is this problem in P?*

2.2 Tree Duality

Another mathematical notion that is closely related to the arc-consistency procedure is *tree duality*. The idea of this concept is that when a digraph H has tree duality, then we can show that there is no homomorphism from a digraph G to H by exhibiting a *tree obstruction* in G. This is formalized in the following definition.

Definition 2.6. A digraph H has *tree duality* if there exists a (not necessarily finite) set $\mathcal{N}$ of orientations of finite trees such that for all digraphs G there is a homomorphism from G to H if and only if no digraph in $\mathcal{N}$ homomorphically maps to G.

We refer to the set $\mathcal{N}$ in Definition 2.6 as an *obstruction set* for $\mathrm{CSP}(H)$. Note that no $T \in \mathcal{N}$ homomorphically maps to H. The pair $(\mathcal{N}, H)$ is called a *duality pair*. We have already encountered such an obstruction set in Exercise 20, where $H = T_2$, and $\mathcal{N} = \{\vec{P}_2\}$. In other words, $(\{\vec{P}_2\}, T_2)$ is a duality pair. Other duality pairs are $(\{\vec{P}_3\}, T_3)$ (Exercise 12), and $(\{Z_1, Z_2, \dots\}, \vec{P}_2)$ (Exercise 22).

Theorem 2.7 is a surprising link between the completeness of the arc-consistency procedure, tree duality, and the power graph, and was discovered by Feder and Vardi [33] in the more general context of constraint satisfaction problems.

Theorem 2.7. *Let H be a finite digraph. Then the following are equivalent.*

1. *H has tree duality;*
2. *$P(H)$ homomorphically maps to H;*
3. *AC solves $\mathrm{CSP}(H)$.*
4. *If every orientation of a tree that homomorphically maps to G also homomorphically maps to H, then G homomorphically maps to H;*

Proof. The equivalence $2 \Leftrightarrow 3$ has been shown in the previous section. We show $3 \Rightarrow 1$, $1 \Rightarrow 4$, and $4 \Rightarrow 2$.

$3 \Rightarrow 1$: Suppose that AC solves $\mathrm{CSP}(H)$. We have to show that H has tree duality. Let $\mathcal{N}$ be the set of all orientations of trees that does not homomorphically map to H. We claim that if a digraph G does not homomorphically map to H, then there is $T \in \mathcal{N}$ that homomorphically maps to G.

By assumption, the arc-consistency procedure applied to G eventually derives the empty list for some vertex of G. We use the computation of the procedure to construct an orientation T of a tree, following the exposition in [48]. When deleting a vertex $u \in V(H)$ from the list of a vertex $x \in V(G)$, we define an orientation of a rooted tree $T_{x,u}$ with root $r_{x,u}$ such that

1. There is a homomorphism from $T_{x,u}$ to G mapping $r_{x,u}$ to x.
2. There is no homomorphism from $T_{x,u}$ to H mapping $r_{x,u}$ to u.

The vertex u is deleted from the list of x because we found an arc $(x_0, x_1) \in E(G)$ with $x_i = x$ for some $i \in \{0, 1\}$ such that there is no arc $(u_0, u_1) \in E(H)$ with $u_i = u$ and u_{1-i} in the list of x_{1-i}. We describe how to construct the tree T for the case that $i = 0$, i.e., for the case that there is an arc $(x, y) \in E(G)$ such that there is no arc $(u, v) \in E(H)$ where $v \in L(y)$; the construction for $i = 1$ is analogous.

Let p, q be vertices and (p, q) an edge of $T_{x,u}$, and select the root $r_{x,u} = p$. If $E(H)$ does not contain an arc (u, v), then we are done, since $T_{x,u}$ clearly satisfies property (1) and (2). Otherwise, for every arc $(u, v) \in E(H)$ the vertex v has already been removed from the list $L(y)$, and hence by induction $T_{y,v}$ having properties (1) and (2) is already defined. We then add a copy of $T_{y,v}$ to $T_{x,u}$ and contract the vertex $r_{y,v}$ with q.

We verify that the resulting orientation of a tree $T_{x,u}$ satisfies (1) and (2). Let f be the homomorphism from $T_{y,v}$ mapping $r_{y,v}$ to v, which exists due to (1). The extension of f to $V(T_{x,u})$ that maps p to x is a homomorphism from $T_{x,u}$ to G, and this shows that (1) holds for $T_{x,u}$. But any homomorphism from $T_{x,u}$ to H that maps $r_{x,u}$ to u would also map the root of $T_{y,v}$ to v, which is impossible, and this shows that (2) holds for $T_{x,u}$. When the list $L(x)$ of some vertex $x \in V(G)$ becomes empty, we can construct an orientation of a tree T by contracting the roots of all $T_{x,u}$ into a vertex r. We then find a homomorphism from T to G by mapping r to x and extending the homomorphism independently on each $T_{x,u}$. But any homomorphism from T to H must map r to some element $u \in V(H)$, and hence there is a homomorphism from $T_{x,u}$ to H that maps x to u, a contradiction.

$1 \Rightarrow 4$: If H has an obstruction set $\mathcal{N}$ consisting of orientations of trees, and if G does not homomorphically map to H, there exists an orientation of a tree $T \in \mathcal{N}$ that maps to G but not to H.

$4 \Rightarrow 2$: To show that $P(H)$ homomorphically maps to H, it suffices to prove that every orientation T of a tree that homomorphically maps to $P(H)$ also homomorphically maps to H. Let f be a homomorphism from T to $P(H)$, and let x be any vertex of T. We construct a sequence $f_0, \ldots, f_n$, for $n = |V(T)|$, where f_i is a homomorphism from the subgraph of T induced by the vertices at distance at most i to x in T, and f_{i+1} is an extension of f_i for all $1 \leq i \leq n$. The mapping f_0 maps x to some vertex from $f(x)$. Suppose inductively that we have already defined f_i. Let y be a vertex at distance $i+1$ from x in T. Since T is an orientation of a tree, there is a unique $y' \in V(T)$ of distance i from x in T such that $(y, y') \in E(T)$ or $(y', y) \in E(T)$. Note that $u = f_i(y')$ is already defined. In case that $(y', y) \in E(T)$, there must be a vertex v in $f(y)$ such that $(u, v) \in E(H)$, since $(f(y'), f(y))$ must be an arc in $P(H)$, and by definition of $P(H)$. We then set $f_{i+1}(y) = v$. In case that $(y, y') \in E(T)$ we can proceed analogously. By construction, the mapping f_n is a homomorphism from T to H. □

2.3 Totally Symmetric Polymorphisms

There is also a characterisation of the power of the arc-consistency procedure which is based on polymorphisms, due to [27].

Definition 2.8. A function $f\colon D^k \to D$ is called *totally symmetric* if

$$f(x_1,\dots,x_k) = f(y_1,\dots,y_k) \text{ whenever } \{x_1,\dots,x_k\} = \{y_1,\dots,y_k\}.$$

Theorem 2.9. *Let H be a finite digraph. Then the following are equivalent.*

1. *$P(H)$ homomorphically maps to H;*
2. *H has totally symmetric polymorphisms of all arities;*
3. *H has a totally symmetric polymorphism of arity $2|V(H)|$.*

Proof. 1. $\Rightarrow$ 2.: Suppose that g is a homomorphism from $P(H)$ to H, and let $k \in \mathbb{N}$ be arbitrary. Let f be defined by $f(x_1,\dots,x_k) = g(\{x_1,\dots,x_k\})$. If $(x_1,y_1),\dots,(x_k,y_k) \in E(H)$, then $\{x_1,\dots,x_k\}$ is adjacent to $\{y_1,\dots,y_k\}$ in $P(H)$, and hence $(f(x_1,\dots,x_k), f(y_1,\dots,y_k)) \in E(H)$. Therefore, f is a polymorphism of H, and it is clearly totally symmetric.

The implication 2. $\Rightarrow$ 3. is trivial. To prove that 3. $\Rightarrow$ 1., suppose that f is a totally symmetric polymorphism of arity $2|V(H)|$. Let $g\colon V(P(H)) \to V(H)$ be defined by

$$g(\{x_1,\dots,x_n\}) := f(x_1,\dots,x_{n-1},x_n,x_n,\dots,x_n)$$

which is well-defined because f is totally symmetric. Let $(U,W) \in E(P(H))$, and let $x_1,\dots,x_p$ be an enumeration of the elements of U, and $y_1,\dots,y_q$ be an enumeration of the elements of W. The properties of $P(H)$ imply that there are $y'_1,\dots,y'_p \in W$ and $x'_1,\dots,x'_q \in U$ such that $(x_1,y'_1),\dots,(x_p,y'_p) \in E(H)$ and $(x'_1,y_1),\dots,(x'_q,y_q) \in E(H)$. Since f preserves E,

$$g(U) = g(\{x_1,\dots,x_p\}) = f(x_1,\dots,x_p,x'_1,\dots,x'_q,x_1,\dots,x_1)$$

is adjacent to

$$g(W) = g(\{y_1,\dots,y_q\}) = f(y'_1,\dots,y'_p,y_1,\dots,y_q,y'_1,\dots,y'_1)\,. \qquad \square$$

Given Theorem 2.9, it is natural to ask whether there exists a k so that the existence of a totally symmetric polymorphism of arity k implies totally symmetric polymorphisms of all arities. The following example shows that this is not the case.

Example 2. For every prime $p \geq 3$, the graph $\vec{C}_p$ clearly does not have a totally symmetric polymorphism of arity p: if $f\colon \{0,\dots,p-1\}^p \to \{0,\dots,p-1\}$ is a totally symmetric operation, then $f(0,1,\dots,p-1) = f(1,\dots,p-1,0)$, and hence f does not preserve the edge relation. On the other hand, if $n < p$ then $\vec{C}_p$ has the totally symmetric polymorphism

$$f(x_1,\dots,x_n) := |S|^{-1} \sum_{x\in S} x \mod p$$

where $S = \{x_1,\dots,x_n\}$. (Note that $|S| < p$ and hence has a multiplicative inverse.) The operation is clearly totally symmetric; the verification that it preserves the edge relation of $\vec{C}_p$ is Exercise 45. $\triangle$

2.4 Semilattice Polymorphisms

Some digraphs have a single binary polymorphism that generates operations satisfying the conditions in the previous theorem, as in the following statement. A binary operation $f\colon D^2 \to D$ is called *commutative* if it satisfies

$$f(x,y) = f(y,x) \text{ for all } x,y \in D\ .$$

It is called *associative* if it satisfies

$$f(x,f(y,z)) = f(f(x,y),z) \text{ for all } x,y,z \in D\ .$$

Definition 2.10. A binary operation is called a *semilattice* operation f if it is associative, commutative, and idempotent.

Examples of semilattice operations are functions from $D^2 \to D$ defined as $(x,y) \mapsto \min(x,y)$; here the minimum is taken with respect to any fixed linear order of D.

Theorem 2.11. *Let H be a finite digraph. Then $P(H) \to H$ if and only if H is homomorphically equivalent to a graph with a semilattice polymorphism.*

Proof. Suppose first that $P(H) \to H$. Thus, H and $P(H)$ are homomorphically equivalent, and it suffices to show that $P(H)$ has a semilattice polymorphism. The mapping $(X,Y) \mapsto X \cup Y$ is clearly a semilattice operation; we claim that it preserves the edges of $P(H)$. Let (U,V) and (A,B) be edges in $P(H)$. Then for every $u \in U$ there is a $v \in V$ such that $(u,v) \in E(H)$, and for every $u \in A$ there is a $v \in B$ such that $(u,v) \in E(H)$. Hence, for every $u \in U \cup A$ there is a $v \in V \cup B$ such that $(u,v) \in E(H)$. Similarly, we can verify that for every $v \in V \cup B$ there is a $u \in U \cup A$ such that $(u,v) \in E(H)$. This proves the claim.

For the converse, suppose that H is homomorphically equivalent to a directed graph G with a semilattice polymorphism f. Let h be the homomorphism from H to G. The operation $(x_1,\dots,x_n) \mapsto f(x_1,f(x_2,f(\dots,f(x_{n-1},x_n)\dots)))$ is a totally symmetric polymorphism of G. Then Theorem 2.9 implies that $P(G) \to G$. The map $S \mapsto \{h(u) \mid u \in S\}$ is a homomorphism from $P(H)$ to $P(G)$. Therefore, $P(H) \to P(G) \to G \to H$, as desired. □

By verifying the existence of semilattice polymorphisms for a concrete class of digraphs, we obtain the following consequence.

Corollary 2.12. AC *solves* CSP(H) *if H is an orientation of a path.*

Proof. Suppose that $1,\dots,n$ are the vertices of H such that either $(i,i+1)$ or $(i+1,i)$ is an arc in $E(H)$ for all $i < n$. It is straightforward to verify that the mapping $(x,y) \mapsto \min(x,y)$ is a polymorphism of H. The statement now follows from Theorem 2.11. □

We want to remark that there are orientations of trees H with an NP-complete H-colouring problem (the smallest known graph has 30 vertices, found by Jana Fischer [35]; see Figure 3). So, unless P=NP, this shows that there are orientations of trees H that do not have tree-duality.

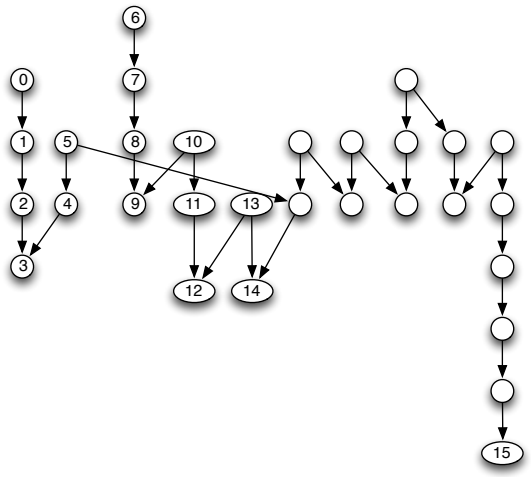

Figure 3: An orientation of a tree H with an NP-complete H-colouring problem [35].

Exercises.

37. Recall that a digraph is called *balanced* if it homomorphically maps to a directed path. Let H be a digraph.
    - Prove: if H is balanced, then $P(H)$ is balanced;
    - Disprove: if H is an orientation of a tree, then $P(H)$ is an orientation of a forest.
38. Up to isomorphism, there is only one unbalanced cycle H on four vertices that is a core and not the directed cycle. Show that AC does not solve CSP(H).
39. Does the digraph
    $$(\{0,1,2,3,4,5\};\{(0,1),(1,2),(0,2),(3,2),(3,4),(4,5),(3,5),(0,5)\})$$
    have tree duality?
40. Show that AC solves CSP(T_n), for every $n \geq 1$.
41. Let H be a finite digraph. Show that $P(H)$ contains a loop if and only if H contains a directed cycle.
42. Show that the previous statement is false for infinite digraphs H.
43. Let G and H be finite digraphs that are homomorphically incomparable and suppose that CSP(G) is NP-hard or CSP(H) is NP-hard. Show that CSP($G \uplus H$) is also NP-hard.
44. Show that an orientation of a tree homomorphically maps to H if and only if it homomorphically maps to $P(H)$.
45. Prove the final statement in Example 2.
46. Let H be a finite directed graph. Then AC_H rejects an orientation of a tree T if and only if there is no homomorphism from T to H (in other words, AC solves CSP(H) when the input is restricted to orientations of trees).

3 The Path-consistency Procedure

The path-consistency procedure is a well-studied generalization of the arc-consistency procedure from artificial intelligence. The path-consistency procedure is also known as the pair-consistency check algorithm in the graph theory literature.

Many CSPs that can not be solved by the arc-consistency procedure can still be solved in polynomial time by the path-consistency procedure. The simplest examples are $H = K_2$ (see Exercise 19) and $H = \vec{C}_3$ (see Exercise 21). The idea is to maintain a list of pairs from $V(H)^2$ for each pair of elements from $V(G)$ (similarly to the arc-consistency procedure, where we maintained a list of vertices from $V(H)$ for each vertex in $V(G)$). We successively remove pairs from these lists when the pairs can be excluded *locally*. Some authors maintain a list only for each pair of *distinct* vertices of $V(G)$, and they refer to our (stronger) variant as the *strong path-consistency procedure*. Our procedure (where vertices need not be distinct) has the advantage that it is at least as strong as the arc-consistency procedure, because the lists $L(x, x)$ and the rules of the path-consistency procedure for $x = y$ simulate the rules of the arc-consistency procedure.

$\mathrm{PC}_H(G)$
Input: a finite digraph G.
Data structure: for all $x, y \in V(G)$ a list $L(x, y)$ of elements of $V(H)^2$
For each $(x, y) \in V(G)^2$
    If $(x, y) \in E(G)$ then $L(x, y) := E(H)$,
    else $L(x, y) := V(H)^2$.
    If $x = y$ then $L(x, y) := L(x, y) \cap \{(u, u) \mid u \in V(H)\}$.
Do
    For all vertices $x, y, z \in V(G)$:
        For each $(u, w) \in L(x, z)$:
            If there is no $v \in V(H)$ such that $(u, v) \in L(x, y)$ and $(v, w) \in L(y, z)$ then
                Remove (u, w) from $L(x, z)$
        If $L(x, z)$ is empty then **reject**
Loop until no list changes

Figure 4: The strong path-consistency procedure for H-colouring

In Subsection 3.1 we will see many examples of digraphs H where the path-consistency procedure solves the H-colouring problem, but the arc-consistency procedure does not. The greater power of the path-consistency procedure comes at the price of a bigger worst-case running time: while the arc-consistency procedure has linear-time implementations, the best known implementations of the path-consistency procedure require cubic time in the size of the input (see Exercise 47).

The k-consistency procedure. The path-consistency procedure can be generalised further to the k-consistency procedure. In fact, arc- and path-consistency procedure are just a special case of the k-consistency for $k = 2$ and $k = 3$, respectively. In other words, for directed graphs H the path-consistency procedure is the 3-consistency procedure and the arc-consistency procedure is the 2-consistency procedure.

The idea of k-consistency is to maintain sets of $(k-1)$-tuples from $V(H)^{k-1}$ for each $(k-1)$-tuple from $V(G)^{k-1}$, and to successively remove tuples by local inference. It is straightforward to generalise also the details of the path-consistency procedure. For fixed H and fixed k, the running time of the k-consistency procedure is still polynomial in the size of G. But the dependency of the running time on k is clearly exponential.

However, we would like to point out that path consistency alias 3-consistency is of *particular* theoretical importance, due to the following recent result.

Theorem 3.1 (Barto and Kozik [6]). *If* CSP(H) *can be solved by k-consistency for some $k \geq 3$, then* CSP(H) *can also be solved by 3-consistency.*

Exercises

47. Show that the path-consistency procedure for the H-colouring problem can (for fixed H) be implemented such that the worst-case running time is cubic in the size of the input graph. (Hint: use a worklist as in AC-3.)

3.1 Majority Polymorphisms

In this section, we present a powerful criterion that shows that for certain graphs H the path-consistency procedure solves the H-colouring problem. Again, this condition was first discovered in more general form by Feder and Vardi [34]; it subsumes many criteria that were studied in artificial intelligence and in graph theory before.

Definition 3.2. Let D be a set. A function f from D^3 to D is called a *majority function* if f satisfies the following equations, for all $x, y \in D$:

$$f(x,x,y) = f(x,y,x) = f(y,x,x) = x$$

Example 3. As an example, let D be $\{1,\ldots,n\}$, and consider the ternary *median* operation, which is defined as follows. Let x, y, z be three elements from D. Suppose that $\{x,y,z\} = \{a,b,c\}$, where $a \leq b \leq c$. Then $median(x,y,z)$ is defined to be b. △

If a digraph H has a polymorphism f that is a majority operation, then f is called a *majority polymorphism of H*.

Example 4. Let H be the transitive tournament on n vertices, T_n. Suppose the vertices of T_n are the first natural numbers, $\{1,\ldots,n\}$, and $(u,v) \in E(T_n)$ if and only if $u < v$. Then the median operation is a polymorphism of T_n, because if $u_1 < v_1$, $u_2 < v_2$, and $u_3 < v_3$, then clearly $median(u_1,u_2,u_3) < median(v_1,v_2,v_3)$. This yields a new proof that the H-colouring problem for $H = T_n$ is tractable. △

Theorem 3.3 (of [34]). *Let H be a finite digraph. If H has a majority polymorphism, then the H-colouring problem can be solved in polynomial time (by the path-consistency procedure).*

For the proof of Theorem 3.3 we need the following lemma.

Lemma 3.4. *Let G and H be finite digraphs. Let f be a polymorphism of H of arity k and let $L := L(x,z)$ be the final list computed by the path-consistency procedure for $x, z \in V(G)$. Then f* preserves *L, i.e., if $(u_1,w_1),\ldots,(u_k,w_k) \in L$, then $(f(u_1,\ldots,u_k), f(w_1,\ldots,w_k)) \in L$.*

Proof. Let $(u_1, w_1), \ldots, (u_k, w_k) \in L$. We prove by induction over the execution of PC_H on G that at all times the pair $(u, w) := (f(u_1, \ldots, u_k), f(w_1, \ldots, w_k))$ is contained in L. Initially, this is true because f is a polymorphism of H. For the inductive step, let $y \in V(G)$. By definition of the procedure, for each $i \in \{1, \ldots, k\}$ there exists v_i such that $(u_i, v_i) \in L(x, y)$ and $(v_i, w_i) \in L(y, z)$. By the inductive assumption, $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)) \in L(x, y)$ and $(f(v_1, \ldots, v_k), f(w_1, \ldots, w_k)) \in L(y, z)$. Hence, $(f(u_1, \ldots, u_k), f(w_1, \ldots, w_k))$ will not be removed in the next step of the algorithm. □

Proof of Theorem 3.3. Let $f\colon V(H)^3 \to V(H)$ be a majority polymorphism of H. Clearly, if the path-consistency procedure derives the empty list for some pair (x, z) from $V(G)^2$, then there is no homomorphism from G to H.

Now suppose that after running the path-consistency procedure on G the list $L(x, z)$ is non-empty for all pairs (x, z) from $V(G)^2$. We have to show that there exists a homomorphism from G to H. A homomorphism h from an induced subgraph G' of G to H is said to *preserve the lists* if $(h(x), h(z)) \in L(x, z)$ for all $x, z \in V(G')$. The proof shows by induction on i that every homomorphism from a subgraph of G with i vertices that preserves the lists can be extended to any other vertex in G such that the resulting mapping is a homomorphism to H that again preserves the lists.

For the base case of the induction, observe that for all vertices $x, z \in V(G)$ every mapping h from $\{x, z\}$ to $V(H)$ such that $(h(x), h(z)) \in L(x, z)$ can be extended to every $y \in V(G)$ such that $(h(x), h(y)) \in L(x, y)$ and $(h(y), h(z)) \in L(y, z)$ (and hence preserves the lists), because otherwise the path-consistency procedure would have removed $(h(x), h(z))$ from $L(x, z)$.

For the inductive step, let h' be any homomorphism from a subgraph G' of G on $i \geq 3$ vertices to H that preserves the lists, and let x be any vertex of G not in G'. Let x_1, x_2, and x_3 be some vertices of G', and h'_j be the restriction of h' to $V(G') \setminus \{x_j\}$, for $1 \leq j \leq 3$. By inductive assumption, h'_j can be extended to x such that the resulting mapping h_j is a homomorphism to H that preserves the lists. We claim that the extension h of h' that maps x to $f(h_1(x), h_2(x), h_3(x))$ is a homomorphism to H that preserves the lists.

For all $y \in V(G')$, we have to show that $(h(x), h(y)) \in L(x, y)$ (and that $(h(y), h(x)) \in L(y, x)$, which can be shown analogously). If $y \notin \{x_1, x_2, x_3\}$, then $h(y) = h'(y) = f(h'(y), h'(y), h'(y)) = f(h_1(y), h_2(y), h_3(y))$, by the properties of f. Since $(h_i(x), h_i(y)) \in L(x, y)$ for all $i \in \{1, 2, 3\}$, and since f preserves $L(x, y)$ by Lemma 3.4, we have $(h(x), h(y)) \in L(x, y)$, and are done in this case.

Clearly, y can be equal to at most one of $\{x_1, x_2, x_3\}$. Suppose that $y = x_1$ (the other two cases are analogous). There must be a vertex $v \in V(H)$ such that $(h_1(x), v) \in L(x, y)$ (otherwise the path-consistency procedure would have removed $(h_1(x), h_1(x_2))$ from $L(x, x_2)$). By the properties of f, we have $h(y) = h'(y) = f(v, h'(y), h'(y)) = f(v, h_2(y), h_3(y))$. Because $(h_1(x), v), (h_2(x), h_2(y)), (h_3(x), h_3(y))$ are in $L(x, y)$, Lemma 3.4 implies that $(h(x), h(y)) = (f(h_1(x), h_2(x), h_3(x)), f(v, h_2(y), h_3(y)))$ is in $L(x, y)$, and we are done.

Therefore, for $i = |V(G)|$, we obtain a homomorphism from G to H. □

Corollary 3.5. *The path-consistency procedure solves the H-colouring problem for $H = T_n$.*

Another class of examples of graphs having a majority polymorphism are *unbalanced cycles*, i.e., orientations of C_n that do not homomorphically map to a directed path [32]. We only prove a weaker result here.

Proposition 3.6. *Directed cycles have a majority polymorphism.*

Proof. Let $\vec{C}_n$ be a directed cycle. Let f be the ternary operation on the vertices of $\vec{C}_n$ that maps u, v, w to u if u, v, w are pairwise distinct, and otherwise acts as a majority operation. We claim that f is a polymorphism of $\vec{C}_n$. Let $(u, u'), (v, v'), (w, w') \in E(\vec{C}_n)$ be arcs. If u, v, w are all distinct, then u', v', w' are clearly all distinct as well, and hence $(f(u, v, w), f(u', v', w')) = (u, u') \in E(\vec{C}_n)$. Otherwise, if two elements of u, v, w are equal, say $u = v$, then u' and v' must be equal as well, and hence $(f(u, v, w), f(u', v', w')) = (u, u') \in E(\vec{C}_n)$. □

```
Majority-Test(H)
Input: a finite digraph H.
Let G := H^3.
For all u, v ∈ V(H), contract the vertices (u,u,v), (u,v,u), (v,u,u), (u,u,u) in G.
If PC_H(G) derives the empty list, reject.
For each x ∈ V(G)
    For each (u,u) ∈ L(x,x)
        Found := False.
        For all y, z ∈ V(G), let L'(y,z) be a copy of L(y,z).
        L'(x,x) := {(u,u)}.
        Run PC_H(G) with the lists L'.
        If this run does not derive the empty list
            For all y, z ∈ V(G), set L(y,z) := L'(y,z).
            Found := True.
    End For.
    If Found = False then reject.
End For.
Accept.
```

Figure 5: A polynomial-time algorithm to find majority polymorphisms.

Theorem 3.7. *There is a polynomial-time algorithm to decide whether a given digraph H has a majority polymorphism.*

Proof. The pseudo-code of the procedure can be be found in Figure 5. Given H, we construct a new graph G as follows. We start from the third power H^3, and then contract all vertices of the form (u, u, v), (u, v, u), and (v, u, u) with the vertex (u, u, u). Let G be the resulting graph. Note that there exists a homomorphism from G to H if and only if H has a majority polymorphism. To decide whether G has a homomorphism to H, we run PC_H on G. If PC_H rejects, then we can be sure that there is no homomorphism from G to H, and hence H has no majority polymorphism. Otherwise, we use the same idea as in the proof of Proposition 2.5: pick $x \in V(G)$ and remove all but one pair (u, u) from $L(x, x)$. Then we continue with the execution of PC_H. If PC_H derives the empty list, we try the same with another pair (v, v) from $L(x, x)$. If we obtain failure for all pairs in $L(x, x)$, then clearly there is no homomorphism from G to H, and we return 'false'. Otherwise, if PC_H does not derive the empty list after removing all pairs but (u, u) from $L(x, x)$, we continue with another vertex $y \in V(G)$, setting $L(y, y)$ to $\{(u, u)\}$ for some $(u, u) \in L(y, y)$. We repeat this procedure until eventually all lists for pairs of the form (x, x) are singleton sets $\{(u, u)\}$; the map that sends x to u is a

homomorphism from G to H. In this case we return '*true*'. If there exists $u \in V(G)$ such that PC_H detects an empty list for all $(u,u) \in L(x,x)$ then the adaptation of PC_H for the precoloured CSP would have given an incorrect answer for the previously selected variable: PC_H did not detect the empty list even though the input was unsatisfiable. Hence, H cannot have a majority polymorphism (see Exercise 52). It is easy to see that the procedure described above has polynomial running time. □

Exercises.

48. A *quasi majority operation* is an operation from V^3 to V satisfying
$$f(x,x,y) = f(x,y,x) = f(y,x,x) = f(x,x,x)$$
for all $x, y \in V$. Use Theorem 1.4 to show that a finite undirected graph H has an H-colouring problem that can be solved in polynomial time if H has a quasi majority polymorphism, and is NP-complete otherwise.

49. Show that every orientation of a path has a majority polymorphism.

50. There is only one unbalanced cycle H on four vertices that is a core and not the directed cycle (we have seen this graph already in Exercise 38). Show that for this graph H the H-colouring problem can be solved by the path-consistency procedure.

51. (∗) Show that every unbalanced orientation of a cycle has a majority polymorphism.

52. Modify the path-consistency procedure such that it can deal with instances of the precoloured H-colouring problem. Show that if H has a majority polymorphism, then the modified path-consistency procedure solves the precoloured H-colouring problem.

53. Modify the path-consistency procedure such that it can deal with instances of the list H-colouring problem. Show that if H has a *conservative* majority polymorphism, then the modified path-consistency procedure solves the list H-colouring problem.

54. An *interval graph* H is an (undirected) graph $H = (V; E)$ such that there is an interval I_x of the real numbers for each $x \in V$, and $(x,y) \in E$ if and only if I_x and I_y have a non-empty intersection. Note that with this definition interval graphs are necessarily *reflexive*, i.e., $(x,x) \in E$. Show that the precoloured H-colouring problem for interval graphs H can be solved in polynomial time. Hint: use the modified path-consistency procedure in Exercise 52.

55. (∗) Let H be an (irreflexive) graph. Show that H has a conservative majority polymorphism if and only if H is an interval graph.

56. (∗) Let H be a reflexive graph. Show that H has a conservative majority polymorphism if and only if H is a *circular arc graph*, i.e., H can be represented by arcs on a circle so that two vertices are adjacent if and only if the corresponding arcs intersect.

57. Show that the digraph $(\mathbb{Z}; \{(x,y) \mid x - y = 1\})$ has a majority polymorphism.

58. Prove that H-colouring can be solved in polynomial time when H is the digraph from the previous exercise.

59. Show that the digraph $H = (\mathbb{Z}; \{(x,y) \mid x - y \in \{1,3\})$ has a majority polymorphism, and give a polynomial time algorithm for its H-colouring problem.

Near-unanimity Polymorphisms Majority functions are a special case of so-called *near-unanimity functions*. A function f from D^k to D is called a *(k-ary) near unanimity* (short, an *nu*) if f satisfies the following equations, for all $x, y \in D$:

$$f(x, \ldots, x, y) = f(x, \ldots, y, x) = \cdots = f(y, x, \ldots, x) = x$$

Obviously, the majority operations are precisely the ternary near-unanimity function. Similarly as in Theorem 3.3 it can be shown that the existence of a k-ary nu polymorphism of H implies that the k-consistency procedure solves CSP(H).

We mention that Theorem 3.1 implies that if H be a directed graph with a near unanimity polymorphism, then already PC solves CSP(H).

3.2 Digraphs with a Maltsev Polymorphism

If a digraph H has a *majority* polymorphism, then the path-consistency procedure solves CSP(H). How about graphs H with a *minority* polymorphisms of H? It turns out that this is an even stronger restriction.

Definition 3.8. An ternary function $f\colon D^3 \to D$ is called

- a *minority operation* if it satisfies

$$\forall x, y \in D.\, f(y, x, x) = f(x, y, x) = f(x, x, y) = y$$

- a *Maltsev operation* if it satisfies

$$\forall x, y \in D.\, f(y, x, x) = f(x, x, y) = y$$

Example 5. Let $D := \{0, \ldots, n-1\}$. Then the function $f\colon D^3 \to D$ given by $(x, y, z) \mapsto x - y + z \mod n$ is a Maltsev operation, since $x - x + z = z$ and $x - z + z = x$. For $n = 2$, this is even a minority operation. If $n > 2$, this function is not a minority, since then $1 - 2 + 1 = 0 \not\equiv 2 \mod n$. Note that f is a polymorphism of $\vec{C}_n$. To see this, suppose that $u_1 - v_1 \equiv 1 \mod n$, $u_2 - v_2 \equiv 1 \mod n$, and $u_3 - v_3 \equiv 1 \mod n$. Then

$$\begin{aligned} f(u_1, u_2, u_3) \equiv u_1 - u_2 + u_3 &\equiv (v_1 + 1) - (v_2 + 1) + (v_3 + 1) \\ &\equiv f(v_1, v_2, v_3) + 1 \mod n\,. \end{aligned}$$

△

The following result appeared in 2011.

Theorem 3.9 (Kazda [47])**.** *If a digraph G has a Maltsev polymorphism then G also has a majority polymorphism.*

Hence, for digraphs H with a Maltsev polymorphism, strong path-consistency solves the H-colouring problem, and in fact even the precoloured H-colouring problem. Theorem 3.9 is an immediate consequence of Theorem 3.14 below; to state it, we need the following concepts.

Definition 3.10. A digraph G is called *rectangular* if $(x, y), (x', y), (x', y') \in E(G)$ implies that $(x, y') \in E(G)$.

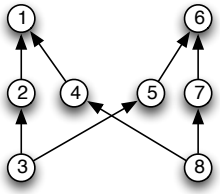

Figure 6: A totally rectangular digraph.

We start with the fundamental observation: digraphs with a Maltsev polymorphism m are rectangular. This follows immediately from the definition of polymorphisms: we must have $(m(x,x',x'),m(y,y,y')) \in E(G)$, but $m(x,x',x') = x$ and $m(y,y,y') = y'$, so $(x,y') \in E(G)$. The converse does not hold, as the following example shows.

Example 6. The graph $\big(\{a,b,c\};\{(a,a),(a,b),(b,c),(c,c)\}\big)$ is rectangular (draw a picture!), but has no Maltsev polymorphism m. Indeed, such an m would have to satisfy $m(a,a,c) = c$ and $m(a,c,c) = a$. Now $(m(a,a,c),m(a,b,c)) \in E(G)$ and $(m(a,b,c),m(a,c,c)) \in E(G)$, but G has no vertex x such that $(c,x) \in E(G)$ and $(x,a) \in E(G)$. △

We are therefore interested in stronger consequences of the existence of a Maltsev polymorphism.

Definition 3.11. A digraph G is called *k-rectangular* if whenever G contains directed paths of length k from x to y, from x' to y, and from x' to y', then also from x to y'. A digraph G is called *totally rectangular* if it is k-rectangular for all $k \geq 1$.

Lemma 3.12. *Every digraph with a Maltsev polymorphism m is totally rectangular.*

Proof. Let $k \geq 1$, and suppose that that G is a digraph with directed paths $(x_1,\ldots,x_k)$, $(y_1,\ldots,y_k)$, and $(z_1,\ldots,z_k)$ such that $x_k = y_k$ and $y_1 = z_1$. We have to show that there exists a directed path $(u_1,\ldots,u_k)$ in G with $u_1 = x_1$ and $u_k = z_k$. It can be verified that $u_i := m(x_i,y_i,z_i)$ has the desired properties. □

An example of a totally rectangular digraph is given in Figure 6. The next lemma points out an important consequence of k-rectangularity.

Lemma 3.13. *Let G be a finite totally rectangular digraph with a cycle of net length $d > 0$. Then G contains a directed cycle of length d.*

Proof. Let $C = (u_0,\ldots,u_{k-1})$ be a cycle of G of net length d; we prove the statement by induction on k. Clearly, C can be decomposed into maximal directed paths, that is, there is a minimal set $\mathcal{D}$ of directed paths such that each pair $(u_0,u_1),(u_1,u_2),\ldots,(u_{k-1},u_0)$ is contained in exactly one of the paths of $\mathcal{D}$. If the decomposition $\mathcal{D}$ consists of a single directed path then we have found a directed cycle and are done. Let P be the shortest directed path of $\mathcal{D}$, leading from u to v in G. Then there are directed paths Q and Q' in $\mathcal{D}$ such that Q starts in u and Q' ends in v, and $P \neq Q$ or $P \neq Q'$. By assumption, $|Q|,|Q'| \geq \ell := |P|$. By ℓ-rectangularity, there exists a directed path P' of length ℓ from the vertex s of Q' at position $|Q'| - \ell$ to the vertex t of Q at position ℓ. Now we distinguish the following cases.

- $Q = Q'$: the cycle that starts in s, follows the path Q until t, and then returns to s via the path P' is shorter than C but still has net length d.
- $Q \neq Q'$. The cycle starting in s, following Q for the final $|Q| - \ell$ vertices of Q, the cycle C until Q', the first $|Q'| - \ell$ vertices of Q' until t, and then P' back to s is a cycle which is shorter than C but still has net length d.

In both cases, the statement follows by induction. □

The following is a strengthening of Theorem 3.9; we only prove that 1 implies 2, and 2 implies 3, which suffices for the already mentioned consequence that for digraphs H with a Maltsev polymorphism, path-consistency solves the H-colouring problem.

Theorem 3.14 (Theorem 3.3 and Corollary 4.12 in [23])**.** *Let G be a finite digraph. Then the following are equivalent.*

1. *G has a Maltsev polymorphism.*
2. *G is totally rectangular.*
3. *If G is acyclic, then the core of G is a simple directed path. Otherwise, the core of G is a disjoint union of simple directed cycles.*
4. *G has a minority and a majority polymorphism.*

Proof. The implication from 4 to 1 is trivial since every minority operation is in particular a Maltsev operation. The implication from 1 to 2 is Lemma 3.12. For the implication from 2 to 3, let us assume that G is connected. The general case then follows by applying the following argument to each of its connected components, and the observation that directed paths homomorphically map to longer directed paths and to directed cycles.

We first consider the case that G is acyclic, and claim that in this case G is balanced, i.e., there exists a surjective homomorphism h from G to $\vec{P}_n$ for some $n \geq 1$. Otherwise, there exist $u, v \in V(G)$ and two paths P and Q from u to v of different net lengths ℓ_1 and ℓ_2 (see Exercise 13). Put these two paths together at u and v to form an unbalanced cycle C. Then Lemma 3.13 implies that G contains a directed cycle contrary to our assumptions.

Now, choose n with $G \to \vec{P}_n$ minimal, and fix $u \in h^{-1}(0)$ and $v \in h^{-1}(n)$. Then it is easy to see from total rectangularity that there must exist a path of length n in G from u to v, and hence the core of G is $\vec{P}_n$.

Now suppose that G contains a directed cycle; let C be the shortest directed cycle of G. We prove that G homomorphically maps to C. It is easy to see that it suffices to show that for any two vertices u, v of G and for any two paths P and Q from u to v we have that their net lengths are congruent modulo $m := |C|$ (see Exercise 14). Suppose for contradiction that there are paths of net length ℓ_1 and ℓ_2 from u to v in G such that $d := \ell_1 - \ell_2 \neq 0$ modulo m; without loss of generality, $\ell_2 < \ell_1$, so $d > 0$. We can assume that u is an element of C, since otherwise we can choose a path S from a vertex of C to u by connectivity of G, and append S to both P and Q. We can also assume that $d < m$ because if not, we can append C to Q to increase the length of Q by a multiple of m, until $d = \ell_1 - \ell_2 < m$. Lemma 3.13 then implies that G contains a directed cycle of length d, a contradiction to the choice of C.

For the missing implication from 3 to 4, we refer to [23] (Corollary 4.11). □

Exercises.

60. Let H be the graph $(\{0,1,\ldots,6\};\{(0,1),(1,2),(3,2),(4,3),(4,5),(5,6)\})$. For which k is it k-rectangular?

4 Logic

A *signature* is a set of relation and function symbols. The relation symbols are typically denoted by $R, S, T, \ldots$ and the function symbols are typically denoted by $f, g, h, \ldots$; each relation and function symbol is equipped with an *arity* from $\mathbb{N}$. A *τ-structure* $\mathfrak{A}$ consists of a set A (the *domain* or *base set*; we typically use the same letter in a different font) together with a relation $R^{\mathfrak{A}} \subseteq A^k$ for each relation symbol R of arity k from τ, and an operation $f^{\mathfrak{A}}\colon A^k \to A$ for each function symbol f of arity k from τ. Functions of arity 0 are allowed; they are also called *constants*. In this text it causes no harm to allow structures whose domain is empty. A τ-structure $\mathfrak{A}$ is called *finite* if its domain A is finite.

A *homomorphism* h from a τ-structure $\mathfrak{A}$ to a τ-structure $\mathfrak{B}$ is a function from A to B that *preserves* each relation and each function: that is,

- if $(a_1,\ldots,a_k)$ is in $R^{\mathfrak{A}}$, then $(h(a_1),\ldots,h(a_k))$ must be in $R^{\mathfrak{B}}$;
- for all $a_1,\ldots,a_k \in A$ we have $h(f^{\mathfrak{A}}(a_1,\ldots,a_k)) = f^{\mathfrak{B}}(h(a_1),\ldots,h(a_k))$.

An *isomorphism* is a bijective homomorphism h such that the inverse mapping $h^{-1}\colon B \to A$ that sends $h(x)$ to x is a homomorphism, too.

A *relational* structure is a τ-structure where τ only contains relation symbols, and an *algebra* (in the sense of universal algebra) is a τ-structure where τ only contains function symbols. In this section, we will work with relational structures only; algebras will appear in Section 7.

4.1 Primitive positive formulas

A first-order τ-formula $\phi(x_1,\ldots,x_n)$ is called *primitive positive* (in database theory also *conjunctive query*) if it is of the form

$$\exists x_{n+1},\ldots,x_\ell(\psi_1 \wedge \cdots \wedge \psi_m)$$

where $\psi_1,\ldots,\psi_m$ are *atomic τ-formulas*, i.e., formulas of the form $R(y_1,\ldots,y_k)$ with $R \in \tau$ and $y_i \in \{x_1,\ldots,x_\ell\}$, of the form $y = y'$ for $y, y' \in \{x_1,\ldots,x_\ell\}$, or $\top$ and $\bot$ (for *true* and *false*). As usual, formulas without free variables are called *sentences*. If $\mathfrak{A}$ is a τ-structure and ϕ a τ-sentence, then we write $\mathfrak{A} \models \phi$ if $\mathfrak{A}$ satisfies ϕ (i.e., ϕ holds in $\mathfrak{A}$).

Note that if we would require that all our structures have a non-empty domain, we would not need the symbol $\top$ since we can use the primitive positive sentence $\exists x.\, x = x$ to express it. It is possible to rephrase the H-colouring problem and its variants using primitive positive sentences.

Definition 4.1. Let $\mathfrak{B}$ be a structure with a finite relational signature τ. Then $\mathrm{CSP}(\mathfrak{B})$ is the computational problem of deciding whether a given primitive positive τ-sentence ϕ is true in $\mathfrak{B}$.

The given primitive positive τ-sentence ϕ is also called an *instance* of CSP($\mathfrak{B}$). The conjuncts of an instance ϕ are called the *constraints* of ϕ. A mapping from the variables of ϕ to the elements of B that is a satisfying assignment for the quantifier-free part of ϕ is also called a *solution* to ϕ.

Example 7 (Disequality constraints)**.** Consider the problem CSP($\{1, 2, \ldots, n\}; \neq$). An instance of this problem can be viewed as an (existentially quantified) set of variables, some linked by disequality[2] constraints. Such an instance holds in $(\{1, 2, \ldots, n\}; \neq)$ if and only if the graph whose vertices are the variables, and whose edges are the inequality constraints, has a homomorphism to K_n. △

Exercises.

61. Show that CSP$\big(\{0,1\}; \{(0,1),(1,0),(0,0)\}, \{(0,1),(1,0),(1,1)\}, \{(1,1),(1,0),(0,0)\}\big)$ can be solved in polynomial time.

62. Generalise the arc-consistency procedure from digraphs to general relational structures.

63. Does the arc-consistency procedure solve CSP($\mathfrak{B}$) where $\mathfrak{B}$ has domain $B = \{0,1,2,3\}$, the unary relation $U_i^{\mathfrak{B}}$ for every $i \in B$, and the binary relations $B^4 \setminus \{(0,0)\}$ and $\{(1,2),(2,3),(3,1),(0,0)\}$?

64. Generalise the path-consistency procedure from digraphs to general relational structures.

65. Verify that the structure $\mathfrak{B}$ from Exercise 63 has the binary idempotent commutative polymorphism $*$ defined as $1*2 = 2$, $2*3 = 3$, $3*1 = 1$, and $0*b = b$ for all $b \in \{1,2,3\}$. Verify that $*$ satisfies 'restricted associativity', i.e., $x*(x*y) = (x*x)*y$ for all $x, y \in B$ (and since it is additionally idempotent and commutative it is called a *2-semilattice*).

66. Does the the structure $\mathfrak{B}$ from Exercise 63 have a majority polymorphism?

67. (*) Does the path-consistency procedure solve CSP($\mathfrak{B}$) for the structure $\mathfrak{B}$ from Exercise 63?

4.2 From Structures to Formulas

To every finite relational τ-structure $\mathfrak{A}$ we can associate a τ-sentence, called the *canonical conjunctive query* of $\mathfrak{A}$, and denoted by $\phi(\mathfrak{A})$. The variables of this sentence are the elements of $\mathfrak{A}$, all of which are existentially quantified in the quantifier prefix of the formula, which is followed by the conjunction of all formulas of the form $R(a_1, \ldots, a_k)$ for $R \in \tau$ and tuples $(a_1, \ldots, a_k) \in R^{\mathfrak{A}}$.

For example, the canonical conjunctive query $\phi(K_3)$ of the complete graph on three vertices K_3 is the formula

$$\exists u \exists v \exists w \, \big(E(u,v) \wedge E(v,u) \wedge E(v,w) \wedge E(w,v) \wedge E(u,w) \wedge E(w,u)\big) \, .$$

The proof of the following proposition is straightforward.

Proposition 4.2. *Let $\mathfrak{B}$ be a structure with finite relational signature τ, and let $\mathfrak{A}$ be a finite τ-structure. Then there is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$ if and only if $\mathfrak{B} \models \phi(\mathfrak{A})$.*

[2]We deliberately use the word *disequality* instead of *inequality*, since we reserve the word *inequality* for the relation $x \leq y$.

4.3 From Formulas to Structures

To present a converse of Proposition 4.2, we define the *canonical structure* $\mathfrak{S}(\phi)$ (in database theory this structure is called the *canonical database*) of a primitive positive τ-formula, which is a relational τ-structure defined as follows. We require that ϕ does not contain $\bot$. If ϕ contains an atomic formula of the form $x = y$, we remove it from ϕ, and replace all occurrences of x in ϕ by y. Repeating this step if necessary, we may assume that ϕ does not contain atomic formulas of the form $x = y$.

Then the domain of $\mathfrak{S}(\phi)$ is the set of variables (both the free variables and the existentially quantified variables) that occur in ϕ. There is a tuple $(v_1, \ldots, v_k)$ in a relation R of $\mathfrak{S}(\phi)$ if and only if ϕ contains the conjunct $R(v_1, \ldots, v_k)$. The following is similarly straightforward as Proposition 4.2.

Proposition 4.3. *Let $\mathfrak{B}$ be a structure with signature τ, and let ϕ be a primitive positive τ-sentence other than $\bot$. Then $\mathfrak{B} \models \phi$ if and only if $\mathfrak{S}(\phi)$ homomorphically maps to $\mathfrak{B}$.*

Due to Proposition 4.3 and Proposition 4.2, we may freely switch between the homomorphism and the logic perspective whenever this is convenient. In particular, instances of CSP($\mathfrak{B}$) can from now on be either finite structures $\mathfrak{A}$ or primitive positive sentences ϕ.

Note that the H-colouring problem, the precoloured H-colouring problem, and the list H-colouring problem can be viewed as constraint satisfaction problems for appropriately chosen relational structures.

4.4 Primitive Positive Definability

Let $\mathfrak{A}$ be a τ-structure, and let $\mathfrak{A}'$ be a τ'-structure with $\tau \subseteq \tau'$. If $\mathfrak{A}$ and $\mathfrak{A}'$ have the same domain and $R^{\mathfrak{A}} = R^{\mathfrak{A}'}$ for all $R \in \tau$, then $\mathfrak{A}$ is called the *τ-reduct* (or simply *reduct*) of $\mathfrak{A}'$, and $\mathfrak{A}'$ is called a *τ'-expansion* (or simply *expansion*) of $\mathfrak{A}$. If $\mathfrak{A}$ is a structure, and R is a relation over the domain of $\mathfrak{A}$, then we denote the expansion of $\mathfrak{A}$ by R by $(\mathfrak{A}, R)$.

If $\mathfrak{A}$ is a τ-structure, and $\phi(x_1, \ldots, x_k)$ is a formula with k free variables $x_1, \ldots, x_k$, then the *relation defined by ϕ* is the relation

$$\{(a_1, \ldots, a_k) \mid \mathfrak{A} \models \phi(a_1, \ldots, a_k)\}.$$

If the formula is primitive positive, then this relation is called *primitive positive definable*.

Example 8. The relation $\{(a, b) \in \{0, 1, 2, 3, 4\}^2 \mid a \neq b\}$ is primitive positive definable in C_5: the primitive positive definition is

$$\exists p_1, p_2 \left(E(x_1, p_1) \wedge E(p_1, p_2) \wedge E(p_2, x_2)\right) \qquad \triangle$$

Example 9. The non-negative integers are primitively positively definable in $(\mathbb{Z}; 0, 1, +, *)$, namely by the following formula $\phi(x)$ which states that x is the sum of four squares:

$$\exists x_1, x_2, x_3, x_4 (x = x_1^2 + x_2^2 + x_3^2 + x_4^2).$$

Clearly, every integer that satisfies $\phi(x)$ is non-negative; the converse is the famous four-square theorem of Lagrange [39]. $\triangle$

The following lemma says that we can expand structures by primitive positive definable relations without changing the complexity of the corresponding CSP. Hence, primitive positive definitions are an important tool to prove NP-hardness: to show that $\mathrm{CSP}(\mathfrak{B})$ is NP-hard, it suffices to show that there is a primitive positive definition of a relation R such that $\mathrm{CSP}(\mathfrak{B}, R)$ is already known to be NP-hard. Stronger tools to prove NP-hardness of CSPs will be introduced in Section 5.

Lemma 4.4 (Jeavons, Cohen, Gyssens [46]). *Let $\mathfrak{B}$ be a structure with finite relational signature, and let R be a relation that has a primitive positive definition in $\mathfrak{B}$. Then $\mathrm{CSP}(\mathfrak{B})$ and $\mathrm{CSP}(\mathfrak{B}, R)$ are linear-time equivalent.*

Proof. It is clear that $\mathrm{CSP}(\mathfrak{B})$ reduces to the new problem. So suppose that ϕ is an instance of $\mathrm{CSP}((\mathfrak{B}, R))$. Replace each conjunct $R(x_1, \ldots, x_l)$ of ϕ by its primitive positive definition $\psi(x_1, \ldots, x_l)$. Move all quantifiers to the front, such that the resulting formula is in *prenex normal form* and hence primitive positive. Finally, equalities can be eliminated one by one: for equality $x = y$, remove y from the quantifier prefix, and replace all remaining occurrences of y by x. Let ϕ' be the formula obtained in this way.

It is straightforward to verify that ϕ is true in $(\mathfrak{B}, R)$ if and only if ϕ' is true in $\mathfrak{B}$, and it is also clear that ϕ' can be constructed in linear time in the representation size of ϕ. □

Recall from Section 1.3 that $\mathrm{CSP}(K_5)$ is NP-hard. Since the edge relation of K_5 is primitively positively definable in C_5 (Example 8), Lemma 4.4 implies that $\mathrm{CSP}(C_5)$ is NP-hard, too.

Exercises.

68. Show that the relation $R := \{(a, b, c) \in \{1, 2, 3\}^3 \mid a = b \text{ or } b = c \text{ or } a = c\}$ has a primitive positive definition over K_3.

69. Show that the relation $\neq$ on $\{1, 2, 3\}$ has a primitive positive definition in the structure $(\{1, 2, 3\}; R, \{1\}, \{2\}, \{3\})$ where R is the relation from the previous exercise.

70. Let R_+, R_* be the relations defined as follows.

$$R_+ := \{(x, y, z) \in \mathbb{Q}^3 \mid x + y = z\}$$
$$R_* := \{(x, y, z) \in \mathbb{Q}^3 \mid x * y = z\}.$$

    Show that R_* is primitive positive definable in the structure $(\mathbb{Q}; R_+, \{(x, y) \mid y = x^2\})$.

71. Let B be any set, and for $n \in \mathbb{N}$ define the relation P_B^{2n} of arity $2n$ as follows.

$$P_B^{2n} := \{(x_1, y_1, x_2, y_2, \ldots, x_n, y_n) \in B^{2n} \mid \bigvee_{i \in \{1, \ldots, n\}} x_i = y_i\}$$

    Show that for every n the relation P_B^{2n} has a primitive positive definition in $(B; P_B^4)$.

72. $(*)$ Let $n \geq 4$. Is there a primitive positive definition of $\neq$ over the structure

$$M_n := (\{1, \ldots, n\}; R, \{1\}, \{2\}, \ldots, \{n\})$$
$$\text{where } R := \{(1, \ldots, 1), (2, \ldots, 2), \ldots, (n, \ldots, n), (1, 2, \ldots, n)\}.$$

4.5 Cores and Constants

An *automorphism* of a structure $\mathfrak{B}$ with domain B is an isomorphism between $\mathfrak{B}$ and itself. The set of all automorphisms α of $\mathfrak{B}$ is denoted by $\mathrm{Aut}(\mathfrak{B})$, and forms a permutation group. When G is a permutation group on a set B, and $b \in B$, then a set of the form

$$S = \{\alpha(b) \mid \alpha \in G\}$$

is called an *orbit* of G (the *orbit of* b). Let $(b_1, \ldots, b_k)$ be a k-tuple of elements of $\mathfrak{B}$. A set of the form

$$S = \{(\alpha b_1, \ldots, \alpha b_k) \mid \alpha \in \mathrm{Aut}(\mathfrak{B})\}$$

is called an *orbit of k-tuples* of $\mathfrak{B}$; it is an orbit of the componentwise action of G on the set B^k of k-tuples from B.

Lemma 4.5. *Let $\mathfrak{B}$ be a structure with a finite relational signature and domain B, and let $R = \{(b_1, \ldots, b_k)\}$ be a k-ary relation that only contains one tuple $(b_1, \ldots, b_k) \in B^k$. If the orbit of $(b_1, \ldots, b_k)$ in $\mathfrak{B}$ is primitive positive definable, then there is a polynomial-time reduction from $\mathrm{CSP}(\mathfrak{B}, R)$ to $\mathrm{CSP}(\mathfrak{B})$.*

Proof. Let ϕ be an instance of $\mathrm{CSP}(\mathfrak{B}, R)$ with variable set V. If ϕ contains two constraints $R(x_1, \ldots, x_k)$ and $R(y_1, \ldots, y_k)$, then replace each occurrence of y_1 by x_1, then each occurrence of y_2 by x_2, and so on, and finally each occurrence of y_k by x_k. We repeat this step until all constrains that involve R are imposed on the same tuple of variables $(x_1, \ldots, x_k)$. Replace $R(x_1, \ldots, x_k)$ by the primitive positive definition θ of its orbit in $\mathfrak{B}$. Finally, move all quantifiers to the front, such that the resulting formula ψ is in prenex normal form and thus an instance of $\mathrm{CSP}(\mathfrak{B})$. Clearly, ψ can be computed from ϕ in polynomial time. We claim that ϕ is true in $(\mathfrak{B}, R)$ if and only if ψ is true in $\mathfrak{B}$.

Suppose ϕ has a solution $s\colon V \to B$. Since $(b_1, \ldots, b_k)$ satisfies θ, we can extend s to the existentially quantified variables of θ to obtain a solution for ψ. In the opposite direction, suppose that s' is a solution to ψ over $\mathfrak{B}$. Let s be the restriction of s' to V. Since $(s(x_1), \ldots, s(x_k))$ satisfies θ, it lies in the same orbit as $(b_1, \ldots, b_k)$. Thus, there exists an automorphism α of $\mathfrak{B}$ that maps $(s(x_1), \ldots, s(x_k))$ to $(b_1, \ldots, b_k)$. Then the extension of the map $x \mapsto \alpha s(x)$ that maps variables y_i of ϕ that have been replaced by x_i in ψ to the value b_i is a solution to ϕ over $(\mathfrak{B}, R)$. □

The definition of cores can be extended from finite digraphs to finite structures: as in the case of finite digraphs, we require that every endomorphism be an automorphism. All results we proved for cores of digraphs remain valid for cores of structures. In particular, every finite structure $\mathfrak{C}$ is homomorphically equivalent to a core structure $\mathfrak{B}$, which is unique up to isomorphism (see Section 1.4). The following proposition can be shown as in the proof of Proposition 1.7.

Proposition 4.6. *Let $\mathfrak{B}$ be a finite core structure. Then orbits of k-tuples of $\mathfrak{B}$ are primitive positive definable.*

Proposition 4.6 and Lemma 4.5 have the following consequence.

Corollary 4.7. *Let $\mathfrak{B}$ be a finite core with a finite relational signature. Let $b_1, \ldots, b_n \in B$. Then $\mathrm{CSP}(\mathfrak{B})$ and $\mathrm{CSP}(\mathfrak{B}, \{b_1\}, \ldots, \{b_n\})$ are polynomial time equivalent.*

Exercises.

73. Show that if m is the number of orbits of k-tuples of a finite structure $\mathfrak{A}$, and $\mathfrak{C}$ is the core of $\mathfrak{A}$, then $\mathfrak{C}$ has at most m orbits of k-tuples.

74. Show that if $\mathfrak{A}$ is a finite structure, and $\mathfrak{C}$ its core, and if $\mathfrak{A}$ and $\mathfrak{C}$ have the same number of orbits of pairs, then $\mathfrak{A}$ and $\mathfrak{C}$ are isomorphic.

4.6 Primitive Positive Interpretations

Primitive positive interpretations are a powerful generalisation of primitive positive definability that can be used to also relate structures with *different* domains.

If C and D are sets and $g\colon C \to D$ is a map, then the *kernel* of g is the equivalence relation E on C where $(c, c') \in E$ if $g(c) = g(c')$. For $c \in C$, we denote by c/E the equivalence class of c in E, and by C/E the set of all equivalence classes of elements of C. The *index* of E is defined to be $|C/E|$.

Definition 4.8. Let σ and τ be relational signatures, let $\mathfrak{A}$ be a τ-structure, and let $\mathfrak{B}$ be a σ-structure. A *primitive positive interpretation* I of $\mathfrak{B}$ in $\mathfrak{A}$ consists of

- a natural number d, called the *dimension* of I,
- a primitive positive τ-formula $\delta_I(x_1, \dots, x_d)$, called the *domain formula*,
- for each atomic σ-formula $\phi(y_1, \dots, y_k)$ a primitive positive τ-formula $\phi_I(\overline{x}_1, \dots, \overline{x}_k)$, called the *defining formulas*, and
- the *coordinate map*: a surjective map $h\colon D \to B$ where

$$D := \{(a_1, \dots, a_d) \in A^d \mid \mathfrak{A} \models \delta_I(a_1, \dots, a_d)\}$$

such that for all atomic σ-formulas ϕ and all tuples $\overline{a}_i \in D$

$$\mathfrak{B} \models \phi(h(\overline{a}_1), \dots, h(\overline{a}_k)) \;\Leftrightarrow\; \mathfrak{A} \models \phi_I(\overline{a}_1, \dots, \overline{a}_k)\ .$$

Note that the dimension d, the set D, and the coordinate map h determine the defining formulas up to logical equivalence; hence, we sometimes denote an interpretation by $I = (d, D, h)$. Note that the kernel of h coincides with the relation defined by $(y_1 = y_2)_I$, for which we also write $=_I$, the defining formula for equality. Also note that the structures $\mathfrak{A}$ and $\mathfrak{B}$ and the coordinate map determine the defining formulas of the interpretation up to logical equivalence. Sometimes, the same symbol is used for the interpretation I and the coordinate map.

Example 10. Let G be a graph and let F be an equivalence relation on $V(G)$. Then G/F is the graph whose vertices are the equivalence classes of F, and where S and T are adjacent if there are $s \in S$ and $t \in T$ such that $\{s, t\} \in E(G)$. If F has a primitive positive definition in G, then G/F has a primitive positive interpretation in G. △

Example 11. The field of rational numbers $(\mathbb{Q}; 0, 1, +, *)$ has a primitive positive 2-dimensional interpretation I in $(\mathbb{Z}; 0, 1, +, *)$. Example 9 presented a primitive positive definition $\phi(x)$ of the set of non-negative integers. The interpretation is now given as follows.

- The domain formula $\delta_I(x, y)$ is $y \geq 1$ (using $\phi(x)$, it is straightforward to express this with a primitive positive formula);
- The formula $=_I (x_1, y_1, x_2, y_2)$ is $x_1 y_2 = x_2 y_1$;
- The formula $0_I(x, y)$ is $x = 0$, the formula $1_I(x, y)$ is $x = y$;
- The formula $+_I(x_1, y_1, x_2, y_2, x_3, y_3)$ is $y_3 * (x_1 * y_2 + x_2 * y_1) = x_3 * y_1 * y_2$;
- The formula $*_I(x_1, y_1, x_2, y_2, x_3, y_3)$ is $x_1 * x_2 * y_3 = x_3 * y_1 * y_2$. △

Theorem 4.9. *Let $\mathfrak{B}$ and $\mathfrak{C}$ be structures with finite relational signatures. If there is a primitive positive interpretation of $\mathfrak{B}$ in $\mathfrak{C}$, then there is a polynomial-time reduction from* CSP($\mathfrak{B}$) *to* CSP($\mathfrak{C}$).

Proof. Let d be the dimension of the primitive positive interpretation I of the τ-structure $\mathfrak{B}$ in the σ-structure $\mathfrak{C}$, let $\delta_I(x_1, \ldots, x_d)$ be the domain formula, and let $h \colon \delta_I(\mathfrak{C}^d) \to D(\mathfrak{B})$ be the coordinate map. Let ϕ be an instance of CSP($\mathfrak{B}$) with variable set $U = \{x_1, \ldots, x_n\}$. We construct an instance ψ of CSP($\mathfrak{C}$) as follows. For distinct variables $V := \{y_1^1, \ldots, y_n^d\}$, we set ψ_1 to be the formula

$$\bigwedge_{1 \leq i \leq n} \delta_I(y_i^1, \ldots, y_i^d) \, .$$

Let ψ_2 be the conjunction of the formulas $\theta_I(y_{i_1}^1, \ldots, y_{i_1}^d, \ldots, y_{i_k}^1, \ldots, y_{i_k}^d)$ over all conjuncts $\theta = R(x_{i_1}, \ldots, x_{i_k})$ of ϕ. By moving existential quantifiers to the front, the sentence

$$\exists y_1^1, \ldots, y_n^d \, (\psi_1 \wedge \psi_2)$$

can be re-written to a primitive positive σ-formula ψ, and clearly ψ can be constructed in polynomial time in the size of ϕ.

We claim that ϕ is true in $\mathfrak{B}$ if and only if ψ is true in $\mathfrak{C}$. Suppose that $f \colon V \to C$ satisfies all conjuncts of ψ in $\mathfrak{C}$. Hence, by construction of ψ, if ϕ has a conjunct $\theta = R(x_{i_1}, \ldots, x_{i_k})$, then

$$\mathfrak{C} \models \theta_I\big((f(y_{i_1}^1), \ldots, f(y_{i_1}^d)), \ldots, (f(y_{i_k}^1), \ldots, f(y_{i_k}^d))\big) \, .$$

By the definition of interpretations, this implies that

$$\mathfrak{B} \models R\big(h(f(y_{i_1}^1), \ldots, f(y_{i_1}^d)), \ldots, h(f(y_{i_k}^1), \ldots, f(y_{i_k}^d))\big) \, .$$

Hence, the mapping $g \colon U \to B$ that sends x_i to $h(f(y_i^1), \ldots, f(y_i^d))$ satisfies all conjuncts of ϕ in $\mathfrak{B}$.

Now, suppose that $f \colon U \to B$ satisfies all conjuncts of ϕ over $\mathfrak{B}$. Since h is a surjective mapping from $\delta_I(\mathfrak{C}^d)$ to B, there are elements $c_i^1, \ldots, c_i^d$ in $\mathfrak{C}$ such that $h(c_i^1, \ldots, c_i^d) = f(x_i)$, for all $i \in \{1, \ldots, n\}$. We claim that the mapping $g \colon V \to C$ that maps y_i^j to c_i^j satisfies ψ in $\mathfrak{C}$. By construction, any constraint in ψ either comes from ψ_1 or from ψ_2. If it comes from ψ_1 then it must be of the form $\delta_I(y_i^1, \ldots, y_i^d)$, and is satisfied since the pre-image of h is $\delta_I(\mathfrak{C}^d)$. If the constraint comes from ψ_2, then it must be a conjunct of a formula $\theta_I(y_{i_1}^1, \ldots, y_{i_1}^d, \ldots, y_{i_k}^1, \ldots, y_{i_k}^d)$ that was introduced for a constraint $\theta = R(x_{i_1}, \ldots, x_{i_k})$ in ϕ. It therefore suffices to show that

$$\mathfrak{C} \models \theta_I\big(g(y_{i_1}^1), \ldots, g(y_{i_1}^d), \ldots, g(y_{i_k}^1), \ldots, g(y_{i_k}^d)\big) \, .$$

By assumption, $R(f(x_{i_1}), \dots, f(x_{i_k}))$ holds in $\mathfrak{B}$. By the choice of $c_1^1, \dots, c_n^d$, this shows that $R(h(c_{i_1}^1, \dots, c_{i_1}^d), \dots, h(c_{i_k}^1, \dots, c_{i_k}^d))$ holds in $\mathfrak{C}$. By the definition of interpretations, this is the case if and only if $\theta_I(c_{i_1}^1, \dots, c_{i_1}^d, \dots, c_{i_k}^1, \dots, c_{i_k}^d)$ holds in $\mathfrak{C}$, which is what we had to show. □

In many hardness proofs we use Theorem 4.9 in the following way.

Corollary 4.10. *Let $\mathfrak{B}$ be a finite relational structure. If there is a primitive positive interpretation of K_3 in $\mathfrak{B}$, then* CSP($\mathfrak{B}$) *is NP-hard.*

Proof. This is a direct consequence of Theorem 4.9 and the fact that CSP(K_3) is NP-hard (see, e.g., [36]). □

Indeed, K_3 is one of the most expressive finite structures, in the following sense.

Theorem 4.11. *If $n \geq 3$ then every finite structure has a primitive positive interpretation in K_n.*

Proof. Let $\mathfrak{A}$ be a finite τ-structure with the domain $A = \{1, \dots, k\}$. Our interpretation I of $\mathfrak{A}$ in K_n is $2k$-dimensional. The domain formula $\delta_I(x_1, \dots, x_k, x'_1, \dots, x'_k)$ expresses that for exactly one $i \leq k$ we have $x_i = x'_i$. Note that this formula is preserved by all permutations of $\{1, \dots, k\}$. We will see in Proposition 5.12 that every such formula is equivalent to a primitive positive formula over K_n. Equality is interpreted by the formula

$$=_I (x_1, \dots, x_k, x'_1, \dots, x'_k, y_1, \dots, y_k, y'_1, \dots, y'_k) := \bigwedge_{i=1}^{k} \big((x_i = x'_i) \Leftrightarrow (y_i = y'_i)\big)$$

Note that $=_I$ defines an equivalence relation on the set of all $2k$-tuples $(u_1, \dots, u_k, u'_1, \dots, u'_k)$ that satisfy δ_I. The coordinate map sends this tuple to i if and only if $u_i = u'_i$. When $R \in \tau$ is m-ary, then the formula $R(x_1, \dots, x_m)_I$ is any primitive positive formula which is equivalent to the following disjunction of conjunctions with $2mk$ variables $x_{1,1}, \dots, x_{m,k}, x'_{1,1}, \dots, x'_{m,k}$: for each tuple $(t_1, \dots, t_m) \in R^{\mathfrak{A}}$ the disjunction contains the conjunct $\bigwedge_{i \leq m} x_{i,t_i} = x'_{i,t_i}$; again, Proposition 5.12 implies that such a primitive positive formula exists. □

Primitive positive interpretations can be composed: if

- $\mathfrak{C}_1$ has a d_1-dimensional pp-interpretation I_1 in $\mathfrak{C}_2$, and
- $\mathfrak{C}_2$ has an d_2-dimensional pp-interpretation I_2 in $\mathfrak{C}_3$,

then $\mathfrak{C}_1$ has a natural $(d_1 d_2)$-dimensional pp-interpretation in $\mathfrak{C}_3$, which we denote by $I_1 \circ I_2$. To formally describe $I_1 \circ I_2$, suppose that the signature of $\mathfrak{C}_i$ is τ_i for $i = 1, 2, 3$, and that $I_1 = (d_1, S_1, h_1)$ and $I_2 = (d_2, S_2, h_2)$. When ϕ is a primitive positive τ_2-formula, let ϕ_{I_2} denote the τ_3-formula obtained from ϕ by replacing each atomic τ_2-formula ψ in ϕ by the τ_3-formula ψ_{I_2}. Note that ϕ_{I_2} is again primitive positive. The coordinate map of $I_1 \circ I_2$ is defined by

$$(a_1^1, \dots, a_{d_2}^1, \dots, a_1^{d_1}, \dots, a_{d_2}^{d_1}) \;\mapsto\; h_1\big(h_2(a_1^1, \dots, a_{d_2}^1), \dots, h_2(a_1^{d_1}, \dots, a_{d_2}^{d_1})\big) \,.$$

Two pp-interpretations I_1 and I_2 of $\mathfrak{B}$ in $\mathfrak{A}$ are called *homotopic*[3] if the relation

$$\{(\bar{x}, \bar{y}) \mid I_1(\bar{x}) = I_2(\bar{y})\}$$

of arity $d_1 + d_2$ is pp-definable in $\mathfrak{A}$. Note that id_C is a pp-interpretation of $\mathfrak{C}$ in $\mathfrak{C}$, called the *identity interpretation* of $\mathfrak{C}$ (in $\mathfrak{C}$).

Definition 4.12. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ with an interpretation I of $\mathfrak{B}$ in $\mathfrak{A}$ and an interpretation J of $\mathfrak{A}$ in $\mathfrak{B}$ are called *mutually pp-interpretable*. If both $I \circ J$ and $J \circ I$ are homotopic to the identity interpretation (of $\mathfrak{A}$ and of $\mathfrak{B}$, respectively), then we say that $\mathfrak{A}$ and $\mathfrak{B}$ are *primitively positively bi-interpretable (via I and J)*.

4.7 Reduction to Binary Signatures

In this section we prove that every structure $\mathfrak{C}$ with a relational signature of maximal arity $m \in \mathbb{N}$ is primitively positively bi-interpretable with a *binary structure* $\mathfrak{B}$, i.e., a relational structure where every relation symbol has arity at most two. Moreover, if $\mathfrak{C}$ has a finite signature, then $\mathfrak{B}$ can be chosen to have a finite signature, too. It follows from Theorem 4.9 that every CSP is polynomial-time equivalent to a binary CSP. This transformation is known under the name *dual encoding* [25, 29]. We want to stress that the transformation works for relational structures with domains of arbitrary cardinality.

A d-dimensional primitive positive interpretation I of $\mathfrak{B}$ in $\mathfrak{A}$ is called *full* if for every $R \subseteq B^k$ we have that R is primitively positively definable in $\mathfrak{B}$ if and only if the relation $I^{-1}(R)$ of arity kd is primitively positively definable in $\mathfrak{A}$. Note that every structure with a primitive positive interpretation in $\mathfrak{A}$ is a reduct of a structure with a full primitive positive interpretation in $\mathfrak{A}$. Full interpretations play an important role in Section 7.3 since they are the counterpart for certain standard algebraic constructions that will be studied later.

Definition 4.13. Let $\mathfrak{C}$ be a structure and $d \in \mathbb{N}$. Then a *d-th full power of* $\mathfrak{C}$ is a structure $\mathfrak{D}$ with domain C^d such that the identity map on C^d is a full d-dimensional primitive positive interpretation of $\mathfrak{D}$ in $\mathfrak{C}$.

In particular, for all $i, j \in \{1, \dots, d\}$ the relation

$$E_{i,j} := \big\{((x_1, \dots, x_d), (y_1, \dots, y_d)) \mid x_1, \dots, x_d, y_1, \dots, y_d \in C \text{ and } x_i = y_j\big\}$$

is primitively positively definable in $\mathfrak{D}$.

Proposition 4.14. *Let $\mathfrak{C}$ be a structure and $\mathfrak{D}$ a d-th full power of $\mathfrak{C}$ for $d \geq 1$. Then $\mathfrak{C}$ and $\mathfrak{D}$ are primitively positively bi-interpretable.*

Proof. Let I be the identity map on C^d which is a full interpretation of $\mathfrak{D}$ in $\mathfrak{C}$. Our interpretation J of $\mathfrak{C}$ in $\mathfrak{D}$ is one-dimensional and the coordinate map is the first projection. The domain formula is *true* and the pre-image of the equality relation in $\mathfrak{C}$ under the coordinate map has the primitive positive definition $E_{1,1}(x, y)$. To define the pre-image of a k-ary relation R of $\mathfrak{C}$ under the coordinate map it suffices to observe that the k-ary relation

$$S := \big\{((a_{1,1}, \dots, a_{1,d}), \dots, (a_{k,1}, \dots, a_{k,d})) \mid (a_{1,1}, \dots, a_{k,1}) \in R\big\}$$

[3] We follow the terminology from [2].

is primitively positively definable in $\mathfrak{D}$ and $J(S) = R$.

To show that $\mathfrak{C}$ and $\mathfrak{D}$ are primitively positive bi-interpretable we prove that $I \circ J$ and $J \circ I$ are pp-homotopic to the identity interpretation. The relation

$$\{(u_0, u_1, \ldots, u_k) \mid u_0 = I(J(u_1), \ldots, J(u_k)), u_1, \ldots, u_k \in C^{k+1}\}$$

has the primitive positive definition $\bigwedge_{i \in \{1,\ldots,k\}} E_{i,1}(u_0, u_i)$ and the relation

$$\{(v_0, v_1, \ldots, v_k) \mid v_0 = J(I(v_1, \ldots, v_k)), v_1, \ldots, v_k \in D^{k+1}\}$$

has the primitive positive definition $v_0 = v_1$. □

Note that for every relation R of arity $k \leq d$ of $\mathfrak{C}$, in a d-th full power $\mathfrak{D}$ of $\mathfrak{C}$ the unary relation

$$R' := \{(a_1, \ldots, a_d) \mid (a_1, \ldots, a_k) \in R\}$$

must be primitively positively definable. We now define a particular full power.

Definition 4.15. Let $\mathfrak{C}$ be a relational structure with maximal arity m and let $d \geq m$. Then the structure $\mathfrak{B} := \mathfrak{C}^{[d]}$ with domain C^d is defined as follows:

- for every relation $R \subseteq C^k$ of $\mathfrak{C}$ the structure $\mathfrak{B}$ has the unary relation $R' \subseteq B = C^d$ defined above, and
- for all $i, j \in \{1, \ldots, d\}$ the structure $\mathfrak{B}$ has the binary relation symbol $E_{i,j}$.

It is clear that the signature of $\mathfrak{B}$ is finite if the signature of $\mathfrak{C}$ is finite. Also note that the signature of $\mathfrak{C}^{[d]}$ is always binary.

Lemma 4.16. *Let $\mathfrak{C}$ be a relational structure with maximal arity m and let $d \geq m$. Then the binary structure $\mathfrak{C}^{[d]}$ is a full power of $\mathfrak{C}$.*

Proof. The identity map is a d-dimensional primitive positive interpretation I of $\mathfrak{B} := \mathfrak{C}^{[d]}$ in $\mathfrak{C}$. Our interpretation J of $\mathfrak{C}$ in $\mathfrak{B}$ is one-dimensional and the coordinate map is the first projection. The domain formula is *true* and the pre-image of the equality relation in $\mathfrak{C}$ under the coordinate map has the primitive positive definition $E_{1,1}(x, y)$. The pre-image of the relation R of $\mathfrak{C}$ under the coordinate map is defined by the primitive positive formula

$$\exists y\big(\bigwedge_{i \in \{1,\ldots,k\}} E_{1,i}(x_i, y) \wedge R'(y)\big).$$

The proof that $I \circ J$ and $J \circ I$ are pp-homotopic to the identity interpretation is as in the proof of Proposition 4.14. □

Corollary 4.17. *For every structure $\mathfrak{C}$ with maximal arity m there exists a structure $\mathfrak{B}$ with maximal arity 2 such that $\mathfrak{B}$ and $\mathfrak{C}$ are primitively positively bi-interpretable. If the signature of $\mathfrak{C}$ is finite, then the signature of $\mathfrak{B}$ can be chosen to be finite, too.*

Proof. An immediate consequence of Lemma 4.16 and Proposition 4.14. □

4.8 The Structure-Building Operators H, C, and I

In the previous three sections we have seen several conditions on $\mathfrak{A}$ and $\mathfrak{B}$ that imply that CSP($\mathfrak{A}$) reduces to CSP($\mathfrak{B}$); in this section we compare them. Let $\mathcal{C}$ be a class of finite structures. We write

1. H($\mathcal{C}$) for the class of structures homomorphically equivalent to structures in $\mathcal{C}$.
2. C($\mathcal{C}$) for the class of all structures obtained by expanding a core structure in $\mathcal{C}$ by singleton relations $\{a\}$. In the setting of relational structures, they play the role of constants (which formally are operation symbols of arity 0).
3. I($\mathcal{C}$) for the class of all structures with a primitive positive interpretation in a structure from $\mathcal{C}$.

Let $\mathcal{D}$ be the smallest class containing $\mathcal{C}$ and closed under H, C, and I. Barto, Opršal, and Pinsker [7] showed that $\mathcal{D} = \mathrm{H}(\mathrm{I}(\mathcal{C}))$. In other words, if there is a chain of applications of the three operators H, C, and I to derive $\mathfrak{A}$ from $\mathfrak{B}$, then there is also a two-step chain to derive $\mathfrak{A}$ from $\mathfrak{B}$, namely by interpreting a structure $\mathfrak{B}'$ that is homomorphically equivalent to $\mathfrak{A}$. This insight is conceptually important for the CSP since it leads to a better understanding of the power of the available tools.

Proposition 4.18 (from [7]). *Suppose that $\mathfrak{B}$ is a core, and that $\mathfrak{C}$ is the expansion of $\mathfrak{B}$ by a relation of the form $\{c\}$ for $c \in B$. Then $\mathfrak{C}$ is homomorphically equivalent to a structure with a primitive positive interpretation in $\mathfrak{B}$. In symbols,*

$$C(\mathcal{C}) \subseteq \mathrm{H}(\mathrm{I}(\mathcal{C}))\,.$$

Proof. By Proposition 4.6, the orbit O of c has a primitive positive definition $\phi(x)$ in $\mathfrak{B}$. We give a 2-dimensional primitive positive interpretation of a structure $\mathfrak{A}$ with the same signature τ as $\mathfrak{C}$. The domain formula $\delta_I(x_1, x_2)$ for $\mathfrak{A}$ is $\phi(x_2)$. If $R \in \tau$ is from the signature of $\mathfrak{B}$ and has arity k then

$$R^{\mathfrak{A}} := \{((a_1, b_1), \ldots, (a_k, b_k)) \in A^k \mid (a_1, \ldots, a_k) \in R^{\mathfrak{B}} \text{ and } b_1 = \cdots = b_k \in O\}.$$

Otherwise, $R^{\mathfrak{C}}$ is of the form $\{c\}$ and we define $R^{\mathfrak{A}} := \{(a, a) \mid a \in O\}$. It is clear that $\mathfrak{A}$ has a primitive positive interpretation in $\mathfrak{B}$.

We claim that $\mathfrak{A}$ and $\mathfrak{C}$ are homomorphically equivalent. The homomorphism from $\mathfrak{C}$ to $\mathfrak{A}$ is given by $a \mapsto (a, c)$:

- if $(a_1, \ldots, a_k) \in R^{\mathfrak{C}} = R^{\mathfrak{B}}$ then $((a_1, c), \ldots, (a_k, c)) \in R^{\mathfrak{A}}$;
- the relation $R^{\mathfrak{C}} = \{c\}$ is preserved since $(c, c) \in R^{\mathfrak{A}}$.

To define a homomorphism h from $\mathfrak{A}$ to $\mathfrak{C}$ we pick for each $a \in O$ an automorphism $\alpha_a \in \mathrm{Aut}(\mathfrak{B})$ such that $\alpha_a(a) = c$. Note that $b \in O$ since $\mathfrak{B} \models \delta(a, b)$, and we define $h(a, b) := \alpha_b(a)$. To check that this is indeed a homomorphism, let $R \in \tau$ be k-ary, and let $t = ((a_1, b_1), \ldots, (a_k, b_k)) \in R^{\mathfrak{A}}$. Then $b_1 = \cdots = b_k =: b \in O$ and we have that $h(t) = (\alpha_b(a_1), \ldots, \alpha_b(a_k))$ is in $R^{\mathfrak{C}}$ since $(a_1, \ldots, a_k) \in R^{\mathfrak{B}} = R^{\mathfrak{C}}$ and α_b preserves $R^{\mathfrak{B}} = R^{\mathfrak{C}}$. If $R^{\mathfrak{A}} = \{(a, a) \mid a \in O\}$, then R is preserved as well, because $h((a, a)) = \alpha_a(a) = c \in \{c\} = R^{\mathfrak{C}}$. □

Theorem 4.19 (from [7]). *Suppose that $\mathfrak{A}$ can be obtained from $\mathcal{C}$ by applying H, C, and* I. *Then $\mathfrak{A} \in \mathrm{H}(\mathrm{I}(\mathcal{C}))$. That is, $\mathfrak{A}$ is homomorphically equivalent to a structure with a primitive positive interpretation in a structure from $\mathcal{C}$.*

Proof. We have to show that $\mathrm{H}(\mathrm{I}(\mathcal{C}))$ is closed under H, C, and I. Homomorphic equivalence is transitive so $\mathrm{H}(\mathrm{H}(\mathcal{C})) \subseteq \mathrm{H}(\mathcal{C})$.

We show that if $\mathfrak{A}$ and $\mathfrak{B}$ are homomorphically equivalent, and $\mathfrak{C}$ has a d-dimensional primitive positive interpretation I_1 in $\mathfrak{B}$, then $\mathfrak{C}$ is homomorphically equivalent to a structure $\mathfrak{D}$ with a d-dimensional primitive positive interpretation I_2 in $\mathfrak{A}$. Let $h_1 \colon A \to B$ be the homomorphism from $\mathfrak{A}$ to $\mathfrak{B}$, and h_2 the homomorphism from $\mathfrak{B}$ to $\mathfrak{A}$. The interpreting formulas of I_2 are the same as the interpreting formulas of I_1; this describes the structure $\mathfrak{D}$ up to isomorphism. We claim that the map $g_1(I_2(a_1, \dots, a_d)) := I_1(h_1(a_1), \dots, h_1(a_d))$ is a homomorphism from $\mathfrak{D}$ to $\mathfrak{C}$. Indeed, for a k-ary relation symbol from the signature of $\mathfrak{C}$ and $\mathfrak{D}$, let $((a_1^1, \dots, a_d^1), \dots, (a_1^k, \dots, a_d^k)) \in R^{\mathfrak{D}}$; hence, the dk-tuple $(a_1^1, \dots, a_d^1, \dots, a_1^k, \dots, a_d^k)$ satisfies the primitive positive defining formula for $R(x_1^1, \dots, x_d^k)$, and

$$(h_1(a_1^1), \dots, h_1(a_d^1), \dots, h_1(a_1^k), \dots, h_1(a_d^k))$$

satisfies this formula, too. This in turn implies that

$$(I_1(h_1(a_1^1), \dots, h_1(a_d^1)), \dots, I_1(h_1(a_1^k), \dots, h_1(a_d^k))) \in R^{\mathfrak{C}}.$$

Similarly, $g_2(I_1(b_1, \dots, b_d)) := I_2(h_2(b_1), \dots, h_2(b_d))$ is a homomorphism from $\mathfrak{C}$ to $\mathfrak{D}$. So we conclude that

$$\begin{aligned} \mathrm{I}(\mathrm{H}(\mathrm{I}(\mathcal{C}))) &\subseteq \mathrm{H}(\mathrm{I}(\mathrm{I}(\mathcal{C}))) \\ &\subseteq \mathrm{H}(\mathrm{I}(\mathcal{C})) \end{aligned}$$

because primitive positive interpretability is transitive, too. Finally, Proposition 4.18 shows that

$$\begin{aligned} \mathrm{C}(\mathrm{H}(\mathrm{I}(\mathcal{C}))) &\subseteq \mathrm{H}(\mathrm{I}(\mathrm{H}(\mathrm{I}(\mathcal{C})))) \\ &\subseteq \mathrm{H}(\mathrm{I}(\mathcal{C}) \end{aligned}$$

where the last inclusion again follows from the observations above. □

There are finite idempotent structures $\mathfrak{B}$ that show that $\mathrm{H}(\mathrm{I}(\mathfrak{B}))$ can be strictly larger than $\mathrm{I}(\mathfrak{B})$.

Example 12. Let $\mathfrak{B}$ be the structure with domain $(\mathbb{Z}_2)^2$ and signature $\{R_{a,b} \mid a, b \in \mathbb{Z}_2\}$ such that

$$R_{a,b}^{\mathfrak{B}} := \{(x, y, z) \in ((\mathbb{Z}_2)^2)^3 \mid x + y + z = (a, b)\}.$$

Let $\mathfrak{B}'$ be the reduct of $\mathfrak{B}$ with the signature $\tau := \{R_{0,0}, R_{1,0}\}$. Let $\mathfrak{A}$ be the τ-structure with domain $\mathbb{Z}_2$ such that for $a = 0$ and $a = 1$

$$R_{a,0}^{\mathfrak{A}} := \{(x, y, z) \in (\mathbb{Z}_2)^3 \mid x + y + z = a\}.$$

Now observe that

- $(x_1, x_2) \mapsto x_1$ is a homomorphism from $\mathfrak{B}'$ to $\mathfrak{A}$, and $x \mapsto (x, 0)$ is a homomorphism from $\mathfrak{A}$ to $\mathfrak{B}'$. Therefore $\mathfrak{A} \in \mathrm{H}(\mathfrak{B}')$.
- Trivially, $\mathfrak{B}' \in \mathrm{I}(\mathfrak{B})$ and consequently $\mathfrak{A} \in \mathrm{H}(\mathrm{I}(\mathfrak{B}))$.
- All polymorphisms of $\mathfrak{B}$ are idempotent.

We finally show that $\mathfrak{A} \notin \mathrm{I}(\mathfrak{B})$. Suppose for contradiction that there is a pp-interpretation of $\mathfrak{A}$ in $\mathfrak{B}$ with coordinate map $c\colon C \to A$ where $C \subseteq B^n$ is primitive positive definable in $\mathfrak{B}$. The kernel K of c has a primitive positive definition ϕ in $\mathfrak{B}$. The two equivalence classes of K are pp-definable relations over $\mathfrak{B}$, too: the formula $\exists x(\phi(x, y) \wedge R_{a,b}(x))$ defines the equivalence class of (a, b). But the relations with a primitive positive definition in $\mathfrak{B}$ are precisely affine linear subspaces of the vector space $(\mathbb{Z}_2)^2$, so their cardinality must be a power of 4. And two powers of 4 cannot add up to a power of 4. △

We will revisit primitive positive interpretations in Section 7 where we study them from a universal-algebraic perspective.

5 Relations and Operations

5.1 Operation Clones

For $n \geq 1$ and a set D (the *domain*), denote by $\mathscr{O}_D^{(n)}$ the set $D^{D^n} := (D^n \to D)$ of n-ary functions on D. The elements of $\mathscr{O}_D^{(n)}$ will typically be called the *operations* of arity n on D, and D will be called the *domain*. The set of all operations on D of finite arity will be denoted by $\mathscr{O}_D := \bigcup_{n \geq 1} \mathscr{O}_D^{(n)}$. An *operation clone* (over D) is a subset $\mathscr{C}$ of $\mathscr{O}_D$ satisfying the following two properties:

- $\mathscr{C}$ contains all projections, that is, for all $1 \leq k \leq n$ it contains the operation $\pi_k^n \in \mathscr{O}_D^{(n)}$ defined by $\pi_k^n(x_1, \ldots, x_n) = x_k$, and
- $\mathscr{C}$ is *closed under composition*, that is, for all $f \in \mathscr{C} \cap \mathscr{O}_D^{(n)}$ and $g_1, \ldots, g_n \in \mathscr{C} \cap \mathscr{O}_D^{(m)}$ it contains the operation $f(g_1, \ldots, g_n) \in \mathscr{O}_D^{(m)}$ defined by

$$(x_1, \ldots, x_m) \mapsto f(g_1(x_1, \ldots, x_m), \ldots, g_n(x_1, \ldots, x_m)) \,.$$

A *clone* is an abstraction of an operation clone that will be introduced later in the course. In the literature, operation clones are often called clones, or *concrete clones*; we prefer to use the terms 'operation clone' and 'clone' in analogy to 'permutation group' and 'group'.

If $\mathscr{C}$ is an operation clone, then $\mathscr{C}'$ is called a *subclone* of $\mathscr{C}$ if $\mathscr{C}'$ is an operation clone and $\mathscr{C}' \subseteq \mathscr{C}$. If $\mathscr{F}$ is a set of functions, we write $\langle \mathscr{F} \rangle$ for the smallest operation clone $\mathscr{C}$ which contains $\mathscr{F}$, and call $\mathscr{C}$ the clone *generated* by $\mathscr{F}$. Note that the set of all clones over a set B forms a lattice: the meet of two operation clones $\mathscr{C}$ and $\mathscr{D}$ is their intersection $\mathscr{C} \cap \mathscr{D}$ (which is again a clone!); the join of $\mathscr{C}$ and $\mathscr{D}$ is the clone generated by their union, $\langle \mathscr{C} \cup \mathscr{D} \rangle$.

5.2 Inv-Pol

The most important source of operation clones in this text are *polymorphism clones* of digraphs and, more generally, structures. For simplicity, we only discuss relational structures; the step to structures that also involve function symbols is straightforward.

Let f be from $\mathscr{O}_B^{(n)}$, and let $R \subseteq B^m$ be a relation. Then we say that f *preserves* R (and that R is *invariant under* f) if $f(r_1, \ldots, r_n) \in R$ whenever $r_1, \ldots, r_n \in R$, where $f(r_1, \ldots, r_n)$ is calculated componentwise. If $\mathfrak{B}$ is a relational structure with domain B then $\mathrm{Pol}(\mathfrak{B})$ contains precisely those operations that preserve $\mathfrak{B}$. It is easy to verify that $\mathrm{Pol}(\mathfrak{B})$ is an operation clone. Conversely, if $\mathscr{F}$ is a set of operations on B, then we write $\mathrm{Inv}(\mathscr{F})$ for the set of all relations on B that are invariant under all functions in $\mathscr{F}$. It will be convenient to define the operator Pol also for sets $\mathcal{R}$ of relations over B, writing $\mathrm{Pol}(\mathcal{R})$ for the set of operations of $\mathscr{O}_B$ that preserve all relations from $\mathcal{R}$.

Proposition 5.1. *Let $\mathfrak{B}$ be any relational structure. Then* $\mathrm{Inv}(\mathrm{Pol}(\mathfrak{B}))$ *contains the set of all relations that are primitive positive definable in $\mathfrak{B}$.*

Proof. Suppose that R is k-ary, has a primitive positive definition $\psi(x_1, \ldots, x_k)$, and let f be an l-ary polymorphism of $\mathfrak{B}$. To show that f preserves R, let $t_1, \ldots, t_l$ be k-tuples from R. Let $x_{k+1}, \ldots, x_n$ be the existentially quantified variables of ψ. Write s_i for the n-tuple which extends the k-tuple t_i such that s_i satisfies the quantifier-free part $\psi'(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n)$ of ψ. Then the tuple $f(s_1, \ldots, s_l)$ satisfies ψ' since f is a polymorphism. This shows that $\mathfrak{B} \models \psi(f(t_1, \ldots, t_l))$ which is what we had to show. □

Theorem 5.2 (of [14,37])**.** *Let $\mathfrak{B}$ be a finite relational structure. A relation R has a primitive positive definition in $\mathfrak{B}$ if and only if R is preserved by all polymorphisms of $\mathfrak{B}$.*

Proof. One direction has been shown in Proposition 5.1. For the other direction, let

$$(a_1^1, \ldots, a_k^1), \ldots, (a_1^w, \ldots, a_k^w)$$

be an enumeration of R. Let $b_1, b_2, \ldots, b_\ell$ be an enumeration of B^w. Let ψ be the canonical query of $\mathfrak{B}^w$ (see Section 4.2). We claim that the primitive positive formula $\psi'(x_1, \ldots, x_k)$ obtained from ψ by adding the conjuncts $x_i = b_j$ if $(a_i^1, \ldots, a_i^w) = b_j$ is a primitive positive definition of R. Indeed, for every $(a_1, \ldots, a_k) \in R$ there exists a $j \in \{1, \ldots, w\}$ such that $(a_1, \ldots, a_k) = (b_1^j, \ldots, b_k^j)$. Then the elements $b_1^j, \ldots, b_\ell^j$ provide witnesses for the existentially quantified variables showing that $(a_1, \ldots, a_k) = (b_1^j, \ldots, b_k^j)$ satisfies the formula.

Conversely, suppose that $(a_1, \ldots, a_k)$ satisfies ψ'. Then $(a_1, \ldots, a_k) = (b_1^j, \ldots, b_k^j)$ for some $j \leq \ell$, and ψ' contains the conjuncts $x_i = b_j$. The witnesses for the existentially quantified variables $b_1, \ldots, b_\ell$ define a homomorphism f from $\mathfrak{B}^w$ to $\mathfrak{B}$. Then f is a polymorphism of $\mathfrak{B}$ and by assumption preserves R. Since the tuples $(a_1^1, \ldots, a_k^1), \ldots, (a_1^w, \ldots, a_k^w)$ are from R and $f((a_1^1, \ldots, a_k^1), \ldots, (a_1^w, \ldots, a_k^w)) = (a_1, \ldots, a_k)$, we obtain that $(a_1, \ldots, a_k) \in R$. □

Corollary 5.3. *The complexity of* $\mathrm{CSP}(\mathfrak{B})$ *only depends on* $\mathrm{Pol}(\mathfrak{B})$*. If $\mathfrak{C}$ is such that* $\mathrm{Pol}(\mathfrak{B}) \subseteq \mathrm{Pol}(\mathfrak{C})$*, then* $\mathrm{CSP}(\mathfrak{C})$ *reduces in polynomial time to* $\mathrm{CSP}(\mathfrak{B})$*.*

Proof. Direct consequence of Theorem 5.2 and Lemma 4.4. □

Exercises.

75. Let R_+, R_* be the relations as defined in Exercise 70. Show that R_* is not primitively positively definable in the structure $(\mathbb{Q}; R_+, \{(x, y) \mid y \geq x^2\})$.

5.3 Essentially Unary Clones

We say that an operation $f\colon B^k \to B$ is *essentially unary* if there is an $i \in \{1,\dots,k\}$ and a unary operation f_0 such that $f(x_1,\dots,x_k) = f_0(x_i)$ for all $x_1,\dots,x_k \in B$. Operations that are not essentially unary are called *essential*.[4] We say that f *depends on argument* i if there are $r,s \in B^k$ such that $f(r) \neq f(s)$ and $r_j = s_j$ for all $j \in \{1,\dots,k\} \setminus \{i\}$.

Lemma 5.4. *Let $f \in \mathscr{O}_B$ be an operation. Then the following are equivalent.*

1. *f is essentially unary.*
2. *f preserves $P_B^3 := \{(a,b,c) \in B^3 \mid a = b \text{ or } b = c\}$.*
3. *f preserves $P_B^4 := \{(a,b,c,d) \in B^4 \mid a = b \text{ or } c = d\}$.*
4. *f depends on at most one argument.*

Proof. Let k be the arity of f. The implication from (1) to (2) is obvious, since unary operations clearly preserve P_B^3.

To show the implication from (2) to (3), we show the contrapositive, and assume that f violates P_B^4. By permuting arguments of f, we can assume that there are 4-tuples $a^1,\dots,a^k \in P_B^4$ with $f(a^1,\dots,a^k) \notin P_B^4$ and $l \leq k$ such that in $a^1,\dots,a^l$ the first two coordinates are equal, and in $a^{l+1},\dots,a^k$ the last two coordinates are equal. Let $c := (a_1^1,\dots,a_1^l,a_4^{l+1},\dots,a_4^k)$. Since $f(a^1,\dots,a^k) \notin P_B^4$ we have $f(a_1^1,\dots,a_1^k) \neq f(a_2^1,\dots,a_2^k)$, and therefore $f(c) \neq f(a_1^1,\dots,a_1^1)$ or $f(c) \neq f(a_2^1,\dots,a_2^k)$. Let $d = (a_1^1,\dots,a_1^k)$ in the first case, and $d = (a_2^1,\dots,a_2^k)$ in the second case. Likewise, we have $f(c) \neq f(a_3^1,\dots,a_3^k)$ or $f(c) \neq f(a_4^1,\dots,a_4^k)$, and let $e = (a_3^1,\dots,a_3^k)$ in the first, and $e = (a_4^1,\dots,a_4^k)$ in the second case. Then for each $i \leq k$, the tuple (d_i,c_i,e_i) is from P_B^3, but $(f(d),f(c),f(e)) \notin P_B^3$.

The proof of the implication from (3) to (4) is again by contraposition. Suppose f depends on the i-th and j-th argument, $1 \leq i \neq j \leq k$. Hence there exist tuples $a_1,b_1,a_2,b_2 \in B^k$ such that a_1,b_1 and a_2,b_2 only differ at the entries i and j, respectively, and such that $f(a_1) \neq f(b_1)$ and $f(a_2) \neq f(b_2)$. Then $(a_1(l),b_1(l),a_2(l),b_2(l)) \in P_B^4$ for all $l \leq k$, but $(f(a_1),f(b_1),f(a_2),f(b_2)) \notin P_B^4$, which shows that f violates P_B^4.

For the implication from (4) to (1), suppose that f depends only on the first argument. Let $i \leq k$ be minimal such that there is an operation g with $f(x_1,\dots,x_k) = g(x_1,\dots,x_i)$. If $i = 1$ then f is essentially unary and we are done. Otherwise, observe that since f does not depend on the i-th argument, neither does g, and so there is an $(i-1)$-ary operation g' such that for all $x_1,\dots,x_n \in B$ we have $f(x_1,\dots,x_n) = g(x_1,\dots,x_i) = g'(x_1,\dots,x_{i-1})$, contradicting the choice of i. □

5.4 Minimal Clones

A *trivial* clone is a clone all of whose operations are projections. Note that it follows from Lemma 5.4 that for any set $B = \{b_1,\dots,b_n\}$ the clone $\mathrm{Pol}(B; P_B^4, \{b_1\},\dots,\{b_n\})$ is trivial.

Definition 5.5. A clone $\mathscr{C}$ is *minimal* if it is non-trivial, and for every non-trivial $\mathscr{E} \subseteq \mathscr{C}$ we have $\mathscr{E} = \mathscr{C}$.

[4] This is standard in clone theory, and it makes sense also when studying the complexity of CSPs, since the essential operations are those that are essential for complexity classification.

Recall that the smallest clone that contains a set of operations $\mathcal{F}$ is called the *clone generated by* $\mathcal{F}$, and denoted by $\langle\mathcal{F}\rangle$; if $g \in \langle\mathscr{F}\rangle$ and $\mathcal{F} = \{f\}$, then we say that f *generates* g.

Definition 5.6. An operation $f \in \mathscr{O}_B$ is *minimal* if f is not a projection and of minimal arity such that every g generated by f is either a projection or generates f.

The following is straightforward from the definitions.

Proposition 5.7. *Every minimal f generates a minimal clone, and every minimal clone contains a minimal operation.*

Theorem 5.8. *Every non-trivial operation clone $\mathscr{C} \subseteq \mathscr{O}_B$ over a finite set B contains a minimal operation.*

Proof. Consider the set of all clones contained in $\mathscr{C}$, partially ordered by inclusion. From this poset we remove the trivial clone; the resulting poset will be denoted by P. We use Zorn's lemma to show that P contains a minimal element. Observe that in P, all descending chains $\mathscr{C}_1 \supseteq \mathscr{C}_2 \supseteq \cdots$ are *bounded*, i.e., for all such chains there exists a $\mathscr{D} \in P$ such that $\mathscr{C}_i \supseteq \mathscr{D}$ for all $i \geq 1$. To see this, observe that the set $\bigcup_{i\geq 1} \mathrm{Inv}(\mathscr{C}_i)$ is *closed under primitive positive definability* in the sense that it is the set of relations that is primitively positively definable over some relational structure $\mathfrak{B}$ (since only a finite number of relations can be mentioned in a formula, and since $\mathrm{Inv}(\mathscr{C}_i)$ is closed under primitive positive definability, for each i). Moreover, one of the relations $P_B^4, \{b_1\}, \ldots, \{b_n\}$, for $B = \{b_1, \ldots, b_n\}$, is not contained in $\bigcup_{i\geq 1} \mathrm{Inv}(\mathscr{C}_i)$; otherwise, there would be a $j \in \mathbb{N}$ such that $\mathrm{Inv}(\mathscr{C}_j)$ contains all these relations, and hence $\mathscr{C}_j$ is the trivial clone contrary to our assumptions. Hence, $\mathrm{Pol}(\mathfrak{B})$ is a non-trivial lower bound of the descending chain $(\mathscr{C}_i)_{i\geq 0}$. By Zorn's lemma, P contains a minimal element, and this element contains a minimal operation in $\mathscr{C}$. □

Remark 5.9. Note that the statement above would be false if B is infinite: take for example the clone over the domain $B := \mathbb{N}$ of the integers generated by the operation $x \mapsto x + 1$. Every operation in this clone is essentially unary, and every unary operation in this clone is of the form $x \mapsto x + c$ for $c \in \mathbb{N}$. Note that for $c > 0$, the operation $x \mapsto x + c$ generates $x \mapsto x + 2c$, but not vice versa, so the clone does not contain a minimal operation.

In the remainder of this section, we show that a minimal operation has one out of the following five types, due to Rosenberg [56]. A k-ary operation f is a *semiprojection* if there exists an $i \leq k$ such that $f(x_1, \ldots, x_k) = x_i$ whenever $|\{x_1, \ldots, x_k\}| < k$.

Lemma 5.10 (Świerczkowski; Satz 4.4.6 in [54])**.** *Every at least 4-ary operation that turns into a projection whenever two arguments are the same is a semiprojection.*

Proof. Let f be an operation of arity at least $n \geq 4$ as in the statement of the lemma. By assumption there exists an $i \in \{1, 3, \ldots, n\}$ such that for all $x_1, x_3, x_4, \ldots, x_n$

$$f(x_1, x_1, x_3, x_4, \ldots, x_n) = x_i\,. \tag{1}$$

Let $k, \ell \in \{1, \ldots, n\}$ be distinct; we also assume that $k < \ell$ and $\ell \neq i$. By assumption, there exists $j \in \{1, \ldots, \ell - 1, \ell + 1, \ldots, n\}$ such that

$$f(x_1, \ldots, x_{\ell-1}, x_k, x_{\ell+1}, \ldots, x_n) = x_j\,. \tag{2}$$

We have to show that $j = i$. Consider the operation defined by

$$f(x_1, x_1, x_3, \dots, x_{k-1}, x_1, x_{k+1}, \dots, x_{\ell-1}, x_1, x_{\ell+1}, \dots, x_n) \tag{3}$$

It follows from (1) that this function returns x_1 if $i = k$, and x_i otherwise. Suppose for contradiction that $i = k \neq 1$. Note that in this case $j = 1$. Choose $p \in \{1, \dots, n\} \setminus \{1, \ell, k\}$ which exists because $n \geq 4$. For the sake of notation, suppose that $\ell < p$. Consider the projection defined by

$$f(x_1, x_2, \dots, x_{p-1}, x_\ell, x_{p+1}, \dots, x_n)\,. \tag{4}$$

Since $i \neq 1$ it follows from (1) that $f(x_\ell, x_\ell, x_3, \dots, x_{p-1}, x_\ell, x_{p+1}, \dots, x_n) = x_i$. Hence, the function in (4) must return x_i. On the other hand, (2) implies that

$$f(x_1, x_2, \dots, x_{l-1}, x_k, x_{l+1}, \dots, x_{p-1}, x_k, x_{p+1}, \dots, x_n) = x_j\,.$$

This implies that the function in (4) must return x_j since $j = 1 \neq k$. Since $i \neq 1 = j$, the projection in (4) cannot return both x_i and x_j, so we have reached a contradiction. Hence, the function in (3) returns x_i, which implies that $j = i$.

Note that the assumption that $k < \ell$ was just for the sake of notation; the case that $\ell < k$ can be shown analogously. This concludes the proof that f is the i-th semiprojection. □

Theorem 5.11. *Let f be a minimal operation. Then f has one of the following types:*

1. *A unary operation, which is either a permutation such that $f^p(x) = x$, for some prime p, or satisfies $f(f(x)) = f(x)$ for all x.*
2. *A binary idempotent operation.*
3. *A majority operation.*
4. *A minority operation.*
5. *A k-ary semiprojection, for $k \geq 3$, which is not a projection.*

Proof. The statement is easy to prove if f is unary, and clear if f is binary. If f is ternary, we have to show that f is majority, Maltsev, or a semiprojection. By minimality of f, the binary operation $f_1(x, y) := f(y, x, x)$ is a projection, that is, $f_1(x, y) = x$ or $f_1(x, y) = y$. Note that in particular $f(x, x, x) = x$. Similarly, the other operations $f_2(x, y) := f(x, y, x)$, and $f_3(x, y) := f(x, x, y)$ obtained by identifications of two variables must be projections. We therefore distinguish eight cases.

1. $f(y, x, x) = x, f(x, y, x) = x, f(x, x, y) = x$.
   In this case, f is a majority.
2. $f(y, x, x) = x, f(x, y, x) = x, f(x, x, y) = y$.
   In this case, f is a semiprojection.
3. $f(y, x, x) = x, f(x, y, x) = y, f(x, x, y) = x$.
   In this case, f is a semiprojection.
4. $f(y, x, x) = x, f(x, y, x) = y, f(x, x, y) = y$.
   The operation $g(x, y, z) := f(y, x, z)$ is a Maltsev operation.

5. $f(y,x,x) = y, f(x,y,x) = x, f(x,x,y) = x$.
   In this case, f is a semiprojection.

6. $f(y,x,x) = y, f(x,y,x) = x, f(x,x,y) = y$.
   In this case, f is a Maltsev operation.

7. $f(y,x,x) = y, f(x,y,x) = y, f(x,x,y) = x$.
   The operation $g(x,y,z) := f(x,z,y)$ is a Maltsev operation.

8. $f(y,x,x) = y, f(x,y,x) = y, f(x,x,y) = y$.
   In this case, f is a Maltsev operation.

We claim that if f is a Maltsev operation, then either it is a minority operation (and we are done) or it generates a Majority operation. Indeed, if f is not a minority then minimality of f implies that $f(x,y,x) = x$. Now consider the function g defined by $g(x,y,z) = f(x,f(x,y,z),z)$. We have

$$\begin{aligned} g(x,x,y) &= f(x,f(x,x,y),y) = f(x,y,y) = x \\ g(x,y,x) &= f(x,f(x,y,x),x) = f(x,x,x) = x \\ g(y,x,x) &= f(y,f(y,x,x),x) = f(y,y,x) = x\,. \end{aligned}$$

Note that every ternary function generated by a majority is again a majority. Also note that a function cannot be a majority and a minority at the same time unless the domain has only one element, so we obtain in this case a contradiction to the minimality of f.

Finally, let f be k-ary, where $k \geq 4$. By minimality of f, the operations obtained from f by identifications of arguments of g must be projections. The lemma of Świerczkowski implies that f is a semiprojection. □

Proposition 5.12. *For all $n \geq 3$, the graph K_n is projective (i.e., all idempotent polymorphisms of K_n are projections). All relations that are preserved by* $\mathrm{Sym}(\{0,\dots,n-1\})$ *are primitive positive definable in K_n.*

This provides for example a solution to Exercise 68.

Proof. By Theorem 5.8, it suffices to show that the clone of idempotent polymorphisms of K_n does not contain a minimal operation. Hence, by Theorem 5.11, we have to verify that $\mathrm{Pol}(K_n)$ does not contain a binary idempotent, a Maltsev, a majority, or a k-ary semiprojection for $k \geq 3$.

1. Let f be a binary idempotent polymorphism of K_n.

   Observation 1. $f(u,v) \in \{u,v\}$: otherwise, $i := f(u,v)$ is adjacent to both u and v, but $f(i,i) = i$ is not adjacent to i, in contradiction to f being a polymorphism.

   Observation 2. If $f(u,v) = u$, then $f(v,u) = v$: this is clear if $u = v$, and if $u \neq v$ it follows from f being a polymorphism.

   By Observation 1, it suffices to show that there cannot be distinct u,v and distinct u',v' such that $f(u,v) = u$ and $f(u',v') = v'$. Suppose for contradiction that there are such u,v,u',v'.

   Case 1. $u = u'$. Since $f(u,v') = f(u',v') = v'$, we have $f(v',u) = u$ by Observation 2. This is in contradiction to $f(u,v) = u$ since $u = u'$ is adjacent to v', and $E(v,u)$.

Case 2. $u \neq u'$.
Case 2.1. $f(u', u) = u$: this is impossible because $f(u, v) = u$, $E(u, u')$, and $E(u, v)$.
Case 2.2. $f(u', u) = u'$: this is impossible because $f(v', u') = u'$, $E(u', v')$, and $E(u', u)$.

2. Since $(1, 0), (1, 2), (0, 2) \in E(K_n)$, but $(0, 0) \notin E(K_n)$, the graph K_n has no Maltsev polymorphism (it is not rectangular; see Section 3.2).

3. If f is a majority, note that $f(0, 1, 2) = f(x_0, x_1, x_2)$ where x_i is some element distinct from i if $f(0, 1, 2) = i$, and $x_i := f(0, 1, 2)$ otherwise. But $(i, x_i) \in E(K_n)$, so f is not a polymorphism of K_n.

4. Finally, let f be a k-ary semiprojection for $k \geq 3$ which is not a projection. Suppose without loss of generality that $f(x_1, \ldots, x_k) = x_1$ whenever $|\{x_1, \ldots, x_k\}| < k$ (otherwise, permute the arguments of f). Since f is not a projection, there exist pairwise distinct $a_1, \ldots, a_k \in V(K_n)$ such that $c := f(a_1, \ldots, a_k) \neq a_1$. Let $b_1, \ldots, b_k$ be such that b_i is any element of $V(K_n) \setminus \{c\}$ if $c = a_i$, and $b_i := c$ otherwise. Note that $b_1 = a_1$ since $c \neq a_1$, and that $f(b_1, \ldots, b_k) = b_1 = a_1$ because f is a semiprojection. But $(a_i, b_i) \in E(K_n)$ for all $i \leq k$, so f is not a polymorphism of K_n.

The second part of the statement follows from Theorem 5.2. □

5.5 Schaefer's Theorem

Schaefer's theorem states that every CSP for a 2-element structure is either in P or NP-hard. By the general results in Section 5.2, most of the classification arguments in Schaefer's article follow from earlier work of Post [55], who classified all clones on a two-element domain. We present a short proof of Schaefer's theorem here.

Note that on Boolean domains, there is precisely one minority operation, and precisely one majority operation.

Theorem 5.13 (Post [55])**.** *Every minimal operation on* $\{0, 1\}$ *is among one of the following:*

- *a unary constant function.*
- *the unary function* $x \mapsto 1 - x$.
- *the binary function* $(x, y) \mapsto \min(x, y)$.
- *the binary function* $(x, y) \mapsto \max(x, y)$.
- *the Boolean minority operation.*
- *the Boolean majority operation.*

Proof. If f is unary the statement is trivial, so let f be a minimal at least binary function above $\mathscr{C}$. Note that $\hat{f}$ defines a function in $\mathscr{C}$, so it must be the identity by minimality of f, and hence f is idempotent.

There are only four binary idempotent operations on $\{0, 1\}$, two of which are projections and therefore cannot be minimal. The other two operations are min and max. Next, note that a semiprojection of arity at least three on a Boolean domain must be a projection. Thus, Theorem 5.11 implies that f is the majority or a minority operation. □

A boolean relation $R \subseteq \{0,1\}^n$ is called *affine* if it is the solution space of a system of linear equalities modulo 2.

Theorem 5.14. *A Boolean relation is affine if and only if it is preserved by the Boolean minority operation.*

Proof. This statement follows from basic facts in linear algebra. Let R be n-ary. We view R as a subset of the Boolean vector space $\{0,1\}^n$. It is well-known that affine spaces are precisely those that are closed under *affine combinations*, i.e., linear combinations of the form $\alpha_1 x_1 + \cdots + \alpha_k x_k$ such that $\alpha_1 + \cdots + \alpha_k = 1$. In particular, if R is affine then it is preserved by $(x_1, x_2, x_3) \mapsto x_1 + x_2 + x_3$ which is the minority operation. Conversely, if R is preserved by the minority operation, then $x_1 + \cdots + x_k$, for odd k, can be written as

$$\text{minority}(x_1, x_2, \text{minority}(x_3, x_4, \ldots \text{minority}(x_{n-2}, x_{k-1}, x_k) \ldots))$$

and hence R is preserved by all affine combinations, and thus affine. □

It is well-known and easy to see (see, for example, [11]) that for every relation $R \subseteq \{0,1\}^n$ there exists a propositional formula $\phi(x_1, \ldots, x_n)$ that defines R, and that ϕ can even be chosen to be in *conjunctive normal form (CNF)*. That is, there is a conjunction of disjunctions of variables or negated variables from $x_1, \ldots, x_n$ such that a tuple $(t_1, \ldots, t_n) \in \{0,1\}^n$ is in R if and only if the formula ϕ evaluates to true after replacing x_i by t_i, for $i \in \{1, \ldots, n\}$. The following definition is useful for proving that certain Boolean relations R can be defined in syntactically restricted propositional logic.

Definition 5.15. When ϕ is a propositional formula in CNF that defines a Boolean relation R, we say that ϕ is *reduced* if the following holds: whenever we remove a literal from a clause in ϕ, then the resulting formula no longer defines R.

Clearly, every Boolean relation has a reduced definition: simply remove literals from any definition in CNF until the formula becomes reduced. A propositional formula in CNF is called *Horn* if every clause contains at most one positive literal.

Lemma 5.16. *A Boolean relation R has a Horn definition if and only if R is preserved by* min.

Proof. It is easy to see that min preserves every relation defined by clauses that contains at most one positive literal, and hence every relation with a Horn definition. Conversely, let R be a Boolean relation preserved by min. Let ϕ be a reduced propositional formula in CNF that defines R. Now suppose for contradiction that ϕ contains a clause C with two positive literals u and v. Since ϕ is reduced, there is an assignment s_1 that satisfies ϕ such that $s_1(u) = 1$, and such that all other literals of C evaluate to 0. Similarly, there is a satisfying assignment s_2 for ϕ such that $s_2(v) = 1$ and all other literals of C evaluate to 0. Then $s_0 \colon x \mapsto \min(s_1(x), s_2(y))$ does not satisfy C, and does not satisfy ϕ, in contradiction to the assumption that min preserves R. □

A binary relation is called *bijunctive* if it can be defined by a propositional formula in CNF where each disjunction has at most two disjuncts.

Lemma 5.17. *A Boolean relation R is preserved by the majority operation if and only if it is bijunctive.*

Proof. We present the proof that when R is preserved by the majority, and ϕ is a reduced definition of R, then all clauses C have at most two literals. Suppose for contradiction that C has three literals l_1, l_2, l_3. Since ϕ is reduced, there must be satisfying assignments s_1, s_2, s_3 to ϕ such that under s_i all literals of C evaluate to 0 except for l_i. Then the mapping $s_0 \colon x \mapsto \text{majority}(s_1(x), s_2(x), s_3(x))$ does not satisfy C and therefore does not satisfy ϕ, in contradiction to the assumption that majority preserves R. □

The following relation is called the *(Boolean) not-all-equal relation.*

$$\text{NAE} := \{(0,0,1), (0,1,0), (1,0,0), (1,1,0), (1,0,1), (0,1,1)\}$$

Theorem 5.18 (Schaefer [57])**.** *Let $\mathfrak{B}$ be a structure over the two-element universe $\{0,1\}$. Then either $(\{0,1\}; \text{NAE})$ has a primitive positive definition in $\mathfrak{B}$, and $\text{CSP}(\mathfrak{B})$ is NP-complete, or*

1. *$\mathfrak{B}$ is preserved by a constant operation.*
2. *$\mathfrak{B}$ is preserved by* min*. In this case, every relation of $\mathfrak{B}$ has a definition by a propositional Horn formula.*
3. *$\mathfrak{B}$ is preserved by* max*. In this case, every relation of $\mathfrak{B}$ has a definition by a* dual-Horn *formula, that is, by a propositional formula in CNF where every clause contains at most one negative literal.*
4. *$\mathfrak{B}$ is preserved by the majority operation. In this case, every relation of $\mathfrak{B}$ is bijunctive.*
5. *$\mathfrak{B}$ is preserved by the minority operation. In this case, every relation of $\mathfrak{B}$ can be defined by a conjunction of linear equations modulo 2.*

In case (1) *to case* (5)*, then for every finite-signature reduct $\mathfrak{B}'$ of $\mathfrak{B}$ the problem $\text{CSP}(\mathfrak{B}')$ can be solved in polynomial time.*

Proof. If $\text{Pol}(\mathfrak{B})$ contains a constant operation, then we are in case one; so suppose in the following that this is not the case. If NAE is primitive positive definable in $\mathfrak{B}$, then $\text{CSP}(\mathfrak{B})$ is NP-hard by reduction from positive not-all-equal-3SAT [36]. Otherwise, by Theorem 5.2 there is an operation $f \in \text{Pol}(\mathfrak{B})$ that violates NAE. If $\hat{f}$ defined as $x \mapsto f(x, \dots, x)$ equals the identity then f is idempotent. Otherwise, $\hat{f}$ equals $\neg$. But then $\neg f \in \text{Pol}(\mathfrak{B})$ is idempotent and also violates NAE. So let us assume in the following that f is idempotent. Then f generates an at least binary minimal operation $g \in \text{Pol}(\mathfrak{B})$.

By Theorem 5.13, the operation g equals min, max, the Boolean minority, or the Boolean majority function.

- $g = \text{min}$ or $g = \text{max}$. If $\mathfrak{B}$ is preserved by min, then by Lemma 5.16 all relations of $\mathfrak{B}$ can be defined by propositional Horn formulas. It is well-known that positive unit-resolution is a polynomial-time decision procedure for the satisfiability problem of propositional Horn-clauses [58]. The case that $g = \text{max}$ is dual to this case.
- $g =$ majority. By Lemma 5.17, all relations of $\mathfrak{B}$ are bijunctive. Hence, in this case the instances of $\text{CSP}(\mathfrak{B})$ can be viewed as instances of the 2SAT problem, and can be solved in linear time [3].

- $g =$ minority. By Theorem 5.14 every relation of $\mathfrak{B}$ has a definition by a conjunction of linear equalities modulo 2. Then CSP($\mathfrak{B}$) can be solved in polynomial time by Gaussian elimination.

This concludes the proof of the statement. □

Exercises.

76. Determine the complexity of the following CSPs.

$$\text{CSP}(\{0,1\};\{(0,0,1,1),(1,1,0,0)\})$$
$$\text{CSP}(\{0,1\};\{(0,0,1),(0,1,0),(1,0,0),(1,1,1)\},\{(0,1),(1,0)\})$$
$$\text{CSP}(\{0,1\};\{0,1\}^3 \setminus \{(1,1,0)\},\{(0,1),(1,0)\}).$$

6 Maltsev Polymorphisms

Recall from Section 3.2 the definition of a Maltsev operation: a ternary operation $f\colon D^3 \to D$ satisfying

$$\forall x, y \in D.\, f(y,x,x) = f(x,x,y) = y\,.$$

As we have seen in Theorem 3.14, every digraph with a Maltsev polymorphism can be solved by the path-consistency procedure. However, when considering arbitrary relational structures then there are many examples with a Maltsev polymorphism that cannot be solved by the path-consistency procedure [4]. In this section, we present the algorithm of Bulatov and Dalmau for CSP($\mathfrak{A}$) when $\mathfrak{A}$ is preserved by a Maltsev polymorphism [19].

Theorem 6.1. *Let $\mathfrak{A}$ be a finite structure with finite relational signature and a Maltsev polymorphism. Then* CSP($\mathfrak{A}$) *can be solved in polynomial time.*

6.1 Examples

The most prominent class of structures $\mathfrak{A}$ with a Maltsev polymorphism comes from groups. For any group G, the operation m given by $(x,y,z) \mapsto x + y^{-1} + z$ is obviously Maltsev and called an *affine Maltsev operation*. Note that if the group G is $F = (\mathbb{Z}_p; +, 0)$ then the k-ary relations preserved by m are precisely the affine subspaces of F^k (with the same argument as given for Theorem 5.14 in the Boolean case). In this case one can use Gaussian elimination to solve CSP($\mathfrak{A}$). If G is Abelian, then $\mathfrak{A}$ is called *affine Maltsev*; these structures will play an important role in Section **??**.

For general finite groups G, and if all relations of $\mathfrak{A}$ are cosets $gH := \{gh \mid h \in H\}$ of subgroups H of G^k, then Feder and Vardi [34] showed how to solve CSP($\mathfrak{A}$) in polynomial time using a previously known algorithm to find small generating sets for a permutation group. We will not discuss this approach, but rather present the more general algorithm of Bulatov and Dalmau which works for all finite structures preserved by a Maltsev polymorphism. First we have a look at two examples of Maltsev polymorphisms that do not come from groups in the way described above.

Example 13. On the domain $\{0,1,2\}$, let m be a minority, and define $m(x,y,z) = 2$ whenever $|\{x,y,z\}| = 3$. Note that m preserves the equivalence relation with the equivalence classes $\{2\}$ and $\{0,1\}$. Also note that m preserves all relations of the form (for $i \in \{0,1\}$)

$$\{(2,\ldots,2)\} \cup \{(x_1,\ldots,x_n) \in \{0,1\}^n \mid x_1 + x_2 + \cdots + x_n = i \bmod 2\}\,. \qquad \triangle$$

Example 14. On the domain $\{0,1,2\}$, let m be a minority, and define $m(x,y,z) = x$ whenever $|\{x,y,z\}| = 3$. Then m preserves for every permutation π of $\{0,1,2\}$ the relation $\{(x,y) \mid \pi(x) = y\}$. Moreover, m preserves the binary relations R that are maximal with the property that $|\operatorname{pr}_1 R|, |\operatorname{pr}_2 R| \leq 2$. △

Exercises.

77. Check the claims made in Example 13. Are all relations that are preserved by m primitive positive definable over the given relations?

78. Check the claims made in Example 14. Are all relations that are preserved by m primitive positive definable over the given relations?

79. Show that if a Maltsev operation m preserves the graph of a Malsev operation m' on the same domain, then $m = m'$.

6.2 Compact Representations of Relations

Our presentation of the proof closely follows that of Bulatov and Dalmau [19].

Definition 6.2 (Forks and Representations)**.** Let $R \subseteq A^n$ be a relation.

- A *fork of* R is a triple (i,a,b) such that there exist $s,t \in R$ with $(s_1,\dots,s_{i-1}) = (t_1,\dots,t_{i-1})$, $s_i = a$, and $t_i = b$. We say that s and t *witness* (i,a,b).
- $R' \subseteq R$ is called a *representation of* R if every fork of R is also a fork of R'.
- A representation R' of R is called *compact* if its cardinality is at most twice the number of forks of R.

Clearly, every relation has a compact representation. Let $R \subseteq A^n$. For $i_1,\dots,i_n \in \{1,\dots,n\}$, we write $\operatorname{pr}_{i_1,\dots,i_k}(R)$ for the k-ary relation on A obtained as the projection of R to the arguments $i_1,\dots,i_k$. For $t = (t_1,\dots,t_n)$, we also write $\operatorname{pr}_{i_1,\dots,i_k}(t)$ for the tuple $(t_{i_1},\dots,t_{i_k})$. For an operation $m\colon A^k \to A$ and a relation R on A, we write $\langle R\rangle_m$ for the smallest relation that contains R and is preserved by m.

Lemma 6.3. *Let A be a finite set and let $m\colon A^3 \to A$ be a Maltsev operation. Let $R \subseteq A^k$ be a relation preserved by m, and let R' be a representation of R. Then $R = \langle R'\rangle_m$.*

Proof. We show by induction on $i \in \{1,\dots,n\}$ that $\operatorname{pr}_{1,\dots,i}\langle R'\rangle_m = \operatorname{pr}_{1,\dots,i}(R)$. Clearly, $\operatorname{pr}_{1,\dots,i}\langle R'\rangle_m \subseteq \operatorname{pr}_{1,\dots,i}(R)$, so we only need to show the converse inclusion. The case $i = 1$ follows from that fact that R has for every $t \in R$ the fork $(1,t_1,t_1)$, and since R' must also have this fork it must contain a tuple t' such that $t'_1 = t_1$.

So let us assume that the statement holds for $i < n$. We have to show that for every $t \in R$ we have $(t_1,\dots,t_{i+1}) \in \operatorname{pr}_{1,\dots,i+1}\langle R'\rangle_m$. By induction hypothesis there exists a tuple $s \in \langle R'\rangle_m$ such that $(s_1,\dots,s_i) = (t_1,\dots,t_i)$. Then $(i+1, s_{i+1}, t_{i+1})$ is a fork of R, so there exist tuples $s', s'' \in R'$ witnessing it. Then the tuple $t' := m(s,s',s'') \in \langle R'\rangle_m$ is such that

$$\begin{aligned}(t'_1,\dots,t'_i,t'_{i+1}) &= (m(t_1,s'_1,s''_1),\dots,m(t_i,s'_i,s''_i),m(s_{i+1},s_{i+1},t_{i+1}))\\ &= (t_1,\dots,t_i,t_{i+1}) \qquad \text{(since } s'_i = s''_i\text{).}\end{aligned}$$

Hence, $(t_1,\dots,t_i,t_{i+1})$ is a tuple from $\operatorname{pr}_{1,\dots,i+1}\langle R'\rangle_m$, as required. □

Procedure *Nonempty*$(R', i_1, \dots, i_k, S)$.

Set $U := R'$.
While $\exists r, s, t \in U$ such that $\mathrm{pr}_{i_1,\dots,i_k} m(r,s,t) \notin \mathrm{pr}_{i_1,\dots,i_k} U$:
    Set $U := U \cup \{m(r,s,t)\}$
If $\exists t \in U$ such that $(t_{i_1}, \dots, t_{i_k}) \in S$ then return t
else return 'No'.

Figure 7: The procedure *Nonempty*.

Exercises.

80. Let A be a finite set. How many forks does the n-ary relation $R := A^n$ have? Explicitly construct a compact representation for R.

81. Let R be the relation $\{(x,y,z,u) \in \{0,1\}^4 \mid x + y + z = 1 \mod 2\}$. Find a smallest possible representation R' for R. Explicitly compute $\langle R' \rangle_m$ where m is the Boolean minority.

6.3 The Bulatov-Dalmau Algorithm

Let $\exists x_1, \dots, x_n(\phi_1 \wedge \cdots \wedge \phi_n)$ be an instance of CSP($\mathfrak{A}$). For $\ell \leq n$, we write R_ℓ for the relation

$$\{(s_1, \dots, s_n) \in A^n \mid \mathfrak{A} \models (\phi_1 \wedge \cdots \wedge \phi_\ell)(s_1, \dots, s_n)\}.$$

The idea of the algorithm is to inductively construct a compact representation R'_ℓ of R_ℓ, adding constraints one by one. Initially, for $\ell = 0$, we have $R_\ell = A^n$, and it is easy to come up with a compact representation for this relation. Note that when we manage to compute the compact representation R'_n for R_n, we can decide satisfiability of the instance: it is unsatisfiable if and only if R'_n is empty. For the inductive step, we need a procedure called *Next* which is more involved; we first introduce two auxiliary procedures.

The procedure *Nonempty*

The procedure *Nonempty* receives as input

- a compact representation R' of a relation R,
- a sequence $i_1, \dots, i_k$ of elements in $[n]$ where n is the arity of R, and
- a k-ary relation S which is also preserved by m.

The output of the procedure is either a tuple $t \in R$ such that $(t_{i_1}, \dots, t_{i_k}) \in S$, or 'No' if no such tuple exists. The procedure can be found in Figure 7. For its correctness we note the following:

- $R' \subseteq U \subseteq R$: initially we start from $U := R' \subseteq R$, and only add tuples to U obtained by applying m to tuples in U, so the added tuples are again in R.

- It follows that if *Nonempty* returns a tuple $(t_{i_1}, \dots, t_{i_k})$, then this tuple is indeed from $\mathrm{pr}_{i_1,\dots,i_k} R$ and the output of the algorithm is correct.
- When the algorithm exits the while loop then $\mathrm{pr}_{i_1,\dots,i_k}\langle U\rangle_m = \mathrm{pr}_{i_1,\dots,i_k} U$. Since $R' \subseteq U$ we have that $\langle U\rangle_m = R$. Hence, every tuple $t \in \mathrm{pr}_{i_1,\dots,i_k} R = \mathrm{pr}_{i_1,\dots,i_k}\langle U\rangle_m$ is contained in $\mathrm{pr}_{i_1,\dots,i_k} U$, and so the answer of the algorithm is also correct when it returns 'No'.

We mention that this procedure does not use the particular properties of a Maltsev polymorphism, but works for any explicitly given polymorphism.

Running time. The number of iterations of the while loop can be bounded by the size $|U|$ of the set U at the end of the execution of the procedure. Hence, when we want to use this procedure to obtain a polynomial-time running time, we have to make sure that the size of U remains polynomial in the input size. The way this is done in the Bulatov-Dalmau algorithm is to guarantee that at each call of *Nonempty* the size L of $\mathrm{pr}_{i_1,\dots,i_k} R$ is polynomial in the input size. Then $|U|$ is bounded by $L + |R'|$ which is also polynomial.

We have to test all tuples $r, s, t \in U$; this can be implemented so that $|U|^3$ steps suffice. In each step we have to compute $m(r, s, t)$ and test whether $\mathrm{pr}_{i_1,\dots,i_k} m(r, s, t) \in \mathrm{pr}_{i_1,\dots,i_k} U$, which can be done in $O(kL)$. In the important case that L is bounded by a constant in the size of the input N, the running time of *Nonempty* is in $O(N^4)$.

The procedure *Fix-values*

The procedure *Fix-values* receives as input

- a compact representation R' of an n-ary relation R preserved by m, and
- a sequence $c_1, \dots, c_k \in A$ for $k \leq n$.

The output of *Fix-values* is a compact representation of the relation

$$R \cap (\{c_1\} \times \cdots \times \{c_k\} \times A \times \cdots \times A).$$

The procedure can be found in Figure 8. The algorithm computes inductively a compact representation U_j of the relation

$$R_j = R \cap (\{c_1\} \times \cdots \times \{c_j\} \times A \times \cdots \times A)$$

This is immediate for $U_0 = R'$, and the set U_k is the relation that we have to compute.

For its correctness, suppose inductively that U_j is a compact representation of R_j. We have to show that the set U_{j+1} computed by the procedure is a compact representation of R_{j+1}:

1. $U_{j+1} \subseteq R_{j+1}$. Suppose that the procedure adds $\{r, m(r, s, t)\}$ to U_{j+1}, where r and s witness the fork (i, a, b) of U_j processed in the for-loop of the procedure. Note that $r \in R_{j+1}$ since $r \in U_j \subseteq R_j$ and $r_{j+1} = c_{j+1}$. Since m preserves R and is idempotent, it also preserves R_j, and since $r, s, t \in R_j$ it follows that $m(r, s, t) \in R_j$. To show that $m(r, s, t) \in R_{j+1}$ it suffices to show that $s_{j+1} = t_{j+1}$ because then $m(r, s, t)_{j+1} = r_{j+1} = c_{j+1}$ since m is Maltsev. If $i > j + 1$ then we have that $s_{j+1} = t_{j+1}$ since s, t witness (i, a, b). Otherwise, we must have $a = b = c_i$ because of the innermost if-clause of the procedure. But then $s = t$ by the stipulation of the algorithm on the choice of s and t.

Procedure *Fix-values*$(R', c_1, \ldots, c_k)$.

Set $j := 0$; $U_j := R'$.
While $j < k$ do:
  Set $U_{j+1} := \emptyset$.
  For each $(i, a, b) \in [n] \times A^2$:
    If $\exists s, t \in U_j$ witnessing (i, a, b) (assuming $s = t$ if $a = b$):
      If $r :=$ Nonempty$(U_j, j+1, i, \{(c_{j+1}, a)\}) \neq$ 'No'
        If $(i > j+1)$ or $(a = b = c_i)$:
          Set $U_{j+1} := U_{j+1} \cup \{r, m(r, s, t)\}$
  Set $j := j + 1$.
Return U_k.

Figure 8: The procedure *Fix-values*.

2. All forks (i, a, b) of R_{j+1} are forks of U_{j+1}. If R_{j+1} has the fork (i, a, b), then by inductive assumption U_j must contain witnesses s, t for (i, a, b). Therefore, the first if-clause of the procedure is positive. Moreover, $s_{j+1} = c_{j+1}$ and $s_i = a$, so $r :=$ *Nonempty*$(U_j, j+1, i, \{(c_{j+1}, a)\}) \neq$ 'No'. Also note that if $i \leq j+1$, then $a = s_i = c_i = t_i = b$. So all the if-clauses of the procedure are positive, and the procedure adds r and $m(r, s, t)$ to U_{j+1}. The tuples r and $m(r, s, t)$ witness (i, a, b). Since s, t witness (i, a, b) we have that $(s_1, \ldots, s_{i-1}) = (t_1, \ldots, t_{i-1})$. Hence, $\mathrm{pr}_{1,\ldots,i-1}(m(r, s, t)) = (r_1, \ldots, r_{i-1})$. Furthermore, we have that $\mathrm{pr}_i(m(r, s, t)) = m(a, a, b) = b$.

3. The representation U_{j+1} of R_{j+1} is compact since at most two tuples are added to U_{j+1} for each fork of R_{j+1}.

Running time. The while loop is performed $k \leq n$ times; the inner for-loop is executed for each $(i, a, b) \in [n] \times A^2$, which is linear for fixed $\mathfrak{A}$. The cost of each iteration is dominated by the cost of calling the procedure *Nonempty*. Note that when calling *Nonempty*, the size of $\mathrm{pr}_{j+1,i} U_j$ is polynomial in the input size (even constant size when $\mathfrak{A}$ is fixed), so the cost of *Nonempty* is in $O(N^4)$ where N is the size of the input. Therefore, the total time complexity of the procedure *Fix-values* is polynomial in the input size (for fixed $\mathfrak{A}$ it is in $O(N^5)$).

The procedure *Next*

Now comes the heart of the algorithm, which is the procedure *Next* that updates a compact representation of the solution space when constraints are added one by one. The input of the procedure is

- a compact representation R' of a relation $R \subseteq A^n$ that is preserved by m,
- a sequence $i_1, \ldots, i_k$ of elements from $[n]$,
- a k-ary relation S which is also preserved by m.

The output of the procedure is a compact representation of the relation

$$R^* := \{t \in R \mid (t_{i_1}, \ldots, t_{i_k}) \in S\}.$$

Procedure $\mathit{Next}(R', i_1, \ldots, i_k, S)$.

Set $U := \emptyset$.
For each $(i, a, b) \in [n] \times A^2$:
    If $\mathit{Nonempty}(R', i_1, \ldots, i_k, i, S \times \{a\}) =: t \neq$ 'No':
        If $\mathit{Nonempty}(\mathit{Fix\text{-}values}(R', t_1, \ldots, t_{i-1}), i_1, \ldots, i_k, i, S \times \{b\}) =: t' \neq$ 'No':
            Set $U := U \cup \{t, t'\}$.
Return U_k.

Figure 9: The procedure *Next*.

The procedure *Next* can be found in Figure 9. Observe that

- the condition $\mathit{Nonempty}(R', i_1, \ldots, i_k, i, S \times \{a\}) \neq$ 'No' from the first if-clause is satisfied if and only if there exists a tuple $t \in R$ such that $(t_{i_1}, \ldots, t_{i_k}) \in S$ and $t_i = a$. Hence, if such a tuple does not exist, then (i, a, b) cannot be a fork of R^*, and nothing needs to be done.
- the condition $\mathit{Nonempty}(\mathit{Fix\text{-}values}(R', t_1, \ldots, t_{i-1}), i_1, \ldots, i_k, i, S \times \{b\}) \neq$ 'No' from the second if-clause is satisfied if and only if there exists a tuple $t' \in R$ such that
  - $(t'_1, \ldots, t'_{i-1}) = (t_1, \ldots, t_{i-1})$,
  - $(t'_{i_1}, \ldots, t'_{i_k}) \in S$, and
  - $t'_i = b$.

  If this condition holds, and since $t_i = a$, we have that t and t' witness (i, a, b). It only remains to show that if (i, a, b) is a fork of R^*, then such a tuple t' must exist. So let r and s be witnesses for (i, a, b) in R^*. Then the tuple $t' := m(t, r, s)$ has the desired properties:
  - for $j < i$ we have that $t'_j = m(t_j, r_j, s_j) = t_j$;
  - $t' \in S$ because $(r_{i_1}, \ldots, r_{i_k}), (s_{i_1}, \ldots, s_{i_k}), (t_{i_1}, \ldots, t_{i_k}) \in S$ and m preserves S.
  - $t'_i = m(t_i, r_i, s_i) = m(a, a, b) = b$.
- The cardinality of U is bounded by twice the number of forks of R^*, so the representation computed by the algorithm is compact.

Running time. The for-loop of the procedure *Next* is performed $n|A|^2$ times and the cost of each iteration is polynomial in the cost of *Nonempty* and *Fix-values*. Also note that k is bounded by the maximal arity of the relations in $\mathfrak{A}$, so constant for fixed $\mathfrak{A}$. It follows that $\mathrm{pr}_{i_1,\ldots,i_k,i}(R)$ is polynomial, so the running time of the calls to *Nonempty* are polynomial. For fixed $\mathfrak{A}$, the global running time of the procedure *Next* is in $O(N^6)$ where N is the size of the input.

Proof of Theorem 6.1. Starting from an empty list of constraints, we add constraints on the variables $x_1, \ldots, x_n$ one by one, and maintain a compact representation of the n-ary relation defined by the constraints considered so far. Initially, we start with a compact representation

of the full relation A^n. In later steps, we use the procedure *Next* to compute a compact representation when a constraint is added, in $O(N^6)$ for fixed $\mathfrak{A}$ and N the size of the input. The instance is unsatisfiable if and only if at the final stage we end up with an empty representation. The entire running time of the algorithm is in $O(N^7)$. □

Exercises.

82. Let $\mathfrak{A}$ be the structure $(\{0,1\}; L_0, L_1)$ where $L_i := \{(x,y,z) \mid x+y+z = i \mod 2\}$, which has the Boolean minority m as polymorphism. Consider the instance

$$\exists x_1, \ldots, x_5 \left(L_1(x_1,x_2,x_3) \wedge L_1(x_2,x_3,x_4) \wedge L_1(x_3,x_4,x_5) \wedge L_0(x_1,x_3,x_5)\right)$$

Compute compact representations R'_ℓ of R_ℓ, for $\ell \in \{1,2,3,4\}$.

83. (∗) Let $\mathfrak{B}$ be a structure with a Maltsev polymorphism f and an *infinite* relational signature. Note that we have defined CSP($\mathfrak{B}$) only if $\mathfrak{B}$ has a finite signature. If we want to define CSP($\mathfrak{B}$) also for structures $\mathfrak{B}$ with an infinite signature, it is important to discuss how the relation symbols in the signature of $\mathfrak{B}$ are represented in the input. We choose to represent a relation symbol R from $\mathfrak{B}$ by listing the tuples in $R^{\mathfrak{B}}$. Adapt the Dalmau algorithm such that it can solve CSP($\mathfrak{B}$) in polynomial time for this choice of representing the relations in $\mathfrak{B}$.

84. (∗) The *graph isomorphism problem (GI)* is a famous computational problem that is neither known to be solvable in polynomial time, nor expected to be NP-hard. An instance of GI consists of two graphs G and H, and the question is to decide whether G and H are isomorphic. Consider the variant of the graph-isomorphism problem where the vertices are coloured, each color appears at most k times for some constant k, and the isomorphism between H and G that we are looking for is required to additionally preserve the colours. Show that this problem can be solved in polynomial time using Dalmau's algorithm (use the previous exercise).

7 Universal Algebra

7.1 Algebras and Clones

In universal algebra, an *algebra* is simply a structure with a purely functional signature. We will typically use bold font letters, like $\mathbf{A}$, to denote algebras, and the corresponding capital roman letters, like A, to denote their domain.

Example 15. A *group* is an algebra with a binary function symbol $\circ$ for composition, a unary function symbol $^{-1}$ for taking the inverse, and a constant denoted by e, satisfying

- $\forall x, y, z.\, x \circ (y \circ z) = (x \circ y) \circ z$,
- $\forall x.\, x \circ x^{-1} = e$,
- $\forall x.\, e \circ x = x$, and $\forall x.\, x \circ e = x$.

Note that all axioms are *universal* in the sense that all the variables are universally quantified (more on that comes later). The group is called *abelian* if it additionally satisfies

$$\forall x, y.\, x \circ y = y \circ x.$$ △

Example 16 (Ring)**.** A *(commutative) ring* is an algebra $\mathbf{A}$ with the signature $\{\cdot, +, -, 0, 1\}$ where $\cdot, +$ are binary, $-$ is unary, and $0, 1$ are constants, such that $(A; +, -, 0)$ is an abelian group and additionally

$$\begin{aligned}
&\forall x, y, z.\ (xy)z = x(yz) && \text{(associativity)}\\
&\forall x.\ 1 \cdot x = x && \text{(multiplicative unit)}\\
&\forall x, y.\ xy = yx && \text{(commutativity)}\\
&\forall x, y, z.\ x(y + z) = xy + xz && \text{(distributivity)}
\end{aligned}$$ △

Example 17 (Module)**.** Let $\mathbf{R}$ be a ring. An $\mathbf{R}$-*module* is an algebra $\mathbf{M}$ with the signature $\{+, -, 0\} \cup \{f_r \mid r \in R\}$ such that $(M; +, -, 0)$ is an abelian group and

$$\begin{aligned}
&\forall x, y.\ f_r(x + y) = f_r(x) + f_r(y) \text{ holds for every } r \in R\\
&\forall x.\ f_{r+s}(x) = f_r(x) + f_s(x) \text{ holds for all } r, s \in R\\
&\forall x.\ f_r(f_s(x)) = f_{rs}(x) \text{ holds for all } r, s \in R.
\end{aligned}$$

An $\mathbf{R}$-module is called *unitary* if it additionally satisfies $\forall x. f_1(x) = x$. △

Example 18 (Semilattice)**.** A *meet-semilattice* $\mathfrak{S}$ is a $\{\leq\}$-structure with domain S such that $\leq^{\mathfrak{S}}$ denotes a partial order where any two $u, v \in S$ have a (unique) *greatest lower bound* $u \wedge v$, i.e., an element w such that $w \leq u$, $w \leq v$, and for all w' with $w' \leq u$ and $w \leq v$ we have $w' \leq w$. Dually, a *join-semilattice* is a partial order with least upper bounds, denoted by $u \vee v$. A *semilattice* is a meet-semilattice or a join-semilattice where the distinction between meet and join is either not essential or clear from the context.

Semilattices can also be characterised as $\{\wedge\}$-algebras where $\wedge$ is a binary operation that must satisfy the following axioms

$$\begin{aligned}
\forall x, y, z\colon\ x \wedge (y \wedge z) = (x \wedge y) \wedge z \qquad & \textit{(associativity)}\\
\forall x, y\colon\ x \wedge y = y \wedge x \qquad & \textit{(commutativity)}\\
\forall x\colon\ x \wedge x = x \qquad & \textit{(idempotency, or idempotence).}
\end{aligned}$$

Clearly, the operation $\wedge^{\mathfrak{S}}$, defined as above in a semilattice $\mathfrak{S}$ viewed as a poset, satisfies these axioms. Conversely, if $(S; \wedge)$ is a semilattice, then the formula $x \wedge y = x$ defines a partial order on S which is a meet-semilattice (and $x \wedge y = y$ defines a partial order on S which is a join-semilattice).

Note that the two ways of formalising semilattices differ when it comes to the notion of a substructure; a *subsemilattice* is referring to the substructure of a semilattice when formalised as an algebraic structure. △

Example 19 (Lattice)**.** A *lattice* $\mathfrak{L}$ is a $\{\leq\}$-structure with domain L such that $\leq^{\mathfrak{L}}$ denotes a partial order such that any two $u, v \in L$ have a largest lower bound $u \wedge v$ and a least upper bound, denoted by $u \vee v$. Lattices can also be characterised as $\{\wedge, \vee\}$-algebras where $\wedge$ and $\vee$ are semilattice operations (Example 18) that additionally satisfy

$$\forall x, y\colon\ x \wedge (x \vee y) = x \text{ and } x \vee (x \wedge y) = x \qquad \textit{(absorption).}$$

If $\mathfrak{L}$ is a lattice and the operations $\wedge$ and $\vee$ are defined as above for semilattices, then these two operations also satisfy the absorption axiom. Conversely, if we are given an algebra

$(S; \wedge, \vee)$ satisfying the mentioned axioms, then the formula $x \wedge y = x$ (equivalently, the formula $x \vee y = y$) defines a partial order on S which is a lattice. Of course, there is potential danger of confusion of the symbols for lattice operations $\wedge$ and $\vee$ with the propositional connectives $\wedge$ for conjunction and $\vee$ for disjunction (which can be seen as lattice operations on the set $\{0, 1\}$) which luckily should not cause trouble here. A lattice $\mathfrak{L} = (L; \wedge, \vee)$ is called *distributive* if it satisfies

$$\forall x, y\colon\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \qquad \text{(distributivity).} \qquad \triangle$$

The clone of an algebra. If $\mathbf{A}$ is an algebra with the signature τ, then a τ-term $t(x_1, \ldots, x_n)$ gives rise to a *term function* $t^{\mathbf{A}}\colon A^n \to A$; the value of $t^{\mathbf{A}}$ at $a_1, \ldots, a_n \in A$ can be obtained by replacing the variables $x_1, \ldots, x_n$ by $a_1, \ldots, a_n$ and evaluating in $\mathbf{A}$.

Example 20. If $\mathbf{A}$ is a group, then the term function for the term $(x \circ y^{-1}) \circ z$ is a Maltsev operation on A. $\triangle$

Example 21. If $t(x_1, x_2)$ is the term that just consists of the variable x_1, then $t^{\mathbf{A}}$ equals the projection π^2_1. $\triangle$

Algebras give rise to clones in the following way. We denote by $\mathrm{Clo}(\mathbf{A})$ the set of all term functions of $\mathbf{A}$. Clearly, $\mathrm{Clo}(\mathbf{A})$ is an operation clone since it is closed under compositions, and contains the projections.

Polymorphism algebras. In the context of complexity classification of CSPs, algebras arise as follows.

Definition 7.1. Let $\mathfrak{B}$ be a relational structure with domain B. An algebra $\mathbf{B}$ with domain B such that $\mathrm{Clo}(\mathbf{B}) = \mathrm{Pol}(\mathfrak{B})$ is called a *polymorphism algebra of* $\mathfrak{B}$.

Note that a structure $\mathfrak{B}$ has many different polymorphism algebras, since Definition 7.1 does not prescribe how to assign function symbols to the polymorphisms of $\mathfrak{B}$.

Any clone $\mathcal{C}$ on a set D can be viewed as an algebra $\mathbf{A}$ with domain D whose signature consists of the operations of $\mathcal{C}$ themselfes; that is, if $f \in \mathcal{C}$, then $f^{\mathbf{A}} := f$. We will therefore use concepts defined for algebras also for clones. In particular, the polymorphism clone $\mathrm{Pol}(\mathfrak{B})$ of a structure $\mathfrak{B}$ might be viewed as an algebra, to which we refer to as *the* polymorphism algebra of $\mathfrak{B}$. Note that the signature of the polymorphism algebra is always infinite, since we have polymorphisms of arbitrary finite arity.

7.2 Subalgebras, Products, Homomorphic Images

In this section we recall some basic universal-algebraic facts that will be used in the following subsections.

Subalgebras. Let $\mathbf{A}$ be a τ-algebra with domain A. A τ-algebra $\mathbf{B}$ with domain $B \subseteq A$ is called a *subalgebra* of $\mathbf{A}$ if for each $f \in \tau$ of arity k we have $f^{\mathbf{B}}(b_1, \ldots, b_k) = f^{\mathbf{A}}(b_1, \ldots, b_k)$ for all $b_1, \ldots, b_k \in B$; in this case, we write $\mathbf{B} \leq \mathbf{A}$. A *subuniverse* of $\mathbf{A}$ is the domain of some subalgebra of $\mathbf{A}$. The smallest subuniverse of $\mathbf{A}$ that contains a given set $S \subseteq A$ is called the *subuniverse of* $\mathbf{A}$ *generated by* S, and the corresponding subalgebra is called the *subalgebra of* $\mathbf{A}$ *generated by* S.

Products. Let $\mathbf{A}, \mathbf{B}$ be τ-algebras with domain A and B, respectively. Then the *product* $\mathbf{A} \times \mathbf{B}$ is the τ-algebra with domain $A \times B$ such that for each $f \in \tau$ of arity k we have $f^{\mathbf{A}\times\mathbf{B}}\big((a_1, b_1), \dots, (a_k, b_k)\big) = \big(f^{\mathbf{A}}(a_1, \dots, a_k), f^{\mathbf{B}}(b_1, \dots, b_k)\big)$ for all $a_1, \dots, a_k \in A$ and $b_1, \dots, b_k \in B$. More generally, when $(\mathbf{A}_i)_{i\in I}$ is a sequence of τ-algebras, indexed by some set I, then $\prod_{i\in I} \mathbf{A}_i$ is the τ-algebra $\mathbf{A}$ with domain A^I such that $f^{\mathbf{A}}\big((a_i^1)_{i\in I}, \dots, (a_i^k)_{i\in I}\big) = \big(f^{\mathbf{A}_i}(a_i^1, \dots, a_i^k)\big)_{i\in I}$ for $a_i^1, \dots, a_i^k \in A_i$.

Lemma 7.2. *Let $\mathbf{A}$ be the polymorphism algebra of a finite structure $\mathfrak{A}$. Then the (domains of the) subalgebras of $\mathbf{A}^k$ are precisely the relations that have a primitive positive definition in $\mathfrak{A}$.*

Proof. A relation $R \subseteq A^k$ is a subalgebra of $\mathbf{A}^k$ if and only if for all m-ary f in the signature of $\mathbf{A}$ and $t^1, \dots, t^m \in R$, we have $\big(f(t_1^1, \dots, t_1^m), \dots, f(t_k^1, \dots, t_k^m)\big) \in R$, which is the case if and only if R is preserved by all polymorphisms of $\mathfrak{A}$, which is the case if and only if R is primitive positive definable in $\mathfrak{A}$ by Theorem 5.2. □

Homomorphic Images. Let $\mathbf{A}$ and $\mathbf{B}$ be τ-algebras. Then a *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a mapping $h\colon A \to B$ such that for all k-ary $f \in \tau$ and $a_1, \dots, a_k \in A$ we have

$$h\big(f^{\mathbf{A}}(a_1, \dots, a_k)\big) = f^{\mathbf{B}}\big(h(a_1), \dots, h(a_k)\big)\,.$$

Definition 7.3. A *congruence* of an algebra $\mathbf{A}$ is an equivalence relation that is preserved by all operations in $\mathbf{A}$.

Lemma 7.4. *Let $\mathfrak{B}$ be a finite structure, and $\mathbf{B}$ be a polymorphism algebra of $\mathfrak{B}$. Then the congruences of $\mathbf{B}$ are exactly the primitive positive definable equivalence relations over $\mathfrak{B}$.*

Proof. A direct consequence of Theorem 5.2. □

Proposition 7.5 (see [22])**.** *Let $\mathbf{A}$ be an algebra. Then E is a congruence of $\mathbf{A}$ if and only if E is the kernel of a homomorphism from $\mathbf{A}$ to some other algebra $\mathbf{B}$.*

Example 22. Let $G = (V, E)$ be the undirected graph with $V = \{a_1, \dots, a_4, b_1, \dots, b_4\}$ such that $a_1, \dots, a_4$ and $b_1, \dots, b_4$ induce a clique, for each $i \in \{1, \dots, 4\}$ there is an edge between a_i and b_i, and otherwise there are no edges in G. Let $\mathbf{A}$ be a polymorphism algebra of G. Then $\mathbf{A}$ homomorphically maps to a two-element algebra $\mathbf{B}$. By Proposition 7.5, it suffices to show that $\mathbf{A}$ has a congruence with two equivalence classes. By Lemma 7.4, it suffices to show that an equivalence relation of index two is primitive positive definable. Here is the primitive positive definition:

$$\exists u, v\, \big(E(x, u) \wedge E(y, u) \wedge E(x, v) \wedge E(y, v) \wedge E(u, v)\big)$$

The equivalence classes of this relation are precisely $\{a_1, \dots, a_4\}$ and $\{b_1, \dots, b_4\}$. △

Example 23. Let $\mathbf{A}$ be the algebra with domain $A := S_3 = \big\{\mathrm{id}, (231), (312), (12), (23), (13)\big\}$ (the symmetric group on three elements), and a single binary operation, the composition function of permutations. Note that $\mathbf{A}$ has the subalgebra induced by $\big\{\mathrm{id}, (123), (321)\big\}$. Also note that $\mathbf{A}$ homomorphically maps to $(\{0, 1\}, +)$ where $+$ is addition modulo 2: the preimage of 0 is $\{\mathrm{id}, (123), (321)\}$ and the preimage of 1 is $\{(12), (23), (13)\}$. △

When $\mathbf{A}$ is a τ-algebra, and $h\colon A \to B$ is a mapping such that the kernel of h is a congruence of $\mathbf{A}$, we define the *quotient algebra* $\mathbf{A}/h$ *of* $\mathbf{A}$ *under* h to be the algebra with domain $h(A)$ where

$$f^{\mathbf{A}/h}(h(a_1), \ldots, h(a_k)) = h(f^{\mathbf{A}}(a_1, \ldots, a_k))$$

where $a_1, \ldots, a_k \in A$ and $f \in \tau$ is k-ary. This is well-defined since the kernel of h is preserved by all operations of $\mathbf{A}$. Note that h is a surjective homomorphism from $\mathbf{A}$ to $\mathbf{A}/h$. The following is well known (see e.g. Theorem 6.3 in [22]).

Lemma 7.6. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be algebras with the same signature, and let* $h\colon \mathbf{A} \to \mathbf{B}$ *be a homomorphism. Then the image of any subalgebra* $\mathbf{A}'$ *of* $\mathbf{A}$ *under* h *is a subalgebra of* $\mathbf{B}$*, and the preimage of any subalgebra* $\mathbf{B}'$ *of* $\mathbf{B}$ *under* h *is a subalgebra of* $\mathbf{A}$*.*

Proof. Let $f \in \tau$ be k-ary. Then for all $a_1, \ldots, a_k \in A'$,

$$f^{\mathbf{B}}(h(a_1), \ldots, h(a_k)) = h(f^{\mathbf{A}}(a_1, \ldots, a_k)) \in h(A') \ ,$$

so $h(A')$ is a subalgebra of $\mathbf{B}$. Now suppose that $h(a_1), \ldots, h(a_k)$ are elements of B'; then $f^{\mathbf{B}}(h(a_1), \ldots, h(a_k)) \in B'$ and hence $h(f^{\mathbf{A}}(a_1, \ldots, a_k)) \in B'$. So, $f^{\mathbf{A}}(a_1, \ldots, a_k) \in h^{-1}(B')$ which shows that $h^{-1}(B')$ induces a subalgebra of $\mathbf{A}$. □

7.3 Pseudovarieties and Varieties

Varieties are a fascinatingly powerful concept to study classes of algebras. The fundamental result about varieties is Birkhoff's theorem, which links varieties with equational theories (Section 7.4). By Birkhoff's theorem, there is also a close relationship between varieties and the concept of an *abstract clone* (Section 7.5).

If $\mathcal{K}$ is a class of algebras of the same signature, then

- $\mathrm{P}(\mathcal{K})$ denotes the class of all products of algebras from $\mathcal{K}$.
- $\mathrm{P}^{\mathrm{fin}}(\mathcal{K})$ denotes the class of all finite products of algebras from $\mathcal{K}$.
- $\mathrm{S}(\mathcal{K})$ denotes the class of all subalgebras of algebras from $\mathcal{K}$.
- $\mathrm{H}(\mathcal{K})$ denotes the class of all homomorphic images of algebras from $\mathcal{K}$.

Note that closure under homomorphic images implies in particular closure under isomorphism. For the operators P, $\mathrm{P}^{\mathrm{fin}}$, S and H we often omit the brackets when applying them to single singleton classes that just contain one algebra, i.e., we write $\mathrm{H}(\mathbf{A})$ instead of $\mathrm{H}(\{\mathbf{A}\})$. The elements of $\mathrm{HS}(\mathbf{A})$ are also called the *factors* of $\mathbf{A}$.

A class $\mathcal{V}$ of algebras with the same signature τ is called a *pseudovariety* if $\mathcal{V}$ contains all homomorphic images, subalgebras, and direct products of algebras in $\mathcal{V}$, i.e., $\mathrm{H}(\mathcal{V}) = \mathrm{S}(\mathcal{V}) = \mathrm{P}^{\mathrm{fin}}(\mathcal{V}) = \mathcal{V}$. The class $\mathcal{V}$ is called a *variety* if $\mathcal{V}$ also contains all (finite and infinite) products of algebras in $\mathcal{V}$. So the only difference between pseudovarieties and varieties is that pseudovarieties need not be closed under direct products of infinite cardinality. The smallest pseudovariety (variety) that contains an algebra $\mathbf{A}$ is called the pseudovariety (variety) *generated* by $\mathbf{A}$.

Lemma 7.7 (HSP lemma)**.** *Let* $\mathbf{A}$ *be an algebra.*

- *The pseudovariety generated by* $\mathbf{A}$ *equals* $\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})$.
- *The variety generated by* $\mathbf{A}$ *equals* $\mathrm{HSP}(\mathbf{A})$.

Proof. Clearly, $\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})$ is contained in the pseudovariety generated by $\mathbf{A}$, and $\mathrm{HSP}(\mathbf{A})$ is contained in the variety generated by $\mathbf{A}$. For the converse inclusion, it suffices to verify that $\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})$ is closed under H, S, and $\mathrm{P}^{\mathrm{fin}}$. It is clear that $\mathrm{H}(\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})) = \mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})$. The second part of Lemma 7.6 implies that $\mathrm{S}(\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})) \subseteq \mathrm{HS}(\mathrm{SP}^{\mathrm{fin}}(\mathbf{A})) = \mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})$. Finally,

$$\mathrm{P}^{\mathrm{fin}}(\mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})) \subseteq \mathrm{H}\,\mathrm{P}^{\mathrm{fin}}\,\mathrm{S}\,\mathrm{P}^{\mathrm{fin}}(\mathbf{A}) \subseteq \mathrm{HSP}^{\mathrm{fin}}\,\mathrm{P}^{\mathrm{fin}}(\mathbf{A}) = \mathrm{HSP}^{\mathrm{fin}}(\mathbf{A})\ .$$

The proof that $\mathrm{HSP}(\mathbf{A})$ is closed under H, S, and P is analogous. □

Exercices.

85. Let B be a subuniverse of an algebra $\mathbf{A}$ generated by $S \subseteq A$. Show that an element $a \in A$ belongs to B if and only if there exists a term $t(x_1, \ldots, x_k)$ and elements $s_1, \ldots, s_k \in S$ such that $a = t^{\mathbf{A}}(s_1, \ldots, s_k)$.
86. Show that the operators HS and SH are distinct.
87. Show that the operators SP and PS are distinct.

Pseudo-varieties are linked to the logic concepts from Section 4.

Theorem 7.8. *Let* $\mathfrak{C}$ *be a finite structure with polymorphism algebra* $\mathbf{C}$*. Then* $\mathfrak{B} \in \mathrm{I}(\mathfrak{C})$ *if and only if there exists* $\mathbf{B} \in \mathrm{HSP}^{\mathrm{fin}}(\mathbf{C})$ *such that* $\mathrm{Clo}(\mathbf{B}) \subseteq \mathrm{Pol}(\mathfrak{B})$.

Proof. We only prove the 'if' part of the statement here; the proof of the 'only if' part is similarly easy. There exists a finite number $d \geq 1$, a subalgebra $\mathbf{D}$ of $\mathbf{C}^d$, and a surjective homomorphism h from $\mathbf{D}$ to $\mathbf{B}$. We claim that $\mathfrak{B}$ has a primitive positive interpretation I of dimension d in $\mathfrak{C}$. All operations of $\mathbf{C}$ preserve D (viewed as a d-ary relation over $\mathfrak{C}$), since $\mathbf{D}$ is a subalgebra of $\mathbf{C}^d$. By Theorem 5.2, this implies that D has a primitive positive definition $\delta(x_1, \ldots, x_d)$ in $\mathfrak{C}$, which becomes the domain formula δ_I of I. As coordinate map we choose the mapping h. Since h is an algebra homomorphism, the kernel K of h is a congruence of $\mathbf{D}$. It follows that K, viewed as a $2d$-ary relation over C, is preserved by all operations from $\mathbf{C}$. Theorem 5.2 implies that K has a primitive positive definition in $\mathfrak{C}$. This definition becomes the formula $=_I$. Finally, let R be a relation of $\mathfrak{B}$ and let f be a function symbol from the signature of $\mathbf{B}$. By assumption, $f^{\mathbf{B}}$ preserves R. It is easy to verify that then $f^{\mathbf{C}}$ preserves $h^{-1}(R)$. Hence, all polymorphisms of $\mathfrak{C}$ preserve $h^{-1}(R)$, and the relation $h^{-1}(R)$ has a primitive positive definition in $\mathfrak{C}$ (Theorem 5.2), which becomes the defining formula for the atomic formula $R(x_1, \ldots, x_k)$ in I. This concludes our construction of the primitive positive interpretation I of $\mathfrak{B}$ in $\mathfrak{C}$. □

7.4 Birkhoff's Theorem

A sentence in a functional signature is called an *identity* if it is of the form

$$\forall x_1, \ldots, x_n \colon s = t$$

where s and t are τ-terms. We present Birkhoff's theorem for the special case of varieties generated by a single finite algebra.

Theorem 7.9 (Birkhoff [8]; see e.g. [44] or [22])**.** *Let τ be a functional signature, and* $\mathbf{A}$ *and* $\mathbf{B}$ *finite algebras with signature τ. Then the following are equivalent.*

1. *All identities that hold in* $\mathbf{B}$ *also hold in* $\mathbf{A}$*;*
2. $\mathbf{A} \in \mathrm{HSP}^{\mathrm{fin}}(\mathbf{B})$.
3. $\mathbf{A} \in \mathrm{HSP}(\mathbf{B})$*;*

Proof. Trivially, 2. implies 3. To show that 3. implies 1., let $\phi = \forall x_1, \dots, x_n.\, s = t$ be an identity that holds in $\mathbf{B}$. Then ϕ is preserved in powers $\mathbf{A} = \mathbf{B}^I$ of $\mathbf{B}$. To see this, let $a_1, \dots, a_n \in A$ be arbitrary. Since $\mathbf{B} \models \phi$ we have $s^{\mathbf{B}}(a_1[j], \dots, a_n[j]) = t^{\mathbf{B}}(a_1[j], \dots, a_n[j])$ for all $j \in I$, and thus $s^{\mathbf{A}}(a_1, \dots, a_n) = t^{\mathbf{A}}(a_1, \dots, a_n)$ by the definition of products. Since $a_1, \dots, a_n$ were chosen arbitrarily, we have $\mathbf{A} \models \phi$. Moreover, ϕ is true in subalgebras of algebras that satisfy ϕ (this is true for universal sentences in general). Finally, suppose that $\mathbf{B}$ is an algebra that satisfies ϕ, and μ is a surjective homomorphism from $\mathbf{B}$ to some algebra $\mathbf{A}$. Let $a_1, \dots, a_n \in A$; by surjectivity of μ we can choose $b_1, \dots, b_n$ such that $\mu(b_i) = a_i$ for all $i \leq n$. Then

$$\begin{aligned} s^{\mathbf{B}}(b_1, \dots, b_n) = t^{\mathbf{B}}(b_1, \dots, b_n) \Rightarrow \quad & \mu(s^{\mathbf{B}}(b_1, \dots, b_n)) = \mu(t^{\mathbf{B}}(b_1, \dots, b_n)) \\ \Rightarrow \quad & t^{\mathbf{A}}(\mu(b_1), \dots, \mu(b_n)) = s^{\mathbf{A}}(\mu(b_1), \dots, \mu(b_n)) \\ \Rightarrow \quad & t^{\mathbf{A}}(a_1, \dots, a_n) = s^{\mathbf{A}}(a_1, \dots, a_n)\ . \end{aligned}$$

1. implies 2.: Let $a_1, \dots, a_k$ be the elements of $\mathbf{A}$, define $m := |B|^k$, and let C be B^k. Let $c^1, \dots, c^m$ be the elements of C; write c_i for $(c_i^1, \dots, c_i^m)$. Let $\mathbf{S}$ be the smallest subalgebra of $\mathbf{B}^m$ that contains $c_1, \dots, c_k$; so the elements of $\mathbf{S}$ are precisely those of the form $t^{\mathbf{B}^m}(c_1, \dots, c_k)$, for a k-ary τ-term t. Define $\mu \colon S \to A$ by

$$\mu(t^{\mathbf{B}^m}(c_1, \dots, c_k)) := t^{\mathbf{A}}(a_1, \dots, a_k).$$

Claim 1: μ is well-defined. Suppose that $t^{\mathbf{B}^m}(c_1, \dots, c_k) = s^{\mathbf{B}^m}(c_1, \dots, c_k)$; then $t^{\mathbf{B}} = s^{\mathbf{B}}$ by the choice of S, and by assumption we have $t^{\mathbf{A}}(a_1, \dots, a_k) = s^{\mathbf{A}}(a_1, \dots, a_k)$.

Claim 2: μ is surjective. For all $i \leq k$, the element c_i is mapped to a_i.

Claim 3: μ is a homomorphism from $\mathbf{S}$ to $\mathbf{A}$. Let $f \in \tau$ be of arity n and let $s_1, \dots, s_n \in S$. For $i \leq n$, write $s_i = t_i^{\mathbf{S}}(c_1, \dots, c_k)$ for some τ-term t_i. Then

$$\begin{aligned} \mu\big(f^{\mathbf{S}}(s_1, \dots, s_n)\big) &= \mu\big(f^{\mathbf{S}}(t_1^{\mathbf{S}}(c_1, \dots, c_k), \dots, t_n^{\mathbf{S}}(c_1, \dots, c_k))\big) \\ &= \mu\big(f^{\mathbf{S}}(t_1^{\mathbf{S}}, \dots, t_n^{\mathbf{S}})(c_1, \dots, c_k)\big) \\ &= \mu\big((f(t_1, \dots, t_n))^{\mathbf{S}}(c_1, \dots, c_k)\big) \\ &= \big(f(t_1, \dots, t_n)\big)^{\mathbf{A}}(a_1, \dots, a_k) \\ &= f^{\mathbf{A}}\big(t_1^{\mathbf{A}}(a_1, \dots, a_k), \dots, t_n^{\mathbf{A}}(a_1, \dots, a_k)\big) \\ &= f^{\mathbf{A}}(\mu(s_1), \dots, \mu(s_n)). \end{aligned}$$

Therefore, $\mathbf{A}$ is the homomorphic image of the subalgebra $\mathbf{S}$ of $\mathbf{B}^m$, and so $\mathbf{A} \in \mathrm{HSP}^{\mathrm{fin}}(\mathbf{B})$. □

Theorem 7.9 is important for analysing the constraint satisfaction problem for a structure $\mathfrak{B}$, since it can be used to transform the 'negative' statement of not interpreting certain finite structures into a 'positive' statement of having polymorphisms satisfying non-trivial identities: this will be the content of Section 7.5 and Section 7.6.

7.5 (Abstract) Clones

Clones (in the literature often *abstract clones*) relate to operation clones in the same way as (abstract) groups relate to permutation groups: the elements of a clone correspond to the functions of an operation clone, and the signature contains composition symbols to code how functions compose. Since an operation clone contains functions of various arities, a clone will be formalized as a multi-sorted structure, with a sort for each arity.

Definition 7.10. A *clone* $\mathbf{C}$ is a multi-sorted structure with sorts $\{C^{(i)} \mid i \in \mathbb{N}\}$ and the signature $\{\pi_i^k \mid 1 \leq i \leq k\} \cup \{\mathrm{comp}_l^k \mid k, l \geq 1\}$. The elements of the sort $C^{(k)}$ will be called the *k-ary operations* of $\mathbf{C}$. We denote a clone by

$$\mathbf{C} = (C^{(0)}, C^{(1)}, \ldots; (\pi_i^k)_{1 \leq i \leq k}, (\mathrm{comp}_l^k)_{k,l \geq 1})$$

and require that π_i^k is a constant in $C^{(k)}$, and that $\mathrm{comp}_l^k \colon C^{(k)} \times (C^{(l)})^k \to C^{(l)}$ is an operation of arity $k+1$. Moreover, it holds that

$$\mathrm{comp}_k^k(f, \pi_1^k, \ldots, \pi_k^k) = f \tag{5}$$

$$\mathrm{comp}_l^k(\pi_i^k, f_1, \ldots, f_k) = f_i \tag{6}$$

$$\mathrm{comp}_l^k\left(f, \mathrm{comp}_l^m(g_1, h_1, \ldots, h_m), \ldots, \mathrm{comp}_l^m(g_k, h_1, \ldots, h_m)\right) = \mathrm{comp}_l^m\left(\mathrm{comp}_m^k(f, g_1, \ldots, g_k), h_1, \ldots, h_m\right). \tag{7}$$

The final equation generalises associativity in groups and monoids, and we therefore refer to it by *associativity*. We also write $f(g_1, \ldots, g_k)$ instead of $\mathrm{comp}_l^k(f, g_1, \ldots, g_k)$ when l is clear from the context. So associativity might be more readable as

$$f(g_1(h_1, \ldots, h_m), \ldots, g_k(h_1, \ldots, h_m)) = f(g_1, \ldots, g_k)(h_1, \ldots, h_m).$$

Every operation clone $\mathscr{C}$ gives rise to an abstract clone $\mathbf{C}$ in the obvious way: $\pi_i^k \in C^{(k)}$ denotes the k-ary i-th projection in $\mathscr{C}$, and $\mathrm{comp}_l^k(f, g_1, \ldots, g_k) \in C^{(l)}$ denotes the composed function $(x_1, \ldots, x_l) \mapsto f(g_1(x_1, \ldots, x_l), \ldots, g_k(x_1, \ldots, x_l)) \in \mathscr{C}$.

Example 24. $\forall x_1, x_2 \colon f(x_1, x_2) = f(x_2, x_1)$ holds in an algebra $\mathbf{A}$ if and only if

$$\mathrm{Clo}(\mathbf{A}) \models \mathrm{comp}_2^2(f^{\mathbf{A}}, \pi_1^2, \pi_2^2) = \mathrm{comp}_2^2(f^{\mathbf{A}}, \pi_2^2, \pi_1^2). \qquad \triangle$$

In the following, we will also use the term 'abstract clone' in situations where we want to stress that we are working with a clone and *not* with an operation clone.

Definition 7.11. Let $\mathbf{C}$ and $\mathbf{D}$ be clones. A function $\xi \colon C \to D$ is called a *(clone) homomorphism* if

- ξ preserves arities of functions, i.e., $\xi(C^{(i)}) \subseteq D^{(i)}$ for all $i \in \mathbb{N}$;
- $\xi((\pi_i^k)^{\mathbf{C}}) = (\pi_i^k)^{\mathbf{D}}$ for all $1 \leq i \leq k$;
- $\xi(f(g_1, \ldots, g_n)) = \xi(f)(\xi(g_1), \ldots, \xi(g_n))$ for all $n, m \geq 1$, $f \in C^{(n)}$, and $g_1, \ldots, g_n \in C^{(m)}$.

Example 25. We write **Proj** for the abstract clone of an algebra with at least two elements all of whose operations are projections; note that any such algebra has the same abstract clone (up to isomorphism), and that **Proj** has a homomorphism into any other clone. $\triangle$

Example 26. All abstract clones of an algebra on a one-element are isomorphic, too, but of course not isomorphic to **Proj**. Any clone homomorphically maps to this trivial clone. △

Proposition 7.12 (Formulation of Birkhoff's theorem for clones)**.** *Let $\mathscr{C}$ and $\mathscr{D}$ be operation clones on finite sets. Then the following are equivalent.*

1. *There is a surjective clone homomorphism from $\mathscr{C}$ to $\mathscr{D}$;*
2. *there are algebras $\mathbf{A}$ and $\mathbf{B}$ with the same signature τ such that $\mathrm{Clo}(\mathbf{A}) = \mathscr{D}$, $\mathrm{Clo}(\mathbf{B}) = \mathscr{C}$, and all universal conjunctive τ-sentences that hold in $\mathbf{B}$ also hold in $\mathbf{A}$;*
3. *there are algebras $\mathbf{A}$ and $\mathbf{B}$ with the same signature such that $\mathrm{Clo}(\mathbf{A}) = \mathscr{D}$, $\mathrm{Clo}(\mathbf{B}) = \mathscr{C}$, and $\mathbf{A} \in \mathrm{HSP}^{\mathrm{fin}}(\mathbf{B})$ (equivalently, $\mathbf{A} \in \mathrm{HSP}(\mathbf{B})$).*

In the study of the complexity of CSPs, the equivalence between (1) and (3) in the above is the most relevant, since (3) is related to our most important tool to prove NP-hardness of CSPs (because of the link between pseudovarieties and primitive positive interpretations from Theorem 7.8), and since (1) is the universal-algebraic property that will be used in the following (see e.g. Theorem 7.18 below).

The following lemma is central for our applications of abstract clones when studying the complexity of CSPs.

Lemma 7.13. *Let $\mathbf{C}$ be a clone and let $\mathbf{F}$ be the clone of a finite algebra such that there is no clone homomorphism from $\mathbf{C}$ to $\mathbf{F}$. Then there is a primitive positive sentence in the language τ of (abstract) clones that holds in $\mathbf{C}$ but not in $\mathbf{F}$.*

Proof. Let $\mathbf{E}$ be the expansion of $\mathbf{C}$ by constant symbols such that every element e of $\mathbf{E}$ is named by a constant c_e. Let V be the set of atomic sentences that hold in $\mathbf{E}$. Let U be the first-order theory of $\mathbf{F}$. Suppose that $U \cup V$ has a model $\mathbf{M}$. There might be elements in $\mathbf{M}$ outside of $\bigcup_i M^{(i)}$. But the τ-reduct of the restriction of $\mathbf{M}$ to $\bigcup_i M^{(i)}$ must be isomorphic to $\mathbf{F}$, since each of the $M^{(i)}$ is finite; we identify it with $\mathbf{F}$. Note that for all constants c_e we have that $c_e^{\mathbf{M}} \in \mathbf{F}$. Since $\mathbf{M}$ satisfies all atomic formulas that hold in $\mathbf{E}$, we have that the mapping $e \mapsto c_e^{\mathbf{M}}$, for e an element of $\mathbf{E}$, is a homomorphism from $\mathbf{C}$ to $\mathbf{F}$, in contradiction to our assumptions.

So $U \cup V$ is unsatisfiable, and by compactness of first-order logic there exists a finite subset V' of V such that $V' \cup U$ is unsatisfiable. Replace each of the new constant symbols in V' by an existentially quantified variable; then the conjunction of the resulting sentences from V is a primitive positive sentence, and it must be false in $\mathbf{F}$. □

A set of identities Σ is called *trivial* if there exists an algebra $\mathbf{A}$ that satisfies Σ and $\mathrm{Clo}(\mathbf{A})$ is isomorphic to **Proj**.

Corollary 7.14. *Let $\mathbf{A}$ be an algebra. If there is no clone homomorphism from $\mathrm{Clo}(\mathbf{A})$ to* **Proj***, then there exists a non-trivial finite set of identities that holds in $\mathbf{A}$.*

7.6 Taylor Terms

The following goes back to Walter Taylor [61]. We slightly deviate from the historic definition in that we do not require idempotency – this allows us to give stronger formulations of several results in the following.

Definition 7.15 (Taylor operations)**.** A function $f\colon B^n \to B$, for $n \geq 2$, is called a *Taylor operation* if for each $1 \leq i \leq n$ there are variables $z_1, \ldots, z_n, z'_1, \ldots, z'_n \in \{x, y\}$ with $z_i \neq z'_i$ such that for all $x, y \in B$

$$f(z_1, \ldots, z_n) = f(z'_1, \ldots, z'_n)\,.$$

Examples for Taylor operations are binary commutative operations, majority operations, and Maltsev operations. We do not insist on idempotency for Taylor operations. Note that an n-ary operation f is Taylor if and only if it satisfies a set of n equations that can be written as

$$f\begin{pmatrix} x & ? & ? & \cdots & ? \\ ? & x & ? & & \vdots \\ \vdots & ? & \ddots & \ddots & \vdots \\ \vdots & & \ddots & x & ? \\ ? & \cdots & \cdots & ? & x \end{pmatrix} = f\begin{pmatrix} y & ? & ? & \cdots & ? \\ ? & y & ? & & \vdots \\ \vdots & ? & \ddots & \ddots & \vdots \\ \vdots & & \ddots & y & ? \\ ? & \cdots & \cdots & ? & y \end{pmatrix}$$

where f is applied row-wise and ? stands for either x or y.

Definition 7.16 (Taylor terms)**.** Let $\mathbf{B}$ be a τ-algebra. A *Taylor term* of $\mathbf{B}$ is a τ-term $t(x_1, \ldots, x_n)$, for $n \geq 2$, such that $t^{\mathbf{B}}$ is a Taylor operation.

Note that a term $t(x_1, \ldots, x_n)$ is a Taylor term in $\mathbf{B}$ if and only if $\mathrm{Clo}(\mathbf{B}) \models \Phi_n(t^{\mathbf{B}})$, where $\Phi_n(f)$ is the sentence

$$\bigwedge_{i \leq n} \exists z, z' \in \{\pi^2_1, \pi^2_2\}^n \left(z_i \neq z'_i \wedge \mathrm{comp}^n_2(f, z_1, \ldots, z_n) = \mathrm{comp}^n_2(f, z'_1, \ldots, z'_n)\right). \tag{8}$$

The following lemma is entirely obvious and I don't know why the formal proof is so long, there must be a better way of presenting it (if giving a proof is necessary at all).

Lemma 7.17. *$\Phi_n(f)$ is equivalent to*

$$\bigwedge_{i \leq n} \exists v, v' \in \{\pi^n_1, \ldots, \pi^n_n\}^n \left(v_i \neq v'_i \wedge \mathrm{comp}^n_n(f, v_1, \ldots, v_n) = \mathrm{comp}^n_n(f, v'_1, \ldots, v'_n)\right).$$

Proof. Let $i \in \{1, \ldots, n\}$ be arbitrary. Suppose that $z, z' \in \{\pi^2_1, \pi^2_2\}^n$ are as in Equation 1. Then define $v, v' \in \{\pi^n_1, \ldots, \pi^n_n\}^n$ as follows: if $z_j = \pi^2_k$ for $k \in \{1, 2\}$, then $v_j := \pi^n_k$, and similarly if $z'_j = \pi^2_k$ for $k \in \{1, 2\}$, then $v'_j := \pi^n_k$. Then v and v' witness that the formula given in the statement is true: $z_i \neq z'_i$ implies that $v_i \neq v'_i$. Moreover, $\mathrm{comp}^n_2(f, z_1, \ldots, z_i) = \mathrm{comp}^n_2(f, z'_1, \ldots, z'_n)$ implies that

$$\mathrm{comp}^2_n(\mathrm{comp}^n_2(f, z_1, \ldots, z_n), \pi^n_1, \pi^n_2) = \mathrm{comp}^2_n(\mathrm{comp}^n_2(f, z'_1, \ldots, z'_n), \pi^n_1, \pi^n_2)$$

We compute

$$\begin{aligned} & \mathrm{comp}^2_n(\mathrm{comp}^n_2(f, z_1, \ldots, z_n), \pi^n_1, \pi^n_2) \\ = \ & \mathrm{comp}^2_n(f, \mathrm{comp}^2_n(z_1, \pi^n_1, \pi^n_2), \ldots, \mathrm{comp}^2_n(z_n, \pi^n_1, \pi^n_2)) \\ = \ & \mathrm{comp}^n_n(f, v_1, \ldots, v_n) \end{aligned}$$

and similarly $\mathrm{comp}_n^2(\mathrm{comp}_2^n(f, z'_1, \dots, z'_n), \pi_1^n, \pi_2^n) = \mathrm{comp}_n^n(f, v'_1, \dots, v'_n)$ and hence

$$\mathrm{comp}_n^n(f, v_1, \dots, v_n) = \mathrm{comp}_n^n(f, v'_1, \dots, v'_n)$$

as required.

Conversely, let $i \leq n$, and suppose that $v, v' \in \{\pi_1^n, \dots, \pi_n^n\}^n$ are as in the formula above. Define $z, z' \in \{\pi_1^2, \pi_2^2\}^n$ as follows. If $v_j = v_i$ define $z_j := \pi_1^2$, and $z_j := \pi_2^2$ otherwise. Similarly, if $v'_j = v'_i$ then $z'_j := \pi_1^2$, and $z'_j := \pi_2^2$ otherwise. Then z and z' witness that $\Phi_n(f)$ is true: since $v_i \neq v'_i$ we have $z_i \neq z'_i$, and

$$\begin{aligned} & \mathrm{comp}_2^n(\mathrm{comp}_n^n(f, v_1, \dots, v_n), z_1, \dots, z_n) \\ = \ & \mathrm{comp}_2^n(f, \mathrm{comp}_2^n(v_1, z_1, \dots, z_n), \dots, \mathrm{comp}_2^n(v_n, z_1, \dots, z_n)) \\ = \ & \mathrm{comp}_2^n(f, z_1, \dots, z_n) \end{aligned}$$

Similarly, $\mathrm{comp}_2^n(\mathrm{comp}_n^n(f, v'_1, \dots, v'_n), z'_1, \dots, z'_n) = \mathrm{comp}_2^n(f, z'_1, \dots, z'_n)$. Hence,

$$\mathrm{comp}_2^n(f, z_1, \dots, z_n) = \mathrm{comp}_2^n(f, z'_1, \dots, z'_n)$$

as required. □

Walter Taylor did not just introduce Taylor operations, but he proved a beautiful statement about their existence. The following is a slightly expanded presentation of the proof of Lemma 9.4 in [43].

Theorem 7.18. *Let* $\mathbf{B}$ *be an idempotent algebra. Then the following are equivalent.*

(1) there is no homomorphism from $\mathrm{Clo}(\mathbf{B})$ *to* **Proj***;*

(2) $\mathbf{B}$ *has a Taylor term.*

Proof. To show the easy implication from (2) to (1), suppose for contradiction that there is a homomorphism ξ from $\mathrm{Clo}(\mathbf{B})$ to **Proj**. Let f be the element of $\mathrm{Clo}(\mathbf{B})$ that is denoted by $t^{\mathbf{B}}$. By definition of **Proj** we have $\xi(f) = \pi_l^n$ for some $l \leq n$. By assumption, $\mathbf{B}$ satisfies

$$\forall x, y.\ t(z_1, \dots, z_n) = t(z'_1, \dots, z'_n) \tag{9}$$

for $z_1, \dots, z_n, z'_1, \dots, z'_n \in \{x, y\}$ such that $z_l \neq z'_l$. Then $\mathrm{Clo}(\mathbf{B})$ satisfies

$$\mathrm{comp}_2^n(f, \pi_{i_1}^2, \dots, \pi_{i_n}^2) = \mathrm{comp}_2^n(f, \pi_{j_1}^2, \dots, \pi_{j_n}^2) \tag{10}$$

for $i_1, \dots, i_n, j_1, \dots, j_n \in \{1, 2\}$ such that $i_l = 1$ if and only if $z_l = x$, $j_l = 1$ if and only if $z'_l = x$, and $i_l \neq j_l$. Since $\xi(f) = \pi_l^n$ we therefore obtain that $\pi_1^2 = \pi_2^2$, which does not hold in **Proj**, a contradiction.

To show the implication from (1) to (2), suppose that $\mathrm{Clo}(\mathbf{B})$ does not homomorphically map to **Proj**. Then Lemma 7.13 implies that there is a primitive positive sentence in the language of clones that holds in $\mathrm{Clo}(\mathbf{B})$ but not in **Proj**. Note that by introducing new existentially quantified variables we can assume that this sentence is of the form $\exists u_1, \dots, u_r.\ \phi$ where ϕ is a conjunction of atoms of the form $y = \mathrm{comp}_l^m(x_0, x_1, \dots, x_m)$ or of the form $y = \pi_l^m$ for $y, x_0, x_1, \dots, x_m \in \{u_1, \dots, u_r\}$ and $l, m \in \mathbb{N}$. For example the equation

$$\mathrm{comp}_2^2(u, \mathrm{comp}_2^2(u, \pi_2^2, \pi_1^2)) = \mathrm{comp}_2^2(\mathrm{comp}_2^2(u, \pi_2^2, \pi_1^2), u)$$

involving the free variable u is equivalent to

$$\begin{aligned}\exists u_1, u_2, u_3\ (u_1 &= \mathrm{comp}^2_2(u, \pi^2_2, \pi^2_1)\\ \wedge\ u_2 &= \mathrm{comp}^2_2(f, u, u_1)\\ \wedge\ u_3 &= \mathrm{comp}^2_2(u, u_1, u)\\ \wedge\ u_2 &= u_3)\ .\end{aligned}$$

For $l, m \in \mathbb{N}$ and $n = lm$ we write $f * g$ as a shortcut for

$$\mathrm{comp}^l_n(f, \mathrm{comp}^m_n(g, \pi^n_1, \dots, \pi^n_m), \dots, \mathrm{comp}^m_n(g, \pi^n_{(l-1)m+1}, \dots, \pi^n_n))\ .$$

Note that

$$\mathrm{Clo}(\mathbf{B}) \models \big(g = \mathrm{comp}^n_l(f * g, \pi^l_1, \dots, \pi^l_l, \pi^l_1, \dots, \pi^l_l, \dots, \pi^l_1, \dots, \pi^l_l)\big) \tag{11}$$

$$\text{and } \mathrm{Clo}(\mathbf{B}) \models \big(f = \mathrm{comp}^n_m(f * g, \pi^l_1, \dots, \pi^l_1, \pi^l_2, \dots, \pi^l_2, \dots, \pi^l_l, \dots, \pi^l_l)\big) \tag{12}$$

since $\mathbf{B}$ is idempotent. For $i \in \{1, \dots, r\}$, let $k_i \in \mathbb{N}$ be the arity of u_i, and define

$$u := u_1 * (u_2 * (\dots (u_{r-1} * u_r) \dots))\ . \tag{13}$$

Observe that for each i we can obtain u_i from u by composing u with projections. In order to formalise this, we need a compact notation for strings of arguments consisting of projection constants. In this notation, (11) reads as $y = \mathrm{comp}^n_l(x * y, (1, \dots, l)^m)$, and (12) reads as $x = \mathrm{comp}^n_m(x * y, 1^m, \dots, l^m)$. Similarly, setting $k := k_1 \cdots k_r$ we have

$$u_i = \mathrm{comp}^k_{k_i}(u, \bar{p}_i) \text{ where } \bar{p}^i := (1^{k_1 \cdots k_{i-1}}, \dots, k_i^{k_1 \cdots k_{i-1}})^{k_{i+1} \cdots k_n} \in \{\pi^{k_i}_1, \dots, \pi^{k_i}_{k_i}\}^k.$$

Let $n := k^2$. Then every term of the form

$$t := \mathrm{comp}^{k_i}_l(u_i, u_{i_1}, \dots, u_{i_{k_i}})$$

can be written as

$$\mathrm{comp}^n_l(u * u, \bar{q}^t)$$

for $\bar{q}^t \in \{\pi^l_1, \dots, \pi^l_l\}^n$ obtained from $\bar{p}^i$ by replacing character $l \leq k_i$ by the string $\bar{p}^{i_l}$; see Figure 10 for an illustration. Similarly, every term of the form $t := u_i$ can be written as $\mathrm{comp}^n_{k_i}(u * u, \bar{q}^t)$ for $\bar{q}^t \in \{\pi^{k_i}_1, \dots, \pi^{k_i}_{k_i}\}^n$ obtained by k times concatenating $\bar{p}^i$ with itself. In this way, every conjunct of ϕ of the form

$$u_j = \mathrm{comp}^{k_i}_{k_j}(u_i, u_{i_1}, \dots, u_{i_{k_i}})$$

can we written in the form

$$\mathrm{comp}^n_{k_j}(u * u, \bar{q}^{t_1}) = \mathrm{comp}^n_{k_j}(u * u, \bar{q}^{t_2})$$

for $t_1 := u_j$ and $t_2 := \mathrm{comp}^{k_i}_{k_j}(u_i, u_{i_1}, \dots, u_{i_{k_i}})$. Conjuncts of ϕ of the form $u_j = \pi^m_l$ can be rewritten similarly. Let ψ be the conjunction of all these equations; note that ϕ implies ψ. Let θ be the formula obtained from ψ by replacing each occurrence of $u * u$ by a variable symbol f. Note that the sentence $\exists f.\theta$ holds in $\mathrm{Clo}(\mathbf{B})$. It suffices to show that every $f \in \mathrm{Clo}(\mathbf{B})$ that satisfies θ is a Taylor operation.

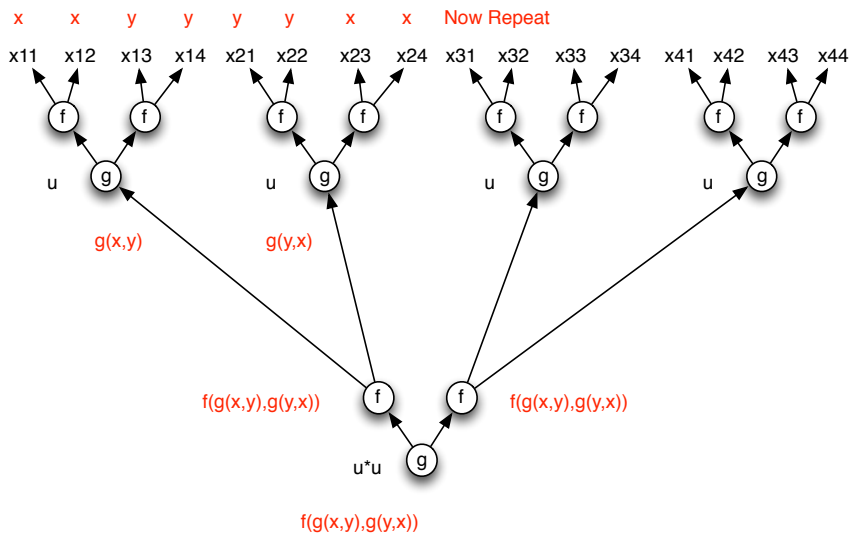

Figure 10: An illustration of $u * u$ and $u := f * g$ for the proof of Taylor's theorem, and how to recover the composed term $f(g(x,y),g(y,x))$ from $u * u$ by identifying arguments.

Suppose for contradiction that for $\ell \in \{1,\dots,n\}$ there are no $\bar{v}, \bar{v}' \in \{\pi^m_1,\dots,\pi^m_m\}^n$, for some $m \leq n$ with $\bar{v}_\ell \neq \bar{v}'_\ell$ such that $\mathrm{comp}^n_m(f,\bar{v}) = \mathrm{comp}^n_m(f,\bar{v}')$ (otherwise, f is a Taylor operation by the argument from Lemma 7.17).

We claim that the assignment ρ that maps u_s to $\mathrm{comp}^n_{k_s}(\pi^n_\ell, \bar{q}^{u_s}) = \bar{q}^{u_s}_\ell$, for $s \in \{1,\dots,r\}$, satisfies all conjuncts of ϕ, contradicting our assumptions. First note that $\bar{q}^{u_s}_\ell = \bar{p}^s_{\ell_1}$ where $\ell_1 \in \{1,\dots,k\}$ is such that $\ell = (\ell_1 - 1)k + \ell_2$ for some $\ell_2 \in \{1,\dots,k\}$. Indeed, consider a conjunct of ϕ

$$u_j = t \text{ for } t = \mathrm{comp}^{k_i}_{k_j}(u_i, u_{i_1}, \dots, u_{i_{k_i}}).$$

By construction, θ contains the conjunct

$$\mathrm{comp}(f, \bar{q}^{u_j}) = \mathrm{comp}(f, \bar{q}^t).$$

So by assumption we must have

$$\mathrm{comp}(\pi^n_\ell, \bar{q}^{u_j}) = \bar{q}^{u_j}_\ell = \bar{q}^t_\ell = \mathrm{comp}(\pi^n_\ell, \bar{q}^t). \tag{14}$$

Then ρ satisfies this conjunct since

$$\begin{aligned}
\mathrm{comp}^{k_i}_{k_j}(\rho(u_i), \rho(u_{i_1}), \dots) &= \mathrm{comp}^{k_i}_{k_j}(\bar{q}^{u_i}_\ell, \bar{q}^{u_{i_1}}_\ell, \dots) \\
&= \mathrm{comp}^{k_i}_{k_j}(\bar{p}^i_{\ell_1}, \bar{p}^{i_1}_{\ell_1}, \dots) \\
&= \bar{q}^t_\ell && \text{(by definition of } \bar{q}^t) \\
&= \bar{q}^{u_j}_\ell && \text{(by (14))} \\
&= \rho(u_j) && \square
\end{aligned}$$

Lemma 7.19. *Let $\mathfrak{B}$ and $\mathfrak{C}$ be homomorphically equivalent structures. Then $\mathfrak{B}$ has a Taylor polymorphism if and only if $\mathfrak{C}$ has a Taylor polymorphism.*

Proof. Let h be a homomorphism from $\mathfrak{B}$ to $\mathfrak{C}$, and g be a homomorphism from $\mathfrak{C}$ to $\mathfrak{B}$. Suppose that f is a Taylor polymorphism for $\mathfrak{B}$ of arity n. It suffices to show that the operation f' given by $(x_1, \dots, x_n) \mapsto h(f(g(x_1), \dots, g(x_n)))$ is a Taylor polymorphism of $\mathfrak{C}$. Indeed, for all $i \leq n$ we have that

$$\begin{aligned} f'(v_{1,i}, \dots, v_{n,i}) &= h\big(f(g(v_{1,i}), \dots, g(v_{n,i}))\big) \\ &= h\big(f(g(v'_{1,i}), \dots, g(v'_{n,i}))\big) \\ &= f'(v'_{1,i}, \dots, v'_{n,i}) \end{aligned}$$

□

Lemma 7.20. *Let $\mathfrak{B} \in \mathrm{C}(\mathfrak{C})$ and suppose that $\mathfrak{C}$ has a Taylor polymorphism. Then $\mathfrak{B}$ has a Taylor polymorphism.*

Proof. Suppose that $\mathfrak{C}$ is a core and that $\mathfrak{B}$ has been obtained by expanding $\mathfrak{C}$ by adding unary singleton relations. Let f be the Taylor polymorphism of $\mathfrak{C}$. Then $\hat{f}(x) := f(x, \dots, x)$ is an automorphism with inverse $i \in \mathrm{Pol}(\mathfrak{C})$. The function $i(f(x, \dots, x))$ is a Taylor operation and idempotent, and therefore a polymorphism of $\mathfrak{B}$. □

Alternatively, one can prove Lemma 7.20 using the fact that $\mathrm{C}(\mathfrak{C}) \subseteq \mathrm{H}(\mathrm{I}(\mathfrak{C}))$, the observation that any $\mathfrak{B}' \in \mathrm{I}(\mathfrak{C})$ has a Taylor polymorphism, and Lemma 7.19.

Corollary 7.21. *Let $\mathfrak{B}$ be a finite structure. Then the following are equivalent.*

- $K_3 \notin \mathrm{H}(\mathrm{I}(\mathfrak{B}))$
- *$\mathfrak{B}$ has a Taylor polymorphism.*

Proof. If $\mathfrak{B}$ has a Taylor polymorphism, then so has any structure in $\mathrm{I}(\mathfrak{B})$, and so has any structure in $\mathrm{H}(\mathrm{I}(\mathfrak{B}))$ by Lemma 7.19. But all polymorphisms of K_3 are essentially unary (Proposition 5.12), so $K_3 \notin \mathrm{H}(\mathrm{I}(\mathfrak{B}))$.

Conversely, suppose that $K_3 \notin \mathrm{H}(\mathrm{I}(\mathfrak{B}))$. If $\mathfrak{B}'$ is the core of $\mathfrak{B}$, and $\mathfrak{C}$ is the expansion of $\mathfrak{B}'$ by all unary singleton relations, then $\mathrm{C}(\mathrm{H}(\mathfrak{B})) \subseteq \mathrm{H}(\mathrm{I}(\mathfrak{B}))$ implies that K_3 is not primitive positive interpretable in $\mathfrak{C}$. Hence, the idempotent clone $\mathrm{Pol}(\mathfrak{C})$ does not homomorphically map to **Proj**. Theorem 7.18 shows that $\mathfrak{C}$ and thus also $\mathfrak{B}'$ must have a Taylor polymorphism. Lemma 7.19 implies that $\mathfrak{B}$ has a Taylor polymorphism. □

Corollary 7.22. *Let $\mathfrak{B}$ be a finite structure. Then $\mathfrak{B}$ has a Taylor polymorphism, or* $\mathrm{CSP}(\mathfrak{B})$ *is NP-hard.*

Proof. If $K_3 \in \mathrm{H}(\mathrm{I}(\mathfrak{B}))$ then $\mathrm{CSP}(\mathfrak{B})$ is NP-hard by Corollary 4.10. Otherwise, $\mathfrak{B}$ has a Taylor polymorphism by Corollary 7.21. □

7.7 The Tractability Conjecture

The following has been conjectured (in slightly different, but equivalent form) by Bulatov, Jeavons, and Krokhin in [21], and is known under the name *tractability conjecture*. Two solutions to this conjecture have been announced in 2017 by Bulatov [18] and by Zhuk [62].

Conjecture 2 (Tractability Conjecture)**.** *Let $\mathfrak{B}$ be a finite relational structure. If* $\mathrm{Pol}(\mathfrak{B})$ *contains a Taylor polymorphism, then* $\mathrm{CSP}(\mathfrak{B})$ *is in P.*

For idempotent algebras $\mathbf{A}$ there is yet another characterisation for the existence Taylor polymorphisms that follows from the following result of Bulatov and Jeavons [20] (Proposition 4.14).

Theorem 7.23. *Let* $\mathbf{B}$ *be an idempotent algebra. Then* $\mathrm{HSP}^{\mathrm{fin}}(\mathbf{B})$ *contains an at least 2-element algebra all of whose operations are projections if and only if* $\mathrm{HS}(\mathbf{B})$ *does.*

Proof. Suppose that $\mathbf{C} \in S(\mathbf{B}^d)$ for some $d \in \mathbb{N}$ has a congruence K such that all operations of $\mathbf{A} := \mathbf{C}/K$ are projections. Let $I \subseteq \{1, \dots, d\}$ be a maximal set such that for each $i \in I$ there exists $b_i \in B$ such that the set

$$C((b_i)_{i \in I}) := \{c \in C \mid c_i = b_i \text{ for all } i \in I\}$$

is not contained in a class of K. Note that such a set exists because for $I = \emptyset$ we have that $C((b_i)_{i \in I}) = C$ is not contained in one class of K since $\mathbf{C}/K$ has at least two elements.

Without loss of generality, we may assume that $\{1, \dots, d\} \setminus I = \{1, \dots, k\}$. Since $\mathbf{B}$ is idempotent, $C((b_i)_{i \in I})$ is the domain of a subalgebra $\mathbf{C}'$ of $\mathbf{C}$. Let $K' := K \cap (C')^2$. All operations of $\mathbf{A}' := \mathbf{C}'/K'$ are restrictions of operations in $\mathbf{A}$ and hence projections. Let

$$B' := \{b \in B \mid (b_1, \dots, b_d) \in C'\}.$$

Then B' is the domain of a subalgebra $\mathbf{B}'$ of $\mathbf{B}$. The image

$$K'' := \{(c_k, e_k) \in (B')^2 \mid (c, e) \in K'\}$$

of K' under the k-th projection is a congruence of $\mathbf{B}'$, and it must have more than one equivalence class: otherwise there would be $b_1, \dots, b_d, b'_1, \dots, b'_{k-1} \in B$ such that the tuples $(b_1, \dots, b_{k-1}, b_k, b_{k+1}, \dots, b_d)$ and $(b'_1, \dots, b'_{k-1}, b_k, b_{k+1}, \dots, b_d)$ are in different K'-classes. But then $I \cup \{k\}$ is such that $C((b_i)_{i \in I \cup \{k\}})$ is not contained in one class of K, contradicting the maximality of I. Therefore, $\mathbf{B}'/K'' \in \mathrm{HS}(\mathbf{B})$ is an algebra with at least two elements all whose operations are projections. □

Since the size of the algebras in $\mathrm{HS}(\mathbf{B})$ is bounded by the size of $\mathbf{B}$, this leads to an algorithm that decides whether a given finite structure $\mathfrak{B}$ satisfies the equivalent conditions in Theorem 7.18. We summarise various equivalent conditions for finite idempotent algebras that were treated in this chapter.

Corollary 7.24. *Let* $\mathbf{B}$ *be a finite idempotent algebra. Then the following are equivalent.*

1. $\mathbf{B}$ *has a Taylor term.*
2. *there is no homomorphism from* $\mathrm{Clo}(\mathbf{B})$ *to* $\mathbf{Proj}$.
3. $\mathbf{B}$ *satisfies some non-trivial finite set of identities.*
4. $\mathrm{HSP}(\mathbf{B})$ *does not contain an at least 2-element algebra all of whose operations are projections.*
5. $\mathrm{HS}(\mathbf{B})$ *does not contain an at least 2-element algebra all of whose operations are projections.*

Proof. The equivalence of (1) and (2) is Theorem 7.18.
The equivalence of (2) and (3) follows from Corollary 7.14.
The equivalence of (2) and (4) follows from Proposition 7.12.
The equivalence of (4) and (5) follows from Theorem 7.9 combined with Theorem 7.23. □

Exercises.

88. Let $\mathbf{A}$ be a finite idempotent algebra. Then there exists an operation $f \in \mathrm{Clo}(\mathbf{A})^{(k)}$ such that for every $B \subseteq A$ and every b_0 in the subalgebra of $\mathbf{A}$ generated by B there exist $b_1, \ldots, b_k \in B$ such that $b_0 = f(b_1, \ldots, b_k)$.

8 Undirected Graphs

This section contains a proof of the dichotomy for finite undirected graphs of Hell and Nešetřil, Theorem 1.4. We prove something stronger, namely that the tractability conjecture (Conjecture 2) is true for finite undirected graphs $\mathfrak{B}$ [16]. More specifically, the following is true.

Theorem 8.1. *Let $\mathfrak{B}$ be a finite undirected graph. Then either*

- $\mathfrak{B}$ *is* bipartite *(i.e., homomorphic to K_2) or has a loop, or*
- $\mathrm{H}(\mathrm{I}(\mathfrak{B}))$ *contains all finite structures.*

In the first case, $\mathfrak{B}$ has a quasi majority polymorphism (which is a Taylor polymorphism; see Exercise 48) and $\mathrm{CSP}(\mathfrak{B})$ is clearly in P, and in the latter case $\mathrm{CSP}(\mathfrak{B})$ is NP-complete by Theorem 4.9.

This theorem has a remarkable consequence in universal algebra, whose significance goes beyond the study of the complexity of CSPs, and which provide a strengthening of Taylor's theorem (Theorem 7.18), discovered by Siggers in 2010 (see Section 8.2).

8.1 The Hell-Nešetřil Theorem

The graph $K_4 - \{0, 1\}$ (a clique with four vertices where one edge is missing) is called a *diamond*. A graph is called *diamond-free* if it does not contain a copy of a diamond as a (not necessarily induced) subgraph. For every $\ell \in \mathbb{N}$, the graph $(K_3)^\ell$ is an example of a diamond-free graph.

Lemma 8.2. *Let $\mathfrak{B}$ be a finite undirected loopless graph which is not bipartite. Then $\mathfrak{B}$ interprets primitively positively a graph that is homomorphically equivalent to a diamond-free core containing a triangle.*

Proof. For a binary relations $R_1, R_2 \subset B^2$, define $R_1 \circ R_2$ to be the binary relation

$$\big\{(x, y) \mid \exists z(R_1(x, z) \wedge R_2(z, y))\big\}.$$

For $R \subseteq B^2$ and $k \geq 1$, define $R^1 := R$ and $R^{k+1} := R^k \circ R$. Note that R^k is primitively positively definable in $(B; R)$. We may assume that

1. $\mathrm{H}(\mathrm{I}(\mathfrak{B}))$ does not contain a non-bipartite loopless graph with fewer vertices than $\mathfrak{B}$, since otherwise we could replace $\mathfrak{B}$ by this graph. In particular, $\mathfrak{B}$ must then be a core.
2. $\mathfrak{B} = (V; E)$ contains a triangle: if the length of the shortest odd cycle in $\mathfrak{B}$ is k, then $(B; E^{k-2})$ is a graph and contains a triangle, so it can replace $\mathfrak{B}$.

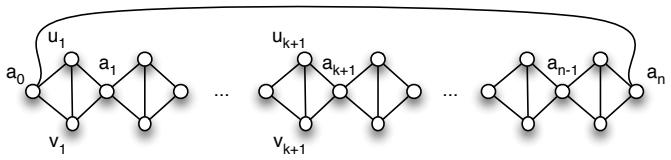

Figure 11: Diagram for the proof of Lemma 8.2.

Claim 1. Every vertex of $\mathfrak{B}$ is contained in a triangle: Otherwise, we can replace $\mathfrak{B}$ by the subgraph of $\mathfrak{B}$ induced by set defined by the primitive positive formula $\exists u, v\,(E(x,u) \wedge E(x,v) \wedge E(u,v))$ which still contains a triangle, contradicting our first assumption.

Claim 2. $\mathfrak{B}$ does not contain a copy of K_4. Otherwise, if a is an element from a copy of K_4, then the subgraph of $\mathfrak{B}$ induced by the set defined by the primitive positive formula $E(a,x)$ is a non-bipartite graph $\mathfrak{A}$, which has strictly less vertices than $\mathfrak{B}$ because $a \notin A$. Moreover, $\mathfrak{A}$ is by Theorem 4.19 homomorphically equivalent to a graph with a primitive positive interpretation in $\mathfrak{B}$, contrary to our initial assumption.

Claim 3. The graph $\mathfrak{B}$ must also be diamond-free. To see this, let R be the binary relation with the primitive positive definition

$$R(x,y) :\Leftrightarrow \exists u, v\,\big(E(x,u) \wedge E(x,v) \wedge E(u,v) \wedge E(u,y) \wedge E(v,y)\big)$$

and let T be the transitive closure of R. The relation T is clearly symmetric, and since every vertex of $\mathfrak{B}$ is contained in a triangle, it is also reflexive, and hence an equivalence relation of $\mathfrak{B}$. Since B is finite, for some n the formula $\exists u_1, \ldots, u_n\,\big(R(x,u_1) \wedge R(u_1,u_2) \wedge \cdots \wedge R(u_n,y)\big)$ defines T, showing that T is primitively positively definable in $\mathfrak{B}$.

We claim that the graph $\mathfrak{B}/T$ (see Example 10) does not contain loops. It suffices to show that $T \cap E = \emptyset$. Otherwise, let $(a,b) \in T \cap E$. Choose (a,b) in such a way that the shortest sequence $a = a_0, a_1, \ldots, a_n = b$ with $R(a_0,a_1)$, $R(a_1,a_2)$, $\ldots$, $R(a_{n-1},a_n)$ in $\mathfrak{B}$ is shortest possible; see Figure 11. This chain cannot have the form $R(a_0,a_1)$ because $\mathfrak{B}$ does not contain K_4 subgraphs. Suppose first that $n = 2k$ is even. Let the vertices u_1, v_1, u_{k+1} and v_{k+1} be as depicted in Figure 11. Let S be the set defined by

$$\exists x_1, \ldots, x_k\,\big(E(u_{k+1},x_1) \wedge E(v_{k+1},x_1) \wedge R(x_1,x_2) \wedge \cdots \wedge R(x_{k-1},x_k) \wedge E(x_k,x)\big).$$

Note that $a_0, u_1, v_1 \in S$ form a triangle. If $a_n \in S$ then we obtain a contradiction to the minimal choice of n. Hence, the subgraph induced by the primitively positively definable set S is non-bipartite and strictly smaller than $\mathfrak{B}$, in contradiction to the initial assumption.

If $n = 2k+1$ is odd, we can argue analogously with the set S defined by the formula

$$\exists x_1, \ldots, x_k\,\big(R(a_{k+1},x_1) \wedge R(x_1,x_2) \wedge \cdots \wedge R(x_{k-1},x_k) \wedge E(x_k,x)\big)$$

and again obtain a contradiction. So we conclude that $\mathfrak{B}/T$ does not contain loops. It also follows that $\mathfrak{B}/T$ contains a triangle, because $\mathfrak{B}$ contains a triangle.

Thus, the initial assumption on $\mathfrak{B}$ then implies that T must be the trivial equivalence relation, which in turn implies that $\mathfrak{B}$ does not contain any diamonds. □

Lemma 8.3 (from [16]). *Let $\mathfrak{B}$ be a diamond-free undirected graph and let $h\colon (K_3)^k \to \mathfrak{B}$ be a homomorphism. Then the image of h is isomorphic to $(K_3)^m$ for some $m \leq k$.*

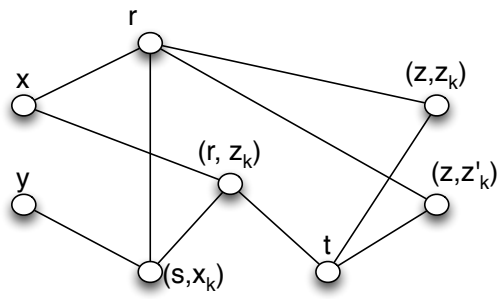

Figure 12: Diagram for the proof of Lemma 8.3.

Proof. Let $I \subseteq \{1, \dots, k\}$ be a maximal set such that $\ker(h) \subseteq \ker(\mathrm{pr}_I)$. Such a set exists, because $\ker(\mathrm{pr}_\emptyset)$ is the total relation. We claim that $\ker(h) = \ker(\mathrm{pr}_I)$; this clearly implies the statement.

By the maximality of I, for every $j \in \{1, \dots, k\} \setminus I$ there are $x, y \in (K_3)^k$ such that $h(x) = h(y)$ and $x_j \neq y_j$. We have to show that for all $z_1, \dots, z_k, z'_j \in \{0, 1, 2\}$

$$h(z_1, \dots, z_j, \dots, z_k) = h(z_1, \dots, z_{j-1}, z'_j, z_{j+1}, \dots, z_k).$$

We may suppose that $z_j \neq x_j$ and $z'_j = x_j$. To simplify notation, we assume that $j = k$. As we have seen in Exercises 7 and 8, any two vertices in $(K_3)^k$ have a common neighbour.

- Let r be a common neighbour of x and $(z, z_k) := (z_1, \dots, z_k)$. Note that r and (z, z'_k) are adjacent, too.
- For all $i \neq k$ we choose an element s_i of K_3 that is distinct from both r_i and y_i. Since x_k is distinct from r_k and y_k we have that $(s, x_k) := (s_1, \dots, s_{k-1}, x_k)$ is a common neighbour of r and y.
- The tuple $(r, z_k) := (r_1, \dots, r_{k-1}, z_k)$ is a common neighbour of both x and (s, x_k).
- Finally, for $i \neq k$ choose t_i to be distinct from z_i and r_i, and choose t_k to be distinct from z_k and from z'_k. Then $t := (t_1, \dots, t_{k-1}, t_k)$ is a common neighbour of (z, z_k), of (z, z'_k), and of (r, z_k).

The situation is illustrated in Figure 12. Since $\mathfrak{B}$ is diamond-free, $h(x) = h(y)$ implies that $h(r) = h(r, z_k)$ and for the same reason $h(z, z_k) = h(z, z'_k)$ which completes the proof. □

Lemma 8.4 (from [16])**.** *If a finite diamond-free graph $\mathfrak{B}$ contains a triangle, then for some $k \in \mathbb{N}$ there is a primitive positive interpretation of $(K_3)^k$ with constants in $\mathfrak{B}$.*

Proof. We construct a strictly increasing sequence of subgraphs $G_1 \subset G_2 \subset \cdots$ of $\mathfrak{B}$ such that G_i is isomorphic to $(K_3)^{k_i}$ for some $k_i \in \mathbb{N}$. Let G_1 be any triangle in $\mathfrak{B}$. Suppose now that G_i has already been constructed. If the domain of G_i is primitively positively definable in $\mathfrak{B}$ with constants, then we are done. Otherwise, there exists an idempotent polymorphism f of $\mathfrak{B}$ and $v_1, \dots, v_k \in G_i$ such that $f(v_1, \dots, v_k) \notin G_i$. The restriction of f to G_i is a homomorphism

from $(K_3)^{k_i}$ to the diamond-free graph $\mathfrak{B}$. Lemma 8.3 shows that $G_{i+1} := f((G_i)^k)$ induces a copy of $(K_3)^{k_{i+1}}$ for some $k_{i+1} \leq k$. Since f is idempotent, we have that $G_i \subseteq G_{i+1}$, and by the choice of f the containment is strict. Since $\mathfrak{B}$ is finite, for some m the set G_m must have a primitive positive definition in $\mathfrak{B}$ with constants. □

Proof of Theorem 8.1. Let $\mathfrak{B}$ be a finite undirected graph that is not bipartite. By Lemma 8.2 $\mathfrak{B}$ interprets primitively positively a graph that is homomorphically equivalent to a diamond-free core $\mathfrak{C}$ containing a triangle. Then Lemma 8.4 applied to $\mathfrak{C}$ implies that for some $k \in \mathbb{N}$ there is a primitive positive interpretation of $(K_3)^k$ with constants in $\mathfrak{C}$. Since $\mathfrak{C}$ is a core, and since $(K_3)^k$ is homomorphically equivalent to K_3, it follows that there is a primitive positive interpretation of a structure that is homomorphically equivalent to K_3 in $\mathfrak{C}$. The structure K_3 interprets all finite structures primitive positively (Theorem 4.11), so Theorem 4.19 implies that H(I($\mathfrak{B}$)) contains all finite structures. □

8.2 Siggers Operations of Arity 6

An operation $s\colon B^6 \to B$ is called *Siggers operation* if

$$s(x, y, x, z, y, z) = s(y, x, z, x, z, y)$$

holds for all $x, y, z \in B$.

Theorem 8.5 (from [60])**.** *Let $\mathfrak{B}$ be a finite structure. Then either $\mathfrak{B}$ interprets all finite structures up to homomorphic equivalence, or $\mathfrak{B}$ has a Siggers polymorphism.*

Proof. Pick $k \geq 1$ and $a, b, c \in B^k$ such that $\{(a_i, b_i, c_i) \mid i \leq k\} = B^3$. Let R be the binary relation on B^k such that $(u, v) \in R$ if and only if there exists a 6-ary $s \in \mathrm{Pol}(\mathfrak{B})$ such that $u = s(a, b, a, c, b, c)$ and $v = s(b, a, c, a, c, b)$. We make the following series of observations.

- The vertices $a, b, c \in B^k$ induce in $(B^k; R)$ a copy of K_3: each of the six edges of K_3 is witnessed by one of the six 6-ary projections from Pol($\mathfrak{B}$).
- The relation R is symmetric: Suppose that $(u, v) \in R$ and let $s \in \mathrm{Pol}(\mathfrak{B})$ be such that $u = s(a, b, a, c, b, c)$ and $v = s(b, a, c, a, c, b)$. Define $s' \in \mathrm{Pol}(\mathfrak{B})$ by $s'(x_1, \ldots, x_6) := s(x_2, x_1, x_4, x_3, x_6, x_5)$; then

  $$v = s(b, a, c, a, c, b) = s'(a, b, a, c, b, c)$$
  $$u = s(a, b, a, c, b, c) = s'(b, a, c, a, c, b)$$

  and hence s' witnesses that $(v, u) \in R$.
- If the graph $(B^k; R)$ contains a loop $(w, w) \in R$, then there exists a 6-ary $s \in \mathrm{Pol}(\mathfrak{B})$ such that

  $$s(a, b, a, c, b, c) = w = s(b, a, c, a, c, b)\,.$$

  The operation s is Siggers: for all $x, y, z \in B$ there exists an $i \leq k$ such that $(x, y, z) = (a_i, b_i, c_i)$, and the above implies that

  $$s(a_i, b_i, a_i, c_i, b_i, c_i) = s(b_i, a_i, c_i, a_i, c_i, b_i)$$

  and we are done in this case.

So we may assume in the following that $(B^k; R)$ is a simple (i.e., undirected and loopless) graph that contains a copy of K_3. The relation R (as a $2k$-ary relation over B) is preserved by $\mathrm{Pol}(\mathfrak{B})$, and hence $(B^k; R)$ has a primitive positive interpretation in $\mathfrak{B}$. By Theorem 8.1 applied to the undirected graph $(B^k; R)$, there is a primitive positive interpretation in $(B^k; R)$ of all finite structures up to homomorphic equivalence, and hence also in $\mathfrak{B}$, and this concludes the proof. □

9 Open Problems

The Feder and Vardi dichotomy conjecture [34] has been the outstanding open problem in the field; it was solved in 2017 by Bulatov [18] and, independently, by Zhuk [62]. However, there are many interesting problems in the field that are still left open. We start with open problems where all the relevant concepts have already been introduced in the course.

1. Is the class of all finite structures, ordered by pp-constructability and factored by the respective equivalence relation, a lattice [12, 13]? Is it countably infinite or uncountably infinite?

2. What is the computational complexity to determine whether a given finite core structure H has tree duality? Is this problem in P? Is it in P if H is a digraph or even an orientation of a tree?

3. Is there a polynomial-time algorithm to determine whether a given core structure H has a Siggers polymorphism? Is this true for the special case where H is a digraph or an orientation of a tree?

4. Is is true that an orientation of a tree has a Siggers polymorphism if and only if it has a binary symmetric (i.e., commutative) polymorphism?

5. Is it true that if H is an orientation of a tree, and $\mathrm{CSP}(H)$ has a Siggers polymorphism, then $\mathrm{CSP}(H)$ can be solved by the path-consistency procedure?

Here come some open problems that require knowledge of concepts that have not been covered in this course; however, references are provided where these concepts are defined formally.

1. Prove that a finite-domain CSP is in P if and only if it can be expressed in *Choiceless Polynomial Time* [9].

2. Prove that a finite-domain CSP is in P if and only if it can be expressed in *fixed point logic* expanded by a *rank operator* (with respect to some finite Abelian group [28, 38]).

3. Is it true that if $\mathrm{CSP}(H)$ is in NL, then $\mathrm{CSP}(H)$ is in linear Datalog (Dalmau [26])? Is this at least true for digraphs H? The same question would already be interesting for orientations of trees.

4. (Egri-Larose-Tesson [30]) Is it true that if $\mathrm{CSP}(H)$ is in L, then $\mathrm{CSP}(H)$ is in symmetric Datalog? Is this at least true for digraphs H? It would already be interesting for orientations of trees.

5. (Larose-Tesson [49]) Is it true that if the polymorphism algebra of H generated a congruence join-semidistributive variety, then CSP(H) is in linear Datalog? Is this at least true for digraphs H? It would already be interesting for orientations of trees.

For CSPs over infinite domains, there are numerous open problems, and I invite the reader to have a look at my book project on that topic [10].

References

[1] S. Aaronson. P=$^?$NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:4, 2017.

[2] G. Ahlbrandt and M. Ziegler. Quasi-finitely axiomatizable totally categorical theories. *Annals of Pure and Applied Logic*, 30(1):63–82, 1986.

[3] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979.

[4] A. Atserias, A. A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009.

[5] L. Barto. The dichotomy for conservative constraint satisfaction problems revisited. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, Toronto, Canada, 2011.

[6] L. Barto and M. Kozik. Constraint satisfaction problems of bounded width. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 595–603, 2009.

[7] L. Barto, J. Opršal, and M. Pinsker. The wonderland of reflections. *Israel Journal of Mathematics*, 223(1):363–398, 2018.

[8] G. Birkhoff. On the structure of abstract algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31(4):433–454, 1935.

[9] A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100(1-3):141–187, 1999.

[10] M. Bodirsky. Complexity of infinite-domain constraint satisfaction. Submitted for publication in the LNL Series, Cambridge University Press, 2020.

[11] M. Bodirsky. Diskrete Strukturen, 2020. Skript zur Vorlesung, TU Dresden.

[12] M. Bodirsky, F. Starke, and A. Vucaj. Smooth digraphs modulo primitive positive constructability, 2020. Preprint available at ArXiv:1906.05699.

[13] M. Bodirsky and A. Vucaj. Two-element structures modulo primitive positive constructability. *Algebra Universalis*, 81(20), 2020. Preprint available at ArXiv:1905.12333.

[14] V. G. Bodnarčuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras, part I and II. *Cybernetics*, 5:243–539, 1969.

[15] A. A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pages 321–330, Ottawa, Canada, 2003.

[16] A. A. Bulatov. H-coloring dichotomy revisited. *Theoretical Computer Science*, 349(1):31–39, 2005.

[17] A. A. Bulatov. Conservative constraint satisfaction re-revisited. *Journal Computer and System Sciences*, 82(2):347–356, 2016. ArXiv:1408.3690.

[18] A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17*, pages 319–330, 2017.

[19] A. A. Bulatov and V. Dalmau. A simple algorithm for Mal'tsev constraints. *SIAM Journal on Computing*, 36(1):16–27, 2006.

[20] A. A. Bulatov and P. Jeavons. Algebraic structures in combinatorial problems. Technical report MATH-AL-4-2001, Technische Universität Dresden, 2001.

[21] A. A. Bulatov, A. A. Krokhin, and P. G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.

[22] S. N. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer Verlag, Berlin, 1981.

[23] C. Carvalho, L. Egri, M. Jackson, and T. Niven. On Maltsev digraphs. *Electr. J. Comb.*, 22(1):P1.47, 2015.

[24] H. Chen and B. Larose. Asking the metaquestions in constraint tractability. *TOCT*, 9(3):11:1–11:27, 2017.

[25] D. A. Cohen, M. C. Cooper, P. G. Jeavons, and S. Živný. Binarisation via dualisation for valued constraints. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3731–3737, 2015.

[26] V. Dalmau. Linear datalog and bounded path duality of relational structures. *Logical Methods in Computer Science*, 1(1), 2005.

[27] V. Dalmau and J. Pearson. Closure functions and width 1 problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, pages 159–173, 1999.

[28] A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *2009 24th Annual IEEE Symposium on Logic In Computer Science*, pages 113–122, Los Angeles, CA, USA, 2009. IEEE.

[29] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[30] L. Egri, B. Larose, and P. Tesson. Symmetric datalog and constraint satisfaction problems in logspace. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pages 193–202, 2007.

[31] M. M. El-Zahar and N. Sauer. The chromatic number of the product of two 4-chromatic graphs is 4. *Combinatorica*, 5(2):121–126, 1985.

[32] T. Feder. Classification of homomorphisms to oriented cycles and of k-partite satisfiability. *SIAM Journal on Discrete Mathematics*, 14(4):471–480, 2001.

[33] T. Feder and M. Y. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 612 – 622, 1993.

[34] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

[35] J. Fischer. CSPs of orientations of trees. Master thesis, TU Dresden, 2015.

[36] M. Garey and D. Johnson. *A guide to NP-completeness*. CSLI Press, Stanford, 1978.

[37] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100, 1968.

[38] E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! *The Journal of Symbolic Logic*, 84(1):54–87, 2019.

[39] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press, 2008. Sixth edition.

[40] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[41] P. Hell and J. Nešetřil. The core of a graph. *Discrete Mathematics*, 109:117–126, 1992.

[42] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, Oxford, 2004.

[43] D. Hobby and R. McKenzie. *The structure of finite algebras*, volume 76 of *Contemporary Mathematics*. American Mathematical Society, 1988.

[44] W. Hodges. *Model theory*. Cambridge University Press, 1993.

[45] W. Hodges. *A shorter model theory*. Cambridge University Press, Cambridge, 1997.

[46] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[47] A. Kazda. Maltsev digraphs have a majority polymorphism. *European Journal of Combinatorics*, 32:390–397, 2011.

[48] B. Larose, C. Loten, and C. Tardif. A characterisation of first-order constraint satisfaction problems. *Logical Methods in Computer Science*, 3(4:6), 2007.

[49] B. Larose and P. Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theoretical Computer Science*, 410(18):1629–1647, 2009.

[50] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[51] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[52] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[53] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[54] R. Pöschel and L. A. Kalužnin. *Funktionen- und Relationenalgebren*. Deutscher Verlag der Wissenschaften, 1979.

[55] E. L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematics Studies*, 5, 1941.

[56] I. G. Rosenberg. Minimal clones I: the five types. *Lectures in Universal Algebra (Proc. Conf. Szeged, 1983), Colloq. Math. Soc. J. Bolyai*, 43:405–427, 1986.

[57] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

[58] U. Schöning. *Logic for Computer Scientists*. Springer, 1989.

[59] Y. Shitov. Counterexamples to hedetniemi's conjecture, 2019. arXiv:1905.02167.

[60] M. H. Siggers. A strong Mal'cev condition for varieties omitting the unary type. *Algebra Universalis*, 64(1):15–20, 2010.

[61] W. Taylor. Varieties obeying homotopy laws. *Canadian Journal of Mathematics*, 29:498–527, 1977.

[62] D. N. Zhuk. A proof of CSP dichotomy conjecture. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17*, pages 331–342, 2017. https://arxiv.org/abs/1704.01914.

A O-notation

The letters o and O stand for the *order* of growth of the function. The *big-O notation* is used to express upper bounds, and the *little-o notation* to express lower bounds. We mention that there exists related notation to describe other kinds of bounds on asymptotic growth, e.g., Θ, Ω, ω, of which we only need Θ in this text, so we skip the definitions of the others.

Let $g\colon \mathbb{R} \to \mathbb{R}$ (we use $\mathbb{R}$ for convenience; similar definitions exist for other domains such that as $\mathbb{N}$ and $\mathbb{Q}$, etc). Then $O(g)$ is the set of all functions $f\colon \mathbb{R} \to \mathbb{R}$ such that there exists $c, x_0 \in \mathbb{R}$ such that $|f(x)| \leq cg(x)$ for all $x \geq x_0$. Note that

$$f \in O(g) \Leftrightarrow \limsup_{x\to\infty} \left|\frac{f(x)}{g(x)}\right| < \infty.$$

In typical usage, the formal definition of $O(g)$ is not used directly; rather, we first use the following simplification rules:

- if $g(x)$ is a sum of several terms, if there is one with largest growth rate, then we drop all other terms;

- if $g(x) = c \cdot f(x)$ and c is a constant that does not depend on x, then c can be omitted.

When we write $O(g)$, we typically choose g to be as simple as possible. O-notation can also be used within arithmetic terms. For example, $h + O(g)$ denotes the set of functions of the form $h + f$ for $f \in O(g)$. In other words, $k \in h + O(g)$ is equivalent to $k - h \in O(G)$.

We write $o(g)$ for the set of all functions $f \colon \mathbb{R} \to \mathbb{R}$ such that for every $\epsilon \in \mathbb{R}_{>0}$ there exists $x_0 \in \mathbb{R}$ such that $|f(x)| \leq \epsilon g(x)$ for all $x \geq x_0$. Informally, $f \in o(g)$ means that g grows much faster than f. For example, $x \mapsto 2x$ is in $o(x \mapsto x^2)$, and $x \mapsto 1/x$ is in $o(1)$. Note that $o(g) \subseteq O(g)$, and that

$$f \in o(g) \Leftrightarrow \lim_{x \to \infty} \frac{f(x)}{g(x)} = 0.$$

Similarly as in the case of the O-notation we may use the o-notation in arithmetic expressions. Note that if $f \in o(g)$ and c is a constant, then $cf \in o(g)$. Frequent notation is to write $f \ll g$ (or $g \gg f$) if $f \in o(g)$.

We write $\Theta(g)$ for the set of all functions f such that there are constants c, C and $x_0 \in \mathbb{R}$ such that $cg(x) \leq f(x) \leq Cg(x)$ for every $x \geq x_0$. In other words, $f \in \Theta(g)$ if $f \in O(g)$ and $g \in O(f)$.

Finally, we write $f(x) \sim g(x)$ if

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1$$

and we say that f and g are *asymptotically equivalent* (for $x \to \infty$).

B What every mathematician needs to know about complexity theory

For a set A, we write A^* for the set of all words over the alphabet A. A *word over* A can be seen as a function from $\{1, \ldots, n\} \to A$, for some $n \in \mathbb{N}$. We write ϵ for the empty word (i.e., for the function with the empty domain).

The most classical setting of complexity theory is the study of the computational complexity of functions f from $\{0,1\}^* \to \{0,1\}$. Alternatively, we may view f as a set of words, namely that set of words w such that $f(w) = 1$; such sets are also called *formal languages*. There are several mathematically rigorous machine models to formalise the set of such functions that are computable or efficiently computable. The first insight is that most of these machine models lead to the same, or to closely related classes of functions. Complexity theory maps out the landscape of the resulting classes of functions. Typically the first machine model that is introduced in introductory courses are *Turing machines*. They strike a good balance between the following two (almost contradictory!) requirements that a theoretician has for these machine models:

- the model should be relatively simple, so that it is easy to show that it can be simulated by many other machine models.
- the model should be relatively powerful, so that it is easy to show that it can simulate many other machine models.

Turing machines are simple, but still the definition does not easily fit into a few lines. On the other hand, today academics are most likely to already have a very good idea of what a computer program can do (in polynomially many steps); and this coincides with what a Turing machine M can do (in polynomially many computational steps). In a nutshell, a Turing machine

- has an unboundedly large memory containing values from $\{-1, 0, 1\}$ (the symbol -1 will be called the *blank* symbol);
- has finitely many *states* Q;
- has a *read-* and *write* head;
- has a finite transition function $\delta\colon Q \times \{-1, 0, 1\} \to \Sigma \times Q \times \{l, r\}$;
- has a *accept* state $y \in Q$.
- has a *start* state $s \in Q$.

Initially, the memory just contains the word $w \in \{0, 1\}^*$, i.e., in the first cell there is w_1, in the second cell there is w_2, etc, and in all further memory cells there is -1, and the machine *is in state* s. Depending on its state $u \in Q$ and the tape content c under the read-write head, let $(v, d, m) := \delta(u, c)$; then

1. the machine changes to state v;
2. the tape content under the read-write head is changed from c to d,
3. the read-write tape moves one cell to the left if $m = l$, and one to the right if $m = r$.

If the machine reaches state y it accepts. Every Turing machine describes a formal language, namely the function $f\colon \{0, 1\}^* \to \{0, 1\}$ such that $f(w) = 1$ if and only if when running the machine on input w it eventually accepts. We also say that M *computes* f, and we then sometimes write $M(f)$ instead of $f(w)$. More generally, Turing machines can be used to describe functions f from $\{0, 1\}^*$ to $\{0, 1\}^*$ where $f(w)$, for a given word w, is the string that is written on the output tape when the Turing machine accepts (here we require that the machine terminates on every input after finitely many steps, and again we say that M *computes* f).

So we will pretend in the following that the reader already knows what Turing machines M are. It turns out that despite the simplicity of Turing machines, they can simulate most of the other machine models, and they can simulate any machine that humans ever constructed (even when neglecting the restriction that we one have some fixed finite maximal memory size in this universe).

In complexity theory we are interested in the number of computation steps that M needs to perform to compute $f(w)$, which corresponds to computation time. For example, we say that a Turing machine runs *in polynomial time* if the number of computation steps is in $O(|w|^k)$ for some $k \in \mathbb{N}$. The class of such functions is denoted by P.

Coding. In the combinatorics course we have met computational complexity for example in the section about colorability. We mentioned that 2-colorability is in P and that k-colorability, for $k \geq 3$, is NP-hard. But these were problems about finite graphs, whereas

in the above we only treated formal languages. But this is just a matter of coding. We first observe that we can simulate any alphabet by our alphabet $\{0,1\}$, by just grouping bits together to represent a richer alphabet. In particular, we will typically use the letter # to separate different numbers in the input. One way to represent a graph as a word is to first write the number n of vertices, followed by the symbol #, followed by a sequence of n^2 bits for the adjacency matrix.

The second most important complexity class is NP.

Definition B.1. NP (for *nondeterministic polynomial time*) stands for the class of all functions $f\colon \{0,1\}^* \to \{0,1\}$ such that there exists a polynomial-time Turing machine M and a $d \in \mathbb{N}$ such that for every $w \in \{0,1\}^*$ there exists a $a \in \{0,1\}^*$ with $|a| \in O(n^d)$ such that $f(w) = M(w\#a)$.

It is a famous open problem whether P = NP, and it is widely conjectured that P $\neq$ NP. To explain the significance of this conjecture, we need a couple of more concepts. Let $f_1, f_2\colon \{0,1\}^* \to \{0,1\}$. A *reduction* from f_1 to f_2 is a function $g\colon \{0,1\}^* \to \{0,1\}^*$ such that $f_1(w) = f_2(g(w))$. A reduction g is *polynomial-time* if g can be computed a Turing machine that runs in polynomial time.

Definition B.2. A function $f\colon \{0,1\}^* \to \{0,1\}$ is *NP-hard* if every function g in NP has a polynomial-time reduction to f. A function is called *NP-complete* if it is in NP and NP-hard.

The class *coNP* is dual to NP: it is the class of all functions f such that $1 - f$ is in NP. There is an analogous definition for any complexity class K: a function is in co-K if $1 - f$ is in K. Clearly, every function in P is both in NP and in coNP.

A class of finite graphs $\mathcal{C}$ is in NP if there exists a formal language in NP such that each word in the language codes a graph in $\mathcal{C}$ (say in the way we described above), and every graph in $\mathcal{C}$ is coded by some word in the language. Unlike the class P, it is possible to define the class of all graph classes in NP transparently and fully formally in a few lines (without any reference to Turing machines).

Theorem B.3 (Fagin)**.** *A class of finite graphs $\mathcal{C}$ is in NP if and only if there exists an existential second-order sentence Φ such that for every finite graph G we have*

$$G \in \mathcal{C} \text{ if and only if } G \models \Phi.$$

We do not define *existential second-order logic* here. The interested reader is referred to a textbook on finite model theory to learn more about such connections between logic and complexity theory, e.g. [50].

We now return to the question why most researchers believe that P $\neq$ NP. In order to show that P=NP is suffices to provide for *any* of the known NP-complete problems a polynomial-time algorithm. There are many NP-complete problems that are of central importance in optimisation, scheduling, cryptography, bioinformatics, artificial intelligence and many more areas. If P=NP, then this would mean a simultaneous breakthrough in all of these areas. It is fair to say that every day, thousands of researchers are directly or indirectly working on proving that P=NP (since they work on things that are related to the better understanding of some NP-complete problem). The fact that nobody has succeeded (not even came close to) is one of the reasons why we believe that P cannot be equal to NP. A world where P = NP would probably be drastically different from the world we live in. On the other hand,

we also have no clue on how to possibly prove that P $\neq$ NP. And quite a bit is known about approaches to proving P $\neq$ NP that must fail (see [1]; great read, free download at `https://www.scottaaronson.com/papers/pnp.pdf`).