Graph Homomorphisms and Universal Algebra
Course Notes

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

December 26, 2015

Contents

1 The Basics
  1.1 Graphs and Digraphs ............................................. 2
  1.2 Graph Homomorphisms ........................................... 4
  1.3 The $H$-coloring Problem and Variants ......................... 6
  1.4 Cores .......................................................... 8
  1.5 Polymorphisms ................................................... 10

2 The Arc-consistency Procedure ...................................... 11
  2.1 The Power Set Graph ............................................. 14
  2.2 Tree Duality ..................................................... 15
  2.3 Totally Symmetric Polymorphisms ............................... 17
  2.4 Semilattice Polymorphisms ..................................... 18

3 The Path-consistency Procedure ..................................... 19
  3.1 Majority Polymorphisms ......................................... 21
  3.2 Near-unanimity Polymorphisms ................................ 23
  3.3 Maltsev Polymorphisms ......................................... 23

4 Logic ............................................................................ 24
  4.1 Canonical Conjunctive Queries .................................. 25
  4.2 Canonical Databases .............................................. 25
  4.3 Primitive Positive Definability ................................ 26
  4.4 Primitive Positive Definability in Cores ......................... 27
  4.5 Primitive Positive Interpretations ............................... 27

5 Relations and Functions .............................................. 28
  5.1 Function Clones .................................................... 28
  5.2 The Galois Connection Inv-Pol .................................. 28
  5.3 Unary Clones ....................................................... 29
  5.4 Minimal Clones ..................................................... 30
  5.5 Schaefer’s Theorem ............................................... 32
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 introduce NP-completeness, even though this concept is 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 [35].

1 The Basics

1.1 Graphs and Digraphs

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, we require that \( V \) is finite, in which case we say that \( G \) is a finite digraph. 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 \((u, u)\). 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.

A digraph \( G = (V, E) \) is called symmetric if \((u, v) \in E\) if and only if \((v, u) \in E\). Symmetric digraphs can be viewed as undirected graphs, and vice versa. Formally, 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 \( V \). For a digraph \( G \), we say that \( G' \) is the undirected graph of \( G \) if \( G' \) is the undirected graph where \((u, v) \in E(G')\) iff \((u, v) \in E(G)\) 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.

Examples of graphs, and corresponding notation.

- We allow that a digraph may have no vertices at all.
• The complete graph on \( n \) vertices \( \{1, \ldots, n\} \), denoted by \( K_n \). This is an undirected graph on \( n \) vertices in which every vertex is joined with any other vertex.

• The cyclic graph on \( n \) vertices, denoted by \( C_n \); this is the undirected graph with the vertex set \( \{0, \ldots, n-1\} \) and edge set
\[
\{(0,1), \ldots, (n-2,n-1), (n-1,0)\} = \{\{u,v\} \mid u - v = 1 \mod n\}.
\]

• The directed cycle on \( n \) vertices, denoted by \( C'_n \); this is the digraph with the vertex set \( \{0, \ldots, n-1\} \) and the arcs \( \{(0,1), \ldots, (n-2,n-1), (n-1,0)\} \).

• The path with \( n+1 \) vertices and \( n \) edges, denoted by \( P_n \); this is an undirected graph with the vertex set \( \{0, \ldots, n\} \) and edge set \( \{(0,1), \ldots, (n-1,n)\} \).

• 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 \( \{(0,1), \ldots, (n-1,n)\} \).

• 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 \( G \sqcup H \) 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 analyze the efficiency of algorithms on graphs.

The following concepts are for undirected graphs only. We say that an undirected graph \( G \) contains a cycle if there 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 cyclic 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 \( G \) iff \((u_i, u_{i+1}) \in E(G) \) for all \( 1 \leq i < k \) and if all vertices \( u_1, \ldots, u_k \) are pairwise distinct. Two vertices \( u, v \in G \) are at distance \( k \) in \( G \) iff 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.

The following definitions that are made for directed graphs. Let \( G \) be a digraph. A sequence \( u_1, \ldots, u_k \) with \((u_i, u_{i+1}) \in E(G) \) for all \( 1 \leq i \leq k-1 \) is called a directed path from \( u_1 \) to \( u_k \). A sequence \( u_1, \ldots, u_k \) is called a directed cycle of \( G \) if it is a directed path and additionally \((u_k, u_1) \) is an arc of \( G \). For some notions for digraphs \( G \) we just use the corresponding notions for undirected graphs applied to the undirected graph of \( G \): A digraph
$G$ is called *weakly connected* if the undirected graph of $G$ is connected. Equivalently, $G$ is weakly 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 *weakly connected component* of $G$ is a maximal weakly connected induced subgraph of $G$. A graph $G$ is called *strongly connected* if for all vertices $x, y \in V(G)$ there is a directed path from $a$ to $b$ in $G$. Two vertices $u, v \in V(G)$ are *at distance* $k$ in $G$ if they are at distance $k$ in the undirected graph of $G$.

Most of the definitions for graphs in this text are analogous for directed and for undirected graphs. We therefore sometimes do not explicitly mention whether we work with directed or undirected graphs. We therefore sometimes do not explicitly mention whether we work with directed or undirected graphs, but just state certain concepts for graphs, for instance in Subsection 1.4 or 1.5. We want to remark that almost all definitions in this section have generalizations to relational structures; however, we focus exclusively on graphs in this course since they allow to reach the key ideas of the underlying theory with a minimum of notation.

### 1.2 Graph Homomorphisms

Let $G$ and $H$ be directed graphs. A *homomorphism* from $G$ to $H$ is a mapping $h: V(G) \rightarrow V(H)$ 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 $G \rightarrow H$. Two directed graphs $G$ and $H$ are *homomorphically equivalent* if there is a homomorphism from $G$ to $H$ and a homomorphism from $H$ to $G$.

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

- a digraph is called *balanced* if it homomorphically maps to a directed path $\overrightarrow{P}_n$;
- 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 graph $H_1 \in C_1$ and a digraph $H_2 \in C_2$ such that $H_1 \rightarrow H_2$. 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*). 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 [27]. There is also a connection of the direct product to category theory [23], 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 $E(H_1 \times H_2)$ if $(u_1, v_1) \in E(H_1)$ and $(u_2, v_2) \in E(H_2)$. Note that the product is associative.

\(^1\)We want to warn the reader that there are several other notions of graph products that have been studied.
and commutative, and we therefore do not have to 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 \( C_1 \lor C_2 \in \mathcal{D} \), the join of \( C_1 \) and \( C_2 \), such that \( C_1 \leq (C_1 \lor C_2) \) and \( C_2 \leq (C_1 \lor C_2) \), and such that for every \( U \in \mathcal{D} \) with \( C_1 \leq U \) and \( C_2 \leq U \) we have \( U \leq C_1 \lor C_2 \).

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

**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 open combinatorial questions.

**Conjecture 1** (Hedetniemi). Suppose that \( G \times H \rightarrow K_n \). Then \( G \rightarrow K_n \) or \( H \rightarrow K_n \).

This conjecture is easy for \( n = 1 \) and \( n = 2 \) (Exercise 4), and has been solved for \( n = 3 \) by Norbert Sauer [15]. For \( n = 4 \) the conjecture is already open.

Clearly, \((\mathcal{D}, \leq)\) has infinite ascending chains, that is, sequences \( C_1, C_2, \ldots \) such that \( C_i \leq C_{i+1} \) for all \( i \in \mathbb{N} \). Take for instance \( C_i := \overrightarrow{P}_i \). 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 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 \( p_0, \ldots, p_n \in \{0, 1\}^* \) and \( q_0, \ldots, q_m \in \{0, 1\}^* \), respectively, the concatenation \( P \circ Q \) of \( P \) and \( Q \) is the oriented path represented by \( p_0, \ldots, p_n, q_1, \ldots, q_m \). 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 \( Z_{k,l} \) 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. Consider the orientation of a path \( 11(01)^{k-1}1 \). Our infinite antichain now consists of the equivalence classes containing the graphs \( Z_{k,k} \) for \( k \geq 2 \).

A strong homomorphism between two directed graphs \( G \) and \( H \) is a mapping from \( V(G) \) to \( V(H) \) such that \((f(u), f(v)) \in E(G)\) if and only if \((u, v) \in E(H)\) for all \( u, v \in V(G) \). An isomorphism between two directed graphs \( G \) and \( H \) is a bijective strong homomorphism from \( G \) to \( H \), i.e., \( f(u) \neq f(v) \) for any two distinct vertices \( u, v \in V(G) \).
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 \((\bar{C}_3)^3\)?

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. Proof the Hedetniemi conjecture for \(n = 1\) and \(n = 2\).

5. Show that a digraph \(G\) homomorphically maps to \(\bar{P}_1\) if and only if \(\bar{P}_2\) does not homomorphically map to \(G\).

6. Construct an orientation of a tree that is not homomorphically equivalent to an orientation of a path.

7. Construct a balanced orientation of a cycle that is not homomorphically equivalent to an orientation of a path.

8. Show that a digraph \(G\) homomorphically maps to \(T_3\) if and only if \(\bar{P}_3\) does not map to \(G\).

1.3 The \(H\)-coloring 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 \(H\)-coloring 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 precolored \(H\)-coloring 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 \(H\)-coloring 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 \(H\)-coloring problem reduces to the precolored \(H\)-coloring problem (it is a special case: the partial map might have an empty domain), and that the precolored \(H\)-coloring problem reduces to the list \(H\)-coloring 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 generalization of all those problems, and many more. It is defined not only for digraphs \(H\), but more generally for relational structures \(S\). Relational structures are the generalization of graphs where have many relations of arbitrary arity instead of just one binary edge relation. The constraint satisfaction problem will be introduced formally in Section 4.

Note that since graphs can be seen as a special case of digraphs, \(H\)-coloring 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 11.

For every finite graph \(H\), the \(H\)-coloring 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 precolored and the list $H$-coloring problem.

We have also seen that the $K_l$-coloring problem is the classical $l$-coloring problem. Therefore, we already know graphs $H$ whose $H$-coloring problem is NP-complete. However, for many graphs and digraphs $H$ (see Exercise 12 and 13) $H$-coloring can be solved in polynomial time. The following research question is open.

**Question 1.** For which digraphs $H$ can the $H$-coloring problem be solved in polynomial time?

As we will see later, if we solve the classification question for the precolored $H$-coloring problem, then we solve Question 1, and vice versa. Quite surprisingly, the same is true classifying the complexity of constraint satisfaction problems: a complete classification into NP-complete and P for the $H$-coloring problem would imply a classification for the class of all CSPs, and vice versa [18].

The list $H$-coloring 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 [9]. Alternative proofs can be found in [2,10]. Also see Section 9.

It has been conjectured by Feder and Vardi [18] that $H$-coloring is for any finite digraph $H$ either NP-complete or can be solved in polynomial time. This is the so-called dichotomy conjecture and it is open as well. It was shown by Ladner that unless P=NP there are infinitely many complexity classes between P and NP; so the conjecture says that for $H$-coloring these intermediate complexities do not appear. It is known that the dichotomy conjecture is true for finite undirected graphs.

**Theorem 1.4** (of [21]). If $H$ homomorphically maps to $K_2$, or contains a loop, then $H$-coloring can be solved in polynomial time. Otherwise, $H$-coloring is NP-complete.

The case that $H$ homomorphically maps to $K_2$ will be the topic of Exercise 12. We will see in Section 6.3 how to obtain the other part of the statement of Theorem 1.4 from more general principles.

**Exercises.**

9. 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).

10. Let $H$ be a finite digraph such that 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.

11. Let $G$ and $H$ be directed graphs, and suppose that $H$ is symmetric. Show that $f: V(G) \rightarrow V(H)$ 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 $H$.

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

13. Prove that CSP($T_3$) can be solved in polynomial time.
14. Prove that CSP($\bar{G}_3$) can be solved in polynomial time.

15. Let $N$ be the set \{Z₁, Z₂, Z₃, ...\}. Show that a digraph $G$ homomorphically maps to $\bar{P}_2$ if and only if no digraph in $N$ homomorphically maps to $G$.

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 $G₁$ and $G₂$ be cores of $H$, and $f₁: H \to G₁$ and $f₂: H \to G₂$ be homomorphisms. Let $e₁$ be the restriction of $f₁$ to $V(G₂)$. Suppose for contradiction that $e₁$ is not an embedding. Since $e₁$ is a homomorphism, there must be distinct non-adjacent $x, y$ in $V(G₂)$ such that either $(e₁(x), e₁(y)) ∈ E$ or $e₁(x) = e₁(y)$. Since $f₂$ is a homomorphism, it follows that $f₂ \circ e₁$ cannot be an embedding either. But $f₂ \circ e₁$ is an endomorphism of $G₂$, contradicting the assumption that $G₂$ is a core.

Hence, $e₁$ is indeed an embedding. Similarly, the restriction $e₂$ of $f₂$ to $V(G₁)$ is an embedding of $G₁$ into $H$. But then it follows that $|V(G₁)| = |V(G₂)|$, and since $H$ is finite it follows that $e₁$ is surjective. We have thus found the desired isomorphism between $G₁$ and $G₂$. □

Since a core $G$ of a finite digraph $H$ is unique up to isomorphism, we call $G$ the core of $H$. Cores can be characterized in many different ways; for some of them, see Exercise 17. There are examples of infinite digraphs that do not have a core in the sense defined above; see Exercise 19. Since a digraph $H$ and its core have the same CSP, it suffices to study CSP($H$) for core digraphs $H$ only. As we will see below, this has several advantages.

The precolored CSP for a digraph $H$ is the following computational problem. Given is a finite digraph $G$ and a partial mapping $c: V(G) \to V(H)$. The question is whether $c$ can be extended to a homomorphism from $G$ to $H$. In other words, we want to find a homomorphism from $G$ to $H$ where some vertices have a pre-described image.

**Proposition 1.6.** Let $H$ be a core. Then CSP($H$) and the precolored CSP for $H$ are linear-time equivalent.

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

We show the converse reduction by induction on the size of the image of partial mapping $c$ in instances of precolored CSP($H$). Let $(G, c)$ be an instance of precolored CSP($H$) where $c$ has an image of size $k ≥ 1$. 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 identify 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 identifying $x \in V(G)$ 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 $f(u')$ to $u$ if vertex $u'$ is a copy of a vertex $u \in V(H)$. 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, is an automorphism $\alpha$ of $H$. Moreover, $\alpha$ fixes $v$ for all $v \in V(H)$ in the image of $c'$. Let $\beta$ be the inverse of $\alpha$, i.e., let $\beta$ 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 $\alpha$. 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 $c$. $\square$

We have already seen in Exercise 10 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 10 requires in the worst-case $|V(G)|^2$ many executions of the decision procedure for CSP($H$). Using the concept of cores and the precolored CSP (and its equivalence to the CSP) we can give a faster method to construct homomorphisms.

**Proposition 1.7.** 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 can assume without loss of generality that $H$ is a core (since $H$ and its core have the same CSP). By Proposition 1.6, there is an algorithm $B$ for precolored 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. $\square$

We want to mention without proof is that it is NP-complete to decide whether a given digraph $H$ is not a core [22].
Exercises.

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

17. 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

18. Prove that the core of a strongly connected graph is strongly connected.

19. Show that the infinite digraph $(\mathbb{Q},<)$ has endomorphisms that are not automorphisms.

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

20. 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 $x \in V(H)$.

21. 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.

22. 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.

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

24. Determine the computational complexity of $\text{CSP}(H)$ for

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

1.5 Polymorphisms

Polymorphisms are a powerful tool for analyzing 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 1.8.** A homomorphism from $H^k$ to $H$, for $k \geq 1$, is called a ($k$-ary) polymorphism of $H$.

In other words, a mapping from $V(H)^k$ to $V(H)$ is a polymorphism of $H$ iff $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)) \in E(H)$ whenever $(u_1, v_1), \ldots, (u_k, v_k)$ are arcs in $E(H)$. Note that any graph $H$ has all projections as polymorphisms, i.e., all mappings $p: V(H)^k \to V(H)$ that satisfy for some $i$ the equation $p(x_1, \ldots, x_k) = x_i$ for all $x_1, \ldots, x_k \in V(H)$. An operation $f: V(H)^k \to V(H)$ is called

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

We say that a $k$-ary operation $f$ depends on an argument $i$ iff there is no $k-1$-ary operation $f'$ such that $f(x_1, \ldots, x_k) = f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k)$. We can equivalently characterize $k$-ary operations that depend on the $i$-th argument by requiring that there are elements $x_1, \ldots, x_k$ and $x'_i$ such that $f(x_1, \ldots, x_k) \neq f(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_k)$. We say that an operation $f$ is essentially unary iff there is an $i \in \{1, \ldots, k\}$ and a unary operation $f_0$ such that $f(x_1, \ldots, x_k) = f_0(x_i)$. Operations that are not essentially unary are called essential.  

**Lemma 1.9.** An operation $f$ is essentially unary if and only if $f$ depends on at most one argument.

**Proof.** It is clear that an operation that is essentially unary depends on at most one argument. Conversely, suppose that $f$ is $k$-ary and depends only on the first argument. Let $i \leq k$ be the maximal such that there is an operation $g$ with $f(x_1, \ldots, x_k) = g(x_1, \ldots, x_i)$. 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 $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_i) = g'(x_1, \ldots, x_{i-1})$, in contradiction to the choice of $i$. \qed

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

An example of a non-projective core is $T_n$: it has the (idempotent) polymorphism $(x, y) \mapsto \min(x, y)$.

**Proposition 1.11.** Let $H$ be a projective core. Then all polymorphisms of $H$ are essentially unary.

**Proof.** Let $f$ be an $n$-ary polymorphism of $H$. Then the function $e(x) := f(x, \ldots, x)$ is an endomorphism of $H$, and since $H$ is a core, a is an automorphism of $H$. Let $i$ be the inverse of $H$. Then $f'$ given by $f'(x_1, \ldots, x_n) := i(f(x_1, \ldots, x_n))$ is an idempotent polymorphism of $H$. By projectivity, there exists a $j \leq n$ such that $f'(x_1, \ldots, x_n) = x_j$. Hence, $f(x_1, \ldots, x_n) = a(f'(x_1, \ldots, x_n)) = a(x_j).$ \qed

**Exercises.**

25. Suppose that CSP($G$) and CSP($H$), for two digraphs $G$ and $H$, can be solved in polynomial time. Show that CSP($G \times H$) and CSP($G \sqcup H$) can be solved in polynomial time as well.

2 The Arc-consistency Procedure

The arc-consistency procedure is one of the most fundamental and well-studied algorithms that is applied for CSPs. This procedure was first discovered for constraint satisfaction problems in Artificial Intelligence [32, 34]: 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 $x$
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 CSP\((H)\).

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\(_H\) solves 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 [32], 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 to remove a value from \(L(x_0)\) or \(L(x_1)\). Whenever we remove a value from a list
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).

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_H does not solve CSP(H). Even in this situation the arc-consistency procedure might be useful for pruning the search space in exhaustive approaches to solve 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 ∈ V(G), and set L(x) to {u} for some u ∈ 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 ∈ V(G), which defines a homomorphism from G to H.
2.1 The Power Set 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$_H$ solves CSP($H$), found by Feder and Vardi [18].

**Definition 2.1.** For a digraph $H$, the (power-) set 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 set 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 $G \not\rightarrow P(H)$.

*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 $f: G \rightarrow P(H)$ 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. $\square$

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

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

Conversely, suppose that $P(H) \rightarrow H$. If AC$_H$ rejects a graph $G$, then Lemma 2.2 asserts that $G \not\rightarrow P(H)$. Since $H \rightarrow P(H)$, we conclude that $G \not\rightarrow H$. If AC$_H$ does accepts $G$, then the lemma asserts that $G \rightarrow P(H)$. Composing homomorphisms, we obtain that $G \rightarrow H$. $\square$

**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 precolored 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 precoloring 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 precolored CSP for $H$, then the classical version of AC$_H$ solves CSP($H$). Hence, it follows that the following are equivalent:

- AC$_H$ solves CSP($H$);
- the above modification of AC$_H$ solves the precolored CSP for $H$;
- $P(H) \rightarrow H$.
Note that the condition given in Theorem 2.3 can be used to decide algorithmically whether CSP$(H)$ can be solved by AC$_H$, because it suffices to test whether $P(H)$ homomorphically maps to $H$. Such problems about deciding properties of 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 CSP$(H)$ by AC$_H$ 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 $P(H) \rightarrow H$ 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 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 detects 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$_H$ still accepts when $L(x) = \{u\}$, then we continue with another element $y$ of $V(P(H))$, setting $L(y)$ to $\{v\}$ for some $v \in L(y)$. If AC$_H$ accepts, we continue like this, 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 precolored 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’. \(\square\)

**Question 2.** What is the computational complexity to decide for a given digraph $H$ whether $H$ has tree duality? What is the computational complexity when we further assume that $H$ is a core? 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 iff there exists a (not necessarily finite) set $N$ 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 $N$ homomorphically maps to $G$.

We think of the set $N$ in Definition 2.6 as an obstruction set for CSP$(H)$. The pair $(N, H)$ is called a duality pair. We have already encountered such an obstruction set in Exercise 13, where $H = T_2$, and $N = \{\overrightarrow{P}_2\}$. In other words, $\{(\overrightarrow{P}_2), T_2\}$ is a duality pair. Other duality pairs are $(\{P_3\}, T_3)$ (Exercise 8), and $(\{Z_1, Z_2, \ldots\}, \overrightarrow{P}_2)$ (Exercise 15).
Theorem 2.7 is a surprising link between the completeness of the arc-consistency procedure, tree duality, and the set graph, and was discovered by Feder and Vardi [17] 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. If every orientation of a tree that homomorphically maps to $G$ also homomorphically maps to $H$, then $G$ homomorphically maps to $H$;

3. $P(H)$ homomorphically maps to $H$;

4. The $AC_H$ solves CSP($H$).

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

$1 \Rightarrow 2$: Suppose $H$ has tree duality, and let $N$ be the tree obstructions from Definition 2.6. Let $G$ be a digraph, and suppose that every tree that homomorphically maps to $G$ also homomorphically maps to $H$. We have to show that $G$ homomorphically maps to $H$. No member of $N$ homomorphically maps to $H$: otherwise by the definition of tree duality $H$ would not be homomorphic to $H$, a contradiction. So, none of the orientations of trees in $N$ homomorphically maps to $G$, and by tree duality, $G$ homomorphically maps to $H$.

$2 \Rightarrow 3$: 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 $H$.

$4 \Rightarrow 1$: Suppose that $AC_H$ solves CSP($H$). We have to show that $H$ has tree duality. Let $N$ 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 $T \in N$ 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 [30]. 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 \( x_1 \ldots i \). 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 identify 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 identifying the roots of all \( T_{x,u} \) to 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.

Therefore, for every graph \( G \) that does not homomorphically map to \( H \) we find an orientation of a tree from \( N \) that homomorphically maps to \( G \), and hence \( H \) has tree-duality. □

### 2.3 Totally Symmetric Polymorphisms

There is also a characterization of the power of arc consistency which is based on polymorphisms, due to [14].

**Definition 2.8.** A function \( f : D^k \to D \) is called totally symmetric if \( f(x_1, \ldots, x_m) = f(y_1, \ldots, y_m) \) whenever \( \{x_1, \ldots, x_m\} = \{y_1, \ldots, y_m\} \).

**Theorem 2.9.** Let \( H \) be a digraph. Then the following are equivalent.

- \( P(H) \) homomorphically maps to \( H \);
- \( H \) has totally symmetric polymorphisms of all arities;
- \( H \) has a totally symmetric polymorphism of arity \( 2|V(H)| \).

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

Conversely, suppose that \( f \) is a totally symmetric polymorphism of arity \( 2|V(H)| \). Let \( g : V(P(H)) \to V(H) \) be defined by \( g(\{x_1, \ldots, x_n\}) := f(x_1, \ldots, x_{n-1}, x_n, x_n, \ldots, x_n) \), which is well-defined by the properties of \( f \). Let \((U, V) \in E(P(H))\), and let \( x_1, \ldots, x_p \) be an enumeration of the elements of \( U \), and \( y_1, \ldots, y_q \) be an enumeration of the elements of \( V \). The properties of \( P(H) \) imply that there are \( y'_1, \ldots, y'_p \in V \) and \( x'_1, \ldots, x'_q \in U \) such that \((x_1, y'_1), \ldots, (x_p, y'_p) \in E(H) \) and \((x'_1, y_1), \ldots, (x'_q, y_q) \in E(H) \). Since \( f \) preserves \( E \),

\[
g(U) = g(\{x_1, \ldots, x_p\}) = f(x_1, \ldots, x_p, x'_1, \ldots, x'_q, x_1, \ldots, x_1)\]
Suppose that 1 \leq i < n.

\begin{align*}
g(V) = g(\{y_1, \ldots, y_q\}) &= f(y_1', \ldots, y_p', y_1, \ldots, y_q, y_1', \ldots, y_1').
\end{align*}

\square

### 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 \( f: D^2 \rightarrow D \) is called \textit{commutative} if it satisfies

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

It is called \textit{associative} if it satisfies

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

\textbf{Definition 2.10.} A binary operation is called a \textit{semilattice operation} \( f \) if it is associative, commutative, and idempotent.

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

\textbf{Theorem 2.11.} Let \( H \) be a finite digraph. If \( H \) has a semilattice polymorphism, then \( P(H) \rightarrow H \). If \( H \) is a core, then \( P(H) \rightarrow H \) if and only if \( H \) is homomorphically equivalent to a directed graph with a semilattice polymorphism.

\textit{Proof.} Suppose that \( H \) has the semilattice polymorphism \( f \). The operation \( g \) defined as \((x_1, \ldots, x_n) \mapsto f(x_1, f(x_2, f(\ldots, f(x_{n-1}, x_n) \ldots)))\) is a totally symmetric polymorphism of \( G \). Then Theorem 2.9 implies that \( P(H) \rightarrow H \).

For the second part of the statement, suppose first that \( P(H) \rightarrow H \). Since \( H \rightarrow P(H) \), we have that \( H \) and \( P(H) \) are homomorphically equivalent. It therefore 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 \( f \). By the first part of the statement, \( P(G) \rightarrow G \). Hence, \( P(G) \rightarrow H \) since \( G \rightarrow H \). Finally, observe that as \( H \) is a subgraph of \( G \), the graph \( P(H) \) is an induced subgraph of \( P(G) \). Therefore, \( P(H) \rightarrow P(G) \rightarrow G \rightarrow H \), as desired. \hfill \square

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

\textbf{Corollary 2.12.} \( \text{CSP}(H) \) can be solved by \( \text{AC}_H \) if \( H \) is an orientation of a path.

\textit{Proof.} Suppose that \( 1, \ldots, n \) 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. \hfill \square
We want to remark that there are orientations of trees $H$ with an NP-complete $H$-coloring problem (the smallest known graph has 45 vertices [24]). So, unless P=NP, this shows that there are orientations of trees $H$ that do not have tree-duality.

Exercises.

26. There is only one unbalanced cycle $H$ on four vertices that is a core and not the directed cycle. Show that for this graph $H$ the constraint satisfaction problem can not be solved by $\text{AC}_H$.

27. Show that $\text{CSP}(T_n)$ can be solved by the arc-consistency procedure, for every $n \geq 1$.

28. Let $H$ be a finite digraph. Show that $P(H)$ contains a loop if and only if $H$ contains a directed cycle.

29. Show that the previous statement is false for infinite digraphs $H$.

30. 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.

31. Show that an orientation of a tree homomorphically maps to $H$ if and only if it homomorphically maps to $P(H)$ (Hint: use parts of the proof of Theorem 2.7).

32. Let $H$ be a finite directed graph. Then $\text{AC}_H$ rejects an orientation of a tree $T$ if and only if there is no homomorphism from $T$ to $H$ (in other words, $\text{AC}_H$ solves $\text{CSP}(H)$ 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 12) and $H = \bar{C}_3$ (see Exercise 14). The idea is to maintain a list of pairs from $V(H)^2$ for each pair of distinct 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.

If we modify the path-consistency procedure as presented in Figure 3 by also maintaining lists $L(x,x)$ (i.e., we maintain lists also for pairs with non-distinct elements), and also process triples $x,y,z$ that are not pairwise distinct, we obtain the so-called strong path-consistency procedure. This procedure is at least as strong as the arc-consistency procedure, because the lists $L(x,x)$ and the rules of the strong path-consistency procedure for $x = y$ simulate the rules of the arc-consistency procedure. We will from now on work with strong path consistency only, which has practical and theoretical properties that are superior to path
**PC$_H(G)$**

Input: a finite digraph $G$.

Data structure: for each $x, y \in V(G)$, a list $L(x, y)$ of pairs from $V(H)^2$

For all $(x, y) \in V(G)^2$

- If $(x, y) \in E(G)$ then $L(x, y) := E(H)$
- else $L(x, y) := V(H)^2$

Do

- For all distinct vertices $x, y, z \in V(G)$:
  - For each $(u, v) \in L(x, y)$:
    - If there is no $w \in V(H)$ such that $(u, w) \in L(x, z)$ and $(w, v) \in L(z, y)$ then
      - Remove $(u, v)$ from $L(x, y)$
  - If $L(x, y)$ is empty then **reject**

Loop until no list changes

Figure 3: The strong path-consistency procedure for $H$-coloring

consistency. Sometimes we might even write path consistency in the following even if strong path consistency is meant.

In Subsection 3.1 we will see many examples of digraphs $H$ where the path-consistency algorithm solves the $H$-coloring problem, but the arc-consistency algorithm does not. The greater power of the path-consistency procedure comes at the price of a larger worst-case running time: while we have seen a linear-time implementation of the path-consistency procedure, the best known implementations of the path-consistency procedure require cubic time in the size of the input (see Exercise 33).

**The $k$-consistency procedure.** The strong path-consistency procedure can be generalised further to the $k$-consistency procedure. In fact, arc- and path-consistency procedure are just the special case 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 generalize 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 strong path consistency alias 3-consistency is of particular theoretical importance, due to the following recent result.

**Theorem 3.1** (Barto and Kozik [4]). If CSP($H$) can be solved by $k$-consistency for some $k \geq 3$, then CSP($H$) can also be solved by 3-consistency.

More on this result can be found in Section 8.

**Exercises**
33. Show that the path-consistency procedure for the $H$-coloring 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 $H$-coloring problem. Again, this condition was first discovered in more general form by Feder and Vardi [18]; 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 $x, y \in D$:

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

As an example, let $D = \{1, \ldots, n\}$, 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 $\text{median}(x, y, z)$ is defined to be $b$.

**Theorem 3.3** (of [18]). Let $H$ be a finite digraph. If $H$ has a polymorphism that is a majority function, then the $H$-coloring problem can be solved in polynomial time (by the strong path-consistency procedure).

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

**Lemma 3.4.** Let $f$ be a $k$-ary polymorphism of a digraph $H$. Let $G$ be an instance of the $H$-coloring problem, and $L(x, y)$ be the final list computed by the strong path-consistency procedure for $x, y \in V(G)$. Then $f$ preserves $L(x, y)$, i.e., if $(u_1, v_1), \ldots, (u_k, v_k) \in L(x, y)$, then $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)) \in L(x, y)$.

**Proof.** We prove by induction over the execution of the algorithm that for all $x, y \in V(G)$ and $(u_1, v_1), \ldots, (u_k, v_k) \in L(x, y)$, at all times in the execution the pair $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k))$ is in the current list for $x, y$. In the beginning, this is true because $f$ is a polymorphism of $H$. For the inductive step, let $x, y, z \in V(G)$ and $(u_1, v_1), \ldots, (u_k, v_k) \in L(x, y)$ be arbitrary. By inductive assumption, $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k))$ is in the current list for $x, y$. We have to show that the condition in the innermost if-statement does not apply to $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k))$. By definition of the procedure, for each $i \in \{1, \ldots, k\}$ there exists a $w_i$ such that $(u_i, w_i) \in L(x, z)$ and $(w_i, v_i) \in L(z, y)$. By inductive assumption, $(f(u_1, \ldots, u_k), f(w_1, \ldots, w_k)) \in L(x, z)$ and $(f(w_1, \ldots, w_k), f(v_1, \ldots, v_k)) \in L(z, y)$. Hence, $(f(u_1, \ldots, u_k), f(v_1, \ldots, v_k))$ will not be removed in the next step of the algorithm.

**Proof of Theorem 3.3.** Suppose that $H$ has a majority polymorphism $f : V(H)^3 \to V(H)$, and let $G$ be an instance of the $H$-coloring problem. Clearly, if the path-consistency procedure derives the empty list for some pair $(u, v)$ 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 $L(x, y)$ is non-empty for all pairs $(x, y)$ from $V(G)^2$. We have to show that there exists a homomorphism from $G$ to $H$. We say that a homomorphism $h$ from an induced subgraph $G'$ of $G$ to $H$ preserves the lists iff $(h(x), h(y)) \in L(x, y)$ for all $x, y \in V(G')$. The proof shows by induction
on \( i \) that any homomorphism from a subgraph of \( G \) on \( i \) 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 preserves the lists.

For the base case of the induction, observe that for all vertices \( x_1, x_2, x_3 \in V(G) \) every mapping \( h \) from \( \{x_1, x_2\} \) to \( V(H) \) such that \((h(x_1), h(x_2)) \in L(x_1, x_2)\) can be extended to \( x_3 \) such that \((h(x_1), h(x_3)) \in L(x_1, x_3)\) and \((h(x_3), h(x_2)) \in L(x_3, x_2)\) (and hence preserves the lists), because otherwise the path-consistency procedure would have removed \((h(x_1), h(x_2))\) from \( L(x_1, x_2)\).

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 \((h(x), h(y)) \in L(x,y)\) (since \( h \) preserves the lists). If \( y \in \{x_1, x_2, x_3\} \) and \((h(x), h(y)) \in L(x,y)\) (and hence preserves the lists), because otherwise the path-consistency procedure would have removed \((h(x), h(y))\) from \( L(x,y)\).

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 \((h(x_1), v) \in L(x, y)\) (otherwise the path-consistency procedure would have removed \((h(x_1), h(x_2))\) from \( L(x, y)\)). By the properties of \( f \), we have \((h(x_1), h(y)) = (f(h(x_1), h(y), h'(y)) = f(v, h_2(y), h_3(y))\). 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 \( H \).

As an example, 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 \( u < v \). 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 \( \text{median}(u_1, u_2, u_3) < \text{median}(v_1, v_2, v_3) \). This yields a new proof that the \( H \)-coloring problem for \( H = T_n \) is tractable.

**Corollary 3.5.** The path-consistency procedure solves the \( H \)-coloring 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 \([16]\). We only prove a weaker result here.

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

**Proof.** Let \( f \) be the ternary operation that maps \( u, v, w \) to the \( u \) if \( u, v, w \) are pairwise distinct, and otherwise acts as a majority operation. We claim that \( f \) is a polymorphism of \( C_n \). Let \((u, u'), (v, v'), (w, w') \in E(C_n)\) be arcs. If \( u, v, w \) are all distinct, then \( u', v', w' \) are clearly all distinct as well, and hence \((f(u, v, w), f(u', v', w')) = (u, u') \in E(C_n)\). Otherwise, if two elements of \( u, v, w \) are equal, say \( u = v \), then \( u' \) and \( v' \) must be equal as well, and hence \((f(u, v, w), f(u', v', w')) = (u, u') \in E(C_n)\). \( \square \)
3.2 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. Hence, Theorem 3.1 implies the following.

**Theorem 3.7.** Let $H$ be a directed graph with a near unanimity polymorphism. Then $PC_H$ solves CSP($H$).

3.3 Maltsev Polymorphisms

**Definition 3.8.** An ternary function $f: D^3 \to D$ 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 1.** Let $D := \{0, \ldots, n - 1\}$. Then the function $f: D^3 \to D$ given by $(x, y, z) \mapsto x - y + z \mod n$ is a Maltsev operation, since $x - x + z = z$ and $x - z + z = z$. For $n = 2$, this is even a minority operation. If $n > 2$, this function is not a minority, since then $1 - 2 + 1 = 0 \not\equiv 2 \mod n$.

The function $f$ from the example above 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

$$f(u_1, u_2, u_3) \equiv u_1 - u_2 + u_3 \equiv (v_1 - 1) - (v_2 - 1) + (v_3 - 1) \equiv f(v_1, v_2, v_3) + 1 \mod n.$$ 

The following surprising result appeared in 2011.

**Theorem 3.9** (Kazda [28]). A digraph $G$ has a Maltsev polymorphism if and only if $G$ is homomorphically equivalent to a directed path, or to a disjoint union of directed cycles.

Hence, for digraphs $H$ with a Maltsev polymorphism, strong path consistency solves the $H$-coloring problem. The following is even more recent that Kazda’s result.

**Theorem 3.10** (Corollary 4.12 in [13]). A digraph $G$ has a conservative Maltsev polymorphism if and only if $G$ has a conservative majority polymorphism.

Hence, when $H$ has a conservative Maltsev polymorphism, then strong path consistency solves even the list $H$-coloring problem (see Exercise 38).
Exercises.

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

35. Show that every orientation of a path has a majority operation.

36. 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 26). Show that for this graph $H$ the $H$-coloring problem can be solved by the path-consistency procedure.

37. Modify the path-consistency procedure such that it can deal with instances of the precolored $H$-coloring problem. Show that if $H$ has a majority operation, then the modified path-consistency procedure solves the precolored $H$-coloring problem.

38. Modify the path-consistency procedure such that it can deal with instances of the list $H$-coloring problem. Show that if $H$ has a conservative majority operation, then the modified path-consistency procedure solves the list $H$-coloring problem.

39. 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 $(x, y) \in E$ iff $I_x$ 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 precolored $H$-coloring problem for interval graphs $H$ can be solved in polynomial time. Hint: use the modified path-consistency procedure in Exercise 37.

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

41. Prove that $H$-coloring can be solved in polynomial time when $H$ is the digraph from the previous exercise.

42. 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 $H$-coloring problem.

4 Logic

Let $\tau$ be a relational signature. A first-order $\tau$-formula $\phi(x_1, \ldots, x_n)$ is called primitive positive (in the database literature also conjunctive query) if it is of the form

$$\exists x_{n+1}, \ldots, x_m (\psi_1 \land \cdots \land \psi_l)$$

where $\psi_1, \ldots, \psi_l$ are atomic $\tau$-formulas, i.e., formulas of the form $R(y_1, \ldots, y_k)$ with $R \in \tau$ and $y_i \in \{x_1, \ldots, x_m\}$, of the form $y = y'$ for $y, y' \in \{x_1, \ldots, x_m\}$, or $\bot$ (for false). As usual, formulas without free variables are called sentences. Note that we do not need a symbol $\top$ for true since we can use the primitive positive sentence $\exists x. x = x$ to express it; we sometimes use $\top$ as a shortcut for $\exists x. x = x$.

It is possible to rephrase the $H$-coloring problem and its variants using primitive positive sentences.
Definition 4.1. Let $\mathfrak{B}$ be a (possibly infinite) structure with a finite relational signature $\tau$. Then $\text{CSP}(\mathfrak{B})$ is the computational problem to decide whether a given primitive positive $\tau$-sentence $\phi$ is true in $\mathfrak{B}$.

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

Example 2 (Disequality constraints). Consider the problem $\text{CSP}([1, 2, \ldots, n]; \neq)$. An instance of this problem can be viewed as an (existentially quantified) set of variables, some linked by disequality constraints. Such an instance is false 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 $K_n$.

4.1 Canonical Conjunctive Queries

To every relational $\tau$-structure $\mathfrak{A}$ we can associate a $\tau$-sentence, called the canonical conjunctive query of $\mathfrak{A}$, and denoted by $Q(\mathfrak{A})$. 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 $Q(K_3)$ of the complete graph on three vertices $K_3$ is the formula

$$\exists u \exists v \exists w \ (E(u, v) \land E(v, u) \land E(v, w) \land E(w, v) \land E(u, w) \land E(w, u)).$$

The proof of the following proposition is straightforward.

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

4.2 Canonical Databases

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

Then the domain of $D(\phi)$ is the set of variables (both the free variables and the existentially quantified variables) that occur in $\phi$. There is a tuple $(v_1, \ldots, v_k)$ in a relation $R$ of $D(\phi)$ iff $\phi$ 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 $\tau$, and let $\phi$ be a primitive positive $\tau$-sentence other than $\bot$. Then $\phi$ is true in $\mathfrak{B}$ if and only if $D(\phi)$ homomorphically maps to $\mathfrak{B}$.

\[\text{We deliberately use the word disequality instead of inequality, since we reserve the word inequality for the relation } x \leq y.\]
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(\mathcal{B}) can from now on be either finite structures \mathfrak{A} or primitive positive sentences \phi.

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

### 4.3 Primitive Positive Definability

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

When \mathfrak{A} is a \tau\text{-}structure, and \phi(x_1, \ldots, x_k) is a formula with \text{k} free variables \text{x}_1, \ldots, \text{x}_k, then the relation defined by \phi 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 3.** The relation \(E' := \{(a, b) \in \{0, 1, 2, 3, 4\}^2 \mid a \neq b\}\) is primitive positive definable in \(\tilde{C}_5\): the primitive positive definition is

\[
\exists p_1, p_2, p_3, q_1, q_2 \left( E(x_1, p_1) \land E(p_1, p_2) \land E(p_2, p_3) \land E(p_3, x_2) \\
\land E(x_1, q_1) \land E(q_1, q_2) \land E(q_2, x_2) \right)
\]

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 CSP(\mathcal{B}) is NP-hard, it suffices to show that there is a primitive positive definition of a relation \(\mathcal{R}\) such that CSP((\mathcal{B}, \mathcal{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.** Let \mathcal{B} be a structure with finite relational signature, and let \mathcal{R} be a relation that has a primitive positive definition in \mathcal{B}. Then CSP(\mathcal{B}) and CSP((\mathcal{B}, \mathcal{R})) are linear-time equivalent. They are also equivalent under deterministic log-space reductions.

**Proof.** It is clear that CSP(\mathcal{B}) reduces to the new problem. So suppose that \phi is an instance of CSP((\mathcal{B}, \mathcal{R})). Replace each conjunct \(R(x_1, \ldots, x_l)\) of \phi 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 \psi be the formula obtained in this way.

It is straightforward to verify that \phi is true in \(\mathcal{B}, R\) if and only if \psi is true in \mathcal{B}, and it is also clear that \psi can be constructed in linear time in the representation size of \phi. The observation that the reduction is deterministic log-space, we need the recent result that undirected reachability can be decided in deterministic log-space [38].
4.4 Primitive Positive Definability in Cores

An automorphism of a structure \( \mathcal{B} \) with domain \( B \) is an isomorphism between \( \mathcal{B} \) and itself. When applying an automorphism \( \alpha \) to an element \( b \) from \( B \) we omit brackets, that is, we write \( ab \) instead of \( \alpha(b) \). The set of all automorphisms \( \alpha \) of \( \mathcal{B} \) is denoted by \( \text{Aut}(\mathcal{B}) \), and \( \alpha^{-1} \) denotes the inverse map of \( \alpha \). Let \( (b_1, \ldots, b_k) \) be a \( k \)-tuple of elements of \( \mathcal{B} \). A set of the form \( S = \{ (ab_1, \ldots, ab_k) \mid \alpha \in \text{Aut}(\mathcal{B}) \} \) is called an orbit of \( k \)-tuples (the orbit of \( (b_1, \ldots, b_k) \)).

**Lemma 4.5.** Let \( \mathcal{B} \) 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 \( \mathcal{B} \) is primitive positive definable, then there is a polynomial-time reduction from \( \text{CSP}(\mathcal{B}, R) \) to \( \text{CSP}(\mathcal{B}) \).

**Proof.** Let \( \phi \) be an instance of \( \text{CSP}(\mathcal{B}, R) \) with variable set \( V \). If \( \phi \) 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 constraints 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 \( \theta \) of its orbits in \( \mathcal{B} \). Finally, move all quantifiers to the front, such that the resulting formula \( \psi \) is in prenex normal form and thus an instance of \( \text{CSP}(\mathcal{B}) \). Clearly, \( \psi \) can be computed from \( \phi \) in polynomial time. We claim that \( \phi \) is true in \( (\mathcal{B}, R) \) if and only if \( \psi \) is true in \( \mathcal{B} \).

Suppose \( \phi \) has a solution \( s : V \rightarrow B \). Let \( s' \) be the restriction of \( s \) to the variables of \( V \) that also appear in \( \phi \). Since \( (b_1, \ldots, b_n) \) satisfies \( \theta \), we can extend \( s' \) to the existentially quantified variables of \( \theta \) to obtain a solution for \( \psi \). In the opposite direction, suppose that \( s' \) is a solution to \( \psi \) over \( \mathcal{B} \). Let \( s \) be the restriction of \( s' \) to \( V \). Because \( (s(x_1), \ldots, s(x_k)) \) satisfies \( \theta \) it lies in the same orbit as \( (b_1, \ldots, b_k) \). Thus, there exists an automorphism \( \alpha \) of \( \mathcal{B} \) 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 \( \phi \) that have been replaced by \( x_i \) in \( \psi \) to the value \( b_i \) is a solution to \( \phi \) over \( (\mathcal{B}, R) \). \( \square \)

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 is an automorphism. All results we proved for cores of digraphs remain valid for cores of structures. In particular, every finite structure \( \mathcal{C} \) is homomorphically equivalent to a core structure \( \mathcal{B} \), which is unique up to isomorphism (see Section 1.4). For core structures, all orbits are primitive positive definable. This can be shown as in the proof of Proposition 1.6.

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

Proposition 4.6 and Lemma 4.5 have the following consequence.

**Corollary 4.7.** Let \( \mathcal{B} \) be a finite core structure with elements \( b_1, \ldots, b_n \) and finite signature. Then \( \text{CSP}(\mathcal{B}) \) and \( \text{CSP}(\mathcal{B}, \{b_1\}, \ldots, \{b_n\}) \) are polynomial time equivalent.

### 4.5 Primitive Positive Interpretations

Primitive positive definability is a powerful tool to study the computational complexity of CSPs, but sometimes we need more. In particular, we would like to have a mathematical tool for studying polynomial-time reductions between CSPs with different finite domains. TODO
Exercises.

43. Show that the relation \( \{(a, b, c) \in \mathbb{Q} \mid a < b \lor a < c\} \) is preserved by the function \( f : \mathbb{Q}^2 \to \mathbb{Q} \) given by \((x, y) \mapsto \max(x, y)\).

44. Let \( a_1, a_2, a_3 \in \mathbb{Q} \) be arbitrary. Show that the relation \( \{(x_1, x_2, x_3) \in \mathbb{Q}^3 \mid a_1x_1 + a_2x_2 + a_3x_3 \leq 0\} \) is preserved by the function \((x, y) \mapsto \max(x, y)\) if and only if at most one of \( a_1, a_2, a_3 \) is positive.

5 Relations and Functions

5.1 Function Clones

For \( n \geq 1 \) and a set \( D \), denote by \( \mathcal{O}^{(n)}_D \) the set \( D^{n^*} := (D^n \to D) \) of \( n \)-ary functions on \( D \). The elements of \( \mathcal{O}^{(n)}_D \) 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 \( \mathcal{O}_D := \bigcup_{n \geq 1} \mathcal{O}^{(n)}_D \). A function clone (over \( D \)) is a subset \( C \) of \( \mathcal{O}_D \) satisfying the following two properties:

- \( C \) contains all projections, that is, for all \( 1 \leq k \leq n \) it contains the operation \( \pi^{n}_k \in \mathcal{O}^{(n)}_D \) defined by \( \pi^{n}_k(x_1, \ldots, x_n) = x_k \), and

- \( C \) is closed under composition, that is, for all \( f \in C \cap \mathcal{O}^{(m)}_D \) and \( g_1, \ldots, g_n \in C \cap \mathcal{O}^{(m)}_D \) it contains the operation \( f(g_1, \ldots, g_n) \in \mathcal{O}^{(m)}_D \) 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 a function clone that will be introduced later in the course. In the literature, function clones are often called clones, or concrete clones; we prefer to use the terms ‘function clone’ and ‘clone’ in analogy to ‘permutation group’ and ‘group’.

When \( C \) is a function clone, then \( C' \) is called a subclone of \( C \) if \( C' \) is a function clone and \( C' \subseteq C \). When \( \mathcal{F} \) is a set of functions, we write \( \langle \mathcal{F} \rangle \) for the smallest function clone \( C \) which contains \( \mathcal{F} \), and call \( C \) the clone generated by \( \mathcal{F} \).

5.2 The Galois Connection Inv-Pol

The most important source of function clones in this text are polymorphism clones of digraphs and, more generally, structures.

Let \( f \) be from \( \mathcal{O}^{(n)}_D \), and let \( R \subseteq D^m \) be a relation. Then we say that \( f \) preserves \( R \) (and that \( R \) is invariant under \( f \)) iff \( f(r_1, \ldots, r_n) \in \rho \) whenever \( r_1, \ldots, r_n \in R \), where \( f(r_1, \ldots, r_n) \) is calculated componentwise. When \( \mathcal{B} \) is a relational structure with domain \( B \) then \( \text{Pol}(\mathcal{B}) \) contains precisely those functions that preserve \( \mathcal{B} \). It is easy to verify that \( \text{Pol}(\mathcal{B}) \) is a function clone. It will be convenient to define the operator \( \text{Pol} \) also for sets \( \mathcal{R} \) of relations over \( B \), writing \( \text{Pol}(\mathcal{R}) \) for the set of operations of \( \mathcal{O}_n \) that preserve all relations from \( \mathcal{R} \).

**Proposition 5.1.** Let \( \mathcal{B} \) be any structure. Then \( \text{Inv}(\text{Pol}(\mathcal{B})) \) contains the set of all relations that are primitive positive definable in \( \mathcal{B} \).
Lemma 5.5.  \[\text{Proof.}\] Direct consequence of Theorem 5.2 and Lemma 4.4.  \[\text{Definition in}\] 5.4.

5.3 Unary Clones

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

\[\text{Proof.}\] Direct consequence of Theorem 5.2 and Lemma 4.4.

5.3 Unary Clones

Definition 5.4.  For any set \(B\), the relations \(P_3\) and \(P_4\) over \(B\) are defined as follows.

\[P_3 = \{(a, b, c) \in B^3 \mid a = b \text{ or } b = c\}\]

\[P_4 = \{(a, b, c, d) \in B^4 \mid a = b \text{ or } c = d\}\]

Lemma 5.5. Let \(f \in \emptyset\) be an operation. Then the following are equivalent.

1. \(f\) is essentially unary.
2. $f$ preserves $P_3$.

3. $f$ preserves $P_4$.

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 $P_3$.

To show the implication from (2) to (3), we show the contrapositive, and assume that $f$ violates $P_4$. By permuting arguments of $f$, we can assume that there are an $l \leq k$ and 4-tuples $a^1, \ldots, a^k \in P_4$ with $f(a^1, \ldots, a^k) \notin P_4$ such that in $a^1, \ldots, a^l$ the first two coordinates are equal, and in $a^{l+1}, \ldots, a^k$ the last two coordinates are equal. Let $c$ be the tuple $(a^1, a^1, a^4, \ldots, a^4)$. Since $f(a^1, \ldots, a^k) \notin P_4$ we have $f(a^1, \ldots, a^1_c) \neq f(a^1, \ldots, a^k)$, and therefore $f(c) \neq f(a^1, \ldots, a^1_c)$ or $f(c) \neq f(a^1, \ldots, a^2_c)$. Let $d = (a^1, \ldots, a^1_c)$ in the first case, and $d = (a^1, \ldots, a^2_c)$ in the second case. Likewise, we have $f(c) \neq f(a^1, \ldots, a^3_c)$ or $f(c) \neq f(a^1, \ldots, a^4_c)$, and let $e = (a^1, \ldots, a^3_c)$ in the first, and $e = (a^1, \ldots, a^4_c)$ in the second case. Then for each $i \leq k$, the tuple $(d_i, c_i, e_i)$ is from $P_3$, but $(f(d), f(c), f(e)) \notin P_3$.

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 $(a^1(l), b_1(l), a^2(l), b_2(l)) \in P_4$ for all $l \leq k$, but $(f(a^1), f(b_1), f(a^2), f(b_2)) \notin P_4$, which shows that $f$ violates $P_4$.

For the implication from (4) to (1), suppose that $f$ depends only on the first argument. Let $i \leq k$ be maximal such that there is an operation $g$ with $f(x_1, \ldots, x_k) = g(x_1, \ldots, x_i)$. 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 $x_1, \ldots, x_n \in B$ we have $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_i) = g'(x_1, \ldots, x_{i-1})$, contradicting the choice of $i$.\hfill\Box

5.4 Minimal Clones

A trivial clone is a clone all of whose operations are projections.

Definition 5.6. A clone $C$ is minimal if it is non-trivial, and for every non-trivial $C \subseteq D$ we have $C = D$.

Definition 5.7. A function $f \in O$ is minimal if $f$ is of minimal arity such every $g$ generated by $f$ is either a projection or generates $f$.

The following is straightforward from the definitions.

Proposition 5.8. Every minimal $f$ generates a minimal clone, and every minimal clone contains a minimal operation.

Lemma 5.9. Let $\mathcal{B}$ be a relational structure and let $R$ be a $k$-ary relation that intersects $m$ orbits of $k$-tuples of $\text{Aut}(\mathcal{B})$. If $\mathcal{B}$ has a polymorphism $f$ that violates $R$, then $\mathcal{B}$ also has an at most $m$-ary polymorphism that violates $R$.

Proof. Let $f'$ be an polymorphism of $\mathcal{B}$ of smallest arity $l$ that violates $R$. Then there are $k$-tuples $t_1, \ldots, t_l \in R$ such that $f'(t_1, \ldots, t_l) \notin R$. For $l > m$ there are two tuples $t_i$ and
different orbits. The assertion now follows from Lemma 5.9. 

For essentially unary clones \( \mathcal{B} \), we can bound the arity of minimal functions above \( \mathcal{B} \).

**Proposition 5.10.** Let \( \mathcal{B} \) be an arbitrary structure with \( r \) orbitals. Then every minimal clone above \( \text{End}(\mathcal{B}) \) is generated by a function of arity at most \( 2r - 1 \).

**Proof.** Let \( \mathcal{C} \) be a minimal clone above \( \text{End}(\mathcal{B}) \). If all the functions in \( \mathcal{C} \) are essentially unary, then \( \mathcal{C} \) is generated by a unary operation together with \( \text{End}(\mathcal{B}) \) and we are done. Otherwise, let \( f \) be an essential operation in \( \mathcal{C} \). By Lemma 5.5 the operation \( f \) violates \( P_3 \) over the domain \( B \) of \( \mathcal{B} \); recall that \( P_3 \) is defined by the formula \( (x = y) \lor (y = z) \). The subset of \( P_3 \) that contains all tuples of the form \((a,a,b)\), for \( a,b \in B \), clearly consists of \( r \) orbit in \( \mathcal{B} \). Similarly, the subset of \( P_3 \) that contains all tuples of the form \((a,b,c)\), for \( a,b \in B \), consists of the same number of orbits. The intersection of these two relations consists of exactly one orbit (namely, the triples with three equal entries), and therefore \( P_3 \) is the union of \( 2r - 1 \) different orbits. The assertion now follows from Lemma 5.9. \( \square \)

In the remainder of this section, we show that a minimal operation has one out of the following five types, due to Rosenberg [39]. A \( k \)-ary operation \( f \) is a semiprojection iff there exists an \( i \leq k \) such that \( f(x_1, \ldots, x_k) = x_i \) whenever \( |\{x_1, \ldots, x_k\}| < k \).

**Lemma 5.11** (´Swierczkowski; Satz 4.4.6 in [36]). Every at least \( 4 \)-ary operation on a finite domain that turns into a projection whenever two arguments are the same, is a semiprojection.

**Proof.** Let \( f \) be an operation on the finite set \( B \) of arity at least \( n \geq 4 \) as in the statement of the lemma. By assumption there exists an \( i \in \{1, \ldots, n\} \) such that

\[
(x_1, x_1, x_3, \ldots, x_n) = x_i
\]

Let \( k, l \in \{1, \ldots, n\} \) be distinct. We assume that \( k \neq i \); this is without loss of generality since otherwise we can change the roles of \( k \) and \( l \). For notational simplicity, we assume in the following that \( k < l \); the case that \( l < k \) is analogous.

We claim that \( f(x_1, \ldots, x_{k-1}, x_l, x_{k+1}, \ldots, n) = x_i \). If \( \{k, l\} = \{1, 2\} \) then the claim is true. If \( \{k, l\} \cap \{1, 2\} = \{k\} \), consider the function defined by \( f(x_1, x_1, x_3, \ldots, x_{l-1}, x_1, x_{l+1}, \ldots, x_n) \). Then (1) implies that this function equals \( x_i \). Since \( f(x_1, x_2, x_3, \ldots, x_{l-1}, x_1, x_{l+1}, \ldots, x_n) \) by assumption is a projection, too, it must be the \( i \)-th projection, which proves the claim. Finally, if \( \{k, l\} \cap \{1, 2\} \), consider the operation defined by

\[
f(x_1, x_1, x_3, \ldots, x_{k-1}, x_l, x_{k+1}, \ldots, x_{l-1}, x_1, x_{l+1}, \ldots, x_n)
\]

Again, (1) implies that this function equals \( x_i \). By assumption the operation \( f(x_1, x_2, x_3, \ldots, x_{k-1}, x_1, x_{k+1}, \ldots, x_{l-1}, x_1, x_{l+1}, \ldots, x_n) \) is a projection, too, so it must be the \( i \)-th projection, which concludes the proof. \( \square \)

**Theorem 5.12.** Let \( f \) be a minimal operation. Then \( f \) has one of the following types:

\[
t_j \text{ that lie in the same orbit of } k \text{-tuples, and therefore } \mathcal{B} \text{ has an automorphism } \alpha \text{ such that } \alpha t_j = t_i. \text{ By permuting the arguments of } f', \text{ we can assume that } i = 1 \text{ and } j = 2. \text{ Then the } (l - 1) \text{-ary operation } g \text{ defined as }
\]

\[
g(x_2, \ldots, x_l) := f'(\alpha x_2, x_2, \ldots, x_l)
\]

is also a polymorphism of \( \mathcal{B} \), and also violates \( R \), a contradiction. Hence, \( l \leq m \). \( \square \)
1. A unary operation with the property that \( f(f(x)) = f(x) \) or \( f(f(x)) = x \).
2. A binary idempotent operation.
3. A Maltsev operation.
4. A majority operation.
5. A \( k \)-ary semiprojection, for \( 3 \leq k \).

**Proof.** There is nothing to show when \( f \) is unary or 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 \( g(x,y,z) := f(y,x,y) \) 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.

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.

5.5 Schaefer’s Theorem

Schaefer’s theorem states that every CSP for a 2-element template is either in P or NP-hard. Via the Inv-Pol Galois connection (Section 5.2), most of the classification arguments in Schaefer’s article follow from earlier work of Post [37], 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 [37]). Every minimal function on a two element set is among one of the following:

- 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. Let \( f \) be a minimal at least binary function above \( \mathcal{C} \). Note that \( \hat{f} \) defines a function in \( \mathcal{C} \). Hence, either \( \hat{f} \) is the identity in which case \( f \) is idempotent, or \( f \) equals \( \neg \) in which case \( \neg f \) is idempotent and minimal above \( \mathcal{C} \) as well. So we can assume without loss of generality that \( f \) 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, consider the function \( g \) defined by \( g(x, y, z) = f(x, f(x, y, z), z) \). We have \( g(x, y, z) = f(x, f(x, x, z), z) = f(x, z, z) = x \), \( g(x, x, y) = f(x, f(x, y, y), y) = f(x, x, y) = y \), and \( g(x, y, x) = f(x, f(x, x, y), x) = f(x, x, x) = x \). Thus, \( g \) is the Boolean majority operation. \( \square \)

Theorem 5.14. A Boolean relation \( R \) is preserved by the minority operation if and only if it has a definition by a conjunction of linear equalities modulo 2.

Proof. The proof of the interesting direction is by induction on the arity \( k \) of \( R \). The statement is clear when \( R \) is unary. Otherwise, let \( R_0 \) be the boolean relation of arity \( k - 1 \) defined by \( R_0(x_2, \ldots, x_k) \iff R(0, x_2, \ldots, x_k) \), and let \( R_1 \subseteq \{0, 1\}^{k-1} \) be defined by \( R_1(x_2, \ldots, x_k) \iff R(1, x_2, \ldots, x_k) \). By inductive assumption, there are conjunctions of linear equalities \( \psi_0 \) and \( \psi_1 \) defining \( R_0 \) and \( R_1 \), respectively. If \( R_0 \) is empty, we may express \( R(x_1, \ldots, x_k) \) by \( x_1 = 1 \land \psi_1 \). The case that \( R_1 \) is empty can be treated analogously. When both \( R_0 \) and \( R_1 \) are non-empty, fix a tuple \((c_2^0, \ldots, c_k^0) \in R_0 \) and a tuple \((c_2^1, \ldots, c_k^1) \in R_1 \). Define \( c^0 \) to be \((0, c_2^0, \ldots, c_k^0) \) and \( c^1 \) to be \((1, c_2^1, \ldots, c_k^1) \). Let \( b \) be an arbitrary tuple from \( \{0, 1\}^k \). Observe that if \( b \in R \), then \( \min(b, c^0, c^1) \in R \), since \( c^0 \in R \) and \( c^1 \in R \). Moreover, if \( \min(b, c^0, c^1) \in R \), then \( \min(\min(b, c^0, c^1), c^0, c^1) = b \in R \). Thus, \( b \in R \) if and only if \( \min(b, c^0, c^1) \in R \). Specialising this to \( b_1 = 1 \), we obtain

\[
(b_2, \ldots, b_k) \in R_1 \iff (\min(b_2, c_2^0, c_2^1), \ldots, \min(b_k, c_k^0, c_k^1)) \in R_0.
\]

This implies

\[
(b_1, \ldots, b_k) \in R \iff (\min(b_2, c_2^0 b_1, c_2^1 b_1), \ldots, \min(b_k, c_k^0 b_1, c_k^1 b_1)) \in R_0.
\]

Thus,

\[
\exists x'_i(\phi_0(x'_2, \ldots, x'_k) \land (x_i + c_i^0 x_1 + c_i^1 x_1 = x'_i))
\]
defines \( R(x_1, \ldots, x_k) \). \( \square \)
The following definition is very useful for proving that certain Boolean relations $R$ can be defined in syntactically restricted propositional logic.

**Definition 5.15.** When $\phi$ is a propositional formula in CNF that defines a Boolean relation $R$, we say that $\phi$ is reduced if when we remove a literal from a clause in $\phi$, the resulting formula is not equivalent to $\phi$.

Clearly, every Boolean relation has a reduced definition: simply remove literals from any definition in CNF until the formula becomes reduced.

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

**Proof.** Let $R$ be a Boolean relation preserved by min. Let $\phi$ be a reduced propositional formula in CNF that defines $\phi$. Now suppose for contradiction that $\phi$ contains a clause $C$ with two positive literals $x$ and $y$. Since $\phi$ is reduced, there is an assignment $s_1$ that satisfies $\phi$ such that $s_1(x) = 1$, and such that all other literals of $C$ evaluate to 0. Similarly, there is a satisfying assignment $s_2$ for $\phi$ such that $s_2(y) = 1$ and all other literals of $C$ evaluate to 0. Then $s_0 : x \mapsto \min(s_1(x), s_2(y))$ does not satisfy $C$, and does not satisfy $\phi$, in contradiction to the assumption that $\min$ preserves $R$.

A binary relation is called *bijunctive* if it can be defined by a propositional formula that is a conjunction of clauses of size two (aka 2CNF formulas).

**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 $\phi$ 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 $\phi$ is reduced, there must be satisfying assignments $s_1, s_2, s_3$ to $\phi$ such that under $s_i$ all literals of $C$ evaluate to 0 except for $l_i$. Then the mapping $s_0 : x \mapsto \text{majority}(s_1(x), s_2(x), s_3(x))$ does not satisfy $C$ and therefore does not satisfy $\phi$, in contradiction to the assumption that majority preserves $R$.

Recall the definition of NAE.

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

**Theorem 5.18** (Schaefer [40]). Let $\mathcal{B}$ be a structure over a two-element universe. Then either ($\{0,1\}; \text{NAE}$) has a primitive positive definition in $\mathcal{B}$, and CSP($\mathcal{B}$) is NP-complete, or

1. $\mathcal{B}$ is preserved by a constant operation.
2. $\mathcal{B}$ is preserved by min. In this case, every relation of $\mathcal{B}$ has a definition by a propositional Horn formula.
3. $\mathcal{B}$ is preserved by max. In this case, every relation of $\mathcal{B}$ 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. $\mathcal{B}$ is preserved by the majority operation. In this case, every relation of $\mathcal{B}$ 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. Let $\mathcal{D}$ be the polymorphism clone of $\mathfrak{B}$. If $\mathcal{D}$ contains a constant operation, then we are in case one; so suppose in the following that $\mathcal{D}$ does not contain constant operations. Let $\mathcal{C}$ be the clone generated by the unary operations in $\mathfrak{B}$; note that $\mathcal{C}$ is either the clone of projections, or the function clone generated by $\neg: x \mapsto 1 - x$. We only give the proof for the first case; the proof for the second case being similar.

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 [19]. Otherwise, by Theorem 5.2 there is an operation $f$ in $\mathcal{D}$ that violates NAE. Since the unary operations in $\mathcal{D}$ preserve NAE, there is an at least binary minimal operation $g$ in $\mathcal{D}$. 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 [41]. The case that $g = \text{max}$ is dual to this case.

- $g = \text{majority}$. 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 [1].

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

This concludes the proof of the statement. \qed

6 Universal Algebra

6.1 Algebras

In universal algebra, an algebra is simply a structure with a purely functional signature.

The clone of an algebra. Algebras give rise to clones in the following way. When $\mathbf{A}$ is an algebra with signature $\tau$ (also called $\tau$-algebra) and domain $A$, we denote by $\text{Clo}(\mathbf{A})$ the set of all term functions of $\mathbf{A}$, that is, functions with domain $A$ of the form $(x_1, \ldots, x_n) \mapsto t(x_1, \ldots, x_n)$ where $t$ is any term over the signature $\tau$ whose set of variables is contained in $\{x_1, \ldots, x_n\}$. Clearly, $\text{Clo}(\mathbf{A})$ is a function 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 6.1. Let $\mathfrak{B}$ be a structure with domain $B$. An algebra $\mathbf{B}$ with domain $B$ such that $\text{Clo}(\mathbf{B}) = \text{Pol}(\mathfrak{B})$ is called a polymorphism algebra of $\mathfrak{B}$. 

35
Note that the signature of a polymorphism algebra is always infinite, since we have polymorphisms of arbitrary finite arity. Also note that a structure \( \mathcal{B} \) has many different polymorphism algebras, since Definition 6.1 does not prescribe how to assign function symbols to the polymorphisms of \( \mathcal{B} \).

### 6.2 Subalgebras, Products, Homomorphic Images

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

**Subalgebras.** Let \( A \) be a \( \tau \)-algebra with domain \( A \). A subalgebra of \( A \) is a \( \tau \)-algebra \( B \) with domain \( B \subseteq A \) such that for each \( f \in \tau \) of arity \( k \) we have \( f^B(b_1, \ldots, b_k) = f^A(b_1, \ldots, b_k) \) for all \( b_1, \ldots, b_k \in B \). When a polymorphism algebra of a structure \( \mathcal{B} \) has a certain subalgebra, what does it tell us about \( \text{CSP}(\mathcal{B}) \)?

**Lemma 6.2.** Let \( \mathcal{B} \) be a structure, \( \mathcal{B} \) a polymorphism algebra of \( \mathcal{B} \), and \( A \) a subalgebra of \( \mathcal{B} \). Suppose that \( \mathfrak{A} \) is a structure such that \( \mathfrak{A} \) is a polymorphism algebra of \( \mathfrak{A} \). Then \( \text{CSP}(\mathfrak{A}) \) reduces to \( \text{CSP}(\mathcal{B}) \).

**Proof.** Let \( \exists x_1, \ldots, x_n. \phi \) be an instance of \( \text{CSP}(\mathfrak{A}) \) where \( \phi \) is quantifier free. Since \( A \) is preserved by all polymorphisms of \( \mathcal{B} \), it has a primitive positive definition \( \psi_A \) over \( \mathcal{B} \). Every relation \( R^A \), viewed as a relation over the larger domain \( B \), is preserved by all polymorphisms of \( \mathcal{B} \), too, and hence has a primitive positive definition \( \psi_R \) over \( \mathcal{B} \). Let \( \phi' \) be the formula obtained from \( \phi \) by replacing atomic formulas of the form \( R(y_1, \ldots, y_k) \) by \( \psi_R(y_1, \ldots, y_k) \). Then \( \mathfrak{A} \models \exists x_1, \ldots, x_n. \phi \) if and only if \( \mathfrak{B} \models \exists x_1, \ldots, x_n(\psi_A(x_1) \land \cdots \land \psi_A(x_n) \land \phi') \). It is straightforward to efficiently compute the latter formula and to bring it into prenex normal form, and therefore we found our polynomial time reduction.

**Example 4.** Suppose we want to deduce the NP-hardness of the precolored \( K_5 \)-coloring problem from the NP-completeness of \( K_3 \)-coloring (which is well-known). The precolored \( K_5 \)-coloring can be modeled as \( \text{CSP}(\mathfrak{A}) \) for \( \mathfrak{A} := \{\{1, 2, 3, 4, 5\}; \neq, C_1, \ldots, C_5\} \) where \( C_i := \{i\} \). Let \( A \) be a polymorphism algebra of \( \mathfrak{A} \); its operations are precisely the idempotent polymorphisms of \( K_5 \). We claim that \( \{1, 2, 3\} \) induces a subalgebra of \( A \). And indeed, the set \( \{1, 2, 3\} \) has the primitive positive definition \( \exists y_4, y_5 (x \neq y_4 \land C_4(y_4) \land x \neq y_5 \land C_5(y_5)) \) over \( \mathfrak{A} \).

It follows via Proposition 1.6 that the \( K_5 \)-coloring problem is also NP-complete.

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

**Lemma 6.3.** Let \( A \) be the polymorphism algebra of a finite structure \( \mathfrak{A} \). Then the (domains of the) subalgebras of \( A^k \) are precisely the relations that have a primitive positive definition in \( \mathfrak{A} \).
Proof. A relation $R \subseteq A^k$ is a subalgebra of $A^k$ if and only if for all $m$-ary $f$ in the signature of $A$ and $t^1, \ldots, t^m \in R$, we have $(f(t^1_1, \ldots, t^m_1), \ldots, f(t^1_k, \ldots, t^m_k)) \in R$, which is the case if and only if $R$ is preserved by all polymorphisms of $A$, which is the case if and only if $R$ is primitive positive definable in $A$ by Theorem 5.2. □

Also algebra products behave well for the complexity of the CSP.

Lemma 6.4. Let $\mathfrak{B}$ be a structure, $\mathfrak{B}$ a polymorphism algebra of $\mathfrak{B}$, and $\mathfrak{A}$ a structure with domain $B^n$ such that all polymorphisms of $\mathfrak{A}$ are operations of $B^n$. Then $\text{CSP}(\mathfrak{A})$ reduces to $\text{CSP}(\mathfrak{B})$.

Proof. Let $\exists x_1, \ldots, x_n, \phi$ be an instance of $\text{CSP}(\mathfrak{A})$ where $\phi$ is quantifier-free. Let $R$ be a relation of $\mathfrak{A}$ (including the binary equality relation). Then $R$ is a subalgebra of $\mathfrak{B}^n$, and hence primitive positive definable in $\mathfrak{B}$. Therefore, when we replace every atomic formula in $\phi$ by the corresponding primitive positive formula over $\mathfrak{B}$, we obtain a sentence $\Psi$ which is true in $\mathfrak{B}$ if and only if $\exists x_1, \ldots, x_n, \phi$ is true in $\mathfrak{A}$. By transforming $\Psi$ into prenex normal form, we obtain an instance of $\text{CSP}(\mathfrak{B})$ that has been computed from $\phi$ in polynomial time; we therefore found a polynomial time reduction from $\text{CSP}(\mathfrak{A})$ to $\text{CSP}(\mathfrak{B})$. □

The following problem, however, is open.

Question 3. Let $\mathfrak{B}_1$ and $\mathfrak{B}_2$ be structures whose $\text{CSP}$ is in $P$ such that they have polymorphism algebras $\mathfrak{B}_1$ and $\mathfrak{B}_2$ with the same signature. Let $\mathfrak{A}$ be a structure with domain $B_1 \times B_2$ such that all polymorphisms of $\mathfrak{A}$ are operations of $B_1 \times B_2$. Is $\text{CSP}(\mathfrak{A})$ in $P$, too?

Homomorphic Images. Let $A$ and $B$ be $\tau$-algebras. Then a homomorphism from $A$ to $B$ is a mapping $h: A \to B$ such that for all $k$-ary $f \in \tau$ and $a_1, \ldots, a_k \in A$ we have

$$h(f^A(a_1, \ldots, a_k)) = f^B(h(a_1), \ldots, h(a_k)).$$

When $g: C \to D$ is a map, then the kernel of $h$ 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 $C$.

Definition 6.5. A congruence of an algebra $A$ is an equivalence relation that is preserved by all operations in $A$.

Lemma 6.6. Let $\mathfrak{B}$ be a finite structure, and $\mathfrak{B}$ a polymorphism algebra of $\mathfrak{B}$. Then the congruences of $\mathfrak{B}$ are exactly the primitive positive definable equivalence relations over $\mathfrak{B}$.

Proof. A direct consequence of theorem 5.2. □

Proposition 6.7 (see [12]). Let $A$ be an algebra. Then $E$ is a congruence of $A$ if and only if $E$ is the kernel of a homomorphism from $A$ to some other algebra $B$.

Example 5. Let $G = (V, E)$ be the undirected graph with $V = \{a_1, \ldots, a_4, b_1, \ldots, b_4\}$ such that $a_1, \ldots, a_4$ and $b_1, \ldots, b_4$ induce a clique, for each $i \in \{1, \ldots, 4\}$ there is an edge between $a_i$ and $b_i$, and otherwise there are no edges in $G$. Let $A$ be a polymorphism algebra of $G$. Then $A$ homomorphically maps to a two-element algebra $B$. Why?
By Proposition 6.7, it suffices to show that $A$ has a congruence with two equivalence classes. By Lemma 6.6, it suffices to show that an equivalence relation of index two is primitive positive definable. Here is the primitive positive definition:

$$\exists u, v \left( E(x, u) \land E(y, u) \land E(x, v) \land E(y, v) \land E(u, v) \right)$$

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

**Example 6.** Let $A$ be the algebra with domain $A := S_3 = \{ \text{id}, (231), (312), (12), (23), (13) \}$ (the symmetric group on three elements), and a single binary operation, the composition function of permutations. Note that $A$ has the subalgebra induced by $\{ \text{id}, (231), (312) \}$. Also note that $A$ homomorphically maps to $\{(0,1), +\}$ where $+$ is addition modulo 2: the preimage of 0 is $\{\text{id}, (231), (312)\}$ and the preimage of 1 is $\{(12), (23), (13)\}$.

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

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

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 $A$. Note that $h$ is a surjective homomorphism from $A$ to $A / h$.

The following is well known (see e.g. Theorem 6.3 in [12]).

**Lemma 6.8.** Let $A$ and $B$ be algebras with the same signature, and let $h : A \to B$ be a homomorphism. Then the image of any subalgebra $A'$ of $A$ under $h$ is a subalgebra of $B$, and the preimage of any subalgebra $B'$ of $B$ under $h$ is a subalgebra of $A$.

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

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

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

When a polymorphism algebra of a structure $\mathfrak{B}$ has a certain homomorphic image, what does it tell us about CSP($\mathfrak{B}$)?

**Lemma 6.9.** Let $\mathfrak{B}$ be a structure, $\mathfrak{B}$ a polymorphism algebra of $\mathfrak{B}$, and $A$ a homomorphic image of $B$. Suppose that $\mathfrak{A}$ is a structure such that $A$ is a polymorphism algebra of $\mathfrak{A}$. Then CSP($\mathfrak{A}$) reduces to CSP($\mathfrak{B}$).

**Proof.** Let $\exists x_1, \ldots, x_n. \phi$ be an instance of CSP($\mathfrak{A}$) where $\phi$ is quantifier free. Let $E$ be the kernel of the homomorphism from $A$ to $B$. Since $E$ is a congruence, it has a primitive positive definition $\psi$ by Lemma 6.6. If $R$ is a $k$-ary relation of $\mathfrak{A}$, then all operations of $A$ preserve $R$, and hence the relation $\{(a_1, \ldots, a_k) \mid (b_1, \ldots, b_k) \in R, (a_i, b_i) \in E\}$ is preserved by all operations of $B$, and therefore has a primitive positive definition $\psi_R$ in $\mathfrak{B}$. Now replace each conjunct of the form $R(x_1, \ldots, x_k)$ in $\phi$ by the formula

$$\exists x'_1, \ldots, x'_k \left( \psi(x_1, x'_1) \land \cdots \land \psi(x_k, x'_k) \land \psi_R(x'_1, \ldots, x'_k) \right)$$

It is straightforward to efficiently perform this replacement and to bring the resulting sentence into prenex normal form. Then $\mathfrak{B}$ satisfies the sentence if and only if $\mathfrak{A}$ satisfies $\phi$. We therefore found our polynomial time reduction.

38
Varieties and Pseudovarieties. When $\mathcal{K}$ is a class of algebras of the same signature, then

- $P(\mathcal{K})$ denotes the class of all products of algebras from $\mathcal{K}$.
- $P^{\text{fin}}(\mathcal{K})$ denotes the class of all finite products of algebras from $\mathcal{K}$.
- $S(\mathcal{K})$ denotes the class of all subalgebras of algebras from $\mathcal{K}$.
- $H(\mathcal{K})$ 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 $P$, $P^{\text{fin}}$, $S$ and $H$ we often omit the brackets when applying them to single algebras, i.e., we write $H(A)$ instead of $H(\{A\})$. The elements of $HS(A)$ are also called the factors of $A$.

A class $\mathcal{V}$ of algebras with the same signature $\tau$ is called a pseudo-variety if $\mathcal{V}$ contains all homomorphic images, subalgebras, and direct products of algebras in $\mathcal{V}$, i.e., $H(\mathcal{V}) = S(\mathcal{V}) = P^{\text{fin}}(\mathcal{V})$. 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 pseudo-varieties and varieties is that pseudo-varieties need not be closed under direct products of infinite cardinality. The smallest pseudo-variety (variety) that contains an algebra $A$ is called the pseudo-variety (variety) generated by $A$.

Lemma 6.10 (HSP lemma). Let $A$ be an algebra.

- The pseudo-variety generated by $A$ equals $HSP^{\text{fin}}(A)$.
- The variety generated by $A$ equals $HSP(A)$.

Proof. Clearly, $HSP^{\text{fin}}(A)$ is contained in the pseudo-variety generated by $A$, and $HSP(A)$ is contained in the variety generated by $A$. For the converse inclusion, it suffices to verify that $HSP^{\text{fin}}(A)$ is closed under $H$, $S$, and $P^{\text{fin}}$. It is clear that $H(HSP^{\text{fin}}(A)) = HSP^{\text{fin}}(A)$. The second part of Lemma 6.8 implies that $S(HSP^{\text{fin}}(A)) \subseteq HS(SP^{\text{fin}}(C)) = HSP^{\text{fin}}(A)$. Finally,

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

The proof that $HSP(A)$ is closed under $H$, $S$, and $P$ is analogous.

Exercises.

45. Show that the operators $HS$ and $SH$ are distinct.

46. Show that the operators $SP$ and $PS$ are distinct.

6.3 The Tractability Conjecture

Lemma 6.2, Lemma 6.4, and Lemma 6.9 can be combined into a single statement.

Corollary 6.11. Let $\mathfrak{A}$ and $\mathfrak{B}$ be finite relational structures, and let $B$ be a polymorphism algebra for $\mathfrak{B}$. If there exists an algebra $A$ in $HSP^{\text{fin}}(B)$ such that $\text{Clo}(A) \subseteq \text{Pol}(\mathfrak{A})$, then there is a polynomial-time reduction from $\text{CSP}(\mathfrak{A})$ to $\text{CSP}(\mathfrak{B})$.

It follows that for finite structures $\mathfrak{B}$, the complexity of $\text{CSP}(\mathfrak{B})$ only depends on the pseudo-variety generated by a polymorphism algebra $B$ of $\mathfrak{B}$. 39
Corollary 6.12. Let $\mathfrak{B}$ be a finite relational structure, and let $\mathfrak{B}$ be a polymorphism algebra for $\mathfrak{B}$. If there exists an algebra $\mathfrak{A}$ in $\text{HSP}_{\text{fin}}(\mathfrak{B})$ such that $\text{Clo}(\mathfrak{A}) \subseteq \text{Pol}([0, 1, 2]; \neq)$, then $\text{CSP}(\mathfrak{B})$ is NP-hard.

Corollary 6.13. Let $\mathfrak{B}$ be a finite relational structure, and let $\mathfrak{B}$ be a polymorphism algebra for $\mathfrak{B}$. If there exists an algebra $\mathfrak{A}$ in $\text{HSP}_{\text{fin}}(\mathfrak{B})$ all of whose operations are projections on a set with at least two elements, then $\text{CSP}(\mathfrak{B})$ is NP-hard.

The following has been conjectured by Bulatov, Jeavons, and Krokhin in [11], and is known under the name tractability conjecture.

Conjecture 2 (Tractability Conjecture). Let $\mathfrak{B}$ be a finite relational structure with an idempotent polymorphism algebra $\mathfrak{B}$. If there is no algebra $\mathfrak{A}$ in $\text{HSP}_{\text{fin}}(\mathfrak{B})$ such that $\text{Clo}(\mathfrak{A}) \subseteq \text{Pol}([0, 1, 2]; \neq)$, then $\text{CSP}(\mathfrak{B})$ is in $P$.

Note that the assumption that the polymorphism algebra of $\mathfrak{B}$ is idempotent is without loss of generality: recall that we can assume without loss of generality that $\mathfrak{B}$ is a core, and that adding the relations of the form $\{a\}$ for elements $a$ of the domain does not change the computational complexity 4.7. The polymorphism algebras of the expanded structure will clearly be idempotent.

6.4 Birkhoff’s Theorem

Varieties (which have been defined in Section 6.2) are a fascinatingly powerful concept to study classes of algebras. The central theorem for the study of varieties is Birkhoff’s HSP theorem, which links varieties with universal conjunctive theories. By Birkhoff’s theorem, there is also a close relationship between varieties and the concept of an abstract clone, as we will see in Section 6.5. We present it here only for varieties generated by a single finite algebra.

Theorem 6.14 (Birkhoff [7]; see e.g. [26] or [12]). Let $\tau$ be a functional signature, and $\mathfrak{A}$ and $\mathfrak{B}$ finite algebras with signature $\tau$. Then the following are equivalent.

1. All universal conjunctive $\tau$-sentences that hold in $\mathfrak{B}$ also hold in $\mathfrak{A}$;

2. $\mathfrak{A} \in \text{HSP}_{\text{fin}}(\mathfrak{B})$;

3. $\mathfrak{A} \in \text{HSP}(\mathfrak{B})$;

Before we show the theorem, we show a simple lemma that is used in the proof. Let $\mathfrak{A}$ and $\mathfrak{B}$ be finite $\tau$-algebras. Then the function $\xi$ from $\text{Clo}(\mathfrak{B})$ to $\text{Clo}(\mathfrak{A})$ that sends for every $\tau$-term $t$ the function $t^\mathfrak{B}$ to the function $t^\mathfrak{A}$, is well-defined surjective clone homomorphism if and only for all $\tau$-terms $s, t$ we have $s^\mathfrak{A} = t^\mathfrak{A}$ whenever $s^\mathfrak{B} = t^\mathfrak{B}$; in this case, we call $\xi$ the natural homomorphism from $\text{Clo}(\mathfrak{B})$ onto $\text{Clo}(\mathfrak{A})$.

Lemma 6.15. Let $\mathfrak{A}$ and $\mathfrak{B}$ be finite $\tau$-algebras, and suppose that the natural homomorphism $\xi$ from $\text{Clo}(\mathfrak{B})$ onto $\text{Clo}(\mathfrak{A})$ exists. Then for all $k \in \mathbb{N}$ there exists an $m \geq 1$ and $C \in B^{k \times m}$ such that for all $k$-ary $f, g \in \text{Clo}(\mathfrak{B})$ we have that $f(C) = g(C)$ implies $\xi(f) = \xi(g)$.

Proof. Take $m := |B|^k$, and let $C$ be $B^k$. Then $f(C) = g(C)$ implies that $f = g$, and hence $\xi(f) = \xi(g)$. $\square$
Proof of Birkhoff’s theorem. Trivially, 2. implies 3. To show that 3. implies 1., let
\[ \phi := \forall x_1, \ldots, x_n (\phi_1 \land \cdots \land \phi_k) \]
be a conjunctive \(\tau\)-sentence that holds in \(B\). Then \(\phi\) is preserved in powers of \(B\). To see this, let \((b_1^1, \ldots, b_m^1), \ldots, (b_1^n, \ldots, b_m^n) \in B^m\) be arbitrary. Then
\[
\begin{align*}
B \models \phi & \iff B \models \forall x_1, \ldots, x_n. \phi_1 \land \cdots \land \forall x_1, \ldots, x_n. \phi_k \\
& \iff B \models \phi_i(b_1^j, \ldots, b_n^j) \text{ for all } j \leq m, i \leq n \\
& \iff B^m \models \phi_i((b_1^1, \ldots, b_m^1), \ldots, (b_1^n, \ldots, b_m^n)) \text{ for all } i \leq n.
\end{align*}
\]
Since \((b_1^1, \ldots, b_m^1), \ldots, (b_1^n, \ldots, b_m^n)\) were chosen arbitrarily, we have that
\[
B^m \models \forall x_1, \ldots, x_n (\phi_1 \land \cdots \land \phi_k).
\]
Moreover, \(\phi\) is true in subalgebras of algebras that satisfy \(\phi\) (this is true for universal sentences in general). Finally, suppose that \(S\) is an algebra that satisfies \(\phi\), and \(\mu\) is a surjective homomorphism from \(S\) to some algebra \(A\). Let \(a_1, \ldots, a_n \in A\) be arbitrary; by surjectivity of \(\mu\) we can choose \(b_1, \ldots, b_n\) such that \(\mu(b_i) = a_i\) for all \(i \leq n\). Suppose that \(\phi_i\) is of the form \(s(x_1, \ldots, x_n) = t(x_1, \ldots, x_n)\) for \(\tau\)-terms \(s, t\). Then
\[
\begin{align*}
\mu(s^B(b_1, \ldots, b_n)) = \mu(t^B(b_1, \ldots, b_n)) & \iff t^A(\mu(b_1), \ldots, \mu(b_n)) = s^A(\mu(b_1), \ldots, \mu(b_n)) \\
& \iff t^A(a_1, \ldots, a_n) = s^A(a_1, \ldots, a_n).
\end{align*}
\]
1. implies 2.: Since all universal conjunctive \(\tau\)-sentences that hold in \(B\) also hold in \(A\), we have that \(s^A = t^A\) whenever \(s^B = t^B\); hence, the natural homomorphism \(\xi\) from \(\text{Clo}(B)\) onto \(\text{Clo}(A)\) exists.

Let \(a_1, \ldots, a_k\) be the elements of \(A\), and let \(m\) and \(C\) be as in Lemma 6.15. Let \(S\) be the smallest subalgebra of \(B^m\) that contains the columns \(c_1, \ldots, c_k\) of \(C\); so the elements of \(S\) are precisely those of the form \(t^A(m(c_1, \ldots, c_k))\), for a \(k\)-ary \(\tau\)-term \(t\). Define \(\mu : S \to B\) by setting \(\mu(t^B(m(c_1, \ldots, c_k))) := t^A(a_1, \ldots, a_k)\). Then \(\mu\) is well-defined, for if \(t^B(c_1, \ldots, c_k) = s^B(c_1, \ldots, c_k)\), then \(t^A(a_1, \ldots, a_k) = s^A(a_1, \ldots, b_k)\) by the property of \(C\) from Lemma 6.15. Since for all \(i \leq k\) the element \(c_i\) is mapped to \(b_i\), the map \(\mu\) is surjective. We claim that \(\mu\) is moreover a homomorphism from \(S\) to \(A\); it then follows that \(A\) is the homomorphic image of the subalgebra \(S\) of \(B^m\), and so \(A \in \text{HSP}^{\text{fin}}(B)\).

To this end, let \(f \in \tau\) be arbitrary, and \(n\) the arity of \(f\), and let \(s_1, \ldots, s_m \in S\). For \(i \leq n\), write \(s_i = t^S_i(c_1, \ldots, c_k)\) for some \(\tau\)-term \(t\). Then
\[
\begin{align*}
\mu(f^S(s_1, \ldots, s_n)) &= \mu(f^S(t^S_1(c_1, \ldots, c_k), \ldots, t^S_m(c_1, \ldots, c_k))) \\
&= \mu(f^S(t^S_1, \ldots, t^S_m)(c_1, \ldots, c_k)) \\
&= \mu((f(t_1, \ldots, t_n))^S(c_1, \ldots, c_k)) \\
&= (f(t_1, \ldots, t_n))^A(a_1, \ldots, a_k) \\
&= f^A(t^A_1(a_1, \ldots, a_k), \ldots, t^A_m(a_1, \ldots, a_k)) \\
&= f^A(\mu(s_1), \ldots, \mu(s_n)).
\end{align*}
\]
\(\square\)
Theorem 6.14 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 the following two sections.

### 6.5 Clones

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

**Definition 6.16.** A clone \( \mathbf{C} \) is a multi-sorted structure with sorts \( \{ C^{(i)} \mid i \in \omega \} \) and the signature \( \{ p_i^k \mid 1 \leq i \leq k \} \cup \{ \text{comp}_k \mid k, l \geq 1 \} \). The elements of the sort \( C^{(k)} \) will be called the \( k \)-ary operations of \( \mathbf{C} \). We denote a clone by

\[
\mathbf{C} = (C^{(0)}, C^{(1)}, \ldots; (p_i^k)_{1 \leq i \leq k}, (\text{comp}_k)_{k,l \geq 1})
\]

and require that \( p_i^k \) is a constant in \( C^{(k)} \), and that \( \text{comp}_k : C^{(k)} \times (C^{(l)})^k \rightarrow C^{(l)} \) is an operation of arity \( k + 1 \). Moreover, it holds that

\[
\text{comp}_k^k(f, p_1^k, \ldots, p_k^k) = f \quad (2)
\]

\[
\text{comp}_k(p_i^k, f_1, \ldots, f_k) = f_i \quad (3)
\]

\[
\text{comp}_l^k(f, \text{comp}_l^m(g_1, h_1, \ldots, h_m), \ldots, \text{comp}_l^m(g_k, h_1, \ldots, h_m)) = \\
\quad \text{comp}_l^m\left(\text{comp}_l^k(f, g_1, \ldots, g_k), h_1, \ldots, h_m\right). \quad (4)
\]

The final equation generalises associativity in groups, and we therefore refer to it by associativity. We also write \( f(g_1, \ldots, g_k) \) instead of \( \text{comp}_l^k(f, g_1, \ldots, g_k) \) when \( l \) is clear from the context.

Every function clone \( \mathcal{C} \) gives rise to an abstract clone \( \mathbf{C} \) in the obvious way: \( p_i^k \in C^{(k)} \) denotes the projection \( \pi_i^k \in \mathcal{C} \), and \( \text{comp}_k(f, g_1, \ldots, g_k) \in C^{(l)} \) denotes the composed function \((x_1, \ldots, x_l) \mapsto f(g_1(x_1, \ldots, x_l), \ldots, g_k(x_1, \ldots, x_l)) \in \mathcal{C} \). 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 a function clone.

**Example 7.** The formula \( \text{comp}_2^2(f, p_1^2, p_2^2) = \text{comp}_2^2(f, p_2^2, p_1^2) \) holds in \( \text{Clo}(\mathbf{A}) \) if and only if \( \forall x_1, x_2, f(x_1, x_2) = f(x_2, x_1) \) holds in \( \mathbf{A} \).

**Proposition 6.17** (Formulation of Birkhoff’s theorem for clones). Let \( \mathcal{C} \) and \( \mathcal{D} \) be function clones on finite sets. Then the following are equivalent.

1. There is a surjective clone homomorphism from \( \mathcal{C} \) to \( \mathcal{D} \);

2. there are algebras \( \mathbf{A} \) and \( \mathbf{B} \) with the same signature \( \tau \) such that \( \text{Clo}(\mathbf{A}) = \mathcal{C} \), \( \text{Clo}(\mathbf{B}) = \mathcal{D} \), and all universal conjunctive \( \tau \)-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 \( \text{Clo}(\mathbf{A}) = \mathcal{C} \), \( \text{Clo}(\mathbf{B}) = \mathcal{D} \), and \( \mathbf{A} \in \text{HSP}^{\text{fin}}(\mathbf{B}) \) (equivalently, \( \mathbf{A} \in \text{HSP}(\mathbf{B}) \)).
In the study of the complexity of CSPs, the equivalence between (1) and (4) in the above is the most relevant, since (4) is related to our most important tool to prove NP-hardness (see Corollary 6.11), and since (1) is the universal-algebraic property that will be used in the following (see e.g. Theorem 6.21 below).

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

**Lemma 6.18.** Let $C$ be a clone and let $D$ be the clone of a finite algebra such that there is no clone homomorphism from $C$ to $D$. Then there is a primitive positive sentence in the language $\tau$ of (abstract) clones that holds in $C$ but not in $D$.

**Proof.** Let $E$ be the expansion of $C$ by constant symbols such that every element $e$ of $E$ is named by a constant $c_e$. Let $U$ be the first-order theory of $D$. We claim that $U$ together with the set of all atomic sentences that holds in $E$ is unsatisfiable. If this theory had a model $M$, consider the restriction of $M$ to $\bigcup_{i\in\mathbb{N}} M^{(i)}$, and consider the $\tau$-reduct of this restriction. Since $M$ satisfies $U$ and each $M^{(i)}$ is finite, the resulting structure must be isomorphic to $D$; we identify it with $D$. All additional constant symbols must denote in $M$ elements from $C$, and since $M$ satisfies all atomic formulas that hold in $E$ we have that the mapping $e \mapsto e^{M}$, for $e$ an element of $E$, is a homomorphism from $C$ to $D$. This is in contradiction to the assumption there is no homomorphism from $C$ to $D$. So by compactness there exists a finite subset $V$ of the atomic formulas that hold in $E$ 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 $D$. \hfill \square

### 6.6 Taylor Terms

The following goes back to Walter Taylor [43].

**Definition 6.19 (Taylor terms).** Let $B$ be a $\tau$-algebra. A Taylor term of $B$ is a $\tau$-term $t(x_1, \ldots, x_n)$, for $n \geq 2$, such that for each $1 \leq i \leq n$ there are $z_1, \ldots, z_n, z'_1, \ldots, z'_n \in \{x, y\}$ with $z_i \neq z'_i$ such that

$$B \models \forall x,y. t(z_1, \ldots, z_n) = t(z'_1, \ldots, z'_n).$$

Operations defined by Taylor terms are called Taylor operations. We do not insist on idempotency for Taylor operations. Note that a term $t(x_1, \ldots, x_n)$ is a Taylor term in $B$ if and only if $\text{Clo}(B) \models \Phi_n(t^B)$, where $\Phi_n(f)$ is the sentence

$$\bigwedge_{i \leq n} \exists u, u' \in \{p_1^2, p_2^2\}^n (u_i \neq u'_i \land \text{comp}_2^n(f, u_1, \ldots, u_n) = \text{comp}_2^n(f, u'_1, \ldots, u'_n)) \quad (5)$$

**Lemma 6.20.** $\Phi_n(f)$ is equivalent to

$$\bigwedge_{i \leq n} \exists v, v' \in \{p_1^n, \ldots, p_n^n\}^n (v_i \neq v'_i \land \text{comp}_n^n(f, v_1, \ldots, v_n) = \text{comp}_n^n(f, v'_1, \ldots, v'_n))$$

**Proof.** Let $i \in \{1, \ldots, n\}$ be arbitrary. Suppose that $u, u' \in \{p_1^2, p_2^2\}^n$ are as in Equation 5. Then define $u, u' \in \{p_1^n, \ldots, p_n^n\}^n$ as follows: if $u_j = p_k^2$ for $k \in \{1, 2\}$, then $v_j := p_k^n$, and similarly if $u'_j = p_k^2$ for $k \in \{1, 2\}$, then $v'_j := p_k^n$. Then $v$ and $v'$ witness that the formula...
given in the statement is true: \( u_i \neq u'_i \) implies that \( v_i \neq v'_i \). Moreover, \( \text{comp}_2^n(f, u_1, \ldots, u_i) = \text{comp}_2^n(f, u'_1, \ldots, u'_i) \) implies that

\[
\text{comp}_2^n(\text{comp}_2^n(f, u_1, \ldots, u_i), p_1^n, p_2^n) = \text{comp}_2^n(\text{comp}_2^n(f, u'_1, \ldots, u'_i), p_1^n, p_2^n)
\]

We compute

\[
\text{comp}_2^n(\text{comp}_2^n(f, u_1, \ldots, u_i), p_1^n, p_2^n) = \text{comp}_2^n(f, \text{comp}_2^n(u_1, p_1^n, p_2^n), \ldots, \text{comp}_2^n(u_n, p_1^n, p_2^n)) = \text{comp}_2^n(f, v_1, \ldots, v_n)
\]

and similarly \( \text{comp}_2^n(\text{comp}_2^n(f, v_1, \ldots, v_n), p_1^n, p_2^n) = \text{comp}_2^n(f, v'_1, \ldots, v'_n) \) and hence

\[
\text{comp}_2^n(f, v_1, \ldots, v_n) = \text{comp}_2^n(f, v'_1, \ldots, v'_n)
\]

as required.

Conversely, suppose that \( v, v' \in \{p_1^n, p_2^n\}^n \) are as in the formula above. Define \( u, u' \in \{p_1^n, p_2^n\}^n \) as follows. If \( v_j = v_i \) define \( u_j := p_1^n \), and \( u_j := p_2^n \) otherwise. Similarly, if \( v'_j = v'_i \) then \( u'_j := p_1^n \), and \( u'_j := p_2^n \) otherwise. For \( j \in \{1, \ldots, n\} \), define \( k_j \) such that \( v_j = p_{k_j}^n \), and set \( w_{k_j} := u_j \). If \( l \in \{1, \ldots, n\} \setminus \{k_1, \ldots, k_n\} \), define \( w_l \in \{p_1^n, p_2^n\} \) arbitrarily. Note that this implies that \( \text{comp}_2^n(v_j, w_{k_1}, \ldots, w_{k_n}) = w_{k_j} = u_j \). Then \( u \) and \( u' \) witness that \( \Phi_n(f) \) is true: since \( v_i \neq v'_i \) we have \( u_i \neq u'_i \), and

\[
\text{comp}_2^n(f, v_1, \ldots, v_n) = \text{comp}_2^n(f, v'_1, \ldots, v'_n)
\]

implies that

\[
\text{comp}_2^n(\text{comp}_2^n(f, v_1, \ldots, v_n), w_1, \ldots, w_n) = \text{comp}_2^n(\text{comp}_2^n(f, v'_1, \ldots, v'_n), u_1, \ldots, u_n)
\]

Now

\[
\text{comp}_2^n(\text{comp}_2^n(f, v_1, \ldots, v_n), u_1, \ldots, u_n) = \text{comp}_2^n(f, \text{comp}_2^n(v_1, u_1, \ldots, u_n), \ldots, \text{comp}_2^n(v_n, u_1, \ldots, u_n)) = \text{comp}_2^n(f, u_1, \ldots, u_n)
\]

Similarly, \( \text{comp}_2^n(\text{comp}_2^n(f, v'_1, \ldots, v'_n), u'_1, \ldots, u'_n) = \text{comp}_2^n(f, u'_1, \ldots, u'_n) \). Hence,

\[
\text{comp}_2^n(f, u_1, \ldots, u_n) = \text{comp}_2^n(f, u'_1, \ldots, u'_n)
\]

as required. \( \square \)

The following is a slightly expanded presentation of the proof of Lemma 9.4 in [25].

**Theorem 6.21.** Let \( B \) be a finite idempotent algebra with signature \( \tau \). Then the following are equivalent.

1. \( \text{HSP}^\text{fin}(B) \) does not contain 2-element algebras all of whose operations are projections;
2. there is no homomorphism from \( \text{Clo}(B) \) to \( 1 \);
3. \( B \) has a Taylor term.
Proof. The equivalence between (1) and (2) is a direct consequence of Proposition 6.17. To show the easy implication from (3) to (2), suppose for contradiction that there were a homomorphism \( \xi \) from \( \text{Clo}(B) \) to 1. Let \( f \) be the element of \( \text{Clo}(B) \) that is denoted by \( t^B \). By definition of 1 we have \( \xi(f) = p^n_l \) for some \( l \leq n \). By assumption, \( B \) satisfies

\[
\forall x, y. t(z_1, \ldots, z_n) = t(z'_1, \ldots, z'_n)
\]

for \( z_1, \ldots, z_n, z'_1, \ldots, z'_n \in \{x, y\} \) such that \( z_i \neq z'_i \). Then \( \text{Clo}(B) \) satisfies

\[
\text{comp}_2^2(f, p^2_1, \ldots, p^2_n) = \text{comp}_2^2(f, p^2_1, \ldots, p^2_n)
\]

for \( i_1, \ldots, i_n, j_1, \ldots, j_n \in \{1,2\} \) such that \( i_i = 1 \) if and only if \( z_i = x \), \( j_i = 1 \) if and only if \( z'_i = x \), and \( i_i \neq j_i \). Since \( \xi(f) = p^n_l \) we therefore obtain that \( p^n_l = p^n_2 \), which does not hold in 1, a contradiction.

To show the implication from (1) to (2), suppose that \( \text{Clo}(B) \) does not homomorphically map to 1. Then Lemma 6.18 implies that there is a primitive positive sentence in the language of clones that holds in \( \text{Clo}(B) \) but not in 1. Note that by introducing new existentially quantified variables we can assume that this sentence is of the form \( \exists u_1, \ldots, u_r. \phi \) where \( \phi \) is a conjunction of atoms of the form \( y = \text{comp}_m^n(x_0, x_1, \ldots, x_m) \) or of the form \( y = p^n_l \) for \( y, x_0, x_1, \ldots, x_m \in \{u_1, \ldots, u_r\} \) and \( l, m \in \mathbb{N} \). For example the equation

\[
\text{comp}_2^2(u, \text{comp}_2^2(u, p^2_2, p^2_1)) = \text{comp}_2^2(\text{comp}_2^2(u, p^2_2, p^2_1), u)
\]

involving the free variable \( u \) is equivalent to

\[
\exists u_1, u_2, u_3 \ (u_1 = \text{comp}_2^2(u, p^2_2, p^2_1) \\
\wedge u_2 = \text{comp}_2^2(f, u, u_1) \\
\wedge u_3 = \text{comp}_2^2(u, u_1, u) \\
\wedge u_2 = u_3)
\]

For \( l, m \in \mathbb{N} \) we write \( x * y \) as a shortcut for

\[
\text{comp}_n^l(x, \text{comp}_n^m(y, p^1_l, \ldots, p^m_l), \ldots, \text{comp}_n^m(y, p^1_{l-1}m+1, \ldots, p^m_l))
\]

where \( n := lm \). Note that

\[
\text{Clo}(B) \models (y = \text{comp}_n^l(x * y, p^1_l, \ldots, p^m_l, p^1_l, \ldots, p^m_l, \ldots, p^m_l)) \tag{8}
\]

and \( \text{Clo}(B) \models (x = \text{comp}_m^n(x * y, p^1_l, \ldots, p^m_l, p^1_l, \ldots, p^m_l, \ldots, p^m_l)) \tag{9} \)

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

\[
u := u_1 * (u_2 * (\ldots (u_{r-1} * u_r) \ldots)) \tag{10}\]

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, (8) reads as \( y = \text{comp}_n^l(x * y, (1, \ldots, l)^m) \), and (9) reads as \( x = \text{comp}_m^n(x * y, (1, \ldots, l)^m) \). Similarly, we have

\[
u := \text{comp}_{k_1 \cdots k_r}^l(u, (1^{k_1 \cdots k_{r-1}}, \ldots, k_i^{k_i \cdots k_{r-1}}, k_{r+1} \cdots k_n)). \tag{11} \]
Let $k := k_1 \cdots k_r$, and let $n = k^2$. Then for every term of the form $\text{comp}^{k_i}_{k} (u_i, v_1, \ldots, v_{k_1})$ with $v_1, \ldots, v_{k_1} \in \{u_1, \ldots, u_r\}$, there exists $\bar{q} \in \{p^{k_1}_{1}, \ldots, p^{k_1}_{1}\}^n$ such that

$$\text{comp}^n (u * u, \bar{q}) = \text{comp}^{k_i}_{k} (u_i, v_1, \ldots, v_{k_1}) .$$

We now present logical consequences of the formula $\phi$ that only involve the expression $u * u$ and the constants $p^n_{1}, \ldots, p^n_{2}$. Suppose that $\phi$ contains the conjunct

$$u_i = \text{comp}^n_{k_i} (u_j, v_1, \ldots, v_{k_j}), \quad v_1, \ldots, v_{k_j} \in \{u_1, \ldots, u_r\} .$$

Then there exist $\bar{q}, \bar{q}' \in \{p^{k_1}_{1}, \ldots, p^{k_1}_{1}\}^n$ such that

$$t_1 := \text{comp}^n (u * u, \bar{q}) = u_i$$

and

$$t_2 := \text{comp}^n (u * u, \bar{q}') = \text{comp}^{k_j}_{k} (u_j, v_1, \ldots, v_{k_j}) .$$

Hence, the expression $t_1 = t_2$ is a logical consequence of $\phi$. Consider the conjunction over all equations that can be obtained in this way from conjuncts of $\phi$ of the form $u_i = \text{comp}^n_{k_i} (u_j, v_1, \ldots, v_{k_j})$, and similarly from conjuncts of the form $y = p^n_{m}$. Let $\psi$ be this conjunction with the additional conjunct

$$\text{comp}^n (u * u, 1^k, \ldots, k^k) = \text{comp}^n (u * u, (1, \ldots, k)^k) . \tag{12}$$

Note that $\phi$ implies $\psi$.

Let $\theta$ be the formula obtained from $\psi$ by replacing each occurrence of $u * u$ by a variable symbol $f$. Note that the sentence $\exists f. \theta$ holds in $\text{Clo}(B)$. By Lemma 6.20, it suffices to show that $\theta$ implies the formula given in the statement of this lemma.

Write $i$ as $i = (i_1 - 1)k + i_2$ with $i_1, i_2 \in \{1, \ldots, k\}$. Suppose first that $i_1 < i_2$. Then $v := (1^k, \ldots, k^k) \in \{p^n_{1}, \ldots, p^n_{1}\}^n$ and $v' := (1, \ldots, k)^k \in \{p^n_{1}, \ldots, p^n_{1}\}^n$ satisfy $v_i = p^n_{i_1} \neq p^n_{i_2} = v'_i$. Because of the conjunct of $\theta$ obtained from the conjunct (12) of $\psi$, we are done in this case. Similarly one can treat the case that $i_1 > i_2$; so suppose in the following that $i_1 = i_2$.

For $l \in \{1, \ldots, r\}$, let $\bar{q} \in \{p^{k_1}_{1}, \ldots, p^{k_1}_{1}\}^n$ be such that $\text{comp}^{k_l}_{k_l} (u, \bar{q}) = u_l$. Define $r_l \in \{1, \ldots, k\}$ to be such that $\bar{q}_{(i_l - 1)k + i_2} = p^{k_l}_{r_l}$. Consider the assignment $\rho$ that maps $u_l$ to $p^{k_l}_{r_l}$, for all $l \in \{1, \ldots, r\}$. Since $\exists u_1, \ldots, u_r. \phi$ does not hold in $1$, there must be a conjunct $u_l = \text{comp}^{k_l}_{k_l} (u_j, v_1, \ldots, v_{k_j})$ of $\phi$ which is false under this assignment, that is,

$$p^{k_l}_{r_l} = \rho(u_l) \neq \rho(\text{comp}^{k_l}_{k_l} (u_j, v_1, \ldots, v_{k_j})) = \rho(v_{r_l}) = p^{k_l}_{r_{r_l}} ,$$

thus $r_l \neq r_{r_l}$. We have already seen that there are $\bar{q}, \bar{q}' \in \{p^n_{1}, \ldots, p^n_{1}\}^n$ such that

$$\text{comp}^n (u * u, \bar{q}) = u_i$$

and

$$\text{comp}^n (u * u, \bar{q}') = \text{comp}^{k_l}_{k_l} (u_j, v_1, \ldots, v_{k_j}) .$$

Note that $\bar{q}_i = \rho(u_i) = p^{k_l}_{r_l} \neq p^{k_l}_{r_{r_l}} = \rho(v_{r_l}) = \bar{q}'_i$. By construction, $\theta$ contains the conjunct $\text{comp}^n (f, \bar{q}) = \text{comp}^n (f, \bar{q}')$. Hence, for every $i \in \{1, \ldots, n\}$ there are $\bar{q}, \bar{q}' \in \{p^n_{1}, \ldots, p^n_{1}\}^n$ with $\bar{q}_i \neq \bar{q}'_i$ such that $\text{comp}^n (f, \bar{q}) = \text{comp}^n (f, \bar{q}')$. Via Lemma 6.20, this shows that $B$ has a Taylor term. □
Figure 4: Overview over the universal-algebraic approach to H-colouring and CSPs.

Lemma 6.22. Let \( \mathcal{B} \) and \( \mathcal{C} \) be homomorphically equivalent structures. Then \( \mathcal{B} \) has a Taylor polymorphism if and only if \( \mathcal{C} \) has a Taylor polymorphism.

Proof. Let \( h \) be a homomorphism from \( \mathcal{B} \) to \( \mathcal{C} \), and \( g \) be a homomorphism from \( \mathcal{C} \) to \( \mathcal{B} \). Suppose that \( f \) is a Taylor polymorphism for \( \mathcal{B} \) of arity \( n \), that is, \( \text{Pol}(\mathcal{B}) = \Phi_n(f) \) for the formula \( \Phi_n \) given in (5) above. It suffices to show that the operation \( f' \) given by \( (x_1, \ldots, x_n) \mapsto h(f(g(x_1), \ldots, g(x_n))) \) is a Taylor operation for \( \mathcal{C} \). Indeed, for all \( i \leq n \) we have that

\[
\begin{align*}
f'(v_{1,i}, \ldots, v_{n,i}) &= h(f(g(v_{1,i}), \ldots, g(v_{n,i}))) \\
&= h(f(g'(v'_{1,i}), \ldots, g'(v'_{n,i}))) \\
&= f'(v'_{1,i}, \ldots, v'_{n,i})
\end{align*}
\]

\( \square \)

Corollary 6.23. Let \( \mathcal{B} \) be a finite structure. Then \( \mathcal{B} \) has a Taylor polymorphism, or \( \text{CSP}(\mathcal{B}) \) is NP-hard.

Proof. Let \( \mathcal{C} \) be the core of \( \mathcal{B} \), and let \( \mathcal{C}' \) be the expansion of \( \mathcal{C} \) by all unary singleton relations. Let \( \mathcal{C} \) be a polymorphism algebra for \( \mathcal{C}' \). If there exists an algebra \( \mathcal{A} \) in \( \text{HSP}^\text{fin}(\mathcal{C}) \) all of whose operations are projections on a set with at least two elements, then \( \text{CSP}(\mathcal{C}') \) is NP-hard by Corollary 6.13. Therefore, \( \text{CSP}(\mathcal{B}) \) is NP-hard by Corollary 4.7.

Otherwise, by Theorem 6.21, \( \mathcal{C} \) has a Taylor term \( t \), and \( t^C \) is a Taylor polymorphism of \( \mathcal{C} \). But then Lemma 6.22 shows that \( \mathcal{B} \) has a Taylor polymorphism, too. \( \square \)
7 Cyclic Terms

In this section, for several results that we present we do not give a proof.

7.1 Digraphs without Sources and Sinks

A source in a digraph is a vertex with no incoming edges, and a sink is a vertex with no outgoing edges. In this section we mention an important result about finite digraphs with no sources and no sinks. Note that undirected graphs \((V,E)\), viewed as directed graphs where for every \(\{u, v\} \in E\) we have \((u, v) \in E\) and \((v, u) \in E\), are examples of such graphs.

**Theorem 7.1** (Barto, Kozik, Nieven [6]). Let \(H\) be a digraph without sources and sinks. If \(H\) has a Taylor polymorphism, then \(H\) is homomorphically equivalent to a disjoint union of cycles.

**Proof.** The proof of this deep theorem is out of the scope of this course. An important part of the proof is using absorption theory, which has been developed for this theorem, and which also plays a role in other recent breakthrough results, such as Theorem 8.1.

If a graph \(H\) is homomorphically equivalent to a disjoint union of cycles, then CSP(\(H\)) is in P (e.g., we can use the algorithm PC\(_H\) to solve it; see Section 3). On the other hand, a digraph without a Taylor polymorphism has an NP-hard CSP. Therefore, Theorem 7.1 shows that the Feder-Vardi conjecture is true for digraphs without sources and sinks: they are either in P or NP-complete.

7.2 Cyclic Terms

An operation \(c : A^n \to A\), for \(n \geq 2\), is cyclic if it satisfies for all \(a_1, \ldots, a_n \in A\) that \(c(a_1, \ldots, a_n) = c(a_2, \ldots, a_n, a_1)\). Cyclic operations are Taylor operations; the converse is a deep result of Barto and Kozic (Theorem 7.4 below). When \(a = (a_0, a_1, \ldots, a_{k-1})\) is a \(k\)-tuple, we write \(\rho(a)\) for the \(k\)-tuple \((a_1, \ldots, a_{k-1}, a_0)\).

**Definition 7.2.** An \(n\)-ary relation \(R\) on a set \(A\) is called cyclic if for all \(a \in A^k\)

\[ a \in R \Rightarrow \rho(a) \in R. \]

**Lemma 7.3** (from [5]). A finite idempotent algebra \(A\) has a \(k\)-ary cyclic term if and only if every nonempty cyclic subalgebra of \(A^k\) contains a constant tuple.

**Proof.** Let \(\tau\) be the signature of \(A\). For the easy direction, suppose that \(A\) has a cyclic \(\tau\)-term \(t(x_1, \ldots, x_k)\). Let \(a = (a_0, a_1, \ldots, a_{k-1})\) be an arbitrary tuple in a cyclic subalgebra \(R\) of \(A^k\). As \(R\) is cyclic, \(\rho(a), \ldots, \rho^{k-1}(a) \in R\), and since \(R\) is a subalgebra, \(b := t^{A^k}(a, \rho(a), \ldots, \rho^{k-1}(a)) \in R\). Since \(t\) is cyclic, the \(k\)-tuple \(b\) is constant.

To prove the converse direction, we assume that every nonempty cyclic subalgebra of \(A^k\) contains a constant tuple. For a \(\tau\)-term \(f(x_0, x_1, \ldots, x_{k-1})\), let \(S(f)\) be the set of all \(a \in A^k\) such that \(f^A(a) = \rho^A(a) = \cdots = \rho^{k-1}(a)\). Choose \(f\) such that \(|S(f)|\) is maximal (here we use the assumption that \(A\) is finite). If \(|S(f)| = |A^k|\), then \(f^A\) is cyclic and we are done. Otherwise, arbitrarily pick \(a = (a_0, b_1, \ldots, a_{k-1}) \in A^k \setminus S(f)\). For \(i \in \{0, \ldots, k-1\}\), define \(b_i := f(\rho^i(a))\), and let \(B := \{b, \rho(b), \ldots, \rho^{k-1}(b)\}\).
We claim that the smallest subalgebra $C$ of $A^k$ that contains $B$ is cyclic. So let $c \in C$ be arbitrary. Since $C$ is generated by $B$, there exists a $\tau$-term $s(x_0, x_1, \ldots, x_{k-1})$ such that $c = s^A(b, \rho(b), \ldots, \rho^{k-1}(b))$. Then $\rho(c) = s^A(\rho(b), \rho^2(b), \ldots, \rho^{k-1}(b), b) \in C$.

According to our assumption, the algebra $C$ contains a constant tuple $d$. Then there exists a $\tau$-term $r(x_0, \ldots, x_{k-1})$ such that $d = r^C(b, \rho(b), \ldots, \rho^{k-1}(b))$. Note that
\[
r^A(b_0) = r^A(b_1) = \cdots = r^A(b_{k-1})
\]
since $d$ is constant (also see Figure 5). It follows that $b \in S(r)$.

Now consider the $\tau$-term $t(x_0, x_1, \ldots, x_{k-1})$ defined by
\[
t(x) := r(t(x), t(\rho(x)), \ldots, t(\rho^{k-1}(x))).
\]
where $x := (x_0, x_1, \ldots, x_{k-1})$. We claim that $S(f) \subseteq S(t)$, but also that $a \in S(t)$. This is clearly in contradiction to the maximality of $|S(f)|$. Let $e \in S(f)$. Then
\[
t^A(\rho^i(e)) = r^A(f^A(\rho^i(e), f^A(\rho^{i+1}(e)), \ldots, f^A(\rho^{i-1}(e)))) \\
= r^A(f^A(e), f^A(e), \ldots, f^A(e)) \\
= f^A(e)
\]
for all $i \in \{0, \ldots, k-1\}$. Therefore, $e \in S(t)$. On the other hand,
\[
t^A(\rho^i(a)) = r^A(f^A(\rho^i(a), f^A(\rho^{i+1}(a)), \ldots, f^A(\rho^{i-1}(a)))) \\
= r^A(b_i, b_{i+1}, \ldots, b_{i-1}) \\
= r^A(\rho^i(b))
\]
which is constant for all $i$ by the choice of $r$. Therefore, $a \in S(t)$ and the contradiction is established. \qed

**Theorem 7.4** (of [5]). Let $A$ be a finite idempotent algebra. Then the following are equivalent.

- $A$ has a Taylor term;
- $A$ has a cyclic term;
- for all prime numbers $p > |A|$, the algebra $A$ has a $p$-ary cyclic term.
Proof. Again, the proof of this theorem is out of the scope of this course. We mention that is also uses absorption theory, and explicitly uses a strong version of Theorem 7.1.

Note that every cyclic term $c$ of $A$ is in particular a \textit{weak near unanimity term}, that is, it satisfies
\[
A \models \forall x, \forall y. c(x, \ldots, x, y) = c(x, \ldots, y, x) = \cdots = c(y, x, \ldots, x).
\]

As an application, we derive the classification of the complexity of $H$-colouring for finite undirected graphs $H$.

\textbf{Theorem 7.5} (of Hell and Nešetřil; proof from [5]). \textit{If $H$ is bipartite or has a loop, then $	ext{CSP}(H)$ is in P. Otherwise, $	ext{CSP}(H)$ is NP-complete.}

\textit{Proof.} If the core $G$ of $H$ equals $K_2$ or has just one vertex, then $	ext{CSP}(H)$ can be solved in polynomial time, e.g. by the Path Consistency Procedure, Section 3. Otherwise, $G$ is not bipartite and there exists a cycle $a_0, a_1, \ldots, a_{2k}, a_0$ of odd length in $H$. If $H$ has no Taylor polymorphism, then by Corollary 6.23, $	ext{CSP}(H)$ is NP-hard.

Otherwise, if $H$ has a Taylor polymorphism, then Theorem 7.4 asserts that there exists a $p$-ary cyclic polymorphism of $H$ where $p$ is a prime number greater than \max\{2k, |A|\}. Since the edges in $H$ are undirected, we can also find a cycle $a_0, a_1, \ldots, a_p, a_0$ in $H$. Then $c(a_0, a_1, \ldots, a_p) = c(a_1, \ldots, a_p, a_0)$, which implies that $H$ contains a loop, a contradiction to the assumption that the core of $H$ has more than one element. \hfill \Box

7.3 Siggers Terms

Interestingly, whether or not a finite idempotent algebra $A$ has a cyclic term can be tested by searching for a single 4-ary term $t$ that satisfies
\[
A \models \forall x, \forall y, \forall z. s(x, y, z, y) = s(y, z, x, x),
\]
a so-called Siggers term. In particular, the question whether a given finite structure has a cyclic term is in NP. Siggers originally found a 6-ary term, which has been improved later to the 4-ary term given above. The proof given below is from [29].

\textbf{Theorem 7.6} (due to [42]; see [29]). \textit{Let $A$ be a finite idempotent algebra. Then $A$ has a cyclic term if and only if $A$ has a Siggers term.}

\textit{Proof.} Suppose that $A$ has a cyclic term. Let $p$ be some prime number larger than $|A|$ of the form $3k + 2$ for some $k \in \mathbb{N}$ and let $c(x_1, \ldots, x_p)$ be a cyclic term of $A$, which exists by Theorem 7.4. Define $s(x, y, z, w)$ to be the term
\[
c(x, x, \ldots, x, y, y, \ldots, y, w, z, \ldots, z),
\]
where the variable $x$ occurs $k + 1$ times, the variable $w$ occurs once, and the variables $y$ and $z$ occur $k$ times each. Then
\[
s(x, y, z, y) = c(x, x, \ldots, x, y, y, \ldots, y, y, z, \ldots, z) \quad (k + 1 \text{ times } x, \ k + 1 \text{ times } y, \ k \text{ times } z) \\
= c(y, y, \ldots, y, z, z, \ldots, z, x, x, \ldots, x) \quad \text{(by cyclicity of } c) \\
= s(y, z, x, x).
\]
Conversely, a Siggers term is a Taylor term, and therefore the other direction follows from Theorem 7.4. \hfill \Box
8 Bounded Width

Equipped with the universal-algebraic approach, we come back to one of the questions that occupied us at the beginning of the course: which $H$-colouring problems can be solved by the path-consistency procedure, $PC_H$, introduced in Section 3? We have seen in Section 3.1 that when $H$ has a majority polymorphism, then $PC_H$ solves the $H$-colouring problem. But this was just a sufficient, not a necessary condition.

A necessary and sufficient polymorphism condition for solvability by $PC_H$ has been found by Barto and Kozik [4]. Their result is much stronger: it characterises not just the strength of $PC_H$, but more generally of $k$-consistency (introduced in Section 3, and not just for $H$-colouring, but for CSPs of finite structures in general. Before we state their result in Theorem 8.1 below, it is convenient to use a more flexible terminology to discuss the idea of $k$-consistency for general relational structures more precisely.

The basic observation is that when generalising 3-consistency for the $H$-colouring problem to $k$-consistency for CSPs of arbitrary finite structures, there are two essential parameters:

- the first is the arity $l$ of the relations maintained for all $l$-tuples of variables in the instance. For $PC_H$, for instance, we have $l = 2$.

- the second is the number of variables considered at a time within the main loop of the algorithm. For $PC_H$, for instance, we have $k = 3$.

Hence, for each pair $(l,k) \in \mathbb{N}^2$, we obtain a different form of consistency, called $(l,k)$-consistency.

Note that it is easy to come up with finite structures $\mathfrak{A}$ whose CSP cannot be solved by $(l,k)$-consistency when $\mathfrak{A}$ might contain relations of arity larger than $k$ (there is no possibility of the $(l,k)$-consistency algorithm to take constraints into account that are imposed on more than $k$ variables). We say that CSP($\mathfrak{A}$) can be solved by local consistency if it can be solved by $(l,k)$-consistency, for some $l,k \in \mathbb{N}$.

**Theorem 8.1** (Barto and Kozik [4]). Let $\mathbf{A}$ be an idempotent finite algebra. Then the following are equivalent.

1. For all structures $\mathfrak{A}$ with maximal arity $k$ such that $\mathbf{A}$ is a polymorphism algebra of $\mathfrak{A}$ the problem $\text{CSP}(\mathfrak{A})$ can be solved by establishing $(2, k+1)$-consistency.

2. There is no homomorphism from $\text{Clo}(\mathbf{A})$ to the clone of a vector space with at least two elements.

3. There exists an $n \in \mathbb{N}$ such that $\mathbf{A}$ has for every $m \geq n$ a weak near unanimity term of arity $m$.

4. $\mathbf{A}$ contains weak near unanimity terms $f$ and $g$ such that

$$\mathbf{A} \models \forall x,y. f(y,x,x) = g(y,x,x,x).$$

A guide to references. For the implication from 1 to 2, suppose for contradiction that there is a homomorphism from $\text{Clo}(\mathbf{A})$ to the clone of a vector space with at least two elements. Then the proof consists of two ideas:

- Linear equations over finite fields cannot be solved by local consistency [18];
The implication from 2 to 3 is due to Maroti and McKenzie [33]. The implication from 3 to 1 is the most difficult one, and has been open for quite some time; again, the solution of Barto and Kozik [4] uses absorption theory, among other techniques. The equivalence of 3 and 4 is from [29].

\[ \forall x, y. \, g(y, x, x) = f(y, x, x) \]

Another remarkable Corollary is that for the \( H \)-coloring problem, \((2, 3)\)-consistency is as powerful as \((2, k)\)-consistency for all \( k \geq 3 \) (we already stated this in Theorem 3.1). One technical step of the proof of Theorem 8.1 is to reduce the argument to an argument about the strength of \((2, 3)\)-consistency, essentially via the following proposition.

Proposition 8.3 (From [3]). Let \( \mathfrak{B} \) be a relational structure whose relations all have arity at most \( k \). Then there is a binary relational structure \( \mathfrak{A} \) with domain \( B^k \) such that \( \text{Pol}(\mathfrak{B}) \) and \( \text{Pol}(\mathfrak{A}) \) are isomorphic. If the signature of \( \mathfrak{B} \) is finite, the signature of \( \mathfrak{A} \) is finite, too.

Proof. First we replace every relation \( R \) in \( \mathfrak{B} \) of arity \( l < k \) by the \( k \)-ary relation

\[ \{(r_1, \ldots, r_l, b_{l+1}, \ldots, b_k) \mid (r_1, \ldots, r_l) \in R, b_{l+1}, \ldots, b_k \in B\} \]

This clearly does not change the set of polymorphisms of \( \mathfrak{B} \), and therefore we may assume that every relation in \( \mathfrak{B} \) has arity precisely \( k \).

Next we introduce the relations in \( \mathfrak{A} \). For every \( k \)-ary relation \( R \) in \( \mathfrak{B} \) we introduce in \( \mathfrak{A} \) the unary relation \( R \) (on \( A = B^k \)), and for all \( i, j \in \{1, \ldots, k\} \) we add a binary relation \( S_{i,j} \) defined as

\[ S_{i,j} := \{((u_1, \ldots, u_k), (v_1, \ldots, v_k)) \mid u_i = v_j\} \]

Let \( f \in \text{Pol}(\mathfrak{B}) \) be arbitrary of arity \( n \). Define the function \( \bar{f} : A^n \to A \) by

\[ \bar{f}(u_1, \ldots, u_k) := (f(u_1, \ldots, u_i), \ldots, f(u_{i-1}, \ldots, u_k)) \]

Since the relation \( S_{i,j} \), viewed as a \( 2k \)-ary relation over \( B \), is primitively positively definable in \( \mathfrak{B} \), it is preserved by \( \bar{f} \). Therefore it is easy to see that \( \bar{f} \) preserves \( S_{i,j} \). Now suppose that \( h : B^n \to B \) is a polymorphism of \( \mathfrak{B} \). Let \( \bar{h} : (A^n)^n \to A \) be the function where \( \bar{h}(u_1^n, \ldots, u_k^n) \) is defined to be the \( i \)-th component of the tuple

\[ h((u_1, \ldots, u_k), \ldots, (u_1, \ldots, u_k)) \in A^n \]

Since \( h \) preserves \( S_{i,j} \), we have that

\[ h_i((u_1, \ldots, u_k), \ldots, (u_1^n, \ldots, u_k^n)) = h_i((v_1, \ldots, v_k), \ldots, (v_1^n, \ldots, v_k^n)) \]

whenever \( u_l^i = v_l^j \) for all \( l \in \{1, \ldots, n\} \). Hence, \( h_i \) depends only on \( u_1^i, \ldots, u_k^i \), and thus there exists an \( n \)-ary operation \( f_i \) on \( A \) such that \( h_i((u_1, \ldots, u_k), \ldots, (u_1^n, \ldots, u_k^n)) = f_i(u_1, \ldots, u_k) \). Note that \( f_i \) is a polymorphism of \( \mathfrak{A} \).

Now, let \( i, j \in \{1, \ldots, k\} \) be distinct. For any \( u_1^i, \ldots, u_k^i \in A \) we choose arbitrarily \( v_1^i, \ldots, v_k^i \) such that \( u_l^i = v_l^j \) for all \( l \in \{1, \ldots, n\} \). Since \( h \) preserves \( S_{i,j} \), it follows that \( f_i(u_1^i, \ldots, u_k^i) = f_j(v_1^i, \ldots, v_k^i) = f_j(u_1^j, \ldots, u_k^j) \). Therefore, \( f_1 = f_2 = \cdots = f_k \). We have shown that \( h = \bar{f} \) for some \( f \in \text{Pol}(\mathfrak{A}) \), and the statement follows.
9 List $H$-coloring

Recall that in the list $H$-coloring 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 $x \in V(G)$.

This can be modelled as the constraint satisfaction problem for the structure $\mathfrak{A}$ with domain $V(G)$ which has, besides the relation $E$, for every subset $S$ of the domain a unary relation $U$ in the signature such that $U^\mathfrak{A} = S$. Note that the polymorphisms of such a structure must be conservative; this is why the corresponding constraint satisfaction problems are called conservative, too.

Bulatov [9] gave a full complexity classification for conservative CSPs, and therefore, a fortiori, for list homomorphism problems. The result confirms the tractability conjecture 2 in this special case.

Theorem 9.1. Let $\mathfrak{A}$ be a finite structure such that all polymorphisms are conservative, and let $A$ be a polymorphism algebra of $\mathfrak{A}$. If every two-element subalgebra of $A$ has a semilattice, Majority, or Minority operation, then CSP($\mathfrak{A}$) is in $P$. Otherwise, CSP($\mathfrak{A}$) is NP-hard.

One direction of this theorem follows by general principles that we have already seen in this course: if $A$ has a two-element subalgebra $B$ without semilattice, Majority, or Minority operation, then CSP($\mathfrak{A}$), then by Lemma 6.2 and Theorem 5.18 CSP($\mathfrak{A}$) is NP-hard. We will comment on the other direction below.

In fact, we can relax the conditions of Bulatov’s result. Suppose that $\mathfrak{A}$ only contains for all $k$-element subsets $S$ of the domain a relation for $S$. The consequence is that for every polymorphism $f$ of $\mathfrak{A}$, $f(x_1, \ldots, x_n) \in \{x_1, \ldots, x_n\}$ whenever $|\{x_1, \ldots, x_n\}| \leq k$. We call such operations $k$-conservative.

Corollary 9.2. Let $\mathfrak{A}$ be a finite structure such that all polymorphisms are $3$-conservative. Then the conclusions of Theorem 9.1 hold as well.

Proof. Let $A$ be a polymorphism algebra of $\mathfrak{A}$. If $A$ has a 2-element subalgebra $B$ without semilattice, Majority, or Minority operation, then CSP($\mathfrak{A}$) is NP-hard. But note that an at most ternary operation is 3-conservative if and only if it is conservative. Hence, we can add all unary relations to the signature of $\mathfrak{A}$, and still satisfy the conditions of Theorem 9.1 that imply polynomial-time tractability. \qed

To prove the main direction of Theorem 9.1, Bulatov used the following construct, which he called the graph of a constraint language, $G_\mathfrak{A}$. The vertices of this directed graph are the elements of the domain. We add a red edge $(a, b)$ if the two-element subalgebra of $A$ with domain $\{a, b\}$ contains a semi-lattice operation $f$ such that $f(a, b) = f(b, a) = a$. We add a yellow edge $(a, b)$ if neither $(a, b)$ nor $(b, a)$ is a red edge, and if the two-element subalgebra with domain $\{a, b\}$ has a majority operation. Note that if $(a, b)$ is a yellow edge, then $(b, a)$ is yellow as well. Finally, $(a, b)$ is a blue edge if neither $(a, b)$ nor $(b, a)$ is a red of yellow edge, and if the two-element subalgebra with domain $\{a, b\}$ has a minority operation. The assumptions of the statement of Theorem 9.1 imply that for all $a, b \in A$, the pair $(a, b)$ is a red, yellow, or blue edge.

Proposition 9.3. Suppose that $A$ is a finite conservative $\tau$-algebra such that every two-element subalgebra of $A$ has a semilattice, majority, or minority operation. Then there are $\tau$-terms $f, g, h$ such that
• For every 2-element subalgebra $B$ of $A$, the operation $f_B$ is a semilattice operation.

• For every 2-element subalgebra $B$ with domain $\{b_1, b_2\}$ of $A$, the operation $g_B$ is a majority operation when $(b_1, b_2)$ is yellow, $p_1^3$ if $(b_1, b_2)$ is blue, and $f_B(f_B(x, y), z)$ if $(b_1, b_2)$ is red.

• For every 2-element subalgebra $B$ with domain $\{b_1, b_2\}$ of $A$, the operation $h_B$ is a minority operation when $(b_1, b_2)$ is blue, $p_1^3$ if $(b_1, b_2)$ is yellow, and $f_B(f_B(x, y), z)$ if $(b_1, b_2)$ is red.

**Proof sketch.** Let $B_1, \ldots, B_n$ be a list of all red edges of $G_3$, and $B_1, \ldots, B_1$ the corresponding subalgebras of $A$, and $f_i$ the semilattice operation of $B_i$ that witnesses that $B_i$ is red. We define $f_1 := f_1$, and for $i \in \{2, \ldots, n\}$

$$f_i(x, y) := f_i(f_i^{-1}(x, y), f_i^{-1}(y, x)).$$

Then $f_n(f^n(x, y), x)$ has the required properties in item 1. The other items can be shown similarly. 

**References**


