Applied Algebra - Graph Homomorphisms

KNU Math 426

Classnotes

Mark Siggers

v. 2020/09/21

The Applied Algebra course is a topics course for fourth year students, taught be three different teachers. These notes are for the first third of the class. Students should have taken abstract algebra, and will benefit from having taken graph theory or combinatorics.

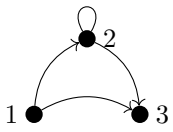
For graph homomorphisms in general, we recommend as a reference the distinctivley smelling book [2] of Hell and Nešetřil. For a much deeper presentation of the material presented in these notes we recommend the course notes [1] of Manuel Bodirsky.

1 Graphs and Homomorphisms

In this class we take a slightly more general and algebraic definition of graphs than we would in a graph theory class.

1.1 Basic Definitions

A *digraph* (short for directed graph) $G = G(V, A)$ consists of a set V of *vertices*, and a binary *arc* relation $A \subset V^2$ on the vertex set. We often introduce a digraph as G and then let $V(G)$ and $A(G)$ denote its vertex set and arc set. Below is a depiction of the digraph $([3], \{(2,2), (1,2), (2,3), (1,3)\})$, (where recall that $[n] = \{1, 2, \ldots, n\}$) .



A digraph $G = (V, A)$ is *symmetric, reflexive,* or *irreflexive* if A has this property as a relation. That is to say, G is symmetric if $(a, b) \in A \implies (b, a) \in A$. The digraph $([3], \{(1,2), (2,1), (2,3)\})$, is properly drawn on the left below. However, we tend to draw the two edges $(1, 2)$ and $(2, 1)$ (which we sometimes call a *diode* as one symmetric edge, as we have done on the right.

A *graph* is a symmetric digraph. It may have loops, so this definition differs from the standard definition of graph, which corresponds (upto viewing diodes as symmetric edges) to irreflexive graphs. We use irreflexive graphs a lot, but we will also consider more the general digraphs we have defined, and then we generalise this definition even more to 'relational systems'. For graphs, the relation A is often denoted E and its elements are called *edges*. (In fact, even for digraphs, we often use E instead of A even though we call them arcs.) **For the first couple chapters, unless we say so explicity, all graphs are irreflexive.**

Some common graphs are given below. Notice we write ij for and edge (i, j). We will do this when it does not cause confusion.

- The complete graph on n vertices, K_n, has $V(K_n) = [n]$ and $E(K_n) = \{ij \mid i, j \in V, i \neq j\}$.
- The n-cycle, C_n, has $V(C_n) = [n]$ and $E(C_n) = \{ij \mid |i - j| = 1 \text{ mod n}\}$.
- The n-path, P_n, has $V(P_n) = \{0, 1, \ldots, n\}$ and $E(P_n) = \{ij \mid |i - j| = 1\}$.
- The complete bipartite graph, $K_{m,n}$, has $V(K_{m,n}) = \{a_1, \ldots a_m\} \cup \{b_1, \ldots, b_n\}$ and $E(K_{m,n}) = \{a_i b_j, b_j a_i \mid i \in [m], j \in [n]\}$.
- The n-cube, Q_n, $V(Q_n)$ is the set of binary strings of length n, vertices u and v are adjacent if they differ in exactly one coordinate.

Definition 1.1. A *homomorphism* $\phi : G \to H$ from a graph G to a graph H is a function $\phi : V(G) \to V(H)$ from the vertex set of G to the vertex set of H which preserves the relation E:

$$(u, v) \in E(G) \to (\phi(u), \phi(v)) \in E(H).$$

If there is a homomorphism ϕ from G to H, we write $G \to H$ (and often say G *maps* to H), if not, we write $G \not\to H$. The main question we are going to ask in this class, given digraphs G and H, is $G \stackrel{?}{\to} H$: 'Does there exists a homomorphism from G to H?'

Not only do we look at answering $G \stackrel{?}{\to} H$ for particular graphs G and H, we look at how hard it is to solve this problem in general.

Exercises

1.1.1 Give (draw) examples of reflexive, irreflexive, symmetric, and reflexive symmetric digraphs.

1.1.2 Draw the graphs $K_4, P_4, C_4, K_{3,4}$ and Q_3.

1.1.3 Where (G, H) is each of the following pairs, decide whether or not $G \to H$.

$$(K_3, K_3), (C_5, C_3), (C_4, C_3), (C_3, C_4), (K_5, K_4) \text{ and } (C_5, K_2).$$

1.1.4 What is the big difference between the question $G \stackrel{?}{\to} H$ for reflexive graphs and for irreflexive graphs? (This is why we are restricting to symmetric irreflexive graphs to start off.)

1.2 Some basic facts about homomorphisms

Many basic concepts and questions from graph theory can be rephrased as questions about homomorphisms, and often the homomorphism point of view provides useful tools to approach these problems.

Recall the following basic definitions about graphs.

Note: Basic Graph Definitions

- A *subgraph* G' of G is a graph G' such that $V(G') \subseteq V(G)$ and $E(G') \subset E(G)$.
- A *walk of length* ℓ in a graph G is a sequence $v_0, v_1, v_2, \ldots, v_\ell$ of vertices of G such that $v_i \sim v_{i+1}$. It is a *circuit of length* ℓ if $v_0 = v_\ell$.
- The *distance* $d(u, v)$ between vertices u and v in G is the length d of the shortest walk $w : P_d \to G$ in G such that $w(0) = u$ and $w(d) = v$.
- A graph is *connected* if the distance between any pair of vertices is finite.
- A *component* of G is maximal connected subgraph.

You will prove the following simple facts in Exercise 1.2.1. From parts *ii.* and *iii.* we see that we can solve the problem $G \stackrel{?}{\to} H$ by solving it for all components of G and H. This allows us to consider only the case that G and H are connected.

Fact 1.2. *The following are true of all graphs G, H and F.*

i. *If S is a subgraph of G and $G \to H$ then $S \to H$.*

ii. *$G \to H$ if and only if $G_i \to H$ for all components G_i of G.*

iii. *If G is connected, then $G \to H$ if and only if $G \to H_i$ for some component H_i of H.*

iv. *If $\alpha : G \to H$ and $\beta : H \to F$ are homomorphisms, then so is the map $\beta \circ \alpha : G \to F$ defined by $[\beta \circ \alpha](g) = \beta(\alpha(g))$.*

v. *For any walk W of length ℓ in a graph H there is a homomorphism $\phi : P_\ell \to H$ such that $W = \phi(P_\ell)$. For any circuit C of length ℓ in H there is a homomorphism $\phi : C_\ell \to H$ such that $C = \phi(C_\ell)$. W (C) is a path (cycle) if and only if ϕ is injective on vertices.*

Recall that a graph $G = (V, E)$ is *bipartite* if there is a partition $V = A \cup B$ of its vertex set such every edge has one vertex in each of A and B. In a graph class, we would prove that a graph is bipartite if and only if it contains no odd cycles. With homomorphisms as a topic, there are a couple other interesting characterisations of bipartite graphs.

Fact 1.3. *For a (connected) irreflexive graph G, the following are equivalent.*

i. *G is bipartite.*

ii. *$G \to K_2$.*

iii. *For all integers $k \geq 1$, $C_{2k+1} \not\to G$.*

iv. *G contains no odd-cycles.*

Proof. We show a couple of the harder implications. The others are left as an exercise.

(*iii.* $\implies$ *i.*) We do the contrapositive. Assume that G is not bipartite. Try to partition its vertices into sets A and B by choosing a vertex v_0 to be in A, and then putting vertices in A if they have even distance from v_0, and in B otherwise. As we are not bipartite, there are two vertices x and y both in A (or both in B) that are adjacent. Taking a shortest walk in G from v_0 to x, continue the walk along the edge xy to y, and then continue along the shortest walk in G from y to v_0. This is a closed walk of odd length, so by part *v.* of Fact 1.2, is the image of a homomorphism from an odd cycle.

(*iv.* $\implies$ *iii.*) We show by induction that an odd closed walk (i.e. a homomorphism from C_{2k+1} in G contains an odd cycle. In the case that $k = 1$ this is trivial as G is irreflexive. So let $k \geq 2$. We are done if no vertex of the walk (except the endpoints) occurs more than once, as then the walk is an odd cycle in G. So assume that there are $0 < i < j < 2k+1$ such that $\phi(i) = \phi(j)$. Then ϕ restricted to the paths $i, i+1, \ldots, j$ and $j, j+1, \ldots \ell, 0, 1, \ldots, i$ are shorter closed walks, and one of them is odd, so by induction contains an odd cycle.

□

Exercises

1.2.1 Prove parts *i.* to *iv.* of Fact 1.2.

1.2.2 True or False. (Give a proof if true and a counter-example if false.)

(a) For every graph G, $G \to K_1$.

(b) If $G \to H$ and $H \to G$, then $G = H$.

(c) If $G \to H$ and $F \to H$, then $G \to F$ or $F \to G$.

(d) If $G \to H$ and $H \to F$, then $G \to F$.

(e) For every graph G, $G \to K_n$ for some integer $n \geq 1$.

1.2.3 Show that no odd cycle admits a homomorphism to K_2. (That is, show that for all $k \geq 1$, we have $C_{2k+1} \not\to K_2$.)

1.2.4 Finish the proof of Fact 1.3. (Hint: I suggest showing the implications *i.* $\implies$ *ii.* $\implies$ *iii.* $\implies$ *iv.*. For *ii.* $\implies$ *iii.* you might use Fact 1.2 and Exercise 1.2.3.)

1.3 The homomorphism order and products

Because homomorphisms compose, as we saw in Fact 1.2, the class of graphs makes up a category, and we get many category theoretic results that we do not always consider in a graph theory class. First among these is an pre-ordering of graph. Consider the relation $\geq$ on the set of all digraphs defined by $H \geq G$ if $G \to H$. It is transitive, and reflexive, so is a pre-order, but fails to be anti-symmetric, so is not a partial order. But there are a couple ways that we can fix this.

Two graphs G and H are *homomorphically equivalent* if $G \to H$ and $H \to G$. Let $[G]$ be the *homomorphism class of* G — the set of all graphs that are homomorphically equivalent to G.

Fact 1.4. *Let G and H be graphs, and $G' \in [G]$ and $H' \in [H]$ be graph homomorphically equivalent to them. Then $G' \to H' \iff G \to H$, so the relation $\leq$ is well-defined on homomorphism classes, and is a partial order.*

Let $G_a, G_b \in [G]$ both be graphs in $[G]$ with a minimum number of vertices. We have homomorphisms $\alpha : G_a \to G_b$ and $\beta : G_b \to G_a$ so $\beta \circ \alpha$ is a homomorphism of G_a to a subgraph. The subgraph cannot have fewer vertices by choice of G_a, and so $\beta \circ \alpha$ simply permutes the vertices of G_a. If the subgraph has fewer edges, then some edge is mapped to a non-edge, which is impossible. So $\beta \circ \alpha$ is an isomorphism of G_a, implying that both α and β are isomorphisms. This gives the following.

Fact 1.5. *There is a unique, upto isomorphism, smallest graph in a homomormphism class $[G]$ of graphs.*

This graph is the *core* $C(G)$ of $[G]$ and of any $G' \in [G]$. A graph is a *core* if it is the core of a homomorphism class. If follows that the homomorphism order $\leq$ on cores is a partial order. The following allows us, when we are cosidering the problem of $G \stackrel{?}{\to} H$ to assume that G and H are cores.

Fact 1.6. *If G and H be graphs, and $C(G)$ and $C(H)$ be their respective cores. then*

$$G \to H \iff C(G) \to C(H).$$

You observe in the exercises that the graph order has infinte chains, and we mention that it also has inifinte antichains. Other interesting question about the graph are are whether or not there are *gaps* (A, B): graphs A and B such that $A > B$ but for which there is no C such that $A > C > B$. It turns out that there is only one gaps in the homomorphism order.

We mention now one last categorical construction, a generalisation of which we will use a lot.

Definition 1.7. The (categorical) product of two graphs G and H is the graph $G \times H$ with vertex set $V(G) \times V(H)$ and such that

$$E(G \times H) = \{((g,h)(g',h')) \mid (g,g') \in E(G) \text{ and } (h,h') \in E(H).$$

Fact 1.8. *Let G and H be graphs.*

i. $G \times H \cong H \times G$.

ii. The projection $\pi_G : G \times H : (g,h) \mapsto g$ is a homomorphism. (And so is π_H.)

Exercises

1.3.1 Show that the relation $\geq$ on graphs is a pre-order: transitive, and reflexive. If it was also anti-symmetric ($x \leq y$ and $y \leq x$ impies $x = y$) it would be a partial order. Show that it is not anti-symmetric.

1.3.2 What graphs are in the homomorphism class $[K_2]$? What is the core of this class?

1.3.3 Prove Fact 1.6.

1.3.4 Show that K_k and C_{2k+1} are cores for all $k \geq 1$.

1.3.5 Show that in the homomorphimsm order of graphs

$$\ldots C_7 \leq C_5 \leq C_3 = K_3 \leq K_4 \leq K_5 \leq K_6 \leq \ldots$$

is an infinite chain. (Does this hold for reflexive graphs?)

1.3.6 Find a pair of incomparible digraphs in the homomorphisms order: digraph A and B such that $A \not\to B$ and $B \not\to A$. (In fact one can find infinitely many graphs that are pairwise incomparible, but this is a bit harder. See if you can find this with the help of Google. Search for Mycielski graph, or sparse incomparibillity.)

1.3.7 Show that (K_2, K_1) is a gap in the homomorphism order. (It is the only one. Google this too.)

1.3.8 Draw $K_2 \times K_2$, $\vec{K_2} \times \vec{K_2}$, and $K_2^r \times K_2^r$ where $\vec{K_2}$ is the digraph on two vertices with only one arc, and K_2^r is K_2 with loops on each vertex.

1.3.9 Show that if G is bipartite and connected then $G \times K_2$ consists of two components, each isomorphic to G.

1.3.10 Show that if G is non-bipartite and connected, then $G \times K_2$ is bipartite and connected.

1.3.11 What is $K_1 \times G$? How about $K_1^r \times G$?

1.3.12 Prove Fact 1.8.

1.3.13 Show that a graph H is a core if and only if its only endomorphisms (that is, homomorphisms to itself) are automorphisms (that is, isomorphisms to itself).

1.3.14 For an endomorphism $r : H \to H$, show that the following conditions are equivalent.

(a) $r^2 = r$. (That is, $r(r(x)) = r(x)$ for all $x \in V(H)$).
(b) $r(x) = x$ for all $x \in r(V(H) := \{r(x) \mid x \in V(H)\}$.

1.3.15 An endomorphism $r : H \to H$ satisfying the equivalent conditions of Exercise 1.3.14 problem is called a *retraction* of H. The subgraph $r(H)$ of H induced by the image $r(V(H))$ is a *retract* of H. Show that the core C of H is, upto isomorphism, a retract of H, and indeed, is the smallest retract of H.

1.4 Graph Colouring and its extensions

The main application of homomorphisms to the study of graphs is to graph colouring.

Recall that a *k-colouring* of an irreflexive graph G is a function $f : V(G) \to [k]$ such that

$$u \sim v \implies f(u) \neq f(v).$$

The *chromatic number* $\chi(G)$ of a graph G is the minimum k such that G has a k-colouring. The chromatic number is one of the most basic graph invariants, and there are thousands of papers dedictated to it. And it is just a homomorphism.

Fact 1.9. *There is a k-colouring of a irreflexive graph G if and only if $G \to K_k$.*

Proof. Let $f : V(G) \to [k]$ be a k-colouring of G. We claim that where K_k has vertex set $[k]$, this is a homomorphism of G to K_k. Indeed, let $u \sim v$ in G, as f is a k-colouring $f(u) \neq f(v)$, and so since these are vertices of K_k, $f(u) \sim f(v)$.

The other direction is just as easy. □

Recall we had the infinite ascending chain

$$K_1 \leq K_2 \leq K_3 \leq K_4 \dots$$

Every graph G is eventually below something in this chain. The chromatic number tells us where G joins the chain. Many graph invariants take this form.

In a graph class you might see several variations of k-colouring. For example, a circular (k, d)-colouring of a graph G is a function $f : V(G) \to \mathbb{Z}_k$ (where $\mathbb{Z}_k$ is the ring $\mathbb{Z}/k\mathbb{Z}$ of integers modulo k) such that for $u \sim v$ in G, $|f(u) - f(v)| > d$. A fractional (k, d)-colouring of a graph G is a function $f : V(G) \to \binom{[k]}{d}$ assigning d colours in $[k]$ to each vertex of G such that for each $u \sim v$, $f(u) \cap f(v) = \emptyset$.

$\binom{S}{d}$ for a set S is the family of d-element subsets of S.

A circular (k, d)-colouring and a fractional (k, d)-colouring of a graph G can be expressed as a homomorphim of G to graphs $C_{k,d}$ and $F_{k,d}$ respectively. Your are asked to find these graphs in the Exercises, and to show that they each yield 'dense' infinte ascending chains containing the chain $K_2 \leq K_3 \leq K_4 \dots$.

Exercises

If a problem is labelled 'maybe hard' it is because I haven't had time to look at it. These ones are well known, important results. Possibly they are trivial. Possibly someone wrote a paper to prove them. I think that likely they are not too hard. If you get a proof for them, let me know. If they are hard an you find a reference for them on the internet, let me know this also. Thanks!

1.4.1 Prove that an irreflexive graph G is k-colourable if and only if $G \to K_k$.

1.4.2 Find the graph $C_{k,d}$ such that $G \to C_{k,d}$ if and only if G has a circular (k, d)-colouring.

1.4.3 Find the graph $F_{k,d}$ such that $G \to F_{k,d}$ if and only if G has a fractional (k, d)-colouring.

1.4.4 (Maybe hard.) Show that if $k/d = 3$ then $C_{k,d}, F_{k,d} \in [K_3]$.

1.4.5 (Maybe hard.) Show that if $k/d < k'/d'$ then $C_{k,d} \to C_{k',d'}$ and $F_{k,d} \to F_{k',d'}$.

1.4.6 For a graph G a digraph $\vec{G}$ we get by throwing away one of (i, j) and (j, i) for every edge (diode) $\{i, j\}$ of G, is called an *orientation* of G.

(a) How many orientations are there of a graph G?

(b) Show that $G \to H$ if and only if there are orientations $\vec{G}$ and $\vec{H}$ of G and H with $\vec{G} \to \vec{H}$.

1.5 The complexity of k-colouring

A *(decision) problem* consists of *instances* and a *decision* such that for every instance one can answer the decision question in finite time and return an answer of YES or NO. Here is how we define problems.

Problem. Col(k) or k-colouring.
Instance: An irreflexive graph G.
Decision: Is there a homomorphism $G \to K_k$?

Every instance has a *size* n, and this is usually clear, so we don't state it. For the problem Col(k) we take this to be the number of vertices of the instance. To see that Col(k) is actually a computational problem, we must observe that there is an algorithm that completes it in finite time. Here is one for an instance of size n having vertex set $[n]$.

Notice that there are k^n possible functions from $[n]$ to $[k]$. Let them be called $f_1, \ldots, f_{k^n}$, and proceed as follows.

Algorithm 1.10. *(k-colouring)*

Input: *An irreflexive graph G.*
Result: YES *if G is k-colourable,* NO *otherwise.*
$i := 1$;
while $i \leq k^n$ **do**
  foreach *edge* $uv \in E(G)$ **do** *CHECK* $f_i(u) = f_i(v)$;
  if TRUE *for all edges* **then exit** *and* **return** YES **else** $i := i + 1$;
end *We have CHECKed all f_i but not exited.*
return NO ;

The *running time* of an algorithm is a functions $f : \mathbb{N} \to \mathbb{N}$ such that any instance of size n can be solved in time at most $f(n)$. For us, 'time' is usually some basic operation such as adding or multiplying two (bounded) integers, or querying a database such as the database of edges of a graph. For Algorithm 1.10 we take the basic operations as CHECKing if two vertices get the same colour, and QUERY, which gets an edge of the graph.

Let's find an upper-bound on the running time of Algorithm 1.10. There are two loops. The outer loop runs over n^k values of i and the inner loop runs over $\binom{n}{2} = O(n^2)$ edges. For each pass of the inner loop we have two basic operations, one QUERY and one CHECK. So the algorithm it has running time

$$k^n \cdot O(n^2) \cdot 2 = O(k^n),$$

Note that constants such as 2 are absorbed into the big-oh notation.

which is exponential in n. This is slow, but finite, so the problem is a computational problem.

This, of course, many not be the best algorithm for solving Col(k). The *complexity* of a problem is the running time of the best possible algorithm for solving it. It is difficult to know if we have the best possible algorithm for a problem, so when we give the complexity of a problem we generally give it as an upper bound. This is why the big-oh notation $O(k^n)$ is commonly used.

In fact we can solve Col(1) in constant time. We just have to CHECK once to see if there is an edge. When $k = 2$ we can solve Col(2) in polynomial time, using a spanning tree.

Algorithm 1.11. *(Spanning Tree)*

Input: *A connected graph G*
Result: *A spanning subtree T of G*
Set $i := 0$, $V = V(G)$, $E = E(G)$, and $E_* = \emptyset$.
Pick a vertex $v_0 \in V(G)$ and set $V(T) = \{v_0\}$ and $E(T) = \emptyset$.
while $i < n$ **do**
  $i := i+1$;
  Find $v \in V(G) \setminus V(T)$ with edge to u in $V(T)$;
  Set $v_i = v$, put it in $V(T)$ and put uv in $E(T)$;
end
return T

The running time of this algorithm is $O(n^3)$ as the loop executes n times, and to find an edge meeting the necessary conditions we have to check at most n^2 edges.

Once we have a spanning tree T, we can 2-colour T in time $O(n)$. To see if G is bipartite, we then check that each edge is between different colour vertices. We can do this in $O(n^2)$ time. So all together, we solve $\mathsf{Col}(2)$ in time $O(n^3)$.

Now. What about $\mathsf{Col}(3)$? This is a harder problem. The best algorithm we know for $\mathsf{Col}(3)$ takes exponential time. But we cannot prove that there is not a faster algorithm, in particular we cannot prove that there is a polynomial time algorithm. We need a little more language to properly talk about this.

1.5.1 NP-completeness

Notice that in Algorithm 1.10, if we get a YES decision, we actually have the k-colouring f_i of G. If you give this to me, I can verify in time $O(n^2)$ that G is k-colourable. This f_i is called a *polynomial certificate* for $\mathsf{Col}(k)$ as it allows one to prove in polynomial time that G is a YES instance of the problem. (We do not have a corresponding certification that G is not k-colourable except if $k = 1$ or 2.) If a problem is such that YES instances have polynomial certificates, the problem is a *non-deterministic polynomial* problem. We shorten this and say the problem 'is in the complexity class NP'. If a problem has polynomial complexity, we say it is 'in the complexity class P'. We have seen that $\mathsf{Col}(k)$ is in NP for all $k \geq 1$, and that $\mathsf{Col}(k)$ is in P for $k = 1, 2$. It is clear that $\mathsf{P} \subseteq \mathsf{NP}$, that is, that any problem in P is also in NP. One of the unsolved Millenium Problems is to determine whether or not $\mathsf{P} = \mathsf{NP}$. It seems that most people feel that $\mathsf{P} \neq \mathsf{NP}$, but we cannot prove it.

If $\mathsf{P} \neq \mathsf{NP}$, then we have $\mathsf{P} \subsetneq \mathsf{NP}$. It is known that if this is true, then there are infinitely many other complexity classes $\mathsf{C}_1, \mathsf{C}_2, \ldots$ such that

$$\mathsf{P} \subsetneq \mathsf{C}_1 \subsetneq \mathsf{C}_2 \subsetneq \cdots \subsetneq \mathsf{NP}.$$

1.5.2 Polynomial Reductions

So we cannot show that $\mathsf{Col}(3)$ is not in P. The best we can do is say that if it is, then $\mathsf{P} = \mathsf{NP}$, so we feel strongly that it is not in P. So how do we show something like this? What we must do, is

show that no problem is NP is much harder than Col(3). We do so with polynomial reductions.

Definition 1.12. For decision problems P and Q, a *polynomial reduction* from P to Q is an algorithm for solving P that creates, possibly several, polynomially sized instances of Q, and whose answer can be computed from the answers to these instances of Q in polynomial time. If such a reduction exists we write $P \leq_{\mathsf{poly}} Q$. If $P \leq_{\mathsf{poly}} Q$ and $Q \leq_{\mathsf{poly}} P$ then we write $P =_{\mathsf{poly}} Q$, and say P and Q are *polynomially equivalent*.

Is this clear? Not so clear, maybe. Let's start with a super easy example showing that $\mathsf{Col}(3) \leq_{\mathsf{poly}} \mathsf{Col}(4)$.

Example 1.13. (Polynomial reduction of Col(3) to Col(4).) Assume we have an algorithm A_4 to solve Col(4). We use this to make an algorithm to solve Col(3). For an instance G of Col(3) let G' be the graph we get from G by adding a new vertex v_0 adjacent to every vertex of G. It only takes $O(n)$ time to add the vertex v_0 to G to make G', this is easily polynomial. The instance G' has size $n+1$ which is also polynomial in n. So we get that $\mathsf{Col}(3) \leq_{\mathsf{poly}} \mathsf{Col}(4)$ by showing (see Exercise 1.5.2)

$$G \to K_3 \iff G' \to K_4.$$

△

Observe that if A_4 has polynomial running time $f_4(n)$, then the running time of this algorithm for Col(3) is at most

$$f_3(n) := O(n) + f_4(n+1)$$

which is also polynomial, (see Exercise 1.5.4).

A problem $Q \in \mathsf{NP}$ is NP-*complete*, or in the complexity class NPC, if for any problem $P \in \mathsf{NP}$ we have that $P \leq_{\mathsf{poly}} Q$. The following is immediate from Exercise 1.5.4.

Fact 1.14. *If P and Q are decision problems with $P \leq_{\mathsf{poly}} Q$ then*

i. If Q is in P, then so is P.

ii. If Q is in NP, then so is P.

iii. If Q is in NP and P is in NPC, then Q is in NPC.

iv. If Q is in NPC and $P =_{\mathsf{poly}} Q$, then P is in NPC.

In 1972, Karp gave a long list of problems that are in NPC. We will talk in the next section a little bit about how he proved it, but for now we just state the following result of his.

Theorem 1.15 (Karp 1972)**.** *The problem Col(3) is in NPC.*

This is a very useful starting point. By Example 1.13 and Fact 1.14 it immediately gives us that Col(4) is in NPC. In fact with a simple adjustment to Example 1.13 you get, (see Exercise 1.5.5) that Col(k) is in NPC for all $k \geq 3$.

Exercises

1.5.1 (a) Find a certificate for the problem Connected of deciding if a graph is connected.

(b) Find a certificate for the problem of deciding if a graph is disconnected.

(c) Show that Connected is in P. (Hint: think about Algorithm 1.11.)

1.5.2 Show that $G \to K_3 \iff G' \to K_4$ in Example 1.13.

1.5.3 Prove Fact 1.14.

1.5.4 Show that if $f(n)$ is a polynomial then $g(n) = f(n+1)$ is a polynomial. Show that if $f(n)$ and $h(n)$ are both polynomials, then $g(n) = f(h(n))$ is a polynomial.

1.5.5 Show that for all $k \geq 3$, $\mathsf{Col}(k) \in \mathsf{NPC}$.

1.5.6 Show that K_3 is the core of K_3^k for any k, and conclude that $\mathsf{Hom}(K_3^k)$ is in NPC.

1.6 H-colouring and dichotomy

As it is k-colouring when $H = K_k$ the following problem, for an irreflexive graph H, is often called H-colouring. The vertices of H are called colours.

Problem. $\mathsf{Hom}(H)$ or *H-colouring.*
Instance: An irreflexive graph G.
Decision: Is there a homomorphims $G \to H$?

We saw that $\mathsf{Col}(1)$ and $\mathsf{Col}(2)$ are in P but $\mathsf{Col}(k)$ is in NPC for $k \geq 3$. But if $\mathsf{P} \neq \mathsf{NP}$ then there are all these different classes in between. It seems strange that we jump from easy to hard. Is it possible that we can find an H that is in some complexity class below NPC but above P?

Nope. Hell and Nešetřil showed the following. It is known as the H-colouring dichotomy.

Theorem 1.16 (Hell, Nešetřil, 1990)**.** *Let H be a irreflexive graph. If H is bipartite, then* $\mathsf{Hom}(H)$ *is in* P*, otherwise* $\mathsf{Hom}(H)$ *is in* NPC*.*

In Exercise 1.6.1 you show $\mathsf{Hom}(H) \in \mathsf{P}$ for bipartite H by showing

$$G \to H \iff G \to K_2.$$

We will prove most of the rest of this theorem, but it will take a while. It uses several polynomial reductions. Lets see one more that is a bit more complicated than Example 1.13. This is called an edge-gadget reduction.

1.6.1 Edge gadgets

Example 1.17. (Edge gadget reduction) We give a polynomial reduction of $\mathsf{Hom}(K_5)$ to $\mathsf{Hom}(C_5)$. As $\mathsf{Hom}(K_5) \in \mathsf{NPC}$ and $\mathsf{Hom}(C_5) \in \mathsf{NP}$ (see Exercise 1.6.3), this tells us that $\mathsf{Hom}(C_5) \in \mathsf{NPC}$.

Let G be an instance of $\mathsf{Hom}(K_5)$. In polynomial time, we construct G^* such that

$$G \to K_5 \iff G^* \to C_5.$$

Indeed, let G^* be the graph we get from G by replacing every edge $e = uv$ with a 3-path of new vertices between u and v.

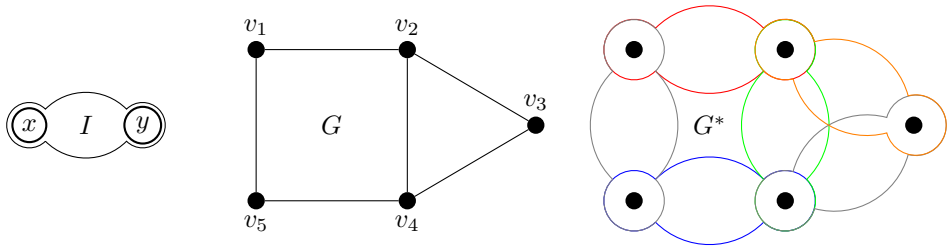
Where G had n vertices and $m \leq n^2$ edges, G^* has $n + 2m$ vertices and $3m$ edges, so its size is polynomial in n.

The main point to observe is that for any two DIFFERENT vertices x and y in C_5 there is a walk of length 3 in C_5 between x and y, but this does not hold if $x = y$.

Now, assume that $\phi : G^* \to C_5$ is a homomorphism. For any edge uv of G there is a path of length 3 between u and v in G^*, so ϕ maps this to a walk of length 3 from $\phi(u)$ to $\phi(v)$ in C_5. As there is no walk of length 3 in C_5 from a vertex to itself, $\phi(u) \neq \phi(v)$. Thus ϕ, restricted to the vertices of G, is a homomorphism from G to K_5.

On the other hand, assume that $\phi : G \to K_5$ is a homomorphism. We define a homomorphism $\phi' : G^* \to C_5$ as follows. For $u \in V(G)$ let $\phi'(u) = \phi(u)$. Now, any other vertex of G^* is in a 3-path P between two adjacent vertices u and v of $V(G)$. As ϕ is a homomorphism $\phi(u) \neq \phi(v)$ so we observed that there is a walk W of length 3 from $\phi(u)$ to $\phi(v)$ in C_5. Let ϕ' map the i^{th} vertex of P to the i^{th} vertex of W. By construction all edges of G^* are in paths between such vertices u and v of G so map to edges of C_5. Thus ϕ' is a homomorphism. △

Let's codify what we did in this example so that it is easier to do again. We had a graph G. To get G^*, we replaced every edge uv of G with a copy of an *edge gadget* I. In the example, I was a path of length 3, but in general, it is any graph $I = I(x, y)$ with two designated vertices x and y which we can identify with the vertices u and v.



From this we define a new graph ${}^*H$ on the same vertices as $V(H)$. (For us H was C_5 and ${}^*H$ was K_5.) The arcs of ${}^*H$ are the pairs (u, v) such that there is a homomorphism of $I(x, y)$ to H taking (x, y) to (u, v). With this setup we get for all instances G of $\mathsf{Hom}({}^*H)$ that:

$$G \to {}^*H \iff G^* \to H.$$

This was a polynomial reduction of $\mathsf{Hom}(H^*)$ to $\mathsf{Hom}(H)$. If we know that $\mathsf{Hom}(H^*)$ is in NPC, this puts $\mathsf{Hom}(H)$ in NPC as well.

Note

When we use this construction to show that $\mathsf{Hom}(H)$ is in NPC, we have to find a gadget I that gives that ${}^*H$ is some graph for which we know $\mathsf{Hom}({}^*H)$ is in NPC. This tends to take ingenuity and luck. (Is there luck in math?) When we are working with reflexive graphs, we have to be careful to keep I symmetric with respect to x and y so that ${}^*H$ is a symmetric graph, rather than a digraph. We also have to be careful that x and y cannot map to the same vertex of H, or ${}^*H$ will have loops, and so $\mathsf{Hom}({}^*H)$ will not be in NPC.

There are several other gadgets, variations on the edge gadget, that we need to finish the proof of Theorem 1.16. We will come back to this. First we take a look at something different.

Exercises

1.6.1 Show that for a bipartite graph H and any graph G,

$$G \to H \iff G \to K_2.$$

(You can use Exercise 1.3.2.) What does this tell you about the complexity of $\mathsf{Hom}(H)$ for bipartite H?

1.6.2 Restate Theorem 1.16 for cores: For an irrflexive graph core H, $\mathsf{Hom}(H)$ is in P if ???? and otherwise in NPC.

1.6.3 Show that $\mathsf{Hom}(H)$ is in NP for all irreflexive graphs H. (Don't show it is in NPC, this is much harder.)

1.6.4 Show that $\mathsf{Hom}(C_{2k+1})$ is in NPC for any $k \geq 5$.

2 Constraint Satisfaction Problems

We generalise the notion of digraph to relational structure, the homomorphism problem of which is called a constraint satisfaction problem (CSP). We start though with boolean satisfaction problems. Along with graphs, these are another 'easiest' version of CSP.

2.1 Boolean Satisfiability

In 1971, Cook, who first defined NPC, showed that the problem SAT, which we define below, is in NPC. We don't do this, but we show how Karp used this, in 1972, to show that Col(k) is in NPC for all $k \geq 3$.

From propositional logic, recall the boolean operators $\wedge$, $\vee$ and $\neg$ acting on the set $\{0, 1\}$. They are called 'wedge' (or 'or' or 'disjunction'), 'vee' (or 'and' or 'conjunction'), and 'negation' (or 'not') respectively. They had operation tables:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

x	$\neg x$
0	1
1	0

A *boolean formula* is a string of variables operated on by $\wedge$, $\vee$ and $\neg$. Its size, or length, is the number of occurences of the variables.

Example 2.1.

$$((x_1 \vee x_2) \wedge x_3) \vee (\neg x_1 \wedge x_3),$$

is a boolean formula of length 5. A solution to the boolean formula is an assigment of values to the variables that evaluates to 1. $(x_1, x_2, x_3) = (1, 0, 1)$ is a solution to the above example. △

Problem. SAT or *Boolean Satisfiability.*
Instance: A boolean formula X.
Decision: Does X have a solution?

Observe that $\wedge$ and $\vee$ are associative so we can write $(x_1 \vee x_2) \vee x_3$ as $(x_1 \vee x_2 \vee x_3)$, and the same can be done for $\wedge$. A *literal* is either a variable or its negation. A boolean formula consisting only of literals and disjunctions ($\wedge$) is a *(disjunctive) clause.* A formula is in *conjunctive normal form* if it is the conjunction of clauses. Like this:

$$(x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4).$$

Let CSAT be the *restriction* of SAT to instances in conjunctive normal form. This means it is the same problem, but we only consider certain instances. Clearly then CSAT $\leq_{\text{poly}}$ SAT. Cook, (and indepently Levin), showed that CSAT is also in NPC. This tends to be more useful than SAT for showing problems are in NPC. Lets do this with a further restriction.

Problem. 3SAT
Instance: A boolean formula X that is the conjuction of clauses, each of length 3.
Decision: Does X have a solution?

Two boolean formula are *equivalent* if they have the same variables, and have the same value for each assigment of values to the variables.

Theorem 2.2. *The problem* 3SAT *is in* NP.

Proof. As 3SAT is a restriction of CSAT we have $3\mathsf{SAT} \leq_{\mathsf{poly}} \mathsf{CSAT}$. We thus show that $\mathsf{CSAT} \leq_{\mathsf{poly}} 3\mathsf{SAT}$, and then it follows from Fact 1.14 that $3\mathsf{SAT} \in \mathsf{NPC}$.

Let X be an instance of CSAT. We construct, in polynomial time, an (not necessarily equivalent) instance X' of 3SAT such that X has a solution if and only if X' has a solution.

First observe that if any clause C of X has length shorter than 3, such as $(\ell_1 \vee \ell_2)$, then we can replace it with an equivalent clause of length 3 by repeating literals: $\ell_1 \vee \ell_2 \vee \ell_2)$; so we may assume that every clause of X has length at least 3. Now say a clause $C = (\ell_1 \vee \cdots \vee \ell_d)$ has length $d > 3$, then we introduce a new variable x_C and replace C with the conjuction C^* of two shorter clauses

$$(\ell_1 \vee \ell_2 \vee x_C) \wedge (\neg x_C \vee \ell_3 \vee \cdots \vee \ell_d).$$

Observe that although C is not equivalent to C^*, C has a solution if and only if C^* does. Indeed, if C as a solution then some literal ℓ_i is true, this makes one clause of C^* true, and one can chose the value of x_C to then make the other clause true. On the other hand, assume that C^* is true. Only one of x_C and $\neg x_C$ are true, and for the one that is false, there must be a literal ℓ_i in the same clause that is true. So C is true.

Now, with this same splitting technique applied recursively to C^* we eventually get to a boolean formula C' consisting of clauses of length 3 which has a solution if and only if C does. Doing this for all clauses of X we get our X'.

We just observe that everytime we split a long clause, we increase the length of the formula by 2. But we also reduce the sum of the lengths of the long clauses by 2. The sum of the lenghts of the long clauses was originally at most n, and so we make at most $n/2$ splits, and so the final length of X' is at most $2n$. This is polynomial in n, and so this is a polynomial reduction, yielding the required $\mathsf{CSAT} \leq_{\mathsf{poly}} 3\mathsf{SAT}$. □

2.1.1 Exercises

2.1.1 Show that SAT is in NP. (You cannot use the fact that it is in NPC!)

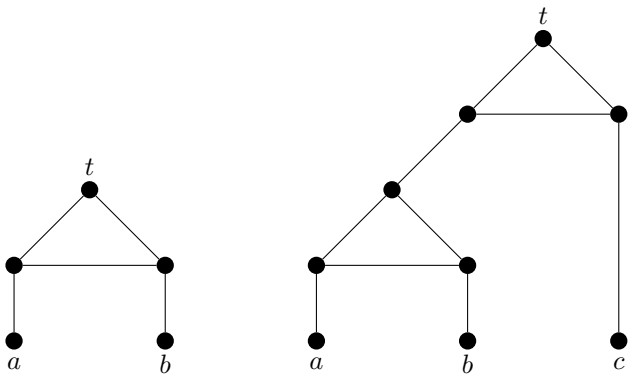
2.1.2 Put the boolean formula from Example 2.1 into conjuntive normal form.

2.1.3 Show that any boolean formula is equivalent to a formula in conjunctive normal form.

2.1.4 A boolean formula is in Disjunctive Normal Form (DNF) if it is the disjunction of conjunctive clauses. Show that the restriction of SAT to DNF instances is in P.

2.1.5 (*) Reduce 3SAT to Col(3) to show that Col(3) is in NPC.

Hint: Consider the what happens when the following graphs are mapped to K_3 (on vertex set $\{0, 1, 1'\}$.)



For the first one, observe that if ϕ is a homomorphism to K_3 with $\phi(a) = \phi(b) = 0$, then $\phi(t) = 0$; so if $\phi(t) = 1$ or $1'$ then at least one of $\phi(a)$ and $\phi(b)$ is not 0. From this you should be able to argue that if ϕ is a homomorphism of the second one to K_3 and $\phi(t) = 1$, then at least one of $\phi(a), \phi(b)$ and $\phi(c)$ is not 0.

2.2 Relational Structures and CSP

In this section we define relational structures, and show how they are a generalisation of digraphs. We define their homomorphism problem as CSP and see how these can be used to define many subproblems of SAT.

2.2.1 Relational Structures

A *k-ary relation* on a set V is a subset $R \subset V^d$. A *relational structure* $\mathcal{G} = (V, \mathcal{R})$ is simply a set $\mathcal{R}$ of relations on a set of vertices V. However to talk about them in a significant way, we need a couple more definitions.

A *signature* is a set of relation symbols such as A, E, R, R_i. Each relational symbol R has an *arity* $\text{arity}(R) \in \mathbb{N}$. A τ-structure, or a *relational structure of signature* τ is a relational structure $\mathcal{G}$ on a vertex set V with a relation $R^{\mathcal{G}}$ of arity $\text{arity}(R)$ for every relation symbol in τ.

Example 2.3. Here are some examples of structure you might have seen in a graph class that can be represented as relational structures.

- A digaph $\mathcal{G} = (V, \{A^{\mathcal{G}}\})$ is a τ-structure where $\tau = \{A\}$ has one relation A of arity 2.
- A relational structure $\mathcal{G} = (V, \{T^{\mathcal{G}}\})$ where T has arity 3 is a 3-uniform directed hypergraph. If the relation $T^{\mathcal{G}}$ is symmetric, then we have our usual hypergraph.
- If $\tau = \{R, B\}$ consists of two symbols of arity 2, then a τ-structure is a digraph with red and blue edges.

△

Yep. This notation seems like over-kill. We will usually write just A instead of $A^{\mathcal{G}}$. But we need to have such notation as we will be talking of homomorphisms between structures, and there has to be language for this. Speaking of which...

A homomorphism $\phi : \mathcal{G} \to \mathcal{H}$ of τ-structures $\mathcal{G}$ and $\mathcal{H}$ is a function $\phi : V(\mathcal{G}) \to V(\mathcal{H})$ that preserves all relations: for each $R \in \tau$, where $k = \text{arity}(R)$

$$(v_1, \ldots, v_k) \in R^{\mathcal{G}} \Rightarrow (\phi(v_1), \ldots, \phi(v_k)) \in R^{\mathcal{H}}.$$

Example 2.4. An instance of SAT can be seen as deciding if there is a homomorphism between two relational structures. Indeed, let X be a boolean formula in conjuctive normal form. We define τ-structures $\mathcal{G}$ and $\mathcal{H}$ such that $\mathcal{G} \to \mathcal{H}$ if and only if X has a solution.

- Let $V(\mathcal{G})$ be the set of literals in X. (So for each variable x_i in X there are up to two vertices, x_i and $\neg x_i$ in $V(\mathcal{G})$). For each clause C of size d, make a relation R_C of arity d containing a d tuple of the literals in C (in any order). Make one more 2-ary relation N and let $N^{\mathcal{G}}$ contain $(x_i, \neg x_i)$ for any variable x_i for which both these literals appear in X.
- Let $\mathcal{H}$ have vertex set $\{0, 1\}$. For each clause C, where $k = \text{arity}(C)$, let $R_c^{\mathcal{H}} = \{0, 1\}^k \setminus \{(0, 0, \ldots, 0)\}$. Let $N^{\mathcal{H}} = \{(0, 1), (1, 0)\}$

Clearly a mapping of $V(\mathcal{G}) \to V(\mathcal{H})$ is an assigment of values 0 and 1 to the variables of X, as long as x_i and $\neg x_i$ go to opposite values. This is what the relation N ensures. The mapping preserves R_c if and only if it takes the tuple of literals in C to a tuple that is not $(0, 0, \ldots, 0)$, and this is exactly what we need to ensure that C is satisfied. So there is a homomorphism $V(\mathcal{G}) \to V(\mathcal{H})$ if and only if X has a solution. △

The problem of deciding if there is a homomorphism $\mathcal{G} \to \mathcal{H}$ between two τ-structures is the problem of deciding if there is an assignment of values from $V(\mathcal{H})$ to variables in $V(\mathcal{G})$ such that certain constraints – that tuples in $R^{\mathcal{G}}$ map to tuples in $R^{\mathcal{H}}$ – are satisfied. So such a problem is called a *constraint satisfaction problem.*

Problem. CSP
Instance: A pair $(\mathcal{G}, \mathcal{H})$ of τ-structures for some signature τ.
Decision: Is there a homomorphism $\mathcal{G} \to \mathcal{H}$?

In the exercises you show that CSP is in NPC. As we do with graph homomorphisms, though, we can ask the complexity of many naturally occuring subproblems– the so-called right-restricted CSPs.

Problem. CSP($\mathcal{H}$) for a relational structure $\mathcal{H}$
Instance: A structure $\mathcal{G}$ of the same signature as $\mathcal{H}$.
Decision: Is there a homomorphism $\mathcal{G} \to \mathcal{H}$?

2.2.2 Examples of CSP($\mathcal{H}$)

Example 2.5. (3SAT) Let $\mathcal{B}_3$ be the structure with $B = V(\mathcal{B}_3) = \{0, 1\}$ and for each $x \in B^3$ let R_x be the 3-ary relation

$$R_x = B^3 \setminus \{x\}.$$

The signature τ consists of eight 3-ary relations.

We show that $\mathsf{CSP}(\mathcal{B}_3)$ is (polynomially equivalent to) $\mathsf{3SAT}$.

For an instance X of $\mathsf{3SAT}$ construct a τ-structure $\mathcal{G}_X$ on the set of variables of X. For a literal ℓ of X let $x(\ell)$ be the variable x_i in ℓ and let

$$p(\ell) = \begin{cases} 0 & \text{if } \ell = \neg x(\ell) \\ 1 & \text{if } \ell = x(\ell). \end{cases}$$

For each clause (ℓ_1, ℓ_2, ℓ_3) of X, put the tuple $(x(\ell_1), x(\ell_2), x(\ell_3))$ in $R_{(p(\ell_1),p(\ell_2),p(\ell_3))}$. That this relation is preserved by a homomorphism $\phi : \mathcal{G}_X \to \mathcal{B}_3$ ensures that the clause gets a value of 1. So if there is such a homomorphism, then X has a solution. On the other hand, a solution of X defines such a homomorphism.

On the other hand, for an instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{B}_3)$ (that it, a τ-structure $\mathcal{G}$) let X be the boolean formula whose variables are the vertices of $\mathcal{G}$, that we get from the conjunction of clauses (ℓ_1, ℓ_2, ℓ_3) for each $(x(\ell_1), x(\ell_2), x(\ell_3)) \in R_{(p(\ell_1),p(\ell_2),p(\ell_3))}$ over all relations R_x. △

How about some graph examples. In a graph class, after you talk about k-colouring and generalisations such as circular and fractional colouring, you also talk about the variation known as list k-colouring.

Example 2.6. (List-homomorphism) A *graph with lists* is graph G and a function $L : V(G) \to 2^{[n]}$ assigning a set of viable colours to each vertex. A list-colouring of a graph G with lists L is an n-colouring $f : V(G) \to [n]$ such that for each vertex $v \in V(G)$ we have that $f(v) \in L(v)$. How do we represent a list-colouring as a homomorphism?

Let τ be a signature consisting of one 2-ary symbol E, and a 1-ary symbol L_s for each subset $S \subseteq [n]$. Let $K = K_n^u$ be the τ-structure on $[n]$ with $E^K = E(K_n)$ and $L_S^K = \{(s) \mid s \in S\}$. For a graph G with lists L, make a τ-structure $\mathcal{G}$ on the same vertex set $V(G)$ with $E^{\mathcal{G}} = E(G)$, and $L_S^{\mathcal{G}} = \{(v) \mid L(v) = S\}$.

We claim that there is a homomophimsm $\mathcal{G} \to K_n^u$ if and only if there is a list-colouring of G with lists L.

Indeed let $\phi : \mathcal{G} \to K_n^u$ be a homomorphism, and apply it to $V(G)$. As it preserves the relation E it is an n-colouring of G. Let $v \in V(G)$ have list $S := L(v)$. Then in $\mathcal{G}$, the tuple (v) is in the relation $L_S^{\mathcal{G}}$ so ϕ must map it to a tuple $(s) \in L_S^K$. But $L_S^K\{(s) \mid s \in S\}$, so $s \in S$. Thus $\phi(v) = s \in S = L(v)$ as needed.

I leave the other direction to you in the exercises. △

Definition 2.7. For a structure $\mathcal{H}$, $\mathsf{ListHom}(\mathcal{H}) := \mathsf{CSP}(\mathcal{H}^u)$ where $\mathcal{H}^u$ is the structure we get from $\mathcal{H}$ by adding a unary relation $U_S = S$ for each subset $S \subset V(\mathcal{H})$.

Another colouring problem that is often considered is when we pre-colour some of the vertices of G and ask if there is a k-colouring of G that agrees with the pre-colouring on these pre-coloured vertices. Like list colouring, these can be useful in inductive proofs. They also can be describes as a homomorpism problem of relational structures.

Example 2.8. (Pre-colour extension) A *pre-k-coloured* graph G is a graph with a function $p : S \to [k]$ for some subset $S \subseteq V(G)$ of the vertices of G. A k-colouring f *extending* p is a k-colouring f of G such that the restriction $f|_S$ of f to S equals p. How do we represent a pre-k-colouring extension as a homomorphism?

Let τ be the signature consisting of the 2-ary symbol E and a 1-ary symbol P_v for each vertex $v \in [k]$. Let $K = K_k^s$ be the τ-structure on $[k]$ with $E^K = E(K_k)$ and $P_i^K = \{v\}$ for all $i \in [k]$. For pre-k-coloured graph (G, p), let G' be the τ-structure with $E^{G'} = E(G)$ and $P_i^{G'} = p^{-1}(i)$ for each $i \in [k]$.

It is an exercise to show that there is an k-colouring of G extending p if and only if $G' \to K_k^s$. △

Definition 2.9. For a structure $\mathcal{H}$, $\mathsf{Ext}(\mathcal{H}) := \mathsf{CSP}(\mathcal{H}^s)$ where $\mathcal{H}^s$ is the structure we get from $\mathcal{H}$ by adding a singleton unary relation $U_x = \{x\}S$ for each $x \in V(\mathcal{H})$.

Lemma 2.10. *If $\mathcal{H}$ is a core, then $\mathsf{CSP}(\mathcal{H}) =_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H}^S)$.*

Proof. As $\mathsf{CSP}(\mathcal{H})$ is a subproblem of $\mathsf{CSP}(\mathcal{H}^s)$ we have $\mathsf{CSP}(\mathcal{H}) \leq_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H}^s)$ so it is enough to get a polynomial reduction of $\mathsf{CSP}(\mathcal{H}^s)$ to $\mathsf{CSP}(\mathcal{H})$. For an instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{H}^s)$ let ${}^*\mathcal{G}$ be the instance of $\mathsf{CSP}(\mathcal{H})$ we get from $\mathcal{G}$ as follows.

- Take a copy G of $\mathcal{G}$ and a disjoint copy H of $\mathcal{H}$.
- For every unary relation U_x in the type of $\mathcal{H}^s$ but not in the type of $\mathcal{H}$, remove U_x from G, and for every $g \in U_x^{\mathcal{G}}$, identify the vertex g of G with the vertex x of H.

We claim that ${}^*\mathcal{G} \to \mathcal{H} \iff \mathcal{G} \to \mathcal{H}^s$. Indeed let $\phi : {}^*\mathcal{G} \to \mathcal{H}$ be a homomorphism. If the restriction $\phi|_H : H \to \mathcal{H}$ of ϕ to H is not the identity, then because $\mathcal{H}$ is a core, it is an automorphism σ. By composing with σ^{-1} we may assume that ϕ restricts to the identity on H. We claim that ϕ restricts on G to a homomorphism of $\mathcal{G}$ to $\mathcal{H}^s$. As it is a homomorphism it preserves all relations in ${}^*\mathcal{G}$ so we only have to observe that it preserves the unary relations U_x not in ${}^*\mathcal{G}$. Let $g \in \mathcal{G}^{U_x}$. Then g was identified with x in H and ϕ takes this to x, so $\phi(g) \in \{(x)\} = \mathcal{H}^{U_x}$ as needed.

On the other hand, let $\phi : \mathcal{G} \to \mathcal{H}^s$ be a homomorphism. Applying ϕ to the copy of G in ${}^*\mathcal{G}$ we get a homomorphism $\phi_G : G \to \mathcal{H}^s$ and so $\phi_G : G \to \mathcal{H}$. On the copy H of $\mathcal{H}$ in ${}^*\mathcal{G}$, let ϕ_H be the identity homomorphism $\mathrm{id} : H \to \mathcal{H} : x \mapsto x$. To verify that ϕ_G and ϕ_H define a homomorphism of ${}^*\mathcal{G}$ to $\mathcal{H}$ we just have to verify that they agree on vertices in G and H. But a vertex g of G was identified with a vertex x of H only if $g \in \mathcal{G}^{U_x}$, and in this case we have that ϕ preserves the relation U_x so

$$\phi_G(g) = \phi(g) = x = \phi_H(x),$$

as needed. □

We can again ask about the complexity of $\mathsf{CSP}(\mathcal{H})$, and in particular, ask if there is a dichotomy like there is for H-colouring. This CSP-dichotomy was conjectured by Feder and Vardi in 1998, likely based on the H-colouring dichotomy of Hell and Nešetřil and a dichotomy for boolean structures ($\mathcal{H}$ having two vertices) by Schaeffer in 1978. Feder and Vardi showed that is was enough to prove

it for digraphs or reflexive graphs. Bulatov proved a list version of it (when $\mathcal{H} = \mathcal{H}^u$) in 2003, and proved it for ternary structures ($\mathcal{H}$ having three vertices) in 2006. Barto Kozik and Niven proved it for digraphs with no sinks or sources in 2011, and many other partial results appeared until it was proved by Bulatov and, independently, Zhuk, in 2017.

Theorem 2.11 (Bulatov '17; Zhuk '17)**.** *For any relational structure $\mathcal{H}$,* $\mathsf{CSP}(\mathcal{H})$ *is either in* P *or in* NPC.

We also know which $\mathcal{H}$ have $\mathsf{CSP}(\mathcal{H})$ in P, but to state this we talk about polymorphisms.

2.2.3 Exercises

Many of these exercises just notation. The the notation is messy, but once you get a handle on it, the problems are easy. Have patience with them, and do enough that you are comfortable with the notation. After that, just read the others and be sure you understand what they are saying.

2.2.1 Finish the proof in Example 2.6.

2.2.2 In Example 2.8 show that there is an k-colouring of G extending p if and only if $G' \to K_k^s$.

2.2.3 Generalise Definition 1.7 to define a product of two τ-structures.

2.2.4 Show that CSP is in NPC. (Hint: Argue that it is in NP, and then observe there are restrictions of the problem that are already in NPC.)

2.2.5 Show that that any instance of $\mathsf{CSP}(\mathcal{H}^s)$ is an instance of $\mathsf{CSP}(\mathcal{H}^u)$, and conclude $\mathsf{CSP}(\mathcal{H}^s) \leq_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H}^u)$.

2.2.6 Show for any relational structure $\mathcal{H}$ show that $\mathcal{H}^s$ is a core. (Use the obvious extension of the definition of a core to relational structures, or of the alternate definition you get from Exercise 1.3.13.)

2.2.7 The left-restricted CSP for a structure $\mathcal{G}$ has instances $\mathcal{H}$ of the same signature and asks if there is a homomorphism $\mathcal{G} \to \mathcal{H}$? (So $\mathcal{G}$ is fixed rather than $\mathcal{H}$.) Show that for any $\mathcal{G}$, this problem is in P. (Hint: How many possible mappings are there from $V(\mathcal{G})$ to $V(\mathcal{H})$.)

2.2.8 Let $\vec{P}_n$ be the irreflexive directed path on $\{0, 1, \ldots, n\}$ with arcs $(i, i+1)$ for all i. Show that $\mathsf{CSP}(\vec{P}_n)$ is in P. (Hint: One way to do it would be to use a greedy algorithm. Another way would be to use a *duality*: show that for any digraph H, $H \to \vec{P}_n \iff \vec{P}_{n+1} \not\to H$; then use Exercise 2.2.7.)

2.2.9 Let T_n be the *transitive tournament* on $[n]$: (i, j) is an arc if an only if $i < j$. Show that $\mathsf{CSP}(T_n)$ is in P, (Hint: Is there a duality for this?)

2.2.10 Recall that $\mathsf{Hom}(H)$ is trivial for a digraph H that has at least one loop— all instances are YES instances. Show that the pre-coloured version of this problem, $\mathsf{Ext}(H) = \mathsf{CSP}(H^s)$ is not trivial for most reflexive graphs H.

2.2.11 For a digraph H, let $\mathsf{Ret}(H)$ be the problem whose instances are digraphs G containing H as an induced subgraph. The decision is to decide if there is a retraction of G to H. Show that $\mathsf{Ret}(H)$ is polynomially equivalent to $\mathsf{Ext}(H)$.

2.3 Polymorphisms

All graphs and relational structures have polymorphisms. But life isn't fair — some have few, and some have many. To over-simplify the compexity classification of CSP problems, structures with few polymorphisms have have hard problems, and structures with many polymorphisms, have easy problems. With a few ideas we show the relationship between a polymorphic paucity and a hard life, we then charge ahead, and state a CSP dichotomy classification. We then go on to describe some notions of clones.

A *polymorphism* of a structure $\mathcal{H}$ is a homomorpism $\mathcal{H}^k \to \mathcal{H}$, where $\mathcal{H}$ is the k time product

$$\mathcal{H} \times \mathcal{H} \times \cdots \times \mathcal{H}$$

for the product you cleverly defined in Exercise 2.2.3. Equivalently, a function $\phi : V(\mathcal{H})^k \to V(\mathcal{H})$ is a (k-ary) polymorphism of the τ-structure $\mathcal{H}$ if, for every $R \in \tau$, where $d = \text{arity}(R)$,

$$\forall i \in [k],\ \vec{a_i} := \begin{bmatrix} a_i^1 \\ a_i^2 \\ \vdots \\ a_i^d \end{bmatrix} \in R \qquad \text{implies} \qquad \phi(\vec{a_1}, \vec{a_2}, \ldots, \vec{a_k}) := \begin{bmatrix} \phi(a_1^1, a_2^1, \ldots, a_k^1) \\ \phi(a_1^2, a_2^2, \ldots, a_k^2) \\ \vdots \\ \phi(a_1^d, a_2^d, \ldots, a_k^d) \end{bmatrix} \in R.$$

(We've done the appalling here and introduced notation as using it. Vertical tuples are denoted with an arrow on top, and a polymorphism applied to a horizontal tuple of them acts row-wise.)

In the exercises you observed for a product of graphs that the project onto each co-ordinate is a homomorphism. So these are polymophisms $\pi_i : G^2 \to G$. This holds more generally.

Fact 2.12. *Let $\mathcal{H}$ be a structure and $k \geq 1$. For each $i \in [k]$ the projection $\pi_i : (x_1, \ldots, x_k) \mapsto x_i$ is a polymorphism of $\mathcal{H}$.*

2.3.1 Few polymorphisms gives a hard CSP

A polymorphism $\phi : \mathcal{H}^k \to \mathcal{H}$ is *idempotent* if $\phi(v, v \ldots, v) = v$ for all $v \in V(\mathcal{H})$. The projections are clearly idempotent. A structure $\mathcal{H}$ is *projective* if its only idempotent polymorphisms, of any arity, are projections.

In the exercises you are asked to show that K_3 is projective, but that K_2 is not. It was shown by Jeavons that for all projective structures $\mathcal{H}$, $\mathsf{CSP}(\mathcal{H}) \in \mathsf{NPC}$. (This was really cool, because it was shown by Lucsak and Nešetřil that 'almost all' (this has a real probablistic meaning) graphs are projective.

Let's show this by reducing $\mathsf{Hom}(K_3)$ to $\mathsf{CSP}(\mathcal{H}^s)$ for any projective structure $\mathcal{H}$. In Exercise 2.3.1 you show that projective structures are cores, and for cores we have $\mathsf{CSP}(\mathcal{H}^s) =_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H})$ from Lemma 2.10. So this shows that $\mathsf{Hom}(K_3) \leq_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H})$, which puts $\mathsf{CSP}(\mathcal{H})$ in NPC. The reason we use $\mathcal{H}^s$ is that, as you show in Exercise 2.3.2, its only polymorphisms are projections.

Example 2.13. Let $\mathcal{H}$ be a projective structure. We show reduce $\mathsf{Hom}(K_3)$ to $\mathsf{CSP}(\mathcal{H}^s)$. First we will assume that $\mathcal{H}$ has at least three vertices, which we may call $1, 2$ and 3. Let G be an

instance of $\mathsf{Hom}(K_3)$. We define an instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{H}^s)$ using the edge gadget $I = (\mathcal{H}^s)^6$ with designated vertices $x = (1,1,2,2,3,3)$ and $y = (2,3,1,3,1,2)$. As the only polymorphisms of $\mathcal{H}^s$ are projections, we have, by the edge gadget construction, that $(\mathcal{H}^s)^*$ is K_3 (with other vertices of $V(\mathcal{H})$ incident to no edges). Thus $\mathsf{Hom}(K_3) \leq_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H})$ as needed. Okay. If $\mathcal{H}$ only has two vertices 0 and 1 we have to make a little adjustment to the edge gadget. This is called the fibre construction, (but not even its authors agree on why!) Let G be an instance of $\mathsf{Hom}(K_3)$. We define an instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{H}^s)$ such that

$$G \to K_3 \iff \mathcal{G} \to \mathcal{H}^s.$$

For each vertex v of G let $\mathcal{G}$ have two vertices v_1 and v_2. Now, let $I = (\mathcal{H}^s)^6$ and designate four vertices of I:

$$x_1 = (1,1,0,0,1,1) \qquad y_1 = (1,0,1,1,1,0)$$
$$x_2 = (0,0,1,1,1,1) \qquad y_2 = (1,1,0,1,0,1)$$

Observe that under the six projections, the pair (x_1, x_2) maps to $(1,0)$ or $(0,1)$ or $(1,1)$ binary representations of the colours $1, 2$ and 3. The pair (y_1, y_2) map to the same colours, but never to the same one as (x_1, x_2) does. In fact these six projections map the pairs (x_1, x_2) and (y_1, y_2) exactly the way the endpoints of an edge map to K_3.

For every edge $e = u, v$ of G take a copy I_A of I and identify its copies of x_1, x_2, y_1, and y_2 with the vertices u_1, u_2, v_1 and v_2 respectively of $\mathcal{G}$, Doing this for all edges we have $\mathcal{G}$. Under a mapping $\phi : \mathcal{G} \to \mathcal{H}^s$, we have that for every edge $e = u, v$ of G, ϕ restricted to I_e maps $(x_1, x_2) = (u_1, u_2)$ and $(y_1, y_2) = (v_1, v_2)$ to different colours in our set of three colours. This yields a 3-colouring of G as needed.

I leave the other direction to you.

△

With a couple easy generalisations of this construction, (and using Theorem 2.17), we can reduce $\mathsf{CSP}(\mathcal{H})$ to 3-colouring for any structure $\mathcal{H}$ for which $\mathsf{CSP}(\mathcal{H})$ is in NPC. This is messy though, so is better done after we have seen some algebra.

Before we get there though, lets see how $\mathcal{H}$ having lots of polymorphisms, or more accurately, certain nice polymorphism, can gives a polynomial time algorithm for solving $\mathsf{CSP}(\mathcal{H})$.

2.3.2 A nice polymorphism gives an easy CSP

Feder and Vardi showed that the existence of a majority polymorphism on $\mathcal{H}$ means that $\mathsf{CSP}(\mathcal{H})$ can be solved by path-consistancy.

Definition 2.14. A 3-ary polymorphism $\phi : \mathcal{H}^3 \to \mathcal{H}$ is a *majority* polymorphism if it for all choices of $x, y \in V(\mathcal{H})$ we have

$$\phi(x,x,y) = \phi(x,y,x) = \phi(y,x,x) = x.$$

We will show how a majority function on a digraph H can be used to give a polynomial time algorithm for finding a homomorphism from G to H if it exists. We break this up into two parts. The first is an algorithm that will decide for us if there is a homomorphism $G \to H$. The second is a proof that if H has a majority function, then the algorithm works. The proof can be converted into an algorithm to find a majority function.

Algorithm 2.15. *(Path-consistancy)*

For a digraphs G and H this algorithm returns, for each arc (uv) of G a list $L(uv)$ of possible images of uv under homomorpism of G to H. If any list is empty, there can be no homomorphism. If all lists contain images, there MAY be a homomorphism.

Input: *Connected digraphs G and H.*
Result: *List $L(x,y) \subset V(H)^2$ for each $x, y \in V(G)$.*
foreach $(x,y) \in V(G)^2$ **do**
  if $x \sim y$ **then** $L(x,y) := E(H)$ **else** $L(x,y) = V(H)^2$*;*
  if $x = y$ **then** $L(x,x) := L(x,x) \cap \{(u,u) | u \in V(H)\}$*;*
end
while *Changes are made* **do**
  foreach $x, y, z \in V(G)$ **do**
    foreach $(u,w) \in L(x,z)$ **do**
      if $\nexists v \in V(H)$ *s.t.* $(u,v) \in L(x,y)$ *and* $(v,w) \in L(y,z)$. **then**
        Remove (u,w) from $L(x,z)$ and **if** $L(x,z) = \emptyset$ **then** *EXIT;*
      end
    end
  end
end

Now if this algorithm returns an empty list, then there can be no homomorphism $G \to H$. If it finishes with all list not empty, it does not necessarily mean there is a homomorphism, but in the case that H has a majority polymorphism, it does mean this.

Theorem 2.16. *Let H be a digraph with a majority polymorphism ϕ. If Algorithm 2.15 returns a set of non-empty lists for a graph G, then there is a homomorphism $G \to H$.*

This proof is beautiful in its utility/length ratio, but a bit tricky. You are welcome to skip it.

Proof. A homomorphism $h : G' \to H$ of an induced subgraph of G *preserves list* if for each $x, z \in V(G')$ $(h(x), h(z)) \in L(x,z)$. We will show the following by induction on i: 'Any list-preserving homomorphism on a subgraph of G induced by i vertices can be extended to a list-preserving on the subgraph of G induced by these vertices and any one more vertex'.

That this is true for $i = 2$ is immediate from the path consistancy of the lists. That it is true for $n - 1$ will tell us that there is a homomorphism of G to H.

Assume that k is in $\{3, \ldots, n-1\}$ and that the statement holds for all values of $i < k$. We show that it holds also for $i = k$. Indeed let $s : S \to H$ be a list preserving homomorphism for some induced subgraph S of G having k vertices, and let c be some vertex of $V(G) \setminus V(S)$. We must extend s to a list preserving homomorphism of the subgraph S^c of G induced by $V(S) \cup \{c\}$, to H.

Let x_1, x_2 and x_3 be vertices of S. For each $i \in [3]$, the restriction s_i of s to $S_i = S \setminus x_i$ is a list-preserving homomorphism, and so by induction on the statement extends to a list-preserving homomorphism s_i^c of the subgraph S_i^c of G induced by $V(S_i) \cup \{c\}$, to H. We claim that $h : S^c \to H : x \mapsto f(s_1^c(x), s_2^c(x), s_3^c(x))$ is a list preserving homomorphism.

If it preserves lists, then it is a homomorphism, so we check for x, z in $V(S^c)$ that $(h(x), h(z)) \in L(x, z)$. For $x \in V(S)$ (ie $x \neq c$), we have that

$$h(x) = f(s_1^c(x), s_2^c(x), s_3^c(x)) = f(s_1(x), s_2(x), s_3(x)) = s(x)$$

as at least two of the $s_i(x)$ are the same as s on x. So if x and z are both in S we have that $(h(x), h(z)) \in L(x, z)$, because s preserved lists. So it is enough to show that $(h(x), h(c)) \in L(x, c)$, for $x \in V(S)$.

Well $(h(x), h(c)) = ((f(s_1^c(x), s_2^c(x), s_3^c(x)), f(s_1^c(c), s_2^c(c), s_3^c(c)))$ and each of the s_i^c preserve lists, so it is enough to observe that f preserves lists: if $(u_1, w_1), (u_2, w_2), (u_3, w_3) \in L(x, z)$ then $(f(u_1, u_2, u_3), f(w_1, w_2, w_3)) \in L(x, z)$. We do this by observing that because f is a polymorphism, it does so for the initial lists in Algorithm 2.15, and that at each step, as (u_i, w_i) stays in $L(x, z)$, for each y there is some v_i such that $(u_i, v_i) \in L(x, y)$ and $(v_i, w_i) \in L(y, z)$. So

$$(f(u_1, u_2, u_3), f(v_1, v_2, v_3)) \in L(x, y) \text{ and } (f(v_1, v_2, v_3), f(w_1, w_2, w_3)) \in L(x, y)$$

and so $(f(v_1, v_2, v_3), f(w_1, w_2, w_3))$ is not removed from $L(x, z)$. □

2.3.3 A fuller statement of the CSP dichotomy

We have seen that the existence of nice polymorphisms on $\mathcal{H}$ can give a polynomial time algorithm for $\mathsf{CSP}(\mathcal{H})$, while if $\mathcal{H}$ has only the necessary polymorphisms, $\mathsf{CSP}(\mathcal{H})$ is in NPC. This is far from solving the problem of which structures yields CSP problems in NPC, but hopefully it makes it believable that the dichotomy of complexity classes can be defined in terms of polymorphisms.

Generalising majority polymorphisms, a k-ary *near-unanimity* (k-NU) polymorphism of a structure $\mathcal{H}$ is an idempotent k-ary polymorphism $\phi : \mathcal{H}^k \to \mathcal{H}$ such that for all choices of $x, y \in V(\mathcal{H})$ we have

$$\phi(x, x, \ldots, x, x, y) = \phi(x, x, \ldots, x, y, x) = \cdots = \phi(y, x, \ldots, x, x, x) = x.$$

Feder and Vardi showed that if $\mathcal{H}$ has a k-NU polymorphism then $\mathsf{CSP}(\mathcal{H})$ is in P, being solved by a generalistion of the above path-consitancy algorirm.

A *weak near-unanimity* (k-WNU)) polymorphism of a structure $\mathcal{H}$ is an idempotent k-ary polymorphism $\phi : \mathcal{H}^k \to \mathcal{H}$ such that for all choices of $x, y \in V(\mathcal{H})$ we have

$$\phi(x, x, \ldots, x, x, y) = \phi(x, x, \ldots, x, y, x) = \cdots = \phi(y, x, \ldots, x, x, x).$$

What Bulatov and Zhuk proved is the following.

Theorem 2.17 (Bulatov '17; Zhuk '17)**.** *For any relational structure $\mathcal{H}$, if $\mathcal{H}$ admits a* WNU *polymorphism, then* $\mathsf{CSP}(\mathcal{H})$ *is in* P*, otherwise, it is in* NPC*.*

2.3.4 Exercises

2.3.1 Show that projective structures are cores. Show that the converse need not be true.

2.3.2 Show that the only polymorphisms of $\mathcal{H}^s$ for any structure $\mathcal{H}$ are idempotent. So if $\mathcal{H}$ is projective the only polymorphisms of $\mathcal{H}^s$ are projections.

2.3.3 Show that the only idempotent 2-ary polymorphisms on irreflexive symmetric triangle K_3 are projections. (Harder.) Show that K_3 is projective.

2.3.4 Show that K_2 is not projective.

2.3.5 Show that on the reflexive n-path P_n^r, the min-but-one function

$$\phi(x_1, x_2, x_3) = z_2$$

where $\{x_1, x_2, x_3\} = \{z_1, z_2, z_3\}$ as multisets, but $z_1 \leq z_2 \leq z_3$, is a majority polymorphism.

Note 2.1

Now, from polymorphisms there are two aspects we need to properly algebracise the notion of a gadget reduction for a structure $\mathcal{H}$. In model theory, one considers the set $\langle\mathcal{H}\rangle$ of all relations we can 'make' from $\mathcal{H}$ using pp-definitions (i.e., gadgets), and if any of these have hard CSP problems then so did the original structure. In universal algebra we can consider, as an algebra, the set $\mathrm{Pol}(\mathcal{H})$ of all polymorphisms that preserve the relations of $\mathcal{H}$. It turns out that $\mathrm{Pol}(\mathcal{H})$ and $\langle\mathcal{H}\rangle$ are dual, in that $\langle\mathcal{H}\rangle$ is exactly the set of relations on $V(\mathcal{H})$ that is preserved by all operations in $\mathrm{Pol}(\mathcal{H})$. We will see that the complexity of $\mathsf{CSP}(\mathcal{H})$ depends only on $\mathrm{Pol}(\mathcal{H})$.
There are certain algebraic operators that preserve the existence of nice polymorphims, and so if an algebra derived by such operators has no nice polymorphisms, neither does $\mathrm{Pol}(\mathcal{H})$. These two approached give nice language to talk about gadget constructions, and certain observations live much more naturally in one language than the other. My goal was to talk about both, but we do not really have the time now. So I will finish by talking only of the model theory point of view.

3 Positive Primitive Formulas

We try to describe some ideas from Model Theory, (or Predicate Logic), that allow us to mimic, and better codify, gadget constructions.

3.1 Definitions

For a signature τ, a *(first order) primitive positive* τ-formula (or pp-formula) is a formula $\phi(x_1, \ldots, x_n)$ with n free variables, of the form

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

where the ψ_i are formulas of the form $R(y_1, \ldots, y_d)$ for some d-ary relation $R \in \tau$ and for $y_i \in \{x_1, \ldots, x_\ell\}$. The formulas $R(y_1, \ldots, y_d)$ are called *atomic formulas*. A pp-formula ϕ with no variables (i.e. with $n = 0$) is a *sentence*.

For a τ-structure $\mathcal{G}$ there is a canonical τ-sentence $\phi(\mathcal{G})$ whose variables are the vertices of $\mathcal{H}$ and whose atomic formulas are $R(y_1, \ldots, y_d)$ for every tuple $(y_1, \ldots, y_d) \in R$ for every relation $R \in \tau$.

Example 3.1. The irreflexive symmetric digraph C_5, having signature $\tau = (E)$ has canonical sentence $\phi(C_5)$ of the form

$$\exists x_1, x_2, x_3, x_4, x_5 \left(E(x_1, x_2) \wedge E(x_2, x_1) \wedge E(x_2, x_3) \wedge \cdots \wedge E(x_1, x_5) \wedge E(x_5, x_1)\right).$$

△

A τ-structure $\mathcal{H}$ *satisfies* or *models* a sentence ϕ, written $\mathcal{H} \models \phi$ if there is a solution to ϕ in $\mathcal{H}$.

Example 3.2. Viewing K_3 as the (E)-structure (irreflexive symmetric digraph) with vertices $[3]$ and $K_3^E = \{(1,2),(2,1),(1,2),(3,1),(2,3),(3,2)\}$, we can ask if $K_3 \models \phi(C_5)$. What would this mean? We have to assign value in K_3 to the variables $x_1, \ldots, x_5$ in $\phi(C_5)$ such that each atomic formula in $\phi(C_5)$ is true. Hey, Donkey.. this is just a 3-colouring of C_5 — a homomorphism of C_5 to K_3. △

More generally, for a τ-structure $\mathcal{H}$, the problem $\mathsf{CSP}_{\mathrm{pp}}(\mathcal{H})$ is the decision problem of deciding for a given instance τ-sentence ϕ, if $\mathcal{H} \models \phi$. An instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{H})$ is decided by the instance $\phi(\mathcal{G})$ of $\mathsf{CSP}_{\mathrm{pp}}(\mathcal{H})$. On the other hand, it is not hard to see that for an instance ϕ of $\mathsf{CSP}_{\mathrm{pp}}(\mathcal{H})$ there is an instance $\mathcal{G}$ of $\mathsf{CSP}(\mathcal{H})$ that decides if for us. So we have the following, which allows us to view an instance of $\mathsf{CSP}(\mathcal{H})$ as either a relational structure or as a pp-sentence.

Fact 3.3. *The problems* $\mathsf{CSP}_{\mathrm{pp}}(\mathcal{H})$ *and* $\mathsf{CSP}(\mathcal{H})$ *are polynomially equivalent.*

3.1.1 Exercises

3.1.1 What are the canonical sentences $\phi(K_3^s)$ and $\phi(K_3^u)$ for $\mathsf{Ext}(K_3) = \mathsf{CSP}(K_3^s)$ and $\mathsf{ListHom}(K_3) = \mathsf{CSP}(K_3^u)$.

3.1.2 Show, with a reasonable sized example, how to make an instance of $\mathsf{CSP}(K_3^s)$ from an instance of $\mathsf{CSP}_{pp}(\mathsf{K}_3^s)$.

3.1.2 Gadgets– new relations from old

Given a τ-structure $\mathcal{H}$ and a τ-formula $\phi(x_1, \ldots, x_k)$ we can define a pp-*definable* k-ary relation

$$R^{\mathcal{H}} = \{(a_1, \ldots, a_k) \mid \mathcal{H} \models \phi(a_1, \ldots, a_k)\}$$

for a new symbol R which is not in τ. Where $\tau' = \tau$, and $\mathcal{H}'$ is τ'-structure we get from $\mathcal{H}$ by adding the relation $R^{\mathcal{H}}$, Jeavons Cohen and Gyssens showed that $\mathsf{CSP}(\mathcal{H}') =_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H})$.

Example 3.4. Let $\mathcal{H}$ be a graph (with relation E) and $\phi_1(x_1)$ be the formula

$$\exists x_2, x_3(E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)).$$

This defines a 1-ary relation R such that $R^{\mathcal{H}}$ contains the tuple (v) for every vertex of $\mathcal{H}$ that is in a triangle.

Let $\phi_2(x_1, x_2)$ be the formula

$$(E(x_1, x_2) \wedge R(x_1) \wedge R(x_2)).$$

The relation this defines is the 2-ary relation E_1 such that $R^{\mathcal{H}}$ is the restriction of E to pairs of vertices that are in triangles.

The fact that $\mathsf{CSP}(\mathcal{H}') =_{\mathsf{poly}} \mathsf{CSP}(\mathcal{H})$ means that if $\mathsf{CSP}(\mathcal{H}') \in \mathsf{NPC}$ then so is $\mathsf{CSP}(\mathcal{H})$. So if we can prove that $\mathsf{Hom}(H')$ is in NPC for every graph H' for which every vertex is in a triangle, then we have that $\mathsf{Hom}(H)$ is in NPC for every graph H that contains a triangle.

This is a common technique with a unary relation. Intersecting it with all relations just restricts the stucture to the sub-structure induced by those vertices in the unary relation. △

Now. we can do exactly the same thing with a gadget construction.

Example 3.5. Let H be a graph containing a triangle, and let H^* be the subgraph induced by vertices contained in a triangle. For an instance G of $\mathsf{Hom}(H^*)$ let ${}^*G$ be the instance of $\mathsf{Hom}(H)$ we get by adding, for every vertex v of G, a new copy K_v of K_3, and identifying one vertex with v.

We observe that

$${}^*G \to H \iff G \to H^*,$$

and so $\mathsf{Hom}(H^*) \leq_{\mathsf{poly}} \mathsf{Hom}(H)$.

Indeed, let $\phi : {}^*G \to H$ be a homomorphism. As every vertex v of ${}^*G$ is in a triangle K_v, $\phi(v)$ must be in a triangle in H, so $\phi(v)$ is in $V(H^*)$. Thus ϕ is a homomorphism of ${}^*G$ to H, and its restriction to vertices of G is a homomorphism of $G \to H$, as needed. On the other hand, let $\phi : G \to H^*$ be a homomorphism, and view it as a partial mapping of ${}^*G$ to H. As for each vertex $v \in V(G)$, $\phi(v)$ is in a triangle, ϕ can be extended to a homomorphism of K_v to H. Doing this for all vertices we get a homomorphism of ${}^*G$ to H.

△

Note 3.1

To me, the second version of this example is more intuitive, but the proof is messier. We have to show homomorphisms extend and yadda yadda. To do this with proper rigour is a lot of work. The pp-formula version seems cleaner.

It might seem that the pp-formula version of the example is saying more, saying $\mathsf{Hom}(H^*) =_{\mathsf{poly}} \mathsf{Hom}(H)$ rather than just $\mathsf{Hom}(H^*) \leq_{\mathsf{poly}} \mathsf{Hom}(H)$. But it is not. Because the problem $\mathcal{H}'$ contains $\mathsf{Hom}(H)$ AND $\mathsf{Hom}(H^*)$. $\mathcal{H}'$ is a τ-structure where τ contains the relation R which is $\mathsf{Hom}(H)$ and R' which is $\mathsf{Hom}(H^*)$.

3.1.3 Exercises

3.1.1 With examples 3.4 and 3.5 we showed that if $\mathsf{Hom}(H)$ is in NPC for every (non-empty) graph H such that every vertex is in a K_3, then $\mathsf{Hom}(H)$ is in NPC for every graph H that contains a triangles. Using an edge gadget, or a pp-formula reduction, shot that if $\mathsf{Hom}(H)$ is in NPC for every (non-empty) graph H such that every edge is in a K_3, then $\mathsf{Hom}(H)$ is in NPC for every (non-empty) graph H containing a triangle.

3.1.2 Show that every relation in K_n^u is pp-definable from K_n^s. From this, conclude $\mathsf{ListHom}(K_n) =_{\mathsf{poly}} \mathsf{Ext}(K_n)$. Then, using that K_n is a core, conclude $\mathsf{ListHom}(K_n) =_{\mathsf{poly}} \mathsf{Hom}(K_n)$. (Do you think this is true for every core graph G in place of K_n?)

3.1.3 Let $I = I(x, y)$ be an edge gadget, and H be a digraph. Show that the digraph H^* on the vertex set $V(H)$ with edgeset

$$\{(u, v) \mid \exists\, \phi\colon I \to H \text{ such that } \phi(x) = u \text{ and } \phi(y) = v\}$$

is pp-definable from H. Conclude that $\mathsf{Hom}(H^*) \leq_{\mathsf{poly}} \mathsf{Hom}(H)$.

4 Finishing up the proof of the H-colouring Dichotomy

Theorem 4.1. *Let H be a non-bipartite graph, the* $\mathsf{Hom}(H)$ *is in* NPC.

We begin the proof. Assume that H is a non-bipartite graph. Our proof is by a series of reductions. Call the proof an induction on the number of vertices, so if any reduction yields a smaller non-bipartite graph, it has worked.

Lemma 4.2. *We may assume that H contains a triangle.*

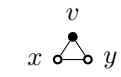
Proof. If H has odd-girth greater than 3, then the pp-formula $\phi(x,y) : \exists u, v, (x \sim u \sim v \sim y)$ which we used in Example 1.17 defines from H a symmetric irreflexive relation H' with smaller odd-girth than H. So (as $\mathsf{Hom}(H') \leq_{\mathsf{poly}} \mathsf{Hom}(H)$) it is enough to prove the theorem for graphs containing a triangle. □

We recall that at any stage, we may assume that H is a core, as so it is enough to show that $\mathsf{Ext}(H)$ is in NPC. So we may use the singleton unary relations $S_v = \{(v)\}$ in H^s.

Lemma 4.3. *We may assume that H contains no K_4 and that every edge is in a triangle.*

Proof. If H has a vertex v in a K_4, then the pp-formula in the signature of H^s

$$\phi(x,y) : \exists u(S_v(u) \wedge E(x,u) \wedge E(x,y) \wedge E(y,u))$$



defines a symmetric binary relation consisting only of the edges of H in triangles with v.

Repeatedly applying this reduction and reducing to cores, we may assume that H contains no K_4 but that every edge is in a K_3. □

Note 4.1

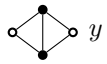
Up to this point, the proof has been essentially that of Hell and Nešetřil. From here, we follow Bulatov.
We have used that if a pp-definable relation has a hard CSP problem, then so does the original structure. But this is not the only way $\mathsf{CSP}(\mathcal{H})$ can be shown to be in NPC. Indeed, in Example 2.13, the gadget in the case that the structure only had two vertices does not give a pp-definable relation. It gives what is called a pp-interpretation. This is where algebra would be more elegant. A pp-interpretation from one structure to another structure on a different vertex corresponds taking a homomorphic image of a power of the polymorphism algebra of the first structure. The proof that there is a reduction between the corresponding CSP problems is either a messy gadget constrution, or the observation that these operations on algebras preserve polymorpisms. We use one of these now, and skimp on the details.

For a binary relations R_1 and R_2 we use $R_1 \circ R_2$ to denote the relation that is *pp*-defined by the formula $\phi(x,y) : \exists x(R_1(x,y) \wedge R_2(y,z)$. The notation R^d for a binary relation R is defined recursively by $R^1 = R$ and $R^d = R \circ R^{d-1}$.

Lemma 4.4. *We may assume that H contains no copy of a diamond:*  .

Proof. Consider the 2-ary relation R that is pp-defined by the formula

$$\phi(x,y) : \exists u, v, (x \sim u, x \sim v, u \sim v, u \sim y, v \sim y).$$



Clearly R is symmetric, and is reflexive as every vertex is in a triangle. Thus its transitive closure T is an equivalence relation on $V(H)$. As H is finite $T = R^d$ for some d, so T is pp-defined. We claim that no equivalence class of T contains the endpoints of an edge. Indeed, assume $T(u,v)$ for $u \sim v$, and let $\ell \leq d$ be the minimum such that $R^\ell(u,v)$. As H is K_4-free, $\ell \geq 2$. If $\ell = 2k$ is even, then there is some z such that $R^k(z,u)$ and $R^k(z,v)$. So $\phi'(x) : (E \circ R^k)(z,x)$ defines a 1-ary relation U containing all neighbours of u and v, in particular, it contains everything in a triangle with u, and it contains the neighbour u of v. But it does not contain z by the minimalily of ℓ, so is a proper subset of $V(H)$. The induced subgraph induced by U is pp-definable as

$$\phi(x,y) : U(x) \wedge U(y) \wedge E(x,y),$$

and so is a smaller graph containing a triangle, so the proof of the claim is done by induction. So no equivalence class of T contains an edge.

If ℓ is odd, something similar works. Try to figure it out your damn self.

Now consider the quotient graph H/T whose vertices are the equivalence classes of T and in which two are adjacent if there is an edge between any two vertices of the equivalence classes. This graph, smaller than H, contains a triangle, as H did. (For any triangle in H the vertices are adjacent so in different equivalence classes, and the equivalence classes are adjacent as they contain adjacent vertices.)

One can now use $E \circ R^d$ as a variation of an edge gadget, and show that $\mathsf{CSP}(H/T) \leq_{\mathsf{poly}} \mathsf{CSP}(H)$. Its messy, and is better done with algebra. Let's just believe it □

We will have one more pp-reduction showing that H contains K_3^k. But before we do that we need to show the following nice little lemma.

Lemma 4.5. *Let $\phi : K_3^k \to H$ be a homomorphism to a diamond-free irreflexive graph H, then $\phi(K_3^k)$ is isomorphic to K_3^d for some $d \leq k$.*

Proof. For an increasing subset $I = \{i_1, i_2, \ldots i_d\} \subset [k]$ of indices let $\pi_I : K_3^k \to K_3^d$ be the multi-projection defined

$$\pi_I(v_1, \ldots, v_k) = (v_{i_1}, \ldots, v_{i_d}).$$

We say that a homomorphism $\phi : K_3^k \to H$ *depends* on an index $i \in [k]$ if for any vertices $x, y \in V(K_3^k)$,

$$x_i \neq y_i \Rightarrow \phi(x) \neq \phi(y).$$

On the other hand ϕ *ignores* i if $\phi(x) = \phi(y)$ whenever x and y differ in *only* the i^{th} coordinate. It is clear that the projection π_I depends on all $i \in I$ and ignores all $j \notin I$. Observing that if ϕ ignores j for each $j \notin I$ then $\phi(x) = \phi(y)$ if and only if $\pi_I(x) = \pi_I(y)$ we get, as you show in Exercise 4.0.1,

that $\phi(K_3^k)$ is isomorphic to if K_3^d for some d if and only if ϕ ignores all indices that it does not depend on.

So suppose ϕ does not depend on some $j \in [k] \setminus I$. That is, suppose there are tuples $x, y \in K_3^k$ such that $x_j \neq y_j$ but $\phi(x) = \phi(y)$. We show that ϕ ignores j. Set $j = k$ to simplify notation.

To further simplify notation, for the rest of the proof we write $(k-1)$-tuples in K_3^{k-1} as uppercase letters such as X, 1-tuples as lowercase letters such as x and k-tuples as concatenations, such as Xx, of a $(k-1)$-tuple and a 1-tuple. For two one tuples x and y let $x * y$ represent an arbitrary 1-tuple different from both of them. As K_3 has three vertices this always exists but may not be unique. $X * Y$ is the same operation componentwise. So $(X * Y)(y * x)$ is adjacent to both Xx and Yy.

With this notation, we have Xx and Yy with $x \neq y$ such that $\phi(Xx) = \phi(Yy)$. To show that ϕ ignores k we show for any Z, z and z', that $\phi(Zz) = \phi(Zz')$. Wlog, we may assume that $z' = x$.

The vertex $Rr := (X * Z)(x * z)$ is adjacent to Xx and to $Sx := ((X * Z) * Y)x$, and Sx is also adjacent to Yy, so $\phi(Rr), \phi(Sx)$ and $\phi(Xx) = \phi(Yy)$ induce a triangle in H. The vertex Rz is also adjacent to Xx and Sx so $\phi(Rz) = \phi(Rr)$ or we have a diamond in H. The vertex $Tt := (Z * R)(z * x)$ is adjacent to Rz, so $\phi(Tt) \sim \phi(Rr)$, and to Zz and Zx. As Zz and Zx are both also adjacent to $Rr = (X * Z)(x * z)$ we have that both $\phi(Zz)$ and $\phi(Zx)$ are in triangles with the edge $\phi(Tt) \sim \phi(Rr)$, and so as H has no diamonds, $\phi(Zz) = \phi(Zx)$ as needed.

□

Recall we can assume now that H is a core, and so we can use precolourings.

Lemma 4.6. *We may assume that H is K_3^k for some k.*

Proof. Let K_3^k be the biggest power of K_3 in H, and let a k-tuple in $V(K_3^k)$ represent the corresponding vertex in a fixed copy K_0 of K_3^k in H. Consider following pre-H-colouring of K_3^{6k}: for every k-tuple $v = (v_1, \ldots, v_k)$ in the subgraph K_0 of H, let the $6k$-tuple

$$(v_1, v_1, v_1, v_1, v_1, v_1 \mid v_2, v_2, v_2, v_2, v_2, v_2 \mid \cdots \mid v_k, v_k, v_k, v_k, v_k, v_k)$$

be pre-coloured with colour v. So any homomorphism of K_3^{6k} to H maps onto K_0; as by the previous lemma, it must be a multi-projection, it is a multi-projection for an index set I containing exactly one of the indices in $[6i + 1, 6i + 6]$ for each $i \in \{0, 1, \ldots, k - 1\}$.

Where x and y are the vertices

$$(1, 1, 2, 2, 3, 3 \mid 1, 1, 2, 2, 3, 3 \mid \cdots \mid 1, 1, 2, 2, 3, 3) \text{ and } (2, 3, 1, 3, 1, 2 \mid 2, 3, 1, 3, 1, 2 \mid \cdots \mid 2, 3, 1, 3, 1, 2)$$

the image of (x, y) under such multi-projections are exactly the edges in K_0. So $K_0 \cong K_3^k$ is pp-defined from the relations of H. □

Now as K_3 is the core of K_3^k, we are done the proof of the Theorem.

4.0.1 Exercises

4.0.1 Where 'depends on' and 'ignores' are as defined in Lemma 4.5 show that an idempotent function $f : K_3^k \to K_3^k$ is a multi-projection if and only if it ignores all indices $i \in [k]$ that it does not depend on. Conclude that the image $f(K_3^k)$ of a homomorphims $f : K_3^k \to H$ is K_3^d for some d if and only if f ignores all indices it does not depend on.

5 A smattering of Universal Algebra

Universal algebra is a broad field of in which the basic structures of classical algebra are abstracted to their presentations as functions on sets, and vastly generalised. Its application to problems of complexity started to arise 30-40 years ago, and has become an invaluable tool in this field. We do not have time to do it in any reasonable amount of detail. So I just to enough in this version of these notes, to convince you that the ideas could be useful.

5.1 Algebras

An *algebra* $\mathbf{A}$ consists of a universe A and a set $\mathcal{F}$ of operations acting on the universe. As with relational structures though, we complicate the definition. A *type* τ is a set of function symbols such as f and g and f_i each having an arity in $\mathbb{N}$. An algebra $\mathbf{A}$ of type τ has an operation $f^{\mathbf{A}} : A^k \to A$ of arity $k = \text{arity}(f)$ for each symbol $f \in \tau$.

Example 5.1. A group $\mathbf{G}$ is an algebra of type $\tau = (\cdot, ^{-1}, 1)$ having arities $(2, 1, 0)$. The operation $\cdot^{\mathbf{G}} : G \times G \to G : (g, h) \mapsto g \cdot^{\mathbf{G}} h$ is the group operation, $(^{-1})^{\mathbf{G}}$ is the inverse, and $1^{\mathbf{G}}$ is the identity. Not every algebra of this type is a group though. To be a group, an algebra of this type must satisfy such identities as

$$\forall x,\ x \cdot 1 = x = 1 \cdot x$$

which is what we need for 1 to actually be the identity. Technically I should have written $x \cdot^{\mathbf{G}} 1^{\mathbf{G}} = x \ldots$ but it looks awful. We drop it if it doesn't cause too much confusion.

Oops, I said that τ was a set but wrote it as a vector. This is a convenience so that we can list the aritites quickly. Sometimes we say that such an algebra is of type $(2, 1, 0)$, conflating the type this vector of arities.

Example 5.2. A ring is an algebra of type $(+, \cdot, -, 1, 0)$ having arities $(2, 2, 1, 0, 0)$.

Now the main kind of algebra that we will be concerned with, is one we get from a relational structure.

Definition 5.3. For a relational structure $\mathcal{H}$, the *polymorphism algebra* $\text{Pol}(\mathcal{H})$ is the algebra wtih universe $V(\mathcal{H})$ whose operations are the polymorphisms of $\mathcal{H}$. The subset of k-ary polymorphism is denoted $\text{Pol}^{[k]}(\mathcal{H})$.

Note 5.1

The following two properties of $\mathrm{Pol}(\mathcal{H})$ mean that is is what is called a 'clone'.

Fact 5.4. *Let $\mathcal{H}$ be a relational structure.*

- *For every $k \geq 1$ and every $i \in [k]$, the projection*

$$\pi_i : \mathcal{H}^k \to \mathcal{H} : (v_1 \ldots, v_k) \mapsto v_i$$

  is in $\mathrm{Pol}(\mathcal{H})$.

- *For every $f \in \mathrm{Pol}^{[k]}(\mathcal{H})$ and every k polymorphisms $g_i \in \mathrm{Pol}^{[\ell]}$ the composition*

$$f(g_1, \ldots, g_k) : (v_1, \ldots, v_\ell) \mapsto f(g_1(v_1, \ldots, v_\ell), g_2(v_1, \ldots, v_\ell), \ldots, g_k(v_1, \ldots, v_\ell))$$

  is in $\mathrm{Pol}(\mathcal{H})$.

It is not too hard to see that a polymorphism f of $\mathcal{H}$ preserves pp-defined relations of $\mathcal{H}$. But it is notational. The proof is the obvious extension of the following example.

Example 5.5. Consider a τ-structure $\mathcal{H}$ with a 2-ary polymorpism f and a relation N pp-defined from τ by a formula such as

$$\phi(x_1, x_2, x_3) : \exists y, z, R_1(x_1, y, x_2) \wedge R_2(x_2, x_3, z).$$

For $\vec{a_1}, \vec{a_2}, \vec{a_3} \in N^{\mathcal{H}}$, there exist y_i and z_i such that

$$\begin{array}{l} R_1(a_1^1, y_1, a_1^2) \wedge R_2(a_1^2, a_1^3, z_1) \\ R_1(a_2^1, y_2, a_2^2) \wedge R_2(a_2^2, a_2^3, z_2) \\ R_1(a_3^1, y_3, a_3^2) \wedge R_2(a_3^2, a_3^3, z_3). \end{array}$$

Applying f row-wise we get

$$R_1(f(a_1^1, a_2^1, a_3^1), f(y_1, y_2, y_3), f(a_1^2, a_2^2, a_3^2)) \wedge R_2(f(a_1^2, a_2^2, a_3^2), f(a_1^3, a_2^3, a_3^3), f(z_1, z_2, z_3)),$$

which shows that $f(\vec{a_1}, \vec{a_2}, \vec{a_3}) \in N^{\mathcal{H}}$. So the polymorphism preserves N. △

This is the easy part of the following. The other direction takes a bit more work, and we skip the proof.

Theorem 5.6. *The set of relations of $V(\mathcal{H})$ that are preserved by all polymorphisms in* $\mathrm{Pol}(\mathcal{H})$ *is exactly the set of relations that are* pp*-definable from* $\mathcal{H}$*.*

An important consequence of this is that, (independent of Theorem 2.17,) the complexity of $\mathsf{CSP}(\mathcal{H})$ depends only on $\mathrm{Pol}(\mathcal{H})$.

An *equation* σ is a string consisting of variables, function symbols, and the equivalence symbol '$\cong$'. For example, when defining majority functions, we saw the equation

$$f(x,x,y) \cong f(x,y,x) \cong f(y,x,x) \cong x.$$

An algebra $\mathbf{A}$ *satisfies* an equation σ, if it contains an operation for each function symbol in the equation, such that the equation holds with these operations for every assignment of values from $V(\mathbf{A})$ to the variables of the equation.

We have seen that if the algebra $\mathrm{Pol}(\mathcal{H})$ satisfies the majority equation given above then $\mathsf{CSP}(\mathcal{H})$ is in P and if $\mathrm{Pol}(\mathcal{H})$ contains only projections then $\mathsf{CSP}(\mathcal{H})$ is in NPC.

Taylor showed that if $\mathrm{Pol}(\mathcal{H})$ contains any operations other than projections, permutations, and thier compositions, then it contains an operation satisfying a certain (pretty long) equation.

The existence of a Taylor operation in a polymorphism algebra has since been shown to be equivalent to the existence operations satisfying various other equations. Among them are

- WNU operations satisfying $f(x,x\ldots,x,y) \cong f(x,x\ldots,y,x) \cong \ldots \cong f(y,x\ldots,x,y)$ of some arity.
- Cycle operations satisfying $f(x_1,\ldots,x_p) \cong f(x_2,\ldots,x_p,x_1) \cong \ldots \cong f(x_p,x_1,\ldots,x_{p-1})$ for some prime $p \geq |V(\mathcal{H})|$.
- Rare-area operations satisfying $f(r,a,r,e) \cong f(a,r,e,a)$.

The proof of the equivalence of this equations is difficult and depends on some big papers: one by Maroti and MacKenzie, and an extension of Theorem 1.16 to smooth digraphs by Barto, Kozik and Niven. They allow for alternate statements of 2.17. In particular, we get that $\mathsf{CSP}(\mathcal{H})$ is in P if $\mathcal{H}$ has a cycle term of prime arity at least n, otherwise $\mathsf{CSP}(\mathcal{H})$ is in NPC.

We finish off with a lovely proof of Theorem 1.16 using this statement, by showing that if a symmetric graph with a cycle term of odd length p has an odd cycle of length at most p, then it has a loop.

Indeed, let H be a graph containing an odd cycle of girth g and a cycle term of length $p \geq g$. It is not hard to see that, using the odd cycle, we can find a closed walk $v_1, v_2, \ldots, v_p$ in H of length p. But then

$$\phi(v_1,\ldots,v_p) = \phi(v_2,\ldots,v_p,v_1).$$

Moreover, as $v_i \sim v_{i+1}$ for each i, we have that

$$\phi(v_1,\ldots,v_p) \sim \phi(v_2,\ldots,v_p,v_1),$$

which means that this vertex has a loop.

Note 5.2

Once we have a group or a ring, we can define subgroups and subrings, products, homomorphisms, isomorphism and what not. We can do the same for all algebras — all of these operations should preserve type, and so knowing this, I think their defintions should mostly be clear. It can quite easily be shown that the family of algebras of a type having operations that satisfy a certain equation is a variety, that is, it is closed under products, sub-algebras, homomorphic images. A variety omits WNU operations if and only if it contains an algebra on at least two vertices all of whose operations are projections, permutations and their compositions.
Universal algebra starts by looking at these varieties.

5.1.1 Exercises

5.1.1 What identities must an algebra of type $(+, \cdot, -, 1, 0)$ satisfy to be a ring.

5.1.2 Show that if a symmetric graph G contains an odd cycle of length g then it contains a closed walk of length d for any $d \geq g$.

References

[1] M. Bodirsky, *Graph Homomorphisms and Universal Algebra Course Notes* Google it.

[2] P. Hell, J. Nešetřil, *Graph Homomorphisms.* Oxford Lecture Series in Mathematics and Its Applications (28). Oxford University Press, 2004.