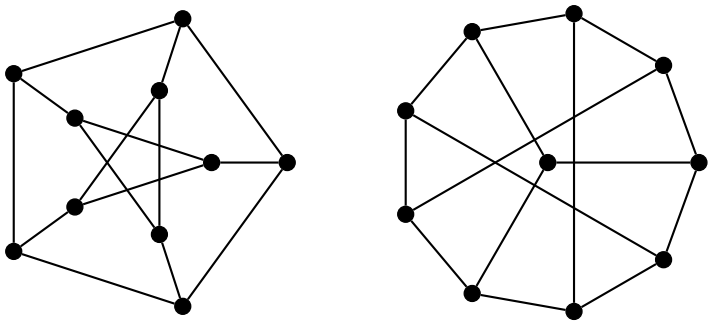
Graph Isomorphism in quasipolynomial time

Aleksandar Makelov
University of Cambridge
April 29 2015

Contents

Introduction

Anyone familiar with drawing pictures of graphs knows it might very well happen that two seemingly different drawings represent the same graph. How can we tell? Even for not very large graphs, such as the ones below, it can take some effort.



For seriously large graphs, inspection by eye is unimaginable; we need a procedure. Naturally, we can number the vertices in the first drawing G_1 with the numbers $1, 2, \ldots, n$ and write down the edges as pairs of numbers. Then we need to tell if we can name the vertices of the second drawing G_2 with $1, 2, \ldots, n$ so that we get the *same* list of edges. This is the *graph isomorphism problem* (abbreviated GI).

PROBLEM 0.1 (Graph isomorphism). Given the descriptions of two graphs G_1, G_2, determine if there exists an isomorphism $G_1 \to G_2$.

There are $n!$ ways to name the vertices of G_2, and we can simply try each of them. But this seems wasteful: it's easy to imagine cases when we can easily refute a great deal of those $n!$ possibilities in advance. For example, we know that the degrees of vertex i in G_1 and G_2 must agree – so if the degrees in G_1 are very evenly distributed, this dramatically restricts which vertices of G_2 can get which names; and if they're not, perhaps that gives us more structure to work with. Can we leverage such invariants to always solve GI with less than $n! = \exp(O(n \log n))$ work?

It turns out that we can do *much* better – but the proof is far from trivial, and combines deep insights from both combinatorics and group theory. An $\exp(c\sqrt{n \log n})$ algorithm due to Luks and Babai [**Luk82**] was found early on in the computational study of GI, but for nearly 35 years nobody could do better (and people tried!). In a remarkable recent result, Babai overcame this barrier by giving a *much* faster **quasipolynomial**, that is $\exp((\log n)^c)$-time, algorithm.

The goal of this essay is to give an exposition of the key ideas behind Babai's algorithm in a top-down manner (as opposed to Babai's more-or-less bottom-up approach), and illuminate their roots in previous work on GI[1].

[1] By the way, both graphs are the Petersen graph, as usual

0.1. Why is graph isomorphism interesting?

As Babai notes, GI has been more or less solved in practice. In fact, our worst-case-guarantee algorithms fail miserably in comparison with extremely naive heuristics which work on almost all graphs. The main interest in GI comes from its special place in the complexity landscape, and as a meeting point of combinatorics and group theory.

0.1.1. To computer scientists. For the computer scientist, the appearance of such a dramatically faster algorithm for a decades-old problem is a rare event. It can perhaps be compared to only one other result in recent memory, the 'PRIMES is in **P**' paper establishing primality testing in polynomial time. As in the case of PRIMES, there were efficient heuristic algorithms for GI long before Babai's result, making it a landmark example of the folklore 'theory follows practice' principle.

Babai's quasipolynomial algorithm also lends more credibility to algorithmic optimism, the belief that there might be efficient algorithms for many problems complexity theorists usually deem intractable – and some algorithmic optimists even believe $\mathbf{P} = \mathbf{NP}$ might be true [2].

However, we shouldn't get too excited – GI is one of few problems believed to be **NP**-intermediate: neither in **P**, nor **NP**-complete, and thus a much faster algorithm for GI is much less surprising than, say, a much faster algorithm for SAT.

Beyond that, GI is an important complete problem for isomorphism of combinatorial structures. It captures the complexity of isomorphism of digraphs, labeled trees, hypergraphs, automata, and relational structures to name a few (see e.g. Zemlyachenko, Korneenko and Tyshkevich [**ZKT85**]).

0.1.2. To mathematicians. For the mathematician, the computational study of GI has lead to deeper understanding of graph invariants, and the structure and interactions between graphs and groups. In particular, efforts to solve GI have given us both negative results on combinatorial and spectral approaches to obtaining canonical forms of graphs [**CFI92**], and new purely combinatorial proofs of group-theoretic results previously accessible only through the classification of finite simple groups [**SW15**].

Another reason why a mathematician might care about complexity in general is that it formalizes a hierarchy of *explicitness* (see e.g. Goldreich, Appendix E [**Gol08**]). To illustrate this idea in the case of GI, if we're in a wishful mood we can imagine there is some closed-form expression we can write down in terms of, say, only the degrees of G_1, G_2, that magically gives us an isomorphism $G_1 \to G_2$ if one exists. That would be a fairly explicit solution of the problem; coincidentally, it corresponds to an algorithm for GI which is extremely economical in terms of computational resources[3]. One can imagine more complicated variations of growing runtime, such as a regularization function we have to iterate several times on the graphs, or a procedure that reduces the problem to $O(n)$ instances of GI on smaller graphs (we'll discuss how both paradigms can solve GI in special cases).

In this way we arrive at the idea that the complexity of an algorithm corresponds in some vague way to the explicitness of the isomorphism we're looking for: how well 'hidden'/ how 'visible' is it among all $n!$ possibilities? Or in other words, how much searching do we have to do before we find it? A key step of Babai's proof is his ability to show that certain *global* symmetries are not very well hidden: we can 'make them reveal themselves' only by computing enough local symmetries.

[2] See e.g. `https://rjlipton.wordpress.com/2009/09/22/its-all-algorithms-algorithms-and-algorithms/`

[3] We've given a vague description, but we can imagine that evaluating a 'formula' will usually be in **L** (logspace) by a straight-line program, and also surely something nice about circuits :)

0.2. Approaches to GI

As we will later see in more detail, there are two natural paradigms for attacking GI; here we'll describe the high-level intuition behind them, and the obstacles they meet.

Combinatorial partitioning: we refine the structure of the graphs by introducing labels which carry information about local invariants. We already mentioned a simple example: coloring each vertex according to its degree. Then any isomorphism φ must respect this *refined* structure, that is, $\varphi(v)$ must have the same color as v. We can then iterate by adding the colors of neighbors to the label of each vertex. The hope is that after not too many rounds of such refinement any potential isomorphism will be evident (e.g., all vertices get different colors).

Group-theoretic divide-and-conquer methods: here we need a little leap of faith. As we will later discuss, GI is polynomial-time equivalent to the problem of finding the automorphism group $\mathrm{Aut}(G)$ of a graph, and there is a well-developed computational theory of permutation groups. Under some natural situations we can constrain $\mathrm{Aut}(G) \leq H$ from above, for example after good combinatorial partitioning. Then we can exploit intransitivity and imprimitivity of G to reduce the isomorphism problem to several smaller instances.

Approaches based solely on combinatorial partitioning have been shown to take $n^{O(n)}$ time to refute isomorphism for some pairs of graphs (Cai, Furer and Immerman [**CFI92**]); and approaches using group theory are blocked by large primitive groups without easily accessible subgroups of small index, where the above divide-and-conquer halts.

0.3. Babai's algorithm

Babai's proof combines combinatorics and group theory in an essential way. As a first step, he switches (following Luks [**Luk82**]) to the more general problem of *string isomorphism*, abbreviated SI, which asks for the possible 'anagrams'

$$\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) = \left\{\sigma \in G \;\middle|\; \forall u \in \Omega : \mathfrak{x}^\sigma(u) = \mathfrak{y}(u)\right\}$$

between two strings $\mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$ when the permutations come from a subgroup $G \leq \mathrm{Sym}(\Omega)$ of all permutations (indeed, for the full symmetric group the problem is trivial). Luks [**Luk82**] gives efficient ways to reduce this problem to significantly smaller instances when G is not a large primitive group.

It was known since the 80s that the obstacle to efficient application of Luks's method were large alternating quotients of permutation groups, and the barrier case is captured by a situation where we have a homomorphism $\varphi : G \to \mathrm{Sym}(\Gamma)$ such that $\mathrm{Alt}(\Gamma) \leq G^\varphi$ (such a φ is called a **giant homomorphism**, or **giant representation**. The main part of Babai's algorithm deals with such a giant homomorphism.

Under Luks's framework, when faced with a giant φ, one finds $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$ by looking for isomorphisms in each coset of $\ker \varphi$, with the cosets represented by liftings of each element $\overline{\sigma} \in G^\varphi$. This corresponds to enumerating the group $G^\varphi \geq \mathrm{Alt}(\Gamma)$, which takes too long for large Γ. Babai's approach is to avoid this oblivious to the strings brute-force search of G^φ by pushing forward invariants of $\mathfrak{x}$ and $\mathfrak{y}$ to Γ through the images of their G-automorphism groups.

If the images of the automorphism groups don't look 'symmetric' enough on many small sets of points, there are classical combinatorial partitioning methods that significantly constrain symmetry. The problem is dealing with the case when there are rich, well-supported automorphisms and potentially many isomorphisms, where we have no ready-made toolbox. The hope is that if such rich structure is found, it will again either allow us to significantly partition Γ by some invariants, or constructively guide our search for isomorphisms. However, it's not clear how to even tell if we're in this case!

The main theorem that makes Babai's algorithm work is the 'unaffected stabilizer theorem' (Theorem 2.11), which allows us to get our hands on global automorphisms by computing efficiently with local information. Specifically, he is able to show that by looking at $O(\log n)$-sized subsets of Γ, we can efficiently decide whether we're in a situation with a few isomorphisms or many isomorphisms, and find constructive evidence of both.

If there's a single idea pervading the proof, it is the unreasonable effectiveness of reductionist algorithmic techniques. Looking at it from a distance, the entire algorithm is an elaborate divide-and-conquer: we find a way to break sets and groups down, solve the smaller problems and piece them together to find global symmetries or refutations thereof.

0.4. This essay

I entered this essay without familiarity with previous work on GI. This will be reflected in the exposition: a GI-seasoned reader will surely be bored at times by the trivialities I remark upon. On the upside, readers who are, like me, making their first steps in this area will hopefully find the exposition more approachable than Babai's paper.

The greatest challenge in writing this essay was getting a high-level view of such a voluminous algorithm. I believe that a completely top-down exposition, where we introduce new ideas only when we need them, is likely to appear unmotivated. This is because there are many paradigms that have been around the GI literature for a long time that permeate the algorithm and guide each next step.

On the other hand, Babai's paper is organized in a bottom-up manner, by first giving a lot of standalone prerequisites, then proving a bunch of theorems, and finally piecing all of them together. It is desirable to have an exposition that feels more like a story in which we know where we're going and hopefully why we're going there.

A nice thing from an expository point of view about the algorithm is that it builds upon some very simple and natural ideas that have been around for a long time. This suggested the following compromise for the structure of the essay:

Chapter 1:

- start off by thinking about graph isomorphism in general, about what works and what doesn't
- develop some simple observations and discuss how they have been used historically with some case studies (we use 3-regular graphs following [**Luk82**])
- at the same time prepare the ground for Babai's generalization of these simple approaches
- reduce to the essential difficulty (large giant actions)

Chapter 2:

- describe the local certificates algorithm, the main algorithm in the paper, in detail. Here we assume the group-theoretical results as given, and focus on the combinatorial aspects.

Chapter 3:

- Describe the design lemma in detail, and justify the use of k-dimensional WL refinement.

There are various details we ignore, so that we can focus on the core ideas in chapter 2 and 3:

- The main detail we ignore is the complexity of the algorithm; it is 'left as exercise'. However, the crucial point for the analysis is the fact that test sets in the Local certificates algorithm are of logarithmic size.

- We don't show how to process the canonical structures found in chapters 2 and 3, but hopefully after the discussion of Luks reduction in chapter 1, it is quite believable that these are the kinds of canonical structures that make the algorithm efficient.

We depart from Babai's paper most significantly in the following ways:

- in terms of organization and synthesis, we put his preliminaries in more context and significantly reorder the presentation, as discussed above.
- We use matrix manipulations to prove regularity properties of binary canonical configurations, as opposed to his purely combinatorial methods. This leads to several cleaner proofs.
- We effectively dispense with the high-dimensional Weisfeiler-Leman canonical refinement by formalizing a notion of explicit regularization that Babai hints at in his proofs.

0.5. Prerequisites

This essay should be approachable by anybody with reasonable command of undergraduate math and computer science. To be specific, on the computer science side, we assume the following:

- a familiarity with basic complexity theory and asymptotic analysis.
- comfort with algorithms and recursion

On the mathematical side, we assume the following:

- experience with undergraduate group theory and permutation groups. Here our principle reference is [**DM96**]
- experience with graph theory
- the most basic concepts of category theory. Category theory is not necessary to understand the essay; it just gives a compact language for abstracting some otherwise elementary and natural arguments that can be managed on a case-by-case basis.
- comfort with combinatorial arguments and overall mathematical maturity.

We *do not* assume knowledge of non-standard topics, such as computational group theory, relational structures, and (coherent) combinatorial configurations.

0.6. Notation and conventions

Notation **sticks**: we're describing a single algorithm, and it greatly improves understanding if we keep the same notation for objects that have the same meaning throughout the entire essay. Moreover, we borrow as much notation as possible from Babai with the goal of making it easier to read his paper and this essay side by side.

All actions are right.

CHAPTER 1

Basic approaches to GI

In this chapter, we will introduce some natural approaches for tackling GI, show some cases in which they have been successful historically, and the obstacles they face. In later chapters, we'll see how Babai overcomes them. At the same time, we will collect useful subroutines, and build familiarity with the kinds of arguments that will later be generalized by Babai's algorithm.

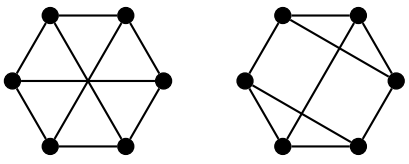
1.1. Colorings

We start off with perhaps the simplest approach to GI: canonical coloring, and see how it leads to the study of *relational structures* and *coherent configurations*. Along the way we will collect some useful examples that will be used later on in Babai's algorithm. As it turns out, we will apply these ideas not to the original graphs, but to structures obtained from strings in the SI problem; this goes to show how pervasive the ideas are.

1.1.1. Vertex and edge coloring. Given graphs X_1, X_2, any potential isomorphism $\varphi : X_1 \to X_2$ must respect the degrees. We can incorporate this information by coloring each vertex according to its degree, and insisting that our isomorphism respects the colors. This can be iterated: for example, no 'blue' vertex all of whose neighbors are 'red' can be mapped to a 'blue' vertex with a 'green' neighbor. So we repeatedly incorporate the colors of each vertex's neighbors in the current colors - we **refine the colors** - until the procedure gives us no new information. This is called **naive vertex coloring**.

However, for regular graphs this is useless. So suppose that instead of vertices we started coloring edges. The natural analogue of the degree is the **co-degree** of an edge $e = (u, v)$: the number of common neighbors of u and v. We can color (u, v) by its co-degree, and iterate this by incorporating the number of common neighbors w where (u, w) is 'blue' and (v, w) is 'red', for all pairs of colors.

This process distinguishes pairs of regular graphs that naive vertex refinement doesn't; for example, consider the two graphs below:



In the left one, all edges will be labeled 0, but on the right we make progress. On the other hand, it's easy to see that the edge coloring algorithm gets stuck on any tree, whereas naive vertex coloring makes progress. So we want something combining the power of both.

When does edge coloring stop? We need every 'blue' edge (u, v) to have the same number of length-2 paths $u \to w \to v$ for any color composition of (u, w) and (w, v) – this is a highly regular situation. The slogan is that

'Color refinement reveals local regularity or refutes it'

As we will shortly see, the colorings stable under a close cousin of edge coloring which incorporates naive vertex coloring have rich structure and strong regularity properties. So we get the idea to explore that regularity to understand isomorphism better. Studying them will lead us to the Weisfeiler-Lehman canonical refinement process, a central regularization tool in Babai's algorithm.

1.1.2. Relational structures and canonical refinement. In this section we formalize the above notions and study so-called classical coherent configurations, which we can intuitively think of as graphs stable under edge coloring.

DEFINITION 1.1 (Binary relational structures). A **binary relation** on a set Ω is a subset $R \subseteq \Omega^2$. A **binary relational structure** $\mathfrak{X} = (\Omega, \mathcal{R})$ consists of a set of **vertices** $\Omega = V(\mathfrak{X})$, and an *ordered* list of non-empty binary relations $\mathcal{R} = (R_1, R_2, \ldots, R_r)$. The **rank** of $\mathfrak{X}$ is r. If $(x, y) \in R_i$, we say that **the color of** (x, y) **is** i and write $c(x, y) = i$. We say $c(x) = c(x, x)$ is the **vertex color** of x.

We want to compare objects like that for isomorphism induced from the isomorphisms of the graphs they came from. The definition is natural:

DEFINITION 1.2. Given binary relational structures $\mathfrak{X} = (\Omega, R_1, \ldots, R_r)$ and $\mathfrak{X}' = (\Omega', R'_1, \ldots, R'_r)$, we say that $\sigma : \Omega \to \Omega'$ is an isomorphism of $\mathfrak{X}$ and $\mathfrak{X}'$ if $R_i^\sigma = R'_i$ for all i, where $R_i^\sigma = \left\{(x^\sigma, y^\sigma) \;\middle|\; (x, y) \in R_i\right\}$.

To combine edge and vertex coloring, we define a special class of binary relational structures that is aware of the distinction between edges and vertices. For the purposes of the essay, it is convenient to make it slightly more general than we need based on the discussion so far.

DEFINITION 1.3 (Binary configuration). We say that the binary relational structure $\mathfrak{X}$ is a **binary configuration** if

(1) The relations partition Ω^2;
(2) A subset of the relations partitions $\mathrm{diag}(\Omega)$
(3) For every R_i, there is some j such that $R_j = \left\{(y, x) \;\middle|\; (x, y) \in R_i\right\}$, the opposite relation.

We call the directed graph $X_i = (\Omega, R_i)$ the **color** i **constituent** of $\mathfrak{X}$. We say $x \in \Omega$ is an **effective** vertex of X_i if it is in the support of the edges. It is an **out-vertex** of X_i if it has positive out-degree in X_i, and an **in-vertex** if it has positive in-degree.

EXAMPLE 1.4. Every graph corresponds to a binary configuration of rank ≤ 3: take $R_1 = \mathrm{diag}(V)$, $R_2 = E$ and R_3 their complement, if any.

In analogy with out edge-coloring algorithm, the refinement process on binary configurations adds to the color $c(x, y)$ the number of z such that $c(x, z) = i, c(z, y) = j$, for all possible pairs i, j, until no further refinement occurs. This algorithm is called **(classical) canonical Weisfeiler-Leman refinement**, abbreviated **WL** from now on.

The word **canonical** refers to the fact that if we have graphs X_1, X_2 and $\sigma : X_1 \to X_2$ is an isomorphism, σ also induces an isomorphism on the WL refinements. In general, when we use **canonical** for some structure on an object, we mean the same thing: being respected by isomorphisms.

So we're interested in understanding the configurations stable under it:

DEFINITION 1.5 (Binary coherent configuration). A binary configuration is **coherent** if it is stable under canonical WL-refinement; in other words, there exist **structure constants** $p_{ijk} \in \mathbb{N}_{\geq 0}$ such that

$$c(x, y) = k \implies \left|\left\{z \;\middle|\; c(x, z) = i, c(z, y) = j\right\}\right| = p_{ijk}$$

The definition of coherent configurations makes it convenient to use some well-known tricks with adjacency matrices of digraphs to understand coherent configurations (here we depart from Babai, who uses first-principles combinatorial arguments):

OBSERVATION 1.6. A binary coherent configuration on vertex set Ω is equivalent to an ordered list of $\Omega \times \Omega$ matrices $A_1, A_2, \ldots, A_r$ with entries in $\{0, 1\}$ such that:

(1) $A_1 + \ldots + A_r = J$, the all-1s $\Omega \times \Omega$ matrix (the relations partition)
(2) $\sum_{i \in D} A_i = I$ for some subset $D \subseteq \{1, 2, \ldots, r\}$ (the diagonal colors)
(3) For every i there is a j such that $A_i^T = A_j$ (opposite-relation closed)
(4) $A_i A_j = \sum_{k=1}^{r} p_{ijk} A_k$ (coherence)

1.1.3. Regularity in coherent configurations. By playing around with these (very special) matrices, we will learn what coherent configurations look like, and establish a range of regularity properties that will turn out to be useful in many contexts throughout the algorithm. In what follows, it helps to interpret matrix multiplication as counting walks on digraphs.

PROPOSITION 1.7. Let $\mathfrak{X}$ be a binary coherent configuration. Then we have

- **vertex-color awareness:** the color $c(x, y)$ determines $c(x)$ and $c(y)$;
- **degree awareness:** all out-vertices of X_i have the same out-degree, and similarly for in-vertices.
- The in-vertices of X_i are a vertex-color class, and similarly for the out-vertices.

PROOF. Observe that

$$A_i = \sum_{j \in D} A_i A_j = \sum_{k=1}^{r} \alpha_{ki} A_k$$

for some non-negative integers α_{ki}. Since the matrices are 'disjoint', it follows that $\alpha_{ii} = 1$ and all others are zero. The only way that can happen is if $A_i = A_i A_j$ for some diagonal color j. A_j is diagonal, so $A_i A_j$ is A_i with the u-th row multiplied by $(A_j)_{uu}$, and if u is an out-vertex (so that the u-th row of A_i is nonzero), $(A_j)_{uu} = 1$, i.e. $c(u) = j$. The in-vertex part is analogous.

Next, we can represent

$$A_i A_i^T = \sum_{k=1}^{r} \alpha_k A_k$$

The u-th entry of the diagonal of $A_i A_i^T$ is the number of $u \to u$ walks that take a step back and a step forward in X_i, i.e. the in-degree of u in X_i. By vertex-color awareness, there's a $j \in D$ such that $c(u) = j$ for any in-vertex u of X_i, hence $\alpha_j > 0$ and $\alpha_k = 0$ for all other vertex colors. Thus all in-vertices of X_i have degree α_j.

Finally, we just got $A_i A_i^T = \alpha_j A_j$, which by inspecting diagonals shows the in-vertices are precisely the color class j. □

It follows that every constituent X_i is either **homogeneous** – its set of effective vertices is a vertex-color class – or **bipartite**, when it's in-vertices form one color class and its out-vertices form another. Also, each X_i is biregular on its support.

The following claims are stated in Babai [**Bab15**] without proof:

PROPOSITION 1.8. If X_i is a bipartite constituent, its connected components equipartition each of the vertex-color classes of its support. If X_i is a homogeneous constituent, each of its weakly connected components is strongly connected, and they equipartition its vertex-color class.

PROOF. We now introduce a routine trick. Suppose X_i is bipartite with out-vertices C_j and in-vertices C_k, and let $M = A_i A_i^T$. Then $u, v \in C_k$ are in the same weakly connected component if there is some alternating backward-forward walk on X_i connecting them, in other words if $M^k_{uv} > 0$ for some k. Hence the weakly-connected component of u in X_i is the support of the vector

$$(I + M + M^2 + \ldots + M^N)e_u$$

for some large enough N, where e_u is the indicator of u. We can write $I + M + \ldots + M^N$ as a linear combination $\sum_{l=1}^{r} \alpha_l A_l$ of the constituents. For each l, $A_l e_u$ is the vector of out-neighbors of u in X_l, and by out-regularity, the size of the support depends only on the α_l and the color of u.

For homogeneous X_i over C_j, the weakly connected components are encoded by the support of $(I + M + \ldots + M^N)e_u$ for $M = A_i + A_i^T$, and the same argument shows they all have the same size.

Now suppose on some weakly connected component we have more than one strongly connected component. Consider the directed graph Y with vertices the SCCs, and edges $S_1 \to S_2$ if S_2 is reachable from S_1 (a classical construction). It is acyclic, and in particular has a vertex S with no in-edges. Start a walk from S; since Y is acyclic, it hits a vertex T with no out-edges. Letting $u \in S, v \in T$, we see that

$$(I + A_i + A_i^2 + \ldots + A_i^N)e_u$$

is supported on at least $S \cup T$ for large enough N, whereas

$$(I + A_i + A_i^2 + \ldots + A_i^N)e_v$$

is supported only on T. But by the above arguments, the support has the same size for u and v - contradiction. □

When we have several connected components of a constituent, regularity exists on the level of components as well. Suppose we have a homogeneous constituent X_i on the color class C, and it is disconnected with connected components $B_1, \ldots, B_m$. Then contracting each B_j to a single vertex preserves bi-regularity of any bipartite component with source vertex-color class C:

PROPOSITION 1.9 (Contracting components). Let X_i be a homogeneous constituent on color class C with components $B_1, \ldots, B_m$, and X_j be bipartite from C to C'. Then the bipartite digraph $Y = ([m], C', E)$, where $(i, y) \in E$ when there is some $x \in B_i$ such that $c(x, y) = j$, is biregular.

PROOF. Fix $y \in C'$. By analogy with the argument from the proof of proposition 1.8, and since the components of X_i are strongly connected, for large enough N the support of

$$(I + A_i + \ldots + A_i^N)A_j^T e_y$$

gives precisely the set of components B_i such that $(i, y) \in E$. The size of this support is independent of $y \in C'$, and the B_i have the same size, so Y is in-regular.

Now fix $x \in B_i$. By similar observations, for large enough N the support of

$$A_j(I + A_i + \ldots + A_i^N)e_x$$

gives the set of y such that $(i, y) \in E$. The size of this support is independent of $x \in C$, so Y is out-regular. □

1.1.4. Strategy and classification of binary coherent configurations. Since we are comparing graphs, if our canonically refined colorings can be divided in some canonical way this will help us restrict isomorphisms. If the coherent configuration $\mathfrak{X}$ has multiple diagonal colors, this is a canonical coloring of the vertex set; if some constituent is disconnected, this is a canonical equipartition of its color class. The importance of regularity in coherent configurations is that it ensures equipartitions, which in turn ensure significant reduction of the problems. The following definitions capture desirable properties:

Definition 1.10. A coherent configuration is **homogeneous** if all vertices have the same color, **primitive** if it is homogeneous and all off-diagonal constituents are connected, and **uniprimitive** if it is primitive has rank ≥ 3, and **clique** if it is primitive and has rank 2.

1.1.5. The orbital configuration. The above properties remind us of orbits and block systems in groups; homogeneous corresponds to transitive, and primitive to primitive.

Here is one connection with groups that will be used in the Aggregation of positive certificates algorithm:

Example 1.11 (The orbital configuration). Suppose G is a group acting on Ω. Then the orbits of the induced action on $\Omega \times \Omega$ define a binary coherent configuration $\mathfrak{X}$ called the **orbital configuration of** G. Indeed, observe that if $c(x, y) = k = c(x', y')$ there is some $\sigma \in G$ such that $x' = x^\sigma, y' = y^\sigma$; then

$$c(x, z) = i, c(z, y) = j \implies c(x', z^\sigma) = i, c(z^\sigma, y') = j$$

and thus the map $z \mapsto z^\sigma$ induces a bijection

$$\{z \mid c(x, z) = i, c(z, y) = j\} \to \{z \mid c(x', z) = i, c(z, y') = j\}$$

Also observe that the diagonal $\mathrm{diag}(\Omega \times \Omega)$ is partitioned according to the Ω-orbits of G, and $\mathfrak{X}$ is clique iff G is doubly transitive.

1.1.6. Computational aspects of coloring. To justify our informal use of color 'names', we make the following conventions. Colors are tuples of numbers (possibly singletons), and the colors are ordered lexicographically after each refinement. Since there are never more than $O(n^2)$ colors, at most $O(n^2)$ rounds, and each round appends at most $O(n^2)$ new numbers to a color, this bookkeeping can be done in polynomial time.

1.1.7. Negative results on the power of coloring. Our dream goal with coloring would be to completely distinguish graphs: that we come up with clever tricks to attach local invariants until, say, each vertex has a different color. However, a landmark result by Cai, Furer and Immerman [**CFI92**] showed that for a generalized variant of coloring called k-dimensional Weisfeiler-Leman refinement which is aware of local invariants for k-tuples, this would take $n^{\Omega(n)}$ time for some pairs of graphs. Moreover, by work of Immerman and Lander [**IL90**], this generalized variant of coloring captures *any* strategy based on counting local invariants.

On the other hand, the pairs of graphs of Cai, Furer and Immerman can be colored so that each isomorphism preserves the colors and each color class has size 4. This variant of GI was already solved in randomized polynomial time by Babai [**Bab79**] (see the discussion in subsection 1.5.3, which basically explains how to do it) using very simple group theory.

1.2. To groups

As an easy first observation, we have the following:

PROPOSITION 1.12. The problem GI is Cook reducible[1] to GI for connected graphs, which is in turn Cook reducible to GRAPH-AUT, the problem of finding generators for the permutation group $\mathrm{Aut}(X)$.

PROOF. For the first reduction, given graphs X_1, X_2, we first reject all cases when only one of the graphs is connected. Then if both are connected, we're done, and if both are disconnected, we take their complements. Indeed, since two graphs are isomorphic iff their complements are, and the complement of a disconnected graph is connected, this works.

For the second reduction, given connected X_1 and X_2, we take their disjoint union $X := X_1 \coprod X_2$, run the GRAPH-AUT algorithm on it, and check for a generator of $\mathrm{Aut}(X)$ which moves a vertex of X_1 to a vertex of X_2. If one is found, then $X_1 \cong X_2$; else, $X_1 \ncong X_2$.

Indeed, $X_1 \cong X_2$ iff there is $\sigma \in \mathrm{Aut}(X)$ swapping X_1 with X_2, by connectedness iff there is some $\sigma \in \mathrm{Aut}(X)$ moving a vertex of X_1 to a vertex of X_2, and the generators already detect this, since otherwise any product of them fixes X_1. □

As it turns out, this reduction is not a dead end (i.e., GRAPH-AUT is not harder than GI), and moreover there is a well-developed theory of computing with permutation groups, as we will soon see.

1.3. The structure of isomorphisms of 3-regular graphs

In this section, we'll make some observations that will let us later solve GI in polynomial time for connected 3-regular graphs. This is a great case study: it is quite simple to get one's head around, yet introduces baby versions of the core paradigms behind Babai's algorithm.

We're following Luks's paper, which also proves the more general result that GI $\in \mathbf{P}$ for bounded-degree graphs [**Luk82**]. However, our reduction to the string isomorphism problem is substantially different, and we emphasize the ideas that are relevant to making sense of Babai's approach.

Unless otherwise specified, all graphs in this section are connected.

1.3.1. Initial observations. First we'll give an intuitive argument that shows the automorphism group is nice, and point to an algorithm to find it. Let's think about the automorphism group of a 3-regular graph $X = (V, E)$ about a single vertex, that is, look at the stabilizer subgroup $\mathrm{Aut}(X)_v$ fixing some $v \in V$. If we unfold the paths starting at v, we can embed $\mathrm{Aut}(X)_v$ in the automorphism group of an almost-regular tree:

PROPOSITION 1.13 (Unfolding 3-regular graphs). The stabilizer $\mathrm{Aut}(X)_v$ can be realized as a subgroup of the automorphism group of a balanced tree in which the root has 3 children, and every other non-leaf has 2 children.

PROOF. Let's 'unfold' X about v, by constructing a tree T where each vertex is labeled by an element of $V(X)$. The root is labeled by v, and the the children of a vertex $\alpha \in T$ with label $u \in V$ are labeled by the possible non-backtracking[2] extensions of the walk in X obtained by concatenating the labels on the path from the root to α. Thus in T the root has three children, and every other node apart from the leaves has 2. We continue this process until all vertices of X have been included in our walks and we have some balanced tree T_X.

[1] Recall that a **Cook reduction** of A to B is simply a polynomial-time algorithm that solves A with oracle access to B.

[2] A **non-backtracking walk** is a walk which doesn't use the same edge in two consequent moves, but can otherwise repeat edges and vertices.

Then we get an injection $\psi : \mathrm{Aut}(X)_v \rightarrowtail \mathrm{Aut}(T_X)$. Indeed, the vertices of T_X correspond to all non-backtracking walks started at v of lengths $0, 1, 2, \ldots$, and every automorpshism σ of X that fixes v permutes all such walks of a given length, so it gives a permutation $\psi(\sigma)$ of (each level of) $V(T_X)$. Under this permutation, every single-edge continuation of a walk is mapped to a single-edge continuation of its image, which is exactly saying that $\psi(\sigma) \in \mathrm{Aut}(T_X)$.

Clearly, ψ is a homomorphism since for $\sigma, \tau \in \mathrm{Aut}(X)$, the permutation on length-k walks induced by $\sigma\tau$ is the composition of the ones induced by σ and τ. Finally, as all vertices of X are present in T_X, we can recover $\sigma \in \mathrm{Aut}(X)$ from $\psi(\sigma)$ by just checking how a complete set of labels is mapped – so ψ is injective. □

The automorphism group of T_X is easily seen to be the iterated wreath product $\underbrace{S_2 \wr \ldots \wr S_2}_{d \text{ times}} \wr S_3$ where $d+1$ is the depth of T_X and we take the product action at each step. However, it would be great to get rid of the 3 at the root, because then we would get a 2-group, which would be even simpler to understand. There's an easy trick to do that:

PROPOSITION 1.14. The group $\mathrm{Aut}(X)_e := \left\{\sigma \in \mathrm{Aut}(X) \;\middle|\; \sigma(e) = e\right\}$ of automorphisms of a 3-regular graph $X = (V, E)$ stabilizing an edge $e \in E$ can be realized as a subgroup of the automorphism group of a balanced binary tree.

PROOF. Introduce an *ideal*[3] vertex v_e on the edge e, giving us a new graph X' in which every vertex apart from v_e has degree 3, and v_e has degree 2. Then $\mathrm{Aut}(X)_e \cong \mathrm{Aut}(X')_{v_e}$. The argument from 1.13 then gives us an injection $\mathrm{Aut}(X')_{v_e} \rightarrowtail S_2 \wr \ldots \wr S_2 \wr S_2$. □

The 'ideal' vertex construction from proposition 1.14 in turn naturally leads to a reduction of the GI problem to the problem of computing $\mathrm{Aut}(X)_e$:

PROPOSITION 1.15. The GI problem for 3-regular graphs is (Cook) reducible to finding $\mathrm{Aut}(X)_e$ for 3-regular graphs.

PROOF. Given 3-regular graphs $X_i = (V_i, E_i)$, let's fix some edge $e_1 \in E_1$. Any isomorphism $X_1 \to X_2$ has to map e_1 to *some* edge of X_2, so we can try them all. Now consider the 3-regular graph G obtained by picking ideal vertices v_{e_1}, v_{e_2} on e_1, e_2 respectively, and taking the disjoint union $X_1 \coprod X_2$ with the edge (v_{e_1}, v_{e_2}).

An isomorphism $X_1 \to X_2$ mapping e_1 to $e_2 \in E_2$ is equivalent to an automorphism of X which transposes the ideal vertices. In analogy with 1.12, this completes the reduction. □

1.3.2. Individualization and ideal domains, informal. The proof of proposition 1.15 illustrates two core paradigms which Babai's algorithm vastly generalizes.

The way we broke symmetry by distinguishing an edge e_1 is called **individualization**. In the general case, when we cannot constrain isomorphism between two structures $\mathfrak{X} \to \mathfrak{Y}$ further, we 'break symmetry' by distinguishing an arbitrary substructure $\mathfrak{X}' \subseteq \mathfrak{X}$, and partition the isomorphisms according to where they send $\mathfrak{X}'$.

The introduction of the two **ideal** vertices and the way in which the restricted action of $\mathrm{Aut}(X)$ on them tells us about isomorphisms $X_1 \to X_2$ will too be generalized. Solving the 'pushforward' automorphism problem on $\{v_{e_1}, v_{e_2}\}$ helped us solve the original isomorphism problem: a lack of symmetry on the ideal set certifies lack of isomorphisms, and vice versa. This will happen in the more general setting.

[3]For now, just think of ideal vertices as convenient formal tools. The term 'ideal' is used by Babai in the Platonic sense, as in being abstract and separate from the 'real', 'concrete' world, which our graphs inhabit, yet 'casting a shadow' on it. He uses it in a different context, as we will see in the next chapter, but it applies equally well here.

1.3.3. Building up the automorphism group of an edge. The next natural observation is that early termination of the unfolding from the proof of proposition 1.13 gives us successive approximations of $\mathrm{Aut}(X)_e$. Indeed, consider the analogous inclusions $\psi_d : \mathrm{Aut}(X_d)_e \to \mathrm{Aut}(T_d)$ where we stop the construction of the binary tree at depth d; this corresponds to looking at the subgraph $X_d = (V_d, E_d)$ induced by the edges and vertices of all paths through e of length at most d.

The groups $\mathrm{Aut}(X_d)_e$ for small values of d are small, and we can brute-force them; so the natural thing to do is to see how we can get from $\mathrm{Aut}(X_d)_e$ to $\mathrm{Aut}(X_{d+1})_e$. The restriction $\pi_d : \mathrm{Aut}(T_{d+1}) \to \mathrm{Aut}(T_d)$ induces a restriction on the embeddings of the automorphism groups:

$$\pi_d : \mathrm{Aut}(X_{d+1})_e \to \mathrm{Aut}(X_d)_e$$

If we know $\mathrm{Aut}(X_d)_e$, to compute $\mathrm{Aut}(X_{d+1})_e$ it suffices to have generators for $\ker \pi_d$ and preimages of $\mathrm{im}\, \pi_d$ (which will give us coset representatives of $\ker \pi_d$).

So the natural thing is to try to build up $\mathrm{Aut}(X)_e$ in steps. If we look at the graph $X_{d+1} \backslash X_d$, every vertex there is connected to 1, 2 or 3 vertices in X_d. If we let A to stand for the set of all 1-, 2- or 3-element subsets of V_d, we can define a mapping

$$f : V_{d+1} \backslash V_d \to A$$

given by the neighbors. We say $v \sim v'$ are **twins** if $f(v) = f(v')$, i.e. they are indistinguishable from the point of view of X_d. It's immediate that $\ker \pi_d$ is precisely the group generated by transpositions of twins. After a bit of combinatorics, it turns out that

Claim 1.16. $\pi_d(\mathrm{Aut}(X_{d+1})_e)$ is precisely the group of elements of $\mathrm{Aut}(X_d)_e$ which stabilize the set $A_1 \subseteq A$ of fathers with one son, the set $A_2 \subseteq A$ of fathers of twins, and the set $A' \subseteq A$ of new edges $(E_{d+1} \backslash E_d) \cap A$.

Abstracting things, we can partition A according to all intersections of the sets A_1, A_2, A'; then we're looking for those elements of $\mathrm{Aut}(X_d)_e$ that fix each color. This leads us to the string isomorphism problem.

1.4. The string isomorphism problem

Abstracting the situation from the 3-regular case, we arrive at the following problem. Suppose Σ is a finite set which we call the **alphabet**, and we consider **strings** over Σ, which are functions $\mathfrak{x} : \Omega \to \Sigma$. An element $\sigma \in \mathrm{Sym}(\Omega)$ acts on strings by the rule

$$\mathfrak{x}^\sigma(x^\sigma) = \mathfrak{x}(x) \quad \text{for all } x \in \Omega$$

Problem 1.17 (String isomorphism). Given strings $\mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$ between finite sets and a permutation group $G \leq \mathrm{Sym}(\Omega)$, the **string isomorphism problem** consists of finding the set

$$\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) = \left\{ \sigma \in G \;\middle|\; \mathfrak{x}^\sigma = \mathfrak{y} \right\}$$

of isomorphisms $\mathfrak{x} \to \mathfrak{y}$.

In fact, we can reduce GI to SI straight away (but notice that we lose the special filtration we got in the trivalent case):

Proposition 1.18. GI is polynomial-time reducible to SI.

Proof. First reduce as before to Graph-Aut, obtaining a graph $X = (V, E)$. Now encode the graph as a string $\mathfrak{x}$ with underlying set $\Omega = \binom{V}{2}$ over the alphabet $\Sigma = \{0, 1\}$, by letting $\mathfrak{x}(u, v) = 1$ iff $(u, v) \in E$. The group G will be the symmetric group $\mathrm{Sym}(V)$ realized as a permutation group by its action on the pairs Ω. □

In the next sections, we'll explain how the specific instance of string isomorphism we encountered in the 3-regular case can be solved.

1.5. Computing with permutation groups

If the size of the GI problem is n, the group $\mathrm{Aut}(G)$ acts on $\Omega(\sqrt{n})$ points and can have size up to $\sqrt{n}!$, more than exponential in n. Thus working with a full list of its elements is not feasible.

As it turns out, there is a well-developed computational theory of permutation groups (see for example Holt, Eick and O'Brien [**HEO05**]) in which groups are represented compactly by sets of generators. For Babai's algorithm, such subroutines are second nature, and we will sacrifice much of the clarity of his ideas if we stop and justify each group-theoretic computation he does. Thus, it will be important to internalize these ideas now, to the point that we are confident enough we can fill any gaps in their informal applications later on.

In this section we give the relevant algorithms following Holt, Eick and O'Brien [**HEO05**] and Luks [**Luk82**] (however notice that Luks uses left actions). We will try not to make the exposition completely unmotivated, by using the natural problems we encountered when we tried to compute $\mathrm{Aut}(G)_e$ as a guide. In fact, this corresponds more closely to how the theory developed historically - aspects of computational group theory were revitalized and re-analyzed in the context of the GI problem (read Luks [**Luk82**]).

1.5.1. Representing permutation groups. Permutations can be stored as arrays of length n and multiplied in $O(n^2)$ time, so we measure the complexity of algorithms by counting the number of group operations.

OBSERVATION 1.19. Every group G has a generating set of size $\leq \log_2 |G|$, and in fact every *minimal* generating set is of size $\leq \log_2 |G|$.

PROOF. Take any $\sigma_1 \in G$, and initialize $S = \{\sigma_1\}$. While the subgroup $H = \langle S \rangle$ is proper, add an element of $G - H$ to S. Since each time the index $[G : H]$ is reduced by at least a factor of 2, when we terminate S is a generating set and $|S| \leq \log_2 |G|$.

If S is a minimal generating set, we in particular have that it is one of the possible generating sets that the above algorithm returns, hence $|S| \leq \log_2 |G|$. □

COROLLARY 1.20. Every permutation group G of degree n has a minimal generating set of size $\leq \log_2 n! < n^2$.

It is crucial that we have a small *minimal* generating set for every group, rather than just a small *minimum* generating set[4].

Indeed, as we saw in the previous sections, there are natural situations when we need to compute some group by joining generating sets for two groups we already know. If we do that too much, we're in danger of losing a uniform upper bound on the size of a generating set representing a group of a given order. The above proposition fixes that: as we'll see, there are efficient algorithms for membership, so we can efficiently clean up any generating set by repeatedly throwing out redundant generators.

DEFINITION 1.21. A **description** of a permutation group G of degree n in a computational context is a minimal generating set S of G of uniformly polynomially bounded size.

In other words, $|S| \leq \mathrm{poly}(n)$ for some fixed polynomial; as we just saw, n^2 works[5].

[4]Recall the difference between minimal and minimum: in our case, the former is a generating set without redundancy, and the latter is a generating set of the smallest possible size.

[5]Babai [**Bab86**] gave a non-trivial argument that $2n$ is sufficient, but any polynomial works for our purposes.

1.5.2. Orbits and blocks. Luks reduction needs efficient algorithms for processing orbits and minimal block systems, and these are quite simple to come up with:

PROPOSITION 1.22. Given the description of a permutation group G on a set Ω, we can in polynomial time compute:

(1) the orbits of G;
(2) a minimal block system if G is transitive.

PROOF. For the orbits, construct the graph on vertex set Ω with edges (x, x^σ) for every generator σ. Its connected components are precisely the orbits of G, and we can find them efficiently by any graph traversal algorithm.

For the blocks, fix $x \in \Omega$ and observe that the smallest G-block containing x and some $y \neq x$, if it exists, is precisely the orbit of $\{x, y\}$ under the induced action of G on the set of unordered pairs $\Omega^{(2)}$ of Ω. We can find the orbits as in (1) in polynomial time, and compute the induced action on the block system by just looking at the action on the blocks that each generator induces. Indeed, under a surjective group homomorphism, a generating set maps to a generating set – so the generating set we use throughout the algorithm doesn't grow.

Repeating this at most $\log_2 n$ times (since the effective domain Ω decreases by at least a factor of 2 every time), we reach a minimal block system. □

1.5.3. Order and membership. In this section, we introduce the central tool of computational group theory, the Schreier-Sims algorithm, which allows efficient computation of order and membership, and provides a so-called **strong generating set** that helps with many other tasks.

Let G be a permutation group on $\Omega = \{x_1, \ldots, x_n\}$. Suppose we want to approximate it in a manner similar to what we did with $\mathrm{Aut}(X)_e$ before. So we need to come up with subgroups that take into account increasing chunks of the G-action. It's natural to let G_i be the pointwise stabilizer $G_{(x_1,x_2,\ldots,x_i)}$ and consider the chain

$$\mathrm{id} = G_n = G_{n-1} \leq \ldots \leq G_1 \leq G_0 = G$$

If we manage to find sets of coset representatives C_i of G_{i+1} in G_i, we will be able to compute $|G| = |C_0||C_1| \ldots$, and test membership by 'filtering' a supposed $\sigma \in G$ to the stabilizers by finding in which coset of G_{i+1} in G_i it resides.

This chain has two crucial properties that allow us to do so efficiently:

- each next group is not much bigger than the previous. Indeed, $[G_i : G_{i+1}] \leq n - i$ by the orbit-stabilizer theorem, since G_i is supported on $n-i$ points, and $G_{i+1} = (G_i)_{x_{i+1}}$.
- Given $\sigma \in G_i$, we can efficiently (in fact in constant time) tell if $\sigma \in G_{i+1}$.

Such a chain allows us to efficiently break down G into manageable chunks.

OBSERVATION 1.23. There is a simple randomized algorithm to find the coset representatives C_i.

PROOF. We just guess enough elements of G_i until we (have very likely) found C_i. We can tell if σ, σ' are in the same right coset by just checking whether $\sigma'\sigma^{-1} \in G_{i+1}$. The number of guesses will be small, sharply concentrated around $O(m \log m)$ where $m = [G_i : G_{i+1}]$ by the coupon collector problem. This is the key observation behind Babai's Las Vegas algorithm for graph isomorphism of colored graphs with bounded color classes [**Bab79**]. □

The above algorithm can be replaced by a deterministic analogue, as was noticed by Sims [**Sim70**] some time before Babai. The core routine executes the membership test we described above on elements of G using incomplete versions of the sets $C_0, C_1, \ldots$ and updates them accordingly when it detects failure (it 'teaches the C_i by examples'):

```
1: procedure FILTER(σ)                                   (: to be used on σ ∈ G :)
2:     for i = 0, 1, ..., n − 1 do
3:         for τ ∈ C_i do                                (: if we reach this point, σ ∈ G_i :)
4:             if στ^{-1} ∈ G_{i+1} then
5:                 σ ← στ^{-1}                           (: no new information, descend :)
6:             else C_i ← C_i ∪ σ; return (: new coset representative found, can't descend further
   :)
7:             end if
8:         end for
9:     end for
10: end procedure
```

There are more efficient ways to proceed from here, but for a polynomial-time algorithm it suffices to apply FILTER enough times. Namely,

```
1: procedure STRONG GENERATORS(G = ⟨S⟩)
2:     Initialize C_0 = C_1 = ... = {id}
3:     for σ ∈ S do
4:         FILTER(σ)
5:     end for
6:     while some C_i increases do
7:         for 0 ≤ i < j ≤ n, σ_i ∈ C_i, σ_j ∈ C_j do
8:             FILTER(σ_i σ_j)
9:         end for
10:    end while
11:    return C_0, C_1, ..., C_{n−1}.
12: end procedure
```

PROOF (SKETCH). In this proof, permutations with index i stand for elements of C_i. It is an invariant of the algorithm that $C_i \subseteq G_i \leq G$, and hence $\langle C_0, C_1, \ldots\rangle \subseteq G$. Our goal is to show that $G = C_{n-1}C_{n-2}\ldots C_0$.

Filtering the generators has the effect that whatever group the C_i generate, it contains G. Filtering C_iC_j for $i < j$ has the purpose of turning $C_{n-1}\ldots C_0$ into a group by allowing us to reorder the indices in a product $\sigma_{n-1}\ldots\sigma_0\tau_{n-1}\ldots\tau_0$. Indeed, if the sets $C_0, C_1, \ldots$ stabilize, this means that whenever we have a product $\sigma_i\sigma_j$ with $i < j$, it survives the filtration, and can be written as $\sigma'_j\sigma_k$ for some $k < j$. Similar ideas work for inverses.

Finally, there are $O(n^2)$ pairs $i < j$, and $|C_0| + |C_1| + \ldots \leq n + (n-1) + \ldots = O(n^2)$ at any time, so we can only have polynomially many rounds with a polynomial amount of work each before the C_i stabilize. □

Here's the promised membership routine:

```
1: procedure MEMBERSHIP(G = ⟨S⟩ ≤ Sym(Ω), σ ∈ Sym(Ω))
2:     Run FILTER on σ, keeping track of coset representatives encountered
3:     if at any point we get false on line 4 of FILTER then
4:         return false
5:     else return expression for σ in terms of generators
6:     end if
7: end procedure
```

1.5.4. Working with homomorphisms. In our situations (such as an induced action on a set of blocks), we often have some homomorphism $\varphi : G \to H$ for a G we know. Under these assumptions, we will need the following:

PROPOSITION 1.24. Given $G = \langle S \rangle$ and a polynomial-time computable homomorphism $\varphi : G \to H$, we can in polynomial time:

- find a strong generating set $\overline{S}$ of $\varphi(G)$ with a set of preimages T of each $\overline{\sigma} \in \overline{S}$;
- **lifting:** for $\overline{\sigma} \in H$ find $\sigma \in \varphi^{-1}(\overline{\sigma})$ if one exists.
- find a strong generating set for $\ker \varphi$.

PROOF. The image $\varphi(S)$ generates $\varphi(G)$, and it can be refined to a strong generating set by our algorithm STRONG GENERATORS. In parallel with executing STRONG GENERATORS on $\varphi(S)$, we keep preimages of all current elements of the C_i. When the procedure finishes, we have a full set of preimages of the strong generators.

Next, given $\overline{\sigma} \in H$, we run MEMBERSHIP on $\varphi(G)$ and $\overline{\sigma}$, which either rejects lifting or gives us an expression $\overline{\sigma}$ in terms of strong generators of $\varphi(G)$, which we pull back via T to get $\sigma \in \varphi^{-1}(\overline{\sigma})$. □

1.6. Luks reduction

In this section we show divide-and-conquer techniques for dealing with intransitive and imprimitive groups in the SI problem, following closely Luks [**Luk82**]. Intransitivity leads to reducing Ω according to the orbits, and imprimitivity leads to reducing G according to the induced action on the blocks.

DEFINITION 1.25 (Windows). Let $\Delta \subseteq \Omega$ and $S \subseteq \mathrm{Sym}(\Omega)$. Then we define

$$\mathrm{Iso}_S^{\Delta}(\mathfrak{x}, \mathfrak{y}) = \left\{ \sigma \in S \;\middle|\; \forall u \in \Delta : \mathfrak{x}(u) = \mathfrak{y}(u^\sigma) \right\}$$

We will usually consider window isomorphisms when the window is S-invariant. Our definitions have some immediate consequences:

PROPOSITION 1.26. For $\sigma \in G$, windows $\Delta, \Delta' \subseteq \Omega$ and subsets $S, S' \subseteq \mathrm{Sym}(\Omega)$, we have the following identities:

(1) **shift identity:** $\mathrm{Iso}_{S\sigma}^{\Delta}(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_S^{\Delta}(\mathfrak{x}, \mathfrak{y}^{\sigma^{-1}})\sigma$, and in particular $\mathrm{Iso}_{S\sigma}(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_S(\mathfrak{x}, \mathfrak{y}^{\sigma^{-1}})\sigma$
(2) **union identity:** $\mathrm{Iso}_{S\cup S'}^{\Delta}(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_S^{\Delta}(\mathfrak{x}, \mathfrak{y}) \cup \mathrm{Iso}_{S'}^{\Delta}(\mathfrak{x}, \mathfrak{y})$
(3) **chain rule:** $\mathrm{Iso}_G^{\Delta\cup\Delta'}(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_{G_1}^{\Delta'}(\mathfrak{x}, \mathfrak{y}^{\sigma^{-1}})\sigma$, where $G' = \mathrm{Iso}_G^{\Delta}(\mathfrak{x}, \mathfrak{y}) = G_1\sigma$ and Δ, Δ' are G-invariant

PROOF. Straightforward from the definitions. □

EXAMPLE 1.27 (Automorphisms vs isomorphisms). Observe that knowing automorphisms is almost like knowing isomorphisms. Suppose $\sigma \in \mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$; then we have the coset relation

$$\mathrm{Aut}_G(\mathfrak{x})\sigma = \mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$$

since $\mathfrak{x}^\tau = \mathfrak{x} \iff \mathfrak{x}^{\tau\sigma} = \mathfrak{x}^\sigma = \mathfrak{y}$.

The following example illustrates the general procedure for reducing the group when we discover canonical structure.

EXAMPLE 1.28 (Alignment). Suppose we're in a situation where we know that any $\sigma \in \mathrm{Aut}_G(\mathfrak{x}, \mathfrak{y})$ must induce an isomorphism of some structures $\mathfrak{X}, \mathfrak{Y}$ canonically associated with $\mathfrak{x}$

and $\mathfrak{y}$ respectively. Furthermore, suppose we can compute $\mathrm{Iso}_G(\mathfrak{X}, \mathfrak{Y})$ efficiently (for example, think vertex colorings). Then we can efficiently reduce the group G since

$$\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_{\mathrm{Iso}_G(\mathfrak{X},\mathfrak{Y})}(\mathfrak{x}, \mathfrak{y}) \text{ and } \mathrm{Iso}_G(\mathfrak{X}, \mathfrak{Y}) = \mathrm{Aut}_G(\mathfrak{X})\sigma$$

for some σ in analogy with the previous example, and then we can apply the shift identity to get

$$\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y}) = \mathrm{Iso}_{\mathrm{Aut}_G(\mathfrak{X})}(\mathfrak{x}, \mathfrak{y}^{\sigma^{-1}})\sigma$$

We refer to this as **alignment**.

PROPOSITION 1.29 (Weak Luks reduction). Let $H \leq G$ and σ_i be representatives of the right cosets of H. Finding $\mathrm{Iso}^{\Delta}_G(\mathfrak{x}, \mathfrak{y})$ reduces to $[G : H]$ instances of finding $\mathrm{Iso}^{\Delta}_H(\mathfrak{x}, \mathfrak{y}^{\sigma})$ for various $\sigma \in G$.

PROOF. Write G as a union $\cup_{i=1}^{m} H\sigma_i$ of right cosets of H. By the union identity and the shift identity from proposition 1.26 we get

$$\mathrm{Iso}^{\Delta}_G(\mathfrak{x}, \mathfrak{y}) = \bigcup_{i=1}^{m} \mathrm{Iso}^{\Delta}_{H\sigma}(\mathfrak{x}, \mathfrak{y}) = \bigcup_{i=1}^{m} \mathrm{Iso}_H(\mathfrak{x}, \mathfrak{y}^{\sigma_i^{-1}})\sigma_i$$

□

The idea when we're faced with a primitive group is to enumerate it and pass to the kernel every time; this is expensive, and that's why the order of primitive groups we encounter matters.

PROPOSITION 1.30 (Strong Luks reduction). Let $\Delta \subseteq \Omega$ be G-invariant and $\{B_1, \ldots, B_m\}$ be a system of G-blocks on Δ. Let $\psi : G \to \overline{G} \leq \mathrm{Sym}(m)$ be the induced action on the blocks, and $N = \ker \psi \leq G$. Then the computation of $\mathrm{Iso}^{\Delta}_G(\mathfrak{x}, \mathfrak{y})$ reduces to $m|\overline{G}|$ instances of finding $\mathrm{Iso}^{B_i}_{M_i}(\mathfrak{x}, \mathfrak{y}^{\sigma_i})$ for certain subgroups $M_i \leq N$ and $\sigma_i \in G$.

PROOF. Coset representatives of N in G can be found by lifting each element of $\overline{G}$. We then apply weak Luks reduction with $N \leq G$. Since N fixes stabilizes every block, we then apply the chain rule m times, treating the blocks $B_1, \ldots, B_m$ as the windows. □

1.6.1. GI for 3-regular graphs revisited. With this machinery, we're ready to finish the 3-regular case. Observe that we need to solve the SI problem for a 2-group $\mathrm{Aut}(X_d)_e$. Any minimal block system of a 2-group consists of exactly 2 blocks and the subgroup stabilizing the blocks is of index 2, hence strong Luks reduction will finish in polynomial time.

1.7. Reducing to Johnson groups

SO, the obstacles are large primitive groups.

THEOREM 1.31 (3.2.1.). Let $G \leq \mathrm{Sym}(n)$ be a primitive group of order $|G| > n^{1+\log_2 n}$ where n is big enough. Then G has a normal subgroup N with $[G : N] \leq n$ such that N has a system of imprimitivity on which N acts as a Johnson group $\mathrm{Alt}(k)^{(t)}$ or $\mathrm{Sym}(k)^{(t)}$ with $k \geq \log_2 n$. Both N and the system of imprimitivity can be found in polynomial time.

The proof it not immediately relevant to understanding of the main algorithm. According to Babai, it has been folklore for a long time; it follows straightforwardly from previous work of Cameron [**Cam81**] and Maroti [**Mar02**] classifying large primitive permutation groups, as well as previous work by Babai, Luks and Seress [**BLS87**] on computational aspects of certain groups.

The usefulness of this theorem comes from the fact that the subgroup N from the statement is easily accessible from G, so we can apply strong Luks reduction (lemma ??) to reduce the case of large primitive groups to Johnson groups.

To summarize our work so far, here is an algorithm that leaves us with the essential obstacle, which we tackle in the next chapters.

procedure REDUCE TO JOHNSON($G \leq \mathrm{Sym}(\Omega), \mathfrak{x}, \mathfrak{y} : \Omega \to \Sigma$)
  if G is intransitive **then**
    Reduce via chain rule
  end if
  Compute a minimal block system $\mathcal{B}$ of G with $m = |\mathcal{B}|$
  Let $\mathfrak{G} \leq \mathrm{Sym}(\mathcal{B})$ be the induced permutation action on the blocks, and $N \leq G$ the kernel of the epimorphism $G \to \mathfrak{G}$.
  if $|\mathfrak{G}| < m^{1+\log_2 m}$ **then** reduce G to N by strong Luks reduction.
  else reduce $\mathfrak{G}$ to a Johnson group by weak Luks reduction (: theorem 1.31 :)
  end if
  return $\Omega, G, \mathcal{B}, \mathfrak{G}$ (: G has a giant action on the blocks $\mathcal{B}$:)
end procedure

1.8. Canonicity and individualization

We noticed that we want to compare graphs for isomorphism by repeatedly refining their structure by colors, with the understanding that any isomorphism must respect the colorings. We will do analogous things with various other structures throughout the proof, so it helps to give some more generality at this point.

More generally, whenever we compare two structures $\mathfrak{X}$ and $\mathfrak{Y}$ for isomorphism, a **canonical** transformation means distinguishing substructure $\mathfrak{X}'$ on $\mathfrak{X}$ and $\mathfrak{Y}'$ on $\mathfrak{Y}$ such that

$$\mathrm{Iso}(\mathfrak{X}', \mathfrak{Y}') = \mathrm{Iso}(\mathfrak{X}, \mathfrak{Y}),$$

i.e. *we don't lose any isomorphisms by distinguishing the substructure.* Here substructure can mean a coloring of vertices/edges/tuples, a partition of the vertex set (so that any isomorphism must preserve the partition but may permute the blocks), or a **colored partition**, where the vertices are colored and each color class is further partitioned.

EXAMPLE 1.32. Let $X = (V, E)$ be a graph. Define $C_i = \left\{ v \in V \;\middle|\; \deg(v) = i \right\}$ and the equivalence relation

$$x \sim y \iff (x, y) \in \mathrm{Aut}(X)$$

Then the color classes C_i together with the equivalence classes of $\sim$ induce a canonical colored partition on V. Indeed, any isomorphism $X \to Y$ respects degree, and moreover must map a $\sim$ equivalence class in X to a $\sim$ equivalence class in Y. Notice that while the colors have an ordering that isomorphisms must preserve, the $\sim$-equivalence classes in the same color class can be permuted.

More generally, we say a structure $\mathfrak{X}'$ is **canonically associated** with $\mathfrak{X}$ if every isomorphism $\mathfrak{X} \to \mathfrak{Y}$ induces also an isomorphism $\mathfrak{X}' \to \mathfrak{Y}'$ in some obvious from the context manner.

EXAMPLE 1.33. For two groups $H_1, H_2 \leq G$ and a G-action $\varphi : G \to \mathrm{Sym}(\Gamma)$, the largest orbit Δ of H_1 is canonically associated with H_1 with respect to isomorphisms $H_1 \to H_2$ induced by G-conjugation whenever Δ is a **dominating** orbit, i.e. $|\Delta| > |\Gamma|/2$. Indeed, if H_1 and H_2 are conjugate via σ, $\overline{\sigma} = \varphi(\sigma)$ maps the H_1 orbits to H_2 orbits. Since the largest orbit is unique, one must be mapped to the other.

Now suppose that we perform naive vertex coloring on X and we get stuck. We can try to continue by artificially introducing irregularity: pick an arbitrary $v \in V(X)$ and give it a unique

color. Then if Y_w denotes the result of the same operation on Y with the vertex w (and the same color), we have

$$\mathrm{Iso}(X, Y) = \bigcup_{w \in V(Y)} \mathrm{Iso}(X_v, Y_w)$$

This idea is called **individualization**. It generalizes to arbitrary structures, and consists of distinguishing an arbitrary substructure $\mathfrak{X}' \subseteq \mathfrak{X}$, so that the isomorphism problem reduces to as many isomorphism problems as there are possible images of $\mathfrak{X}'$ in $\mathfrak{Y}$. We say that the structure $\mathfrak{X}$, $\mathfrak{X}'$ is **canonical with respect to** v.

There is a way to formalize canonicity in the language of category theory, but for the most part it's very easy to understand intuitively what it means. One way to think about it is that it is a refinement of the structure of $\mathfrak{X}$ that is *natural*, in the sense that it doesn't depend on any arbitrary choices. Another way to think about it is to imagine how the isomorphism of the refined structures follows from the isomorphism of the initial structures.

Canonicity is more of a way of thinking than a mathematical statement, so it's better to get a feel for it through examples; and while the categorical framework is more succinct, in all our situations we can deal with canonicity from first principles.

Example 1.34. For a graph X, the set of induced subgraphs on k vertices isomorphic to a given graph Z on k vertices is canonically associated with X. The ordered tuple of vertices of a dominating connected component is *not* canonically associated: an isomorphism need not preserve the order in general.

1.8.1. Categorical formulation. Here we sketch how canonicity is formalized through categories. All the categories we consider are concrete, and all their morphisms are isomorphisms; and most of the time, they will only have two objects, associated with the two strings we are testing for isomorphism. The definition of canonicity is then simply expressed as being the mapping of objects associated with a function between such categories:

Definition 1.35. Let $\square(\cdot)$ denote the underlying set functor for a concrete category. Then a **canonical embedding** of objects from category $\mathcal{D}$ into objects from category $\mathcal{C}$ is a functor $F : \mathcal{C} \to \mathcal{D}$ such that $\square(F(X)) \subseteq \square(X)$ and $Ff : F(X) \to F(Y)$ is the restriction of f to $\square(F(X))$.

For example, this formalizes our concept of colored partition. Individualization over a category $\mathcal{C}$ is handled analogously, by considering *all* possibilities at once; this can be implemented by e.g. a category in which to every object X of $\mathcal{C}$ we assign an object which is the set of all individualized versions of X.

1.9. Twins and symmetry defect

A final note we need to make is about twins, which we already mentioned in the previous section. The **twin equivalence relation** on a structure $\mathfrak{X}$ is given by

$$x \sim y \iff \tau = (x, y) \in \mathrm{Aut}(\mathfrak{X})$$

including the case $x = y$ when $(x, x) = \mathrm{id}$ trivially belongs to $\mathrm{Aut}(\mathfrak{X})$. We call the equivalence classes of the twin relation **symmetric sets**.

The *symmetry defect* is a robust partitioning tool that we'll use:

Definition 1.36. We say that $\mathfrak{X}$ has **(relative) symmetry defect** α if the largest symmetric set in $\mathfrak{X}$ has (relative) size $\leq 1 - \alpha$.

Example 1.37. A non-trivial (not empty and not clique) regular graph has symmetry defect $\leq 1/2$, since by simple counting there can be no independent set of size $> 1/2$

CHAPTER 2

Local certificates

[1]As we saw at the end of the previous chapter, we have efficiently reduced the SI problem to the case when G is transitive, and in particular there is a giant representation $\varphi : G \to \mathrm{Sym}(\Gamma)$ for some set Γ with $N = \ker \varphi$.

In the SI problem, we saw that significant reduction of the group or Ω leads to efficient computation. We're going to try to do the same in the case of a giant representation, except that we will recurse on both Γ and Ω. Often, we will be looking for the following objects:

DEFINITION 2.1. A **colored partition** of Ω is a coloring of the elements of Ω along with a partition of each colored class. It is a **colored equipartition** if each color class is equipartitioned (notice that the parts in different color classes don't need to be the same size).

We say a colored partition Π is **admissible** if the blocks have size ≥ 2, where possible (i.e. in the color classes which have more than one element). We denote the size of the largest block by $\rho(\Pi)$. A **colored α-partition** is an admissible colored partition with $\rho(\Pi) \leq \alpha n$.

Note: it is allowed that a color class is partitioned with only one part!

Recall that, with Luks reduction, the natural thing do in such a situation was to go over all permutations $\overline{\sigma} \in G^\varphi \cong G/N$, lift them to some preimage $\sigma \in \varphi^{-1}(\overline{\sigma})$, and solve the isomorphism problems for the arising cosets $N\sigma$. However, our giant action $G^\varphi \geq \mathrm{Alt}(\Gamma)$ is too big for that when m is large enough.

Under the above strategy, we are essentially saying that *whatever* G-isomorphism $\mathfrak{x}^\sigma = \mathfrak{y}$ there might be, it must map to *something* in G^φ – so we check everything. But this is completely oblivious to the strings we have! Can we somehow use information about invariants of $\mathfrak{x}$ and $\mathfrak{y}$ to reject some symmetries in G^φ *before* we reduce to the kernel N? It starts to sound like we have some 'pushforward' SI instance on Γ for which we want to reject isomorphisms.

2.1. The strategy: pushing to the ideal domain

This leads us to try to break the symmetry on Γ by finding canonical structures on Γ associated with the strings $\mathfrak{x}$ and $\mathfrak{y}$. This is Babai's first idea: to carry our toolkit for finding canonical structures over to the set Γ, which he calls the **ideal** domain.

Right now, it's hard to visualize what's actually happening on the ideal domain and what these canonical structures might look like, since our strings live on Ω. So let's try to push our (mental) picture to Γ.

OBSERVATION 2.2. If $\sigma \in \mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$, $\mathrm{Aut}_G(\mathfrak{x})^\varphi$ and $\mathrm{Aut}_G(\mathfrak{y})^\varphi$ are conjugate via $\varphi(\sigma)$.

PROOF. Suppose $\mathfrak{x}^\sigma = \mathfrak{y}$, and let $\tau \in \mathrm{Aut}_G(\mathfrak{x})$. Then

$$\mathfrak{y}^{\sigma^{-1}\tau\sigma} = \mathfrak{x}^{\tau\sigma} = \mathfrak{x}^\sigma = \mathfrak{y}$$

[1]Babai claims that we need G to be imprimitive for us to apply the entire local certificates algorithm from this section, and treats the primitive case by other (easier) methods. I still don't understand why the local certificates approach fails in this case.

and hence $\mathrm{Aut}_G(\mathfrak{y}) = \sigma^{-1}\,\mathrm{Aut}_G(\mathfrak{x})\sigma$. Taking images under φ, we get the claim. □

In other words, the $\overline{\sigma} \in \mathrm{Sym}(\Gamma)$ it suffices to lift are the ones which induce a permutation isomorphism between the permutation groups $\mathrm{Aut}_G(\mathfrak{x})^\varphi$ and $\mathrm{Aut}_G(\mathfrak{y})^\varphi$. So we can think of these permutation groups as the strings' avatars on Γ, and we need to come up with ways to constrain the possible isomorphisms between them.

From this point of view, Babai's philosophy is a natural generalization of what we've been doing so far: we will attempt to find canonical structures on Γ induced by the images of the automorphism groups.

Here is a high-level description of the strategy:

(1) If the automorphisms are sufficiently rich, they affect a significant chunk of Γ. Then either:
   (a) find canonical structures, such as largest orbits and constituents of orbital configurations, to significantly partition Γ or Ω; or
   (b) the automorphisms induce a giant, from which we induce a nice coloring of Ω.

(2) If the automorphisms are not sufficiently rich, there is a significant chunk of Γ on which their action has a large symmetry defect. In particular, we can find a canonical relational structure with large symmetry defect on that chunk, and from it extract obstacles to isomorphism that reduce Γ.

This is admittedly vague, and we will clarify it over the next sections. First of all, unlike in the case where we compare two explicit graphs for isomorphism, we can't really get our hands on the groups $\mathrm{Aut}_G(\mathfrak{x})^\varphi$! So to show that they are rich, we need to come up with some efficient lower bound – that is, demonstrate many automorphisms that move many points of Γ around; and to show they are few, we need to give an upper bound that constrains the symmetry defect. We will shortly see the hard part is constructing global symmetry.

2.1.1. Discovery of canonical structure and pulling back to Ω. Looking a little bit ahead, here's what we can expect (though we will see how we get there in a more motivated manner).

Proposition 2.3 (Canonical structure, informal). We can roughly divide the canonical structures we will discover in three classes, according to their effect:

(1) colored partitions of Γ with no dominant color. In this case we can pull back the colored partition to Ω by remembering that Ω has a block system corresponding to a Johnson action. This will have the effect of significantly reducing $n = |\Omega|$.
(2) canonically embedded structures called *Johnson schemes* for which we can test isomorphism efficiently. In this case we reduce significantly $m = |\Gamma|$.
(3) colored partitions of Γ with a dominant color class C on which G acts as a giant. This significantly reduces Γ, but puts us in the same miserable situation of having to deal with a giant action $G \to \mathrm{Sym}(\Gamma')$ for some significant chunk of Γ. **Notice that this giant action doesn't come with a canonical Johnson action, unlike the one we obtained in 1.7**. If we are to deal with situation (1), which may come up when we recurse on the current case (3), we need to be able to find a canonical way to spot a Johnson action from a giant representation. This is the content of Babai's 'Main structure theorem', Theorem 8.5.1. in **[Bab15]**.

2.1.2. A note on running time. We won't be concerned with the complexity estimates too much, however, by reading the justificaition of the algorithms it should be fairly evident that we're making significant progress. We will also not be too explicit about how we take canonical structures into account; this is intuitive by example 1.28, and by the fact that isomorphism can be

tested in polynomial time for colored partitions and Johnson schemes, which are the end products of the entire algorithm.

Specifically, we reduce both $n = |\Omega|$, $m = |\Gamma|$ and the group G. Our golden rule for significant reduction is quasipolynomial multiplicative cost: we can reduce the SI problem on domains of size n to $q(n)$ SI problems on domain of size significantly smaller than n, which we take to mean something like $9n/10$.

2.2. Detecting asymmetry, and the definition of fullness

In a sense, bounding the symmetry defect as described above is *easy*. It suffices to find enough subsets $A_i \subseteq \Gamma$ such that

(1) none are too big;
(2) they span a large chunk of Γ;
(3) $\mathrm{Aut}_G(\mathfrak{x})$ does not induce a giant action on any of them.

Then we'll be in good position to find significant irregularity on Γ: a largest strongly symmetric set for $\mathrm{Aut}_G(\mathfrak{x})$ will be of size $< |A_i|$ for all i. An alternative way to view things is that we can be very *demanding* when we look for symmetry, and require that $\mathrm{Aut}_G(\mathfrak{x})$ induce a giant action on various A_i. This motivates the following definition, one of the most essential in the algorithm:

DEFINITION 2.4. We call a subset $A \subseteq \Gamma$ **full** if φ induces a giant representation of the G-automorphisms of $\mathfrak{x}$ on A; in other words, if $\mathrm{Alt}(A) \leq \mathrm{Aut}_G(\mathfrak{x})^A_A$.

2.2.1. Local to global asymmetry. If we find enough sets A_i satisfying (1)-(3) above, we can infer a canonical relational structure with large symmetry defect.

As a first observation, the *set* of subsets $A \subseteq \Gamma$ of a given size which are not full is canonically associated with $\mathfrak{x}$.

How can we exploit that? The non-full *sets* themselves don't carry the non-fullness information; we need to consider the *tuples* on each A, and color them according to whether one can be carried to another. This will be a relational structure aware of non-fullness: it won't have a strongly symmetric set of size $\geq k$ whenever we have a non-full set A with $|A| = k$. At this point, our relational structure is not canonical since the assignment of colors is arbitrary. To force canonicity, we can explicitly compare the non-full sets for $\mathfrak{x}$ and $\mathfrak{y}$, compute the G-isomorphisms between them, and build the relational structures from that refined information.

Note that we haven't yet explained how to test fullness.

2.3. Building global symmetry, and the affected/unaffected dichotomy

As we discussed in section 2.1, if we fail to find enough local asymmetry, we *need* to get our hands on the automorphism groups in order to compute canonical structures. We will do this by finding constructive evidence of fullness (local symmetry), and aggregating it to global symmetry. The aggregation consists of simply taking the group generated by the local symmetries; the hard part is to construct automorphisms of the strings that certify fullness of test sets. Our ability to do so is the key step of the entire algorithm.

DEFINITION 2.5. In this section we often say $A \subseteq \Gamma$ is a **test** set, which means it has size $\log_2 n$ unless otherwise specified.

So suppose we have some $A \subseteq \Gamma$ which we want to test for fullness, so that we're looking for a subgroup of G_A of automorphisms of $\mathfrak{x}$ which induces a giant action on A. First, we observe that

OBSERVATION 2.6. The representation $\psi_A : G_A \to \mathrm{Sym}(A)$ is giant.

PROOF. Since $|A| \leq |\Gamma| - 2$ for large enough m, the alternating group on Γ can permute A in all ways. □

Suppose $H \leq G_A$ induces a giant on A; which points $x \in \Omega$ can H move? If $\mathrm{Alt}(A) \not\leq H_x^{\psi_A}$, H certainly needs to move x. This motivates a simple definition with deep consequences:

DEFINITION 2.7. Given a representation $\psi : K \to \mathrm{Sym}(S)$ for $K \leq \mathrm{Sym}(\Omega)$ and $H \leq K$, we say that $x \in \Omega$ is **affected** by (H, ψ) if $\mathrm{Alt}(S) \not\leq H_x^{\psi}$. We denote the set of points affected by (H, ψ) by $\mathrm{Aff}(H, \psi)$.

Given the crucial role of this definition in the entire algorithm, we should stop and think about its consequences for a bit.

OBSERVATION 2.8. It is immediate from the definition that if $H \leq G_A$ induces a giant on $\mathrm{Sym}(A)$, so does every stabilizer H_x for an unaffected point $x \notin \mathrm{Aff}(G_A, \psi_A)$.

OBSERVATION 2.9. The set $\mathrm{Aff}(H, \psi)$ is invariant under H. Indeed, every two stabilizers H_x, H_y for x, y in the same H-orbit are conjugate, so ψ maps them to conjugate subgroups in $\mathrm{Sym}(S)$; since $\mathrm{Alt}(S)$ and $\mathrm{Sym}(S)$ are self-conjugate, x is affected iff y is. Thus we can speak of **affected orbits**.

OBSERVATION 2.10. If $H_1 \leq H_2$, $\mathrm{Aff}(H_1, \psi) \supseteq \mathrm{Aff}(H_2, \psi)$, since $(H_1)_x^{\psi} \leq (H_2)_x^{\psi}$

Life would be very simple if we could stabilize all the unaffected points *at once* and still have the above observation; then the affected points would be where the action takes place. Babai's key insight is that for large enough A this is the case:

THEOREM 2.11 (Unaffected stabilizer theorem). Let $K \leq \mathrm{Sym}(\Omega)$ be a permutation group and $\varphi : K \to \mathrm{Sym}(S)$ a giant representation. Assume $k > \max\{8, 2 + \log_2 n\}$, and let $D \subseteq \Omega$ be the set of elements not affected by φ. Then $\mathrm{Alt}(S) \leq G_{(D)}^{\varphi}$.

The first important thing to notice is that

OBSERVATION 2.12. The unaffected stabilizer theorem says there exist affected points, since $G_{(\Omega)}^{\varphi} = \mathrm{id}$.

Now suppose that we look at $W = \mathrm{Aff}(G_A, \psi_A)$. Our wishful running hypothesis is that if we consider the automorphisms of $\mathfrak{x}$ permuting the G_A-invariant set W, their set of affected points is again W, and they induce a giant on $\mathrm{Sym}(A)$.

If this happens to be the case, that is, if both $\mathrm{Aff}(\mathrm{Aut}_G^W(\mathfrak{x})_A, \psi_A) = W$ and $\mathrm{Alt}(A) \leq \mathrm{Aut}_G^W(\mathfrak{x})_A^{\psi_A}$, then by stabilizing $\mathrm{Aut}_G^W(\mathfrak{x})_A^{\psi_A}$ on $\overline{W}$ we in fact get a giant action on A by automorphisms of the complete string $\mathfrak{x}$. This means that A is full!

If this fails to happen, there are two possible reasons:

- $\mathrm{Alt}(A) \not\leq \mathrm{Aut}_G^W(\mathfrak{x})_A^{\psi_A}$. Since W is G_A-invariant, we have $\mathrm{Aut}_G(\mathfrak{x})_A \leq \mathrm{Aut}_G^W(\mathfrak{x})_A$, and thus $\mathrm{Alt}(A) \not\leq \mathrm{Aut}_G(\mathfrak{x})_A^A$, which certifies that A is *not full*.
- $W' = \mathrm{Aff}(\mathrm{Aut}_G^W(\mathfrak{x})_A, \psi_A)$ is bigger than W. Then we repeat the process with our new hypothesis being that $\mathrm{Aff}(\mathrm{Aut}_G^{W'}(\mathfrak{x})_A, \psi_A) = W'$.

In the next section, we will formalize this procedure.

2.4. The local certificates algorithm

Babai refers to the procedure we just outlined as 'growing the beard', the beard being the set W of affected points. At the first iteration, W is non-empty by the unaffected stabilizer theorem.

In subsequent iterations, the restricted automorphism group $H(W) = \mathrm{Aut}_G(\mathfrak{x}^W)_A$ decreases and the set of affected points increases by observation 2.10. When beard stops growing, we either find a refutation in the face of local symmetry $\mathrm{Aut}_G(\mathfrak{x}^W)_A \geq \mathrm{Aut}_G(\mathfrak{x})_A$ which is already not giant, or global symmetry $\mathrm{Aut}_G(\mathfrak{x})^A_A \geq \mathrm{Alt}(A)$ by stabilizing unaffected points.

1: **procedure** LOCAL CERTIFICATES(test set A)
2: $W \leftarrow \emptyset$
3: **while** $\mathrm{Alt}(A) \leq H(W)^A$ and $\mathrm{Aff}(H(W), \psi_A) \neq W$ **do**
4: $W \leftarrow \mathrm{Aff}(H(W), \psi_A)$
5: Recompute $H(W) \leftarrow \mathrm{Aut}^W_G(\mathfrak{x})$
6: **end while**
7: $W(A) \leftarrow W$
8: **if** $\mathrm{Alt}(A) \leq H(W)^A$ **then**
9: $K(A) \leftarrow H(W)_{(\overline{W})}$ # $K(A) \leq \mathrm{Aut}_G(\mathfrak{x})$
10: **return** certificate of fullness $W(A), K(A)$
11: **else** $M(A) \leftarrow H(W)^A$
12: **return** certificate of non-fullness $W(A), M(A)$
13: **end if**
14: **end procedure**

2.4.1. Recomputing $H(W)$: the affected orbit lemma. It's not clear we've accomplished anything yet, since for all we know W might turn out to be all of Ω on the first iteration, and we would need to compute $\mathrm{Aut}_G(\mathfrak{x})_A$ from scratch, which is exactly what we are trying to avoid.

So suppose we're on line 5 of LOCAL CERTIFICATES. Let W_{old} be the value of W before line 4 is executed, and W_{new} after. Then at line 5, $H(W) = H(W_{old}) = \mathrm{Aut}^{W_{old}}_G(\mathfrak{x})_A$, and we want to replace it by $H(W_{new}) = \mathrm{Aut}^{W_{new}}_G(\mathfrak{x})_A \leq H(W_{old})$. The idea is to search for $H(W_{new})$ in $H(W_{old})$ by strong Luks reduction to the kernel N of the representation $H(W_{old}) \to \mathrm{Sym}(A)$, which is a giant (since we reached line 5).

Babai shows the orbits of N are small, making the reduction efficient.

THEOREM 2.13 (Affected orbits). Suppose $\varphi : H \to \mathrm{Sym}(A)$ is a giant action, and $|A|$ is large enough. Then for any affected orbit $\Delta \subseteq \mathrm{Aff}(H, \varphi)$, $\ker\varphi$ is not transitive on Δ and each orbit of $\ker\varphi$ in Δ has size $\leq |\Delta|/|A|$.

PROOF. For $x \in \Delta$ we have $\mathrm{Alt}(A) \not\leq H^\varphi_x$. Letting $N = \ker\varphi$, we have $|\Delta| = [H : H_x]$ and $|x^N| = [N : N_x]$. Furhtermore, applying the correspondence theorem and the second isomorphism theorem, we have $[H : NH_x] = [H^\varphi : H^\varphi_x]$ and

$$[N : N_x] = [N : (N \cap H_x)] = [NH_x : H_x] = \frac{|\Delta|}{[H^\varphi : H^\varphi_x]} \leq |\Delta|/|A|$$

since for large enough A, maximal non-giant subgroups of $\mathrm{Alt}(A)$ and $\mathrm{Sym}(A)$ both have index $\geq |A|$. □

Now we're ready to give the procedure to recompute $H(W)$.

1: **procedure** RECOMPUTE $H(W)$
2: $N \leftarrow \ker(\psi_A : H(W_{old}) \to \mathrm{Sym}(A))$ # ψ_A is giant
3: $L \leftarrow \emptyset$
4: **for** $\overline{\sigma} \in \psi_A(H(W_{old}))$ **do**
5: Lift $\overline{\sigma}$ to $\sigma \in H(W_{old})$
6: $L \leftarrow L \cup \mathrm{Aut}^{W_{new}}_{N\sigma}(\mathfrak{x})$ # strong Luks reduction

end for
$H(W) \leftarrow L$
end procedure

Here's what we've accomplished:

THEOREM 2.14 (Local certificates). Let $A \subseteq \Gamma$ be a test set. Then by making quasipolynomially many calls to SI on instances of size $O(n/\log_2 n)$ and performing additional quasipolynomial computation, we decide if A is full and return

(1) A subgroup $K(A) \leq \mathrm{Aut}_G(\mathfrak{x})$ such that $K(A)^A$ is giant if A is full;
(2) A subgroup $M(A) \leq \mathrm{Sym}(A)$ such that $\mathrm{Aut}_G(\mathfrak{x})_A \leq M(A)$ and $M(A)$ is not giant if A is not full.

2.5. Aggregation of certificates: general strategy

In this section, we find different canonical structures on Γ depending on whether many or few of our test sets are full.

The set of local certificates is canonically associated with the string $\mathfrak{x}$, and this will guide our construction of canonical structures. We let $F(\mathfrak{x})$ be the group generated by the positive certificates $K(A)$ over all full test sets.

PROPOSITION 2.15. Isomorphic strings $\mathfrak{x} \cong \mathfrak{y}$ would lead to permutationally isomorphic groups $F(\mathfrak{x}), F(\mathfrak{y})$.

PROOF. Recall that if $\mathfrak{x} \cong \mathfrak{y}$, the automorphism groups $\mathrm{Aut}_G(\mathfrak{x})$ and $\mathrm{Aut}_G(\mathfrak{y})$ are permutationally isomorphic. Under this permutation isomorphism and its pushforward to Γ, each test set A for $\mathfrak{x}$ has some corresponding test A' for $\mathfrak{y}$, so that when we run the LOCAL CERTIFICATES algorithm on A and A' in parallel, the quantities (subsets of Ω and Γ, subgroups of $\mathrm{Sym}(\Omega)$ and $\mathrm{Sym}(\Gamma)$) correspond under this isomorphism. If we want to be extra careful, we can show this by induction, but the point is that at no stage in the algorithm did we make an arbitrary choice that would break the correspondence. □

We need to understand how our certificates permute the points of Γ, and to this end we define the 'pushforward' versions of the affected/unaffected dichotomy to Γ:

DEFINITION 2.16. For a full test set A, let $\widetilde{W}(A)$ denote the **sideburn** of A, the set of affected points $\mathrm{Aff}(K(A)^\Gamma, \pi)$ where $\pi_A : K(A)^\Gamma \to \mathrm{Sym}(A)$ is the restriction map.

Implications of the unaffected stabilizer/affected orbits theorems naturally carry over to Γ:

PROPOSITION 2.17. For a test set A, we have that

(1) $A \subseteq \widetilde{W}(A)$: the points of A are affected by $K(A)^\Gamma$
(2) The group $K(A)^\Gamma$ has minimal degree[2] at most $|\widetilde{W}(A)|$.
(3) The orbits of $\ker \pi_A$ on $\widetilde{W}(A)$ have length $\leq m/k$.

PROOF. Part (1) is because stabilizing some $x \in A$ prevents the alternating group from acting on A (i.e., x is affected). For (2), the points $U = \Gamma \backslash \widetilde{W}(A)$ are unaffected, so the pointwise stabilizer $K(A)^\Gamma_{(U)}$ restricts to a giant on A by the unaffected stabilizer theorem. Part (3) follows from the affected orbits theorem and part (1). □

[2]Here **minimal degree** is the smallest possible support of an element of a permutation group.

2.6. Aggregation of positive certificates

,Here is the procedure in the case when the positive certificates dominate, interspersed with the justification:

procedure AGGREGATE POSITIVE CERTIFICATES
(: **Positive certificates of big support and small orbits** :)
if the support of F^Γ has size $\geq m/5$ and no orbit of F^Γ has size $> 4m/5$ **then**
We get a colored 4/5-partition $\mathfrak{X}$ of Γ. The color classes are $(\mathrm{supp}(F^\Gamma), \Gamma \setminus \mathrm{supp}(F^\Gamma))$, and the blocks are the orbits on $\mathrm{supp}(F^\Gamma)$.
return $\mathfrak{X}$.
end if
(: **Positive certificates of big support with a large orbit** :)
if F^Γ has an orbit $C \subseteq \Gamma$ of size $|C| > 4m/5$ **then**
Being the biggest orbit of the group F^Γ canonically associated with $\mathfrak{x}$, C is canonical.

(: The key observation is that we can find canonical structure with large symmetry defect from the orbital configuration of F^Γ. As we saw, the orbital will be a UPCC unless F^Γ acts 2-transitively. However, a classical theorem from group theory tells us that since the minimal degree of F^Γ is small, it has to be a giant:

THEOREM 2.18 (Bochert's theorem). If $G \leq \mathrm{Sym}(n)$ is doubly transitive and not giant, its minimal degree is $n/8$. :)

(: The minimal degree of F^Γ is at most the minimal degree of its subgroup $K(A)^\Gamma$ for some test set A, and by proposition 2.17 the latter is at most $|\widetilde{W}(A)|$. Since $\widetilde{W}(A)$ is canonically associated with A, if $\widetilde{W}(A)$ is big we can individualize A and induce a significant partitioning; we deal with the case when $|\widetilde{W}(A)| \geq m/5$ for some A in this manner as a sub-case: :)
(: **Long beards** :)
if $|\widetilde{W}(A)| \geq m/5$ for some A **then**
Since $\widetilde{W}(A)$ is canonically associated with A, we can get a significant partition $\mathfrak{X}$ at moderate cost by individualizing each vertex of A. While the sideburn might be big, by proposition 2.17 the orbits of $\ker \pi_A$ on it have size $\leq m/k \ll m/5$, and thus the $\ker \pi_A$ orbits on Γ give us a 4/5-partition, with the color classes being $(\widetilde{W}(A), \Gamma \setminus \widetilde{W}(A))$, and the partition coming from the orbits of $\ker \pi_A$.
return $\mathfrak{X}$
end if
(: **Short beards, uniprimitive case** :)
if F^C is not doubly transitive **then**
Since C is an orbit of F^C, the orbital configuration $\mathfrak{X} = (C, R_1, \ldots, R_r)$ of F^C on C is a canonical UPCC for $r \leq m$, as we saw in subsection 1.1.5. Every off-diagonal constituent $\mathfrak{X}_i$ is a non-trivial biregular graph. Since the colors are arbitrary, the ordering of $\mathfrak{X}_i$ is **not** canonical, but the set of constituents is.
Individualize some $\mathfrak{X}_i$ at multiplicative cost $r \leq m$.
return $\mathfrak{X}_i$. (: $\mathfrak{X}_i$ has symmetry defect $\geq 1/2$:)
end if
(: **Short beards, giant case** :)
if F^C is doubly transitive **then**
F^C is giant on C by Bochert's theorem and short beards.
(: Recall C is canonical being the largest orbit :)

29: **return** Canonical coloring $\mathfrak{X} = (C, \Gamma\backslash C)$ and giant representation $G \to \mathrm{Sym}(C)$. (: we deal with this case in section 2.8 :)
30: **end if**
31: **end if**
32: **end procedure**

2.7. Canonically pulling back structure to Ω

We want to pull the canonical colorings we obtained in the AGGREGATE POSITIVE CERTIFICATES algorithm back to a coloring of Ω, by remembering that the giant action $\varphi : G \to \mathrm{Sym}(\Gamma)$ comes from a Johnson group. Specifically, the blocks are labeled by subsets $\binom{\Gamma}{t}$ for some $t \geq 1$ and partition Ω so that G acts on the block system as a Johnson group (recall section 1.7).

However, as we discussed in item (3) of proposition 2.3, we might face situations when we need to deal with a giant action that doesn't come from some canonical Johnson action, such as on line 29 of the algorithm. Babai deals with this by establishing a converse of sorts to the way we obtained a giant representation from a Johnson action: a canonical Johnson action can be found from any giant representation on a sufficiently big set.

THEOREM 2.19 (Main structure theorem). Let $G \leq \mathrm{Sym}(\Omega)$ and $\varphi : G \to \mathrm{Sym}(S)$ be a giant representation with $|S| \geq 2\log_2 n$. Then:

(1) For every $x \in \Omega$ there is a unique $T(x) \subseteq S$ such that $|T(x)| < |S|/4$ and
$$\mathrm{Alt}(S)_{(T(x))} \leq G_x^\varphi \leq \mathrm{Sym}(S)_{T(x)}$$
(2) An element $x \in \Omega$ is affected by (G, φ) iff $|T(x)| \geq 1$
(3) $|T(x)| = t_\Delta$ is invariant within an orbit Δ
(4) At least one orbit is affected
(5) For every orbit Δ, the equivalence relation $x \sim y \iff T(x) = T(y)$ splits Δ into $\binom{|S|}{t_\Delta}$ blocks of imprimitivity, labeled by the set $\binom{S}{t_\Delta}$. These are the **standard blocks** for φ. The action of G on the set of standard blocks in Δ is $\mathrm{Alt}(S)^{(t_\Delta)}$ or $\mathrm{Sym}(S)^{(t_\Delta)}$. If Δ is affected, this is a Johnson action, and the kernel of the action on the blocks is $\ker\varphi$
(6) The standard blocks form the unique largest system of imprimitivity on which the kernel of the G-action is $\ker\varphi$

The last item is what ensures canonicity; moreover observe that the standard blocks can be computed efficiently.

Finally, here's the claim that allows us to lift colorings to Ω canonically:

PROPOSITION 2.20 (Effect of coloring on t-tuples). Given a coloring of Γ with color classes $C_1, \ldots, C_l$, it induces a canonical coloring of $\binom{\Gamma}{t}$ where the color of $T \in \binom{\Gamma}{t}$ is the tuple of its color distribution $(|T \cap C_1|, |T \cap C_2|, \ldots, |T \cap C_l|)$. Then any α-coloring of Γ induces a $\max(2/3, \alpha)$-coloring of $\binom{\Gamma}{t}$.

PROOF. The essential bound we need is that $\binom{m_1}{t_1}\binom{m_2}{t_2} \leq 2/3\binom{m}{t}$ whenever $m_1 + m_2 = m, t_1 + t_2 = t$ and $t_i > 0$, which follows by standard arguments[3]. □

[3]You should find it very believable though, since by counting $\binom{m}{t} = \sum_{i=0}^{t}\binom{m_1}{i}\binom{m_2}{t-i}$ has many terms similar to $\binom{m_1}{t_1}\binom{m_2}{t_2}$.

2.8. Processing giant automorphisms

2.8.1. General strategy. In this section we sketch why it's easy to spot when a giant action $\varphi : G \to \mathrm{Sym}(\Gamma)$ is giant on $\mathrm{Aut}_G(\mathfrak{x})$ for transitive G, and if this is the case, to find $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$.

Suppose we have generators S for $\mathrm{Alt}(\Gamma)$. We can efficiently lift them to coset represenatives σ_i of $\varphi^{-1}(\overline{\sigma}_i)$, and then determination of $\varphi^{-1}(\overline{\sigma}_i) \cap \mathrm{Aut}_G(\mathfrak{x})$ reduces to a string isomorphism problem with group $N = \ker \varphi$ by shifting, since $\varphi^{-1}(\overline{\sigma}_i)$ is a coset $N\sigma_i$ and $\mathrm{Aut}_G(\mathfrak{x}) = \mathrm{Iso}_G(\mathfrak{x}, \mathfrak{x})$.

The crucial point in terms of efficiency is that Ω is affected, since by the unaffected stabilizer theorem there are affected points, and G is transitive. By the affected orbits theorem, it follows that the orbits of N are of equal small size $\leq n/m$; this allows efficient application of weak Luks reduction.

Now we have found $\varphi^{-1}(\overline{\sigma}) \cap \mathrm{Aut}_G(\mathfrak{x})$; if some of them is empty, it's not possible that φ is giant on $\mathrm{Aut}_G(\mathfrak{x})$; however, if none of them are empty, this means $\mathrm{Aut}_G(\mathfrak{x})$ induces a giant, and moreover it is (almost) generated by the elements of the cosets $\varphi^{-1}(\overline{\sigma}_i) \cap \mathrm{Aut}_G(\mathfrak{x})$. Indeed, it is if $\mathrm{Aut}_G(\mathfrak{x})^{\varphi} = \mathrm{Alt}(\Gamma)$ and almost is if it is the whole $\mathrm{Sym}(\Gamma)$; we can deal with the latter case by weak Luks reduction with cosets of $\mathrm{Alt}(\Gamma)$ in $\mathrm{Sym}(\Gamma)$.

We swept some technicalities about cosets that come up when we want to determine $\mathrm{Iso}_G(\mathfrak{x}, \mathfrak{y})$ under the rug, but they can be dealt with in the spirit of example 1.27

2.8.2. Giant automorphisms on big orbit. We can apply this to deal with the result of line 29 of the AGGREGATE POSITIVE CERTIFICATES algorithm, by lifting the coloring to Ω as described in section 2.7, and processing the windows starting from the window corresponding to the dominant color, which satisfies the assumptions of subsection 2.8.1, which means that even though it's big we have a special procedure to solve string isomorphism on it. The remaining windows are processed in the usual way, which is efficient as they are small.

2.9. Aggregation of negative certificates

The remaining case is that F^{Γ} has a large 'defect' as witnessed by a large set $|D| < 4m/5$ of fixed points of F^{Γ}. So any test set $A \subseteq D$ is not full. Here we'll execute the strategy outlined in subsection 2.2.1.

As we discussed, we will build the wanted canonical k-ary relational structures by explicitly forcing canonicity. This will happen by incorporating information about which ordered test sets A for $\mathfrak{x}$ can go to which ordered test sets A' for $\mathfrak{y}$. In addition we want the relational structure to be aware of non-fullness, so we also compare tests sets for the same string as well.

Recall that we have a transitive giant action $\varphi : G \to \mathrm{Sym}(\Gamma)$, and let $t = t_{\Omega}$ be given from the structure theorem 2.19. We can get away with computing the local isomorphisms on $W(A)$ and the standard blocks from the structure theorem 2.19, because they are canonically associated with A and already are aware of non-fullness. This can be done in a manner analogous to the way we recomputed $H(W) = \mathrm{Aut}_G^W(\mathfrak{x})_A$ during the LOCAL CERTIFICATES algorithm.

We need to introduce some notation:

DEFINITION 2.21. For a giant action $\varphi : G \to \mathrm{Sym}(\Gamma)$ of a transitive group G and $S \subseteq \Gamma$, let $\Omega(S)$ be the union of the standard blocks corresponding to the elements $\binom{S}{t}$.

NOTATION 2.22. For a string $\mathfrak{x} : \Omega \to \Sigma$ and a test set $A \subseteq \Gamma$, define $\mathfrak{x}_A : \Omega \to \Sigma \times \{0, 1\}$ by

$$\mathfrak{x}_A(u) = \begin{cases} (\mathfrak{x}(u), 1), & \text{if } u \in \Omega(A) \\ (\mathfrak{x}(u), 0), & \text{otherwise} \end{cases}$$

and $\mathfrak{x}^W : \Omega \to \Sigma \cup \{*\}$ by

$$\mathfrak{x}^W(u) = \begin{cases} \mathfrak{x}(u), & \text{if } u \in W \\ *, & \text{otherwise} \end{cases}$$

PROPOSITION 2.23 (Comparison of local certificates). For a pair of test sets $A, A' \subseteq \Gamma$ and strings $\mathfrak{x}, \mathfrak{x}' : \Omega \to \Sigma$ we can compute the isomorphism group $\mathrm{Iso}_G\left(\mathfrak{x}_A^{W(A)}, (\mathfrak{x}')_{A'}^{W'(A')}\right)$ by making quasipolynomially many calls to SI on domains of size $O(n/\log n)$ and performing additional quasipolynomial computation.

REMARK 2.24. The W' in $W'(A')$ refers to the fact that the algorithm that computes W depends on the string $\mathfrak{x}'$.

PROOF. We run the LOCAL CERTIFICATES routine on A, A' in parallel, and when we recompute $H(W)$, we compute in parallel a current approximation to $\mathrm{Iso}_G\left(\mathfrak{x}_A^{W(A)}, (\mathfrak{x}')_{A'}^{W'(A')}\right)$ in a way analogous to how we computed the current $\mathrm{Aut}_G^W(\mathfrak{x})_A$. The idea is similar to how we turn an Iso_G computation to an Aut_G computation in Luks reduction using coset representatives. □

With this we're ready to give the aggregation procedure, to be executed if AGGREGATE POSITIVE CERTIFICATES doesn't terminate:

---

procedure AGGREGATE NEGATIVE CERTIFICATES($\mathfrak{x}, \mathfrak{y}$)
  Rename $\mathfrak{x}_1 \leftarrow \mathfrak{x}$ and $\mathfrak{x}_2 \leftarrow \mathfrak{y}$.
  $D_i \leftarrow$ set of fixed points of $F(\mathfrak{x}_i)$
  (: **Since AGGREGATE POSITIVE CERTIFICATES didn't terminate,** $|D_1| > 4m/5$:)
  if $|D_1| \neq |D_2|$ **then**
    reject isomorphism, **return** $\emptyset$
  end if
  (: **Construct canonical** k**-ary relational structures** :)
  $\mathcal{P}_i \leftarrow D_i^{\langle k \rangle}$ where $S^{\langle k \rangle}$ denotes the set of ordered k-tuples of distinct elements of S.
  $\mathcal{P} \leftarrow (\mathcal{P}_1 \times \{1\}) \coprod (\mathcal{P}_2 \times \{2\})$.
  (: W_i **is the beard** W **with respect to** $\mathfrak{x}_i$:)
  Compute $\mathrm{Iso}_G\left((\mathfrak{x}_i)_A^{W_i(A)}, (\mathfrak{x}_j)_{A'}^{W_j(A')}\right)$ for all pairs of test sets A, A' by Proposition 2.23
  For a k-tuple $\vec{u} = (u_1, \ldots, u_k) \in \mathcal{P}_i$, let $A(\vec{u}) = \{u_1, \ldots, u_k\}$ be its underlying test set.
  Define a relation over the elements of $\mathcal{P}$ by

$$(\vec{u}, i) \sim (\vec{v}, j) \iff \exists \sigma \in \mathrm{Iso}_G\left((\mathfrak{x}_i)_A^{W_i(A)}, (\mathfrak{x}_j)_{A'}^{W_j(A')}\right) \text{ such that } \vec{u}^\sigma = \vec{v}$$

  where $A = A(\vec{u}), A' = A(\vec{v})$.
  Observe that $\sim$ is an equivalence relation (easy and believable: Proposition 2.25)
  Let $Q_1, \ldots, Q_r$ be the equivalence classes of $\sim$.
  Define the k-ary relational structures

$$\mathfrak{X}_i = (D_i, Q_1 \cap \mathcal{P}_i, \ldots, Q_r \cap \mathcal{P}_i)$$

  (: **Then** $\mathfrak{X}_i$ **is canonically associated with** $\mathfrak{x}_i$**, proposition 2.26** :)
  Since no $A \subseteq D_i$ is full for $\mathfrak{x}_i$, $\mathfrak{X}_i$ has symmetry defect $\geq |D_i| - k > m/2$, proposition
  return $\mathfrak{X}_1, \mathfrak{X}_2$
end procedure

Here are the promised proofs:

PROPOSITION 2.25. The relation $\sim$ from the AGGREGATE NEGATIVE CERTIFICATES algorithm is an equivalence relation.

PROOF. This follows because we have a category with objects the labeled test sets (A, i) for $i \in \{1, 2\}$ and morphisms the elements of $\mathrm{Iso}_G\left((\mathfrak{r}_i)_A^{W(A)}, (\mathfrak{r}_j)_{A'}^{W(A')}\right)$. However, it's easy to see it from first principles:

- $\mathrm{id} \in \mathrm{Iso}_G((\mathfrak{r}_1)_A^{W(A)}, (\mathfrak{r}_1)_A^{W(A)}) = \mathrm{Aut}_G((\mathfrak{r}_1)_A^{W(A)}, (\mathfrak{r}_1)_A^{W(A)})$
- If $\sigma \in \mathrm{Iso}_G\left((\mathfrak{r}_i)_A^{W(A)}, (\mathfrak{r}_j)_{A'}^{W(A')}\right)$, $\sigma^{-1} \in \mathrm{Iso}_G\left((\mathfrak{r}_j)_{A'}^{W(A')}, (\mathfrak{r}_i)_A^{W(A)}\right)$ and $\vec{v}^{\sigma^{-1}} = \vec{u}$
- If $(\vec{u}, i) \sim (\vec{v}, j)$ via σ and $(\vec{v}, j) \sim (\vec{w}, l)$ via τ, σ and τ are composable G-string isomorphisms, and the composition $\sigma\tau$ certifies that $(\vec{u}, i) \sim (\vec{w}, l)$.

□

PROPOSITION 2.26. $\mathfrak{X}_i$ defined on line 17 are canonically associated with $\mathfrak{r}_i$.

PROOF. To illustrate how canonicity works from first principles, we can be very explicit about this; in the end, it's just an elaborate change of coordinates.

Let $\sigma \in \mathrm{Iso}_G(\mathfrak{r}_1, \mathfrak{r}_2)$. Observe that D_i is canonically associated with $\mathfrak{r}_i$, so σ projects to $\overline{\sigma} = \varphi(\sigma)$ taking D_1 to D_2. Now suppose $(\vec{u}, 1) \sim (\vec{v}, 1)$ for $A = A(\vec{u}), A' = A(\vec{v})$. As we noted before in observation 2.2, since σ is an isomorphism, it induces a conjugation between the automorphism groups $\mathrm{Aut}_G(\mathfrak{r}_1)^\Gamma$ and $\mathrm{Aut}_G(\mathfrak{r}_2)^\Gamma$. Conjugation in permutation groups is just a 'change of coordinates', a renaming of the points. The LOCAL CERTIFICATES algorithm and related routines will then produce 'conjugate quantities' when we run them on $\mathfrak{r}_1$ and $\mathfrak{r}_2$; this can be shown by induction on each step of the algorithms. In particular, the coset

$$C_{A,A'} := \mathrm{Iso}_G\left((\mathfrak{r}_1)_A^{W_1(A)}, (\mathfrak{r}_1)_{A'}^{W_1(A')}\right)$$

is conjugate via $\overline{\sigma}$ to the coset

$$C_{B,B'} = \mathrm{Iso}_G\left((\mathfrak{r}_2)_B^{W_2(B)}, (\mathfrak{r}_2)_{B'}^{W_2(B')}\right)$$

where $B = \overline{\sigma}(A), B' = \overline{\sigma}(A')$ are the test sets corresponding to A, A' under the isomorphism σ. Now let $\tau \in C_{A,A'}$ be such that $\vec{u}^\tau = \vec{v}$; then we have that

$$(\vec{u}^{\overline{\sigma}})^{\overline{\sigma}^{-1}\tau\overline{\sigma}} = \vec{v}^{\overline{\sigma}} \text{ and } \overline{\sigma}^{-1}\tau\overline{\sigma} \in C_{B,B'}$$

and hence $\vec{u}^{\overline{\sigma}} \sim \vec{v}^{\overline{\sigma}}$. But the isomorphism σ induces the map $\vec{w} \mapsto \vec{w}^{\overline{\sigma}}$ on tuples, which means that σ respects $\sim$; this is precisely the meaning of canonicity in this context. □

PROPOSITION 2.27. $\mathfrak{X}_i$ defined on line 17 has relative symmetry defect $\geq 1/2$.

PROOF. Suppose $A \subseteq D_i$ is a set of twins; it follows that for $\vec{u}$ with $A(\vec{u}) = A$, every permutation of $\vec{u}$ is the restriction of some G-automorphism of $\mathfrak{r}_i^{W_i(A)}$ to A; but this means A is full, hence $|A| \leq k - 1 < k$. Thus $\mathfrak{X}_i$ has symmetry defect $> |D_i| - k > m/2$ in our range of parameters. □

This completes the aggregation of local certificates.

CHAPTER 3

The design lemmas

In this chapter, we deal with the aggregate of negative certificates, by showing how the reduction of k-ary relational structures to Johnson schemes and colored partitions works.

This has two parts: the design lemma reduces k-ary structures to UPCCs, and the extended design lemma reduces UPCC to Johnson schemes for which we can test isomorphism efficiently (and thus perform alignment, as outlined in example 1.28

DEFINITION 3.1. For $t \geq 2, k \geq 2t+1$, the **Johnson scheme** $\mathfrak{J}(k,t)$ is a symmetric coherent configuration (each constituent is undirected) with vertex set $\binom{\Gamma}{t}$ for some $|\Gamma| = k$, and relation R_i of the pairs (T_1, T_2) for which $|T_1 \backslash T_2| = i$.

We prove the basic design lemma in full; the extended design lemma has a longer proof which however uses the same basic techiques, and many of the regularity results we proved for binary coherent configurations in chapter 1.

Also, the design lemma uses the k-dimensional version of WL refinement without proof of its nice properties, and it is interesting to see how we can avoid having to prove them using first principles. This will also give us a strategy to avoid WL altogether for the purposes of the entire algorithm (even though knowing how WL works guides us).

3.1. The design lemma

THEOREM 3.2. Let $1/2 \leq \alpha < 1$ and $\mathfrak{X} = (\Omega, R_1, \ldots, R_r)$ be a k-ary relational structure with $n = |\Omega|$ vertices, $k \leq n/2$, and relative strong symmetry defect $\geq 1 - \alpha$. We can in time $n^{O(k)}$ find a sequence S of $\leq k-1$ points such that individualizing each element of S gives us either

(a) a canonical relative to S colored α-partition of Ω, or
(b) a canonically relative to S embedded UPCC $\mathfrak{X}^*$ on $W \subseteq \Omega$ with $|W| \geq \alpha n$.

We want to find nice regular things, and the first simplification is to regularize $\mathfrak{X}$ by using a generalization of Weisfeiler-Leman refinement.

3.2. k-dimensional canonical WL refinement

The machinery we developed in section 1.1 on binary configurations and coherence generalizes to k-ary structures.

DEFINITION 3.3. A **k-ary relational structure** is the obvious generalization of a binary relational structure. We say that a k-ary relational structure is a **k-ary configuration** if

(1) The relations partition Ω^k
(2) If $c(x_1, \ldots, x_k) = c(x'_1, \ldots, x'_k)$ the relations $i \sim j \iff x_i = x_j$ and $i \sim' j \iff x'_i = x'_j$ are the same
(3) For every i and $\pi \in \mathrm{Sym}(k)$, $R_i^\pi = R_j$ for some j.

The way the edge coloring algorithm from 1.1 generalizes to k dimensions is as follows (see for example Immerman and Lander [**IL90**]): color each induced ordered subgraph[1] on k vertices according to its isomorphism type, and refine the color of $\vec{u}$ according to the number of tuples of 'conic' neighborhoods

$$\big\{(x_1, \dots, x_{j-1}, y, x_{j+1}, \dots, x_k) \;\big|\; 1 \le j \le k\big\}$$

of a given color composition. This leads to the generalization of stable configurations:

Definition 3.4. A k-ary configuration is **coherent** if there are structure constants $p_{i_0,\dots,i_k} \in \mathbb{N}_{\ge 0}$ such that

$$c(\vec{x}) = i_0 \implies \big|\big\{y \in \Omega \;\big|\; c(x_1, \dots, x_{j-1}, y, x_{j+1}, \dots, x_k) = i_j\big\}\big| = p(i_0, \dots, i_k)$$

Similarly, **k-dimensional WL refinement**, which we abbreviate as k-WL, is defined by adding the above information to the color of $\vec{x}$ for all possible $i_1, \dots, i_k$.

The core slogan is that

'k-dimensional WL is aware of regularity/irregularity at $O(k)$ scale'

Apart from hand-wavy intuition, this is supported by two major results from the history of GI:

- the theorem of Immerman and Lander [**IL90**] that any two graphs with the same k-WL refinement agree on all *first-order logic with counting* formulas on k variables;
- the negative result of Cai, Furer and Immerman [**CFI92**] that one needs $\Omega(n)$ rounds of k-WL to distinguish all pairs of graphs.

Babai uses k-WL in a very 'freestyle' manner, not as a set of specific statements, but rather as a machine that can give us various sorts of local regularity, just like we saw with 2-WL in 1.1 (such as equipartitions, twins, and often more specific structure) whenever we need it in an argument. Moreover, he doesn't really prove anything about how and why it works or point to a place that does (so apparently the tool is folklore for people working on GI). So if you're a newcomer, reading the paper might leave you distressed. Fortunately Babai also hints at how we can completely avoid k-WL for our purposes (for example, in the proof of the Design Lemma, Theorem 6.1.1. in [**Bab15**]), and this is what we're going to formalize now.

3.3. Avoiding WL: Explicit canonical regularization

The basic idea is quite simple: supposing that k-WL fails to deliver some sort of (canonical!) regularity we want, we can use this failure to *properly* canonically refine our configuration, by explicitly attaching invariants of interest to the colors. Since proper refinement can't keep happening forever, either at some point our regularity will be present or we will hit a discrete configuration.

As long as we can force the regularity in $n^{O(k)}$ time, we will be in the same ballpark complexity; not surprisingly in view of the intuition from the previous section, this runtime corresponds to doing polynomial work for every $O(k)$-sized subset.

For a k-ary configuration $\mathfrak{X}$, let $WL(\mathfrak{X})$ stand for the result of k-WL refinement. Let $\mathscr{R}$ be a predicate on $\mathfrak{X}$ which is true iff $\mathfrak{X}$ has the kind of regularity we want, such that $\mathscr{R}(\mathfrak{D})$ is true for any discrete configuration $\mathfrak{D}$ (one in which every k-tuple has a different color; it is not unique because the order of the colors matters). Notice that this is a reasonable condition to ask from $\mathscr{R}$, since if we're down to a discrete configuration, isomorphism is trivial. In practice, $\mathscr{R}$ will often have a much stronger property,

$$\mathscr{R}(\mathfrak{X}) \text{ is true} \implies \mathscr{R}(\mathfrak{X}') \text{ is true for any (canonical) refinement } \mathfrak{X}' \text{ of } \mathfrak{X} \quad \spadesuit$$

[1] Here ordered means we require isomorphisms to respect the ordering.

This reflects the fact that our regularity boils down to the colors on $\mathfrak{X}$ being 'aware' of the regularity. However, as we shall shortly see in the proof of the design lemma, establishing ♠ is not obvious for some predicates. Suppose also that $\mathscr{R}$ comes with a procedure $\textsc{Refine}(\cdot, \mathscr{R})$ which exploits failure of $\mathscr{R}_i$ by properly canonically refining the relational structure.

The way we will apply explicit canonical regularization in practice will be by collecting a ($O(1)$-long) sequence of predicates $\mathscr{R}_1, \mathscr{R}_2, \dots$ together with their associated refinement procedures during a proof. We then apply regularization 'from the past', so that in the proof we're already looking at the refined version having all the properties we need.

Remark 3.5. It might be somewhat surprising (and confusing) that we don't need our predicates to satisfy ♠; this means that during refinement the property $\mathscr{R}_i$ can come and go, but as we shall soon see it is bound to be true in the end.

The following algorithm formalizes this:

```
1: procedure Explicit canonical regularization(𝔛, ℛ_1, ..., ℛ_α)   (: such that ℛ_i(𝔇) is
   true for all discrete 𝔇 and α = O(1) :)
2:     𝔛 ← WL(𝔛)
3:     while ℛ_i(𝔛) is false for some i do
4:         for i such that ℛ_i(𝔛) is false do
5:             (: properly refine colors by exploiting failure of ℛ_i :)
6:             𝔛 ← Refine(𝔛, ℛ_i)
7:             𝔛 ← WL(𝔛)
8:         end for
9:     end while
10:    return 𝔛
11: end procedure
```

The following proposition sums up the practical consequences of this algorithm:

Proposition 3.6 (Best of both worlds). Suppose that for all i, $\textsc{Refine}(\mathfrak{X}, \mathscr{R}_i)$ induces a proper canonical refinement whenever $\mathscr{R}_i(\mathfrak{X})$ is false, and evaluating $\mathscr{R}_i$ and $\textsc{Refine}(\cdot, \mathscr{R}_i)$ can be done in $n^{O(k)}$ time. Then the Explicit canonical regularization algorithm returns a canonical coherent configuration satisfying $\mathscr{R}_i$ for every i in time $n^{O(k)}$.

Proof. Observe that we cannot enter the while loop more than n^k times, since every time we get a new color (because some $\mathscr{R}_i$ is false and the corresponding Refine gives new colors), and there are at most n^k colors. On the other hand, we also leave the loop after at most n^k iterations since we hit the discrete configuration and all $\mathscr{R}_i$ are true. The amount of work in one iteration of the while loop is $O(n^{O(k)}) = n^{O(k)}$.

We return iff we get past the while loop, and in this case the returned $\mathfrak{X}$ satisfies $\mathscr{R}_i$ for all i. Moreover, since we return either from line 2 or line 7, the result is the WL refinement of some configuration, hence coherent. Finally, since the result is a composition of canonical refinements, it is again canonical. □

The sad truth of life is that lines 3-9 will never be executed[2] if what Babai claims about k-WL is true, but they give us a simple proof of why it works as we want it to.

Example 3.7. Back to 2-WL, suppose we are too lazy to prove that binary coherent configurations are aware of the strong twin relation. Then apply Explicit canonical regularization where $\mathscr{R}$ is the statement that our binary configuration is aware of twins, and Refine consists

[2]just like the entire algorithm described in this essay

of attaching 0 to $c(x, y)$ if x, y are not twins, and 1 if they are. Since for any pair of twins $c(x, y) = c(y, x)$, this will be canonical: any isomorphism of configurations will have to send blue-twin edges to blue-twin edges. Observe that after refining in this way, $\mathscr{R}$ becomes true. The other conditions are easy to check.

Note that the position of the 'twin' color in the tuples encoding the colors will also be canonical, and will in all subsequent refinements carry this information.

One thing that is not clear in Babai's paper is that the input to the design lemma is a k-ary relational structure, but it is treated as if it is a k-ary configuration. Moreover, individualization and explicit canonical regularization also has the effect of potentially breaking some of the axioms of a configuration. We can deal with this in the spirit of EXPLICIT CANONICAL REGULARIZATION:

PROPOSITION 3.8 (Automatic handling of non-configurations). In time $n^{O(k)}$ we can turn a k-ary relational structure to a k-ary configuration. This can be used to augment k-WL, so that it works on any k-ary relational structure.

PROOF. We need to enforce items (1)-(3) of definition 3.3.

1: **procedure** RELATIONAL STRUCTURE TO CONFIGURATION($\mathfrak{X}$)
2: Partition Ω^k according to the relations and their complement, and make new colors for every k-tuple $\vec{x}$ which encode the set of R_i which contain $\vec{x}$. This is clearly canonical.
3: Refine the color of each k-tuple according to its equivalence class under

$$\vec{x} \sim \vec{y} \iff (x_i = x_j \iff y_i = y_j),$$

encoded in some canonical way (e.g., as a k-tuple of integers encoding the equality type where each i appears for the first time after $1, 2, \ldots, i-1$ have appeared). This is again canonical, and preserves the result of the previous step.
4: Refine each $c(\vec{x})$ by the list of colors of $\vec{x}^\pi$ for a canonical enumeration of $\mathrm{Sym}(k)$. This is canonical, and preserves the results of the previous steps, since if $\mathfrak{X}$ is aware of the relation $\sim$ above, any refinement of $\mathfrak{X}$ also is.
5: **end procedure**

The running time is $O(n^k k^k) = n^{O(k)}$. □

This shows that

PROPOSITION 3.9. The EXPLICIT CANONICAL REFINEMENT algorithm works with k-ary relational structures instead of configurations at every stage with the same $n^{O(k)}$ complexity.

PROOF. We can augment the algorithm with the RELATIONAL STRUCTURE TO CONFIGURATION procedure executed whenever we get a new relational structure that is not a configuration at some step in the algorithm. □

3.4. Proving the lemma

The way to obtain a binary structure from a k-ary structure is similar to how we obtain vertex colors in a binary configuration:

DEFINITION 3.10. For a k-ary relational structure $\mathfrak{X} = (\Omega, R_1, \ldots, R_r)$, the **$t$-skeleton** of $\mathfrak{X}$ is the t-ary relational structure $\mathfrak{X}^{(t)} = (\Omega, R_1^{(t)}, \ldots)$ where

$$R_i^{(t)} = \left\{(x_1, \ldots, x_t) \;\middle|\; (x_1, x_2, \ldots, x_t, x_t, \ldots, x_t) \in R_i\right\}$$

and the empty relations $R_i^{(t)}$ are omitted.

OBSERVATION 3.11. The skeleton of a configuration is a configuration, and the skeleton is a canonical construction.

---

procedure SPLIT OR UPCC($\mathfrak{X}$)
  for $S \subseteq \Omega$ with $|S| \leq k-1$ in increasing order of $|S|$ **do**
    $\mathfrak{X}' \leftarrow \mathfrak{X}$ with each element of S individualized
    $\mathfrak{X}_{(S)} \leftarrow$ EXPLICIT CANONICAL REGULARIZATION$(\mathfrak{X}', \mathscr{R}_1, \mathscr{R}_2, \ldots)$ for $\mathscr{R}_i$ to be specified later
    if vertex-color classes of $\mathfrak{X}_{(S)}$ induce α-coloring **then**
      return vertex-color classes
    end if
    $C(S) \leftarrow$ a vertex-color class of size $> \alpha n$ (: canonical dominant class since $\alpha \geq 1/2$:)
    $\mathfrak{X}^*(S) \leftarrow$ restriction of the 2-skeleton $\mathfrak{X}^{(2)}_{(S)}$ (: $\mathfrak{X}^*(S)$ is a homogeneous binary coherent configuration :)
    if $\mathfrak{X}^*(S)$ is imprimitive **then**
      return Colored partition with classes $C(S), \Omega \backslash C(S)$ and blocks on $C(S)$ given by equipartition into connected components of disconnected off-diagonal constituent
    end if (: after this line $\mathfrak{X}^*(S)$ is primitive :)
    if $\mathfrak{X}^*(S)$ is uniprimitive **then**
      return $\mathfrak{X}^*(S)$ (: we found a canonical UPCC :)
    end if
  end for
end procedure

The claim is that this procedure returns. Since the only case we didn't cover is when $\mathfrak{X}^*(S)$ is a clique, it follows that we need to prove the following:

PROPOSITION 3.12. For some S in SPLIT OR UPCC, $\mathfrak{X}^*(S)$ is not the clique configuration.

Assume the contrary. Since $\mathfrak{X}_{(S)}$ refines $\mathfrak{X}_{(\emptyset)}$, it must be the case that $C(S) \subseteq C = C(\emptyset)$, and since everything in S has a unique color in $\mathfrak{X}_{(S)}$, the large color class $C(S)$ is disjoint from S, and hence

$$C(S) \subseteq C \backslash S$$

As S grows, intuitively $C(S)$ shrinks (imagine $S \subseteq S'$; individualizing a larger set should intuitively lead to more refinement, thus smaller classes). So let's look at the first S for which $C(S) \subsetneq C \backslash S$.

PROPOSITION 3.13. There is a smallest S such that $C(S) \subsetneq C \backslash S$ and $|S| < k$.

PROOF. The symmetry defect tells us that since C is large, not all of its vertices are twins, but by assumption $\mathfrak{X}_{(\emptyset)}$ is a clique. So if some two $x, y \in C$ are twins, all of them are since, the color $c(x, y)$ is aware of the twin relation by Example 3.7.

Now take some non-twins $x, y \in C$ witnessed by $\vec{z}$ such that $c(\vec{z}) = i$ but $c(\vec{z}^{(x,y)}) \neq i$. Then after individualizing each vertex of $S = Z \backslash \{x, y\}$ where $Z = \{z_1, \ldots, z_k\}$ it cannot be that x, y are of the same color. This is because WL will be aware of the local asymmetry between x and y: no automorphism of $\mathfrak{X}$ sends x to y, since it has to fix the individualized points S and would thus send $\vec{z}$ to $\vec{z}^{(x,y)}$.

However, we don't need to prove this or believe WL. Let $\mathscr{R}_1$ be the predicate that says that x, y are of different colors whenever the above situation arises, and let REFINE$(\cdot, \mathscr{R}_1)$ be the procedure that when given S, refines the vertex color $c(x)$ of each point by adding its equivalence

class under

$$x \sim y \iff c(\vec{z}) = c(\vec{z}^{(x,y)}) \text{ for all } \vec{z} \text{ with } S = \{z_1, \ldots, z_k\} \backslash \{x, y\}$$

which will be encoded as the list of the above colors. If $\mathscr{R}_1$ fails, this induces a proper refinement. The possible $\vec{z}$ are precisely the k-tuples of elements of S with one or two 'holes', and thus can be canonically enumerated with respect to S. Thus the whole procedure is canonical relative to S, since any isomorphism σ sending $S \to S'$ will be forced to respect the canonical enumeration and the colors. This procedure takes time $O(n^2 k^k)$. □

Going back, let S be as above; it suffices to assume $|C(S)| > \alpha n$ or else we return. We claim that then we get contradiction because we succeed one step before S:

PROPOSITION 3.14. There is some $x \in S$, such that if we let $Q = S \backslash \{x\}$, $\mathfrak{X}^*_{(Q)}$ is not a clique.

PROOF. Assume $\mathscr{X}^*_{(Q)}$ is a clique. The picture is that $C \backslash Q = C(Q)$ by minimality of S, hence $C \backslash Q$ is a color class in $\mathfrak{X}^*_{(Q)}$. Here we play with the symmetry between the points we could have used instead of x to increase Q.

If there exists $x \in S \backslash C$, let B be the vertex-color class of x in $\mathfrak{X}^*_{(Q)}$ and for $y \in B$ let $N(y) = C \backslash Q \backslash C(Q \cup \{y\})$. We consider the hypergraph $\mathcal{H}$ on vertex set the color class $C \backslash Q$ and edges the sets $N(y)$ for $y \in B$. It is regular and uniform. Again, this follows by explicit canonical regularization:

CLAIM 3.15. $\mathcal{H}$ is uniform.

PROOF. Let $\mathscr{R}_2$ be the predicate that checks if $\mathcal{H}$ is uniform. The precise statement is that for every color class in $\mathfrak{X}^*_{(Q)}$ outside $C \backslash Q$, the size of $N(y)$ is uniform across the color class. This predicate can be tested efficiently (it requires considering $O(n)$ vertices, and individualizing an additional vertex on top of Q to compute $N(y)$). The refinement routine consists of adding $|N(y)|$ to the color of y, which induces a proper refinement, and is clearly canonical since C, Q are canonically associated with Q.

Notice that in this case it was not at all obvious whether $\mathscr{R}_2$ has property ♠, but that doesn't matter. □

Analogously we can come up with a predicate $\mathscr{R}_3$ that $\mathcal{H}$ is regular. Since we're assuming $\mathfrak{X}^*_{(Q)}$ is a clique, the off-diagonal color on $C \backslash Q$ is aware of the co-degrees in $\mathcal{H}$, and similar arguments show that in fact every two vertices of $\mathcal{H}$ belong to the same number of edges (the refinement in this case attaches co-degrees to edges over $C \backslash Q$). Hence $\mathcal{H}$ is a balanced incomplete block design, which by Fisher's inequality means that $|B| \geq |C \backslash Q|$, contradiction with the fact that $C \backslash Q$ is a dominating color class.

Now we're reduced to the case when $S \subseteq C$; let $x \in S$ be arbitrary. In this case we find a canonically embedded digraph over $C \backslash Q$. In analogy with the previous argument, we let

$$N(y) = C \backslash (C(Q \cup \{y\}) \cup Q \cup \{y\})$$

and we consider the digraph $X(Q)$ on vertex set $C \backslash Q$ where the out-edges from y point to each element of $N(y)$. Since $X(Q)$ is canonically associated with Q, any isomorphism has to respect it; the key is that WL detects this because $|Q| < k$. In this case, somewhat weirdly, the predicate $\mathscr{R}_4$ will be that $Q \subseteq C$ and $\mathfrak{X}^*_{(Q)}$ is *not a clique*. The refinement procedure will include whether or not $y \to z$ is an edge of $X(Q)$ in the edge colors over $C \backslash Q$. Since $X(Q)$ is a nontrivial graph (neither clique nor empty), this will induce a proper refinement. This completes the proof of the design lemma. □

3.5. The extended design lemma

In this section, we briefly sketch some aspects of the proof of the extended design lemma to illustrate more canonical partitioning ideas.

The main content of the extended design lemma is the following:

THEOREM 3.16. Let $3/4 \leq \alpha < 1$ and $\mathcal{X} = (\Omega, \mathcal{R})$ a k-ary relational structure with $|\Omega| = n$ and relative strong symmetry defect $\geq 1 - \alpha$. Then at multiplicative cost $q(n)n^{O(k)}$, for $q(n)$ a quasipolynomial function, we can find either

(a) a canonical colored α-partition of Ω, or
(b) a canonically embedded nontrivial Johnson scheme on a subset $W \subseteq \Omega$ with $|W| \geq \alpha n$.

This will follow by reducing the following two results to one another recursively:

THEOREM 3.17. Let $\mathfrak{X} = (V; R_1, \ldots, R_r)$ be a UPCC with n vertices and let $2/3 \leq \beta < 1$ be a parameter. Then at quasipolynomial multiplicative cost (independent of β) we can find either:

(a) a canonical colored β-partition of V;
(b) a canonically embedded non-trivial Johnson scheme on a subset V of size $\geq \beta n$.

THEOREM 3.18. Let $X = (V_1, V_2; E, f)$ be a vertex-colored bipartite graph with $|V_1| \geq 2$ and let $2/3 \leq \alpha < 1$ be a parameter. Assume $|V_2| < \alpha|V_1|$. Assume moreover that the symmetry defect of X on V_1 is $\geq 1 - \alpha$. Then at quasipolynomial multiplicative cost (independent of α) we can find either:

(a) a canonical colored α-partition of V_1, or
(b) a canonically embedded nontrivial Johnson scheme on a subset of V_1 of size $\geq \alpha|V_1|$.

First, we sketch how theorem 3.17 reduces to theorem 3.18.

PROPOSITION 3.19. Let $\mathcal{X} = (V, \mathcal{R})$ be a UPCC on n vertices and $2/3 \leq \beta < 1$. Then we can efficiently either

(a) achieve (a) of 3.17, or
(b) compute an instance of 3.18 for which solutions give corresponding solutions to 3.17

PROOF SKETCH. We individualize some $x \in V$; this makes its sets of neighbors C_i in each constituent canonical relative to x. If each component is small, we achieve (a). Otherwise, there is a dominating component C_2. We consider some bipartite constituent Y from C_2 to C_j; it is a semiregular nontrivial graph since $C_2 \gg C_j$ and the out-degree of C_j is one of the small degrees trumped by $|C_2|$.

By easy counting, this means that Y has symmetry defect $\geq 1/2$; if the symmetry defect is $> \alpha$, we apply theorem 3.18; if not, we can do other canonical tricks. □

After that, Babai reduces theorem 3.18 via various canonical partitioning routines like the ones we've seen so far to the case of *homogeneous* coherent configurations: imprimitive, clique, uniprimitive, and explicit Johnson scheme.

In the imprimitive case, a crucial role is played by the Contracting components lemma 1.9 to process the disconnected components of an off-diagonal constituent.

CHAPTER 4

Conclusion

In the course of following Babai's algorithm, we saw how crucial the interplay between group theory and combinatorics was in the ideal domain Γ. The key fact that starts everything going, the unaffected stabilizer theorem, is very special in the sense that the test sets are exactly the right size $\log n$ so that they are both not too big (which allows quasipolynomial time) and not too small (since Babai has counterexamples for the unaffected stabilizer theorem for smaller values). Therefore it seems that a polynomial algorithm for SI would have to be sufficiently different.

On the other hand, we have other interesting open problems lying around, such as whether SI reduces back to GI[1] It is therefore intriguing to see if there's a way to extract a simpler proof from Babai's paper that only shows a quick algorithm for GI[2].

Finally, there is potential in simplifying the SI proof. It's easy to see that our work in chapter 3 on explicit canonical regularization can be used to get completely rid of WL refinement in the proof.

[1]See e.g. `https://rjlipton.wordpress.com/2015/12/07/permutation-problems-with-strings/`

[2]For example, one idea that I've been thinking about over the last day is using the extended design lemma to directly solve isomorphism of binary coherent configurations, which would solve it for graphs.

Bibliography

[Bab79] László Babai. Monte-carlo algorithms in graph isomorphism testing. *Université tde Montréal Technical Report, DMS*, (79-10), 1979.

[Bab86] László Babai. On the length of subgroup chains in the symmetric group. *Communications in Algebra*, 14(9):1729–1736, 1986.

[Bab15] László Babai. Graph isomorphism in quasipolynomial time. *arXiv preprint arXiv:1512.03547*, 2015.

[BLS87] László Babai, Eugene Luks, and Ákos Seress. Permutation groups in nc. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 409–420. ACM, 1987.

[Cam81] Peter J Cameron. Finite permutation groups and finite simple groups. *Bull. London Math. Soc*, 13(1):1–22, 1981.

[Cam03] Peter J Cameron. Coherent configurations, association schemes and permutation groups. *Groups, combinatorics and geometry*, pages 55–72, 2003.

[CFI92] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[DM96] John D Dixon and Brian Mortimer. *Permutation groups*, volume 163. Springer Science & Business Media, 1996.

[Dou11] Brendan L Douglas. The weisfeiler-lehman method and graph isomorphism testing. *arXiv preprint arXiv:1101.5211*, 2011.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM SIGACT News*, 39(3):35–39, 2008.

[HEO05] Derek F Holt, Bettina Eick, and Eamonn A O'Brien. *Handbook of computational group theory*. CRC Press, 2005.

[IL90] Neil Immerman and Eric Lander. *Describing graphs: A first-order approach to graph canonization*. Springer, 1990.

[Irn05] Christophe-André Mario Irniger. Graph matching–filtering databases of graphs using machine learning techniques. 2005.

[Luk82] Eugene M Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of computer and system sciences*, 25(1):42–65, 1982.

[Mar02] Attila Maróti. On the orders of primitive groups. *Journal of Algebra*, 258(2):631–640, 2002.

[Sim70] Charles C Sims. Computational methods in the study of permutation groups. In *Computational problems in abstract algebra*, pages 169–183, 1970.

[SW15] Xiaorui Sun and John Wilmes. Faster canonical forms for primitive coherent configurations. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 693–702. ACM, 2015.

[ZKT85] Viktor N Zemlyachenko, Nickolay M Korneenko, and Regina I Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, 1985.