Université Paris Cité

École doctorale 386 Sciences Mathématiques de Paris Centre
IMJ-PRG
Équipe Logique Mathématique

The Role of Permutation Groups in the Search for a Logic for Polynomial Time.

Par Anatole DAHAN

Thèse de doctorat de Mathématiques : Logique et Fondements de l'Informatique

Dirigée par Arnaud DURAND
Et par Luc SEGOUFIN

Présentée et soutenue publiquement le 1er Juillet 2025

Devant un jury composé de :

Martin GROHE, Pr.	RWTH Aachen, Allemagne	Rapporteur
Mamadou KANTÉ, Pr.-HDR	Université Clermont-Auvergne	Rapporteur
Nadia CREIGNOU, Pr.-HDR	Université Aix-Marseille II	Examinatrice
Anuj DAWAR, Pr.	University of Cambridge	Examinateur
Tamara SERVI, MCF	Université Paris-Cité	Examinatrice
Cristina SIRANGELO, Pr.-HDR	Université Paris Cité	Examinatrice
Charles PAPERMAN, MCF-HDR	Université de Lille	Membre invité
Arnaud DURAND, Pr.	Université Paris Cité	Directeur de thèse
Luc SEGOUFIN, DR	INRIA Paris	Codirecteur de thèse

Titre : La place des Groupes de Permutations dans la Recherche d'une Logique pour le Temps Polynomial.

Mots clefs : Complexité descriptive ; Théorie des modèles finis ; Théorie des groupes ; Logique pour P

Résumé : Du fait des nombreuses difficultés auxquelles la théorie de la complexité fait face depuis des décennies, plusieurs champs de recherche proposent d'étudier ce domaine sous des angles variés, dans l'espoir d'en comprendre les mécanismes profonds. Parmi eux, la Complexité Descriptive fait correspondre des classes de complexité à des logiques. Cependant, cette approche se heurte, elle aussi, à des obstacles importants, et voilà plus de quarante ans que la question, pourtant centrale, de l'existence d'une logique caractérisant le temps polynomial déterministe reste sans réponse. Alors que la logique de point-fixe capture le temps polynomial sur les structures ordonnées, en l'absence d'un tel ordre, aucune logique ne semble satisfaire cette propriété. Face à cette situation, l'étude d'opérateurs algébriques avec lesquels augmenter le pouvoir d'expression de la logique de point-fixe constitue une piste de recherche féconde. Un opérateur ayant été particulièrement étudié est l'opérateur de rang, qui introduit des notions d'algèbre linéaire dans la logique de point-fixe, en permettant le calcul du rang de toute matrice définissable. Mais il a été démontré récemment que la logique de point-fixe avec opérateur de rang ne capture pas le temps polynomial. Dans cette thèse, nous étudions la pertinence d'une autre structure algébrique sur laquelle fonder de tels opérateurs : les groupes de permutations. Pour ce faire, nous identifions une opération fondamentale sur les groupes de permutations, calculer l'ordre du groupe engendré par un ensemble donné de permutations, et étudions l'expressivité de la logique de point-fixe augmentée par cet opérateur d'ordre. Notre résultat principal est que la logique de point-fixe avec opérateur d'ordre est strictement plus expressive que la logique de rang. Plus précisément, nous démontrons que le problème séparant la logique de rang du temps polynomial est définissable dans cette nouvelle logique. Notre technique repose sur la traduction au sein de la logique de point-fixe d'un algorithme de canonisation des structures fondé sur la théorie des groupes. En effet, la théorie des groupes de permutations s'est avérée être un allié de taille dans la résolution des problèmes d'isomorphisme et de canonisation des graphes, ce qui constitue d'ailleurs une de nos motivations premières d'étudier l'opérateur d'ordre. Cependant, l'application de ces techniques au sein de la logique de point-fixe est une entreprise délicate. Notre travail révèle en effet que la représentation même des groupes de permutations par des ensembles de générateurs ne saurait être aussi expressive dans le cadre logique qu'elle ne l'est dans le contexte algorithmique. Il est donc nécessaire de concevoir de nouvelles représentations des groupes. C'est l'étude d'une telle représentation pour les groupes abéliens qui permet la définition d'une procédure de canonisation au sein de la logique d'ordre. Notre travail constitue ainsi une première étude des différentes représentations possibles des groupes de permutations dans un contexte d'invariance par isomorphisme.

Title: The Role of Permutation Groups in the Search for a Logic for Polynomial Time.

Keywords: Descriptive complexity; Finite Model Theory; Group Theory ; Logic for P

Abstract: Given the numerous challenges that complexity theory has faced for decades, several research fields study this domain from various perspectives, in the hope of understanding its underlying mechanisms. Among these, Descriptive Complexity establishes correspondences between complexity classes and logics. However, this approach also encounters significant obstacles, and the central question of the existence of a logic characterizing deterministic polynomial time has remained unanswered for over forty years. While fixed-point logic captures polynomial time on ordered structures, in the absence of such an order, no logic seems to satisfy this property. In light of this situation, the study of algebraic operators to enhance the expressive power of fixed-point logic represents a promising research direction. One operator that has been extensively studied is the rank operator, which introduces linear algebra concepts into fixed-point logic by enabling the computation of the rank of any definable matrix. However, it was recently demonstrated that fixed-point logic with the rank operator does not capture polynomial time. In this thesis, we explore the relevance of another algebraic structure as a foundation for such operators: permutation groups. To this end, we identify a fundamental operation on permutation groups — computing the order of the group generated by a given set of permutations — and investigate the expressiveness of fixed-point logic augmented with this order operator. Our main result is that fixed-point logic with the order operator is strictly more expressive than rank logic. Specifically, we prove that the problem separating rank logic from polynomial time is definable in this new logic. Our technique relies on translating, within fixed-point logic, a canonization algorithm for structures based on group theory. Indeed, permutation group theory has shown great use in the design of efficient algorithms for the graph isomorphism and canonization problems, which was one of our primary motivations for studying the order operator. However, applying these techniques within fixed-point logic is a delicate endeavor. Our work reveals that the representation of permutation groups by sets of generators cannot be as expressive in the logical framework as it is in the algorithmic context. Thus, it is necessary to devise new representations of groups. The study of such a representation for abelian groups enables the definition of a canonization procedure within order logic. Our work thus constitutes a preliminary investigation into the various possible representations of permutation groups in the context of isomorphism invariance.

Résumé

Voilà presque cent ans que furent introduits les premiers modèles de calcul ; ceux-là même qui donnèrent naissance, à la moitié du siècle dernier, aux balbutiements de l'informatique. Cette révolution des fondements des mathématiques engendra donc une avancée technologique majeure : la construction de machines bien plus aptes à calculer que nous. Il est alors naturel d'étudier les bornes de cette aptitude, de comprendre ce qu'une machine peut, et ne peut pas, calculer.

Si l'on appréhende assez bien les limites théoriques et absolues du calcul — ce qui relève de la théorie de la Calculabilité — de multiples zones d'ombre subsistent dès que des limites sont imposées sur les diverses ressources nécessaires au calcul, en particulier le temps (le nombre d'étapes de calcul), l'espace (la quantité de mémoire nécessaire) ou le non-déterminisme (la possibilité de « deviner »). Ceci constitue le champ d'étude de la Théorie de la Complexité.

Face aux nombreuses questions qui y demeurent ouvertes, plusieurs approches ont été développées pour étudier, au travers de prismes spécifiques et divers, les classes de complexité. La complexité de circuits, la complexité algébrique, ou encore la complexité implicite sont tant de démarches. Notre travail s'inscrit dans une telle démarche : la Complexité Descriptive. Ce domaine de recherche entreprend de caractériser des classes de complexité par des *logiques*. Dans ce contexte, la notion de logique doit être entendue en un sens très général : une logique $\mathcal{L}$ est la donnée, pour toute signature Σ d'un ensemble d'*énoncés* $\mathrm{SEN}_{\mathcal{L}}(\Sigma)$, et d'une relation de satisfaction entre énoncés et structures relationnelles $\mathrm{SAT}_{\mathcal{L}}(\Sigma)$. Si φ est un $\mathcal{L}[\Sigma]$-énoncé (i.e. $\varphi \in \mathrm{SEN}_{\mathcal{L}}(\Sigma)$), et $\mathfrak{A}$ une Σ-structure, on écrit $\mathfrak{A} \models \varphi$ lorsque $(\varphi, \mathfrak{A}) \in \mathrm{SAT}_{\mathcal{L}}(\Sigma)$. Les fonctions $\mathrm{SEN}_{\mathcal{L}}$ et $\mathrm{SAT}_{\mathcal{L}}$ doivent toutes deux être décidables. Enfin, on attend d'une logique qu'elle soit *invariante par isomorphisme* : si $\mathfrak{A}$ et $\mathfrak{B}$ sont deux Σ-structures isomorphes, les ensembles d'énoncés satisfaits respectivement par $\mathfrak{A}$ et $\mathfrak{B}$ doivent être identiques.

Ainsi, tout $\mathcal{L}[\Sigma]$-énoncé φ délimite une classe de Σ-structures qui satisfont l'énoncé, dénotée $\mathrm{Mod}(\varphi)$. Intuitivement, une classe de complexité $\mathcal{C}$ est *caractérisée* par $\mathcal{L}$ (on dit aussi que $\mathcal{L}$ *capture* $\mathcal{C}$) si, pour toute signature Σ, et toute classe de Σ-structures $\mathcal{K}$ close par isomorphisme, $\mathcal{K}$ est décidable dans $\mathcal{C}$ si et seulement si $\mathcal{K}$ est définie par un énoncé de $\mathcal{L}$.[1]

Donner une définition générale de la notion de capture est difficile, tant cette notion dépend de la définition de la classe de complexité $\mathcal{C}$ à l'étude. La définition précise de ce que serait une logique capturant P — la classe des problèmes décidables en temps polynomial déterministe — est donnée dans [Gur88], et reproduite à la Définition 1.28.

[1] Par soucis de précision, notons qu'afin d'exclure des cas pathologiques, il est nécessaire que l'implication réciproque soit effective : il doit exister une procédure qui, étant donné un $\mathcal{L}[\Sigma]$-énoncé, produit un algorithme décidant $\mathrm{Mod}(\varphi)$ dans le respect des conditions définissant la classe $\mathcal{C}$.

À titre d'illustration supplémentaire, mentionnons le théorème de Fagin, théorème fondateur de la Complexité Descriptive : NP — la classe des problèmes décidables en temps polynomial *non*-déterministe — est caractérisée par la logique du second-ordre existentielle (SO∃), c'est à dire l'extension de la logique du premier-ordre par quantification existentielle sur des relations.

La recherche d'une logique capturant P constitue un problème ouvert majeur de la Complexité Descriptive. Après cette rapide introduction des termes du problème, il semble important de revenir sur une condition mentionnée plus haut avec une rapidité qui ne rend pas justice à son importance : l'invariance par isomorphisme. En effet, cette condition constitue une composante essentielle de la Complexité Descriptive : peut-on concevoir un modèle du calcul (en temps polynomial) agissant directement sur les structures, et non sur leurs encodages ? Lorsque cette part du problème est évacuée — lorsqu'un encodage de la structure étudiée est définissable — le théorème d'Immerman-Vardi fourni une réponse positive à cette question : la logique de point-fixe, capture P sur toute structure linéairement ordonnée (cet ordre linéaire permettant la définition, au sein de la logique, d'un encodage de la structure, et vidant la condition d'invariance par isomorphisme de tout effet). Cette logique, notée FP, joue un rôle central dans cette thèse et est définie formellement p. 15. Pourtant, si FP capture P sur les structures ordonnées, elle ne peut exprimer le fait d'avoir un domaine de cardinalité paire (propriété pourtant bien décidable en temps polynomial) sur des structures arbitraires, et ne capture donc pas P dans le cas général. La quête d'une logique pour P dans le cas général a mené la communauté à considérer successivement des extensions de FP de plus en plus puissantes pour répondre aux résultats de séparations entre P et les logiques candidates précédentes. D'abord, face à l'inexpressibilité de Parité qui vient d'être mentionnée, un mécanisme de comptage a été ajouté à FP, donnant naissance à la logique FPC (cf. Définition 1.29). Dans [CFI92], Cai, Fürer et Immerman séparent FPC de P. L'étude des structures utilisées (dorénavant appelées *structures de CFI*) dans ce résultat mit en exergue l'incapacité de FPC à exprimer la satisfaisabilité de systèmes d'équations linéaires sur un corps — pourtant à la portée de P via le pivot de Gauss — motivant l'introduction [Daw+09] d'une nouvelle extension de FP munie d'un opérateur permettant de calculer le rang d'une matrice définissable (FP + rk, cf. Définition 1.33). Enfin, Lichter a récemment séparé FP + rk de P [Lic23]. Ce résultat s'appuie sur une généralisation de la construction de Cai-Fürer-Immerman. Là encore, la difficulté du problème utilisé par Lichter réside dans la résolution d'un système d'équations linéaires ; cependant, la construction employée par Lichter permet la définition de systèmes d'équations linéaires sur des anneaux unitaires finis, en lieu des corps finis employés dans la construction originelle. À nouveau, la satisfaisabilité d'un système d'équations linéaires sur un anneau est décidable en temps polynomial (à l'aide de la forme normale de Smith).

Cette progression semble indiquer l'importance des structures algébriques (et des opérations qui peuvent être effectuées sur celles-ci en temps polynomial) dans la recherche d'une logique pour P. Dans cette thèse, nous avons étudié la pertinence des structures de groupes, et plus particulièrement des groupes de permutations, dans la recherche d'extensions plus expressives de FP. Une telle étude des groupes de permutations est aussi motivée par leur importance dans l'obtention d'algorithmes efficaces pour l'Isomorphisme et la Canonisation de Graphes. Ces deux problèmes jouent en effet

un rôle central dans la recherche d'une logique pour P. La canonisation de graphe[2] est un problème de fonction, qui consiste à calculer une *fonction de canonisation* f. Une fonction de canonisation est une fonction qui à tout graphe $\mathfrak{G}$ associe l'encodage d'un graphe isomorphe à $\mathfrak{G}$, avec la propriété supplémentaire que si $\mathfrak{G} \simeq \mathfrak{H}$, $f(\mathfrak{G}) = f(\mathfrak{H})$. Par analogie, une logique *canonise* les structures d'une classe $\mathcal{C}$ si une formule définit, pour toute structure dans $\mathcal{C}$, une structure ordonnée qui lui est isomorphe. Le théorème d'Immerman-Vardi a pour corollaire que, si $\mathcal{L} \geq \mathsf{FP}$ canonise les structures dans $\mathcal{C}$, $\mathcal{L}$ exprime toutes les propriétés dans P sur $\mathcal{C}$. Ainsi, plusieurs résultats de capture *partielle* ont été obtenus en montrant qu'une extension de FP canonise certaines structures. Par exemple, FPC capture P sur toute classe de structures excluant un mineur [Gro17].

Le pouvoir de canonisation de FPC coïncide exactement avec celui de l'algorithme de coloration de Weisfeiler-Lehman, comme le montrent Cai, Fürer et Immerman [CFI92]. Cet algorithme constitue un test d'isomorphisme partiel purement combinatoire. Cette limitation a motivé le développement de méthodes plus puissantes pour tester l'isomorphisme de structures, en particulier l'approche basée sur les groupes de permutations initiée par Babai et Luks [Bab79 ; Luk82], en proposant des algorithmes en temps polynomial pour les graphes à classes de couleur bornées (*Bounded colour-class graphs*). Cette approche accorde un rôle central au groupe d'automorphismes du graphe considéré. À titre d'exemple, pour tester l'isomorphisme entre deux graphes $\mathfrak{G}$ et $\mathfrak{H}$, il suffit d'examiner le graphe disjoint $\mathfrak{G} \sqcup \mathfrak{H}$ et de déterminer si son groupe d'automorphismes $\mathrm{Aut}(\mathfrak{G} \sqcup \mathfrak{H})$ contient une permutation qui échange $\mathfrak{G}$ et $\mathfrak{H}$ (en supposant, par exemple, que les graphes sont connexes, quitte à considérer leurs complémentaires).

Dans le cadre de la canonisation d'un graphe $\mathfrak{G} = (V, E)$, cette idée se traduit par l'introduction d'une structure auxiliaire au cours du calcul progressif d'un encodage canonique de la structure donnée en entrée. Cette structure auxiliaire est un *labeling coset* $\sigma\Lambda \subseteq \mathrm{Sym}(V)$, où $\Lambda \leq \mathrm{Sym}(V)$. Au fur et à mesure que l'on restreint les potentiels encodages canoniques de $\mathfrak{G}$, on actualise la valeur de $\sigma\Lambda$ de sorte que toute permutation $f \in \sigma\Lambda$ constitue un isomorphisme entre $\mathfrak{G}$ et un de ces encodages canoniques. Bien sûr, ces restrictions successives doivent toutes être canoniques, c'est-à-dire indépendantes de l'ordre dans lequel les sommets de V sont fournis en entrée. L'entretien d'une telle structure de labeling coset permet justement une étude structurelle du groupe Λ, d'où de telles restrictions canoniques peuvent être dérivées. En particulier, si Λ n'agit pas transitivement sur V^2, on peut définir un ordre canonique sur la partition de V^2 en orbites, et canoniser l'encodage de $\mathfrak{G}$ séquentiellement, orbite par orbite. Un mécanisme similaire permet de traiter le cas où l'action de Λ sur V^2 est transitive imprimitive. Lorsque aucun de ces cas ne se produit, on applique une recherche brute du labeling coset minimisant l'encodage. Cette méthode de canonisation, introduite dans [BL83], permet la canonisation de nombreuses classes de graphes en temps polynomial, en utilisant la structure des classes en question pour produire une restriction initiale à un labeling coset canonique. En particulier, les graphes de degré borné et les graphes à coloriage borné (*bounded colour-class graphs*) peuvent-être ainsi canonisés efficacement.

Cette brève introduction à l'utilisation des groupes dans la recherche d'algorithmes efficaces pour l'isomorphisme et la canonisation de graphes motive et précise l'objet de notre recherche : comment cet ensemble de techniques peut-il être traduit dans une

[2]Ou de structures sur n'importe quelle signature, ces problèmes se réduisant l'un à l'autre par des réductions de très faible complexité

extension de FP ? Quels obstacles l'invariance par isomorphisme dresse-t-elle dans la représentation de ces structures, et comment peut-on y remédier ?

C'est à ces questions que nous avons commencé de donner une réponse dans cette thèse. Dans un premier temps, nous avons introduit et étudié une représentation des groupes de permutations naïvement adaptée de leur représentation algorithmique (cf. Définition 2.15). Les algorithmes rapidement présentés plus haut tirent en effet parti de la possibilité de représenter des groupes de permutations par des ensemble de générateurs de taille polynomiale. En effet, tout groupe $G \leq \mathrm{Sym}(D)$ admet un ensemble générateur de taille $O(|D|^2)$. De plus, l'algorithme de Schreier-Sims [Sim70 ; Sim71] permet d'effectuer, en temps polynomial, plusieurs opérations primitives sur des groupes ainsi représentés, parmi lesquelles le test d'appartenance, le calcul de l'ordre du groupe, et enfin l'obtention d'un ensemble générateur (de taille polynomiale) pour n'importe quel sous-groupe d'indice polynomial (à condition qu'on puisse reconnaitre l'appartenance d'une permutation au sous-groupe en temps polynomial). Ces primitives sont fondamentales à l'implémentation des algorithmes décrits plus haut. Cependant, la procédure de Schreier-Sims dépend crucialement de la présence d'un ordre linéaire sur le domaine D sur lequel les permutations agissent, et il semble impossible de définir cette opération dans FP.[3]

Ainsi, ces opérations constituent des candidats naturels d'opérateurs à ajouter à FP pour étendre son pouvoir d'expression au-delà de celui de $\mathsf{FP} + \mathsf{rk}$. En particulier, le calcul de l'ordre du groupe généré par un ensemble de permutations admet une définition relativement naturelle au sein de FP (cf. Section 2.4), et permet, dans FP, de vérifier l'appartenance d'une permutation au groupe engendré par un ensemble de permutations (cf. Lemme 3.2). Nous dénotons cet opérateur ord.

Au cours du Chapitre 3, nous initions l'étude de la logique $\mathsf{FP} + \mathsf{ord}$, et de cette représentation des groupes à l'aide de petits ensembles de permutations les générant. En particulier, nous montrons les limites de cette représentation : si, comme mentionné plus haut, tout groupe $G \leq \mathrm{Sym}(A)$ admet un ensemble générateur S de taille polynomiale, ceci n'est plus vrai dès qu'on attend de S qu'il soit symétrique, i.e. qu'il respecte les automorphismes de la structure dans laquelle G est définissable, au sens où $S^\sigma = S$ pour tout $\sigma \in \mathrm{Aut}(\mathfrak{A})$ (cf. Théorème 3.6). Pire, on peut construire une structure $\mathfrak{A}$ où un groupe $G \leq \mathrm{Sym}(A)$ admet un ensemble de générateurs définissable dans FPC, où $H \leq G$ est accessible — au sens où l'algorithme de Schreier-Sims permet de dériver un ensemble générateur pour H à partir d'un ensemble de générateurs pour G — mais tel qu'aucun ensemble générateur de H de taille polynomiale ne respecte les symétries de $\mathfrak{A}$. Intuitivement, ce résultat signifie que l'adaptation naïve de la représentation computationnelle des groupes de permutations se heurte en elle-même aux limitations de l'invariance par isomorphisme, et d'autres représentations sont nécessaires. Ceci motive l'introduction de la *définissabilité par morphismes* en Section 3.3. Lorsque $H \trianglelefteq G$, si G admet un ensemble de générateurs définissable, ce formalisme offre une autre manière de représenter H : en définissant un morphisme $m : G \to K$ tel que $\ker(m) = H$. Dans $\mathsf{FP} + \mathsf{ord}$, si H_1 et H_2 sont morphisme-définissables depuis G, on peut définir leurs ordres, vérifier l'appartenance, et surtout, définir par morphisme depuis G le groupe $H_1 \cap H_2$. Cette nouvelle représentation des groupes de permutations jouera un

[3] Comme nous le verrons plus loin, on peut déduire l'indéfinissabilité de ces opérations dans $\mathsf{FP}+\mathsf{rk}$ de nos résultats

rôle crucial dans le Chapitre 4. Enfin, nous montrons en section 3.4 que $\mathsf{FP}+\mathsf{ord}$ définit l'opérateur de rang rk, et ainsi, que $\mathsf{FP}+\mathsf{rk} \leq \mathsf{FP}+\mathsf{ord}$.

Le Chapitre 4 est consacré à la séparation entre $\mathsf{FP}+\mathsf{rk}$ de $\mathsf{FP}+\mathsf{ord}$. Cette démonstration s'appuie sur la récente percée de Lichter [Lic23], séparant $\mathsf{FP}+\mathsf{rk}$ de P. Comme mentionnées plus haut, les structures employées par Lichter sont des structures de CFI généralisées. En particulier, elles ont une propriété qui avait déjà été étudiée par Pakusa dans sa thèse de doctorat [Pak15] : on peut y définir (canoniquement) un *coloriage abélien* (cf. Définition 4.1). Ainsi, nous démontrons dans ce chapitre que $\mathsf{FP}+\mathsf{ord}$ canonise les structures à coloriage abélien, ce qui implique que $\mathsf{FP}+\mathsf{ord}$ capture P sur toute classe de structure où un coloriage abélien est définissable, ce qui, avec le résultat de Lichter, conclut la preuve de séparation entre $\mathsf{FP}+\mathsf{rk}$ et $\mathsf{FP}+\mathsf{ord}$. Pakusa avait déjà démontré la capacité d'une autre logique (CPT) à canoniser les structures à coloriage abélien. Sa démonstration employait l'algorithme de canonisation de Babai & Luks décrit plus haut, spécialisé au cadre ici présent. Plus précisément, les coloriages abéliens sont suffisamment restreints pour rendre superflu l'utilisation de l'imprimitivité du groupe Λ, et il suffit d'employer la récursion liée à l'intransitivité. La difficulté majeure à définir cet algorithme dans une logique réside dans la représentation des labeling cosets. En effet, la représentation usuelle d'une classe à gauche par un représentant et un ensemble générateur est, par essence, non-invariante par isomorphisme. En effet, le choix du représentant ne respecte pas les symétries de la structure considérée. De plus, si dans le cadre algorithmique (où la structure donnée en entrée est ordonnée), un labeling coset $\sigma\Lambda$ est un sous-ensemble de $\mathrm{Sym}(V)$, ce n'est plus le cas dans le cadre logique : en l'absence d'un ordre sur V, on ne peut représenter les labelings que comme des fonctions $f : V \to \{0, 1, ..., |V| - 1\}$ (des *étiquetages*), et de telles fonctions ne peuvent être composées, ne constituant donc pas un groupe.

Pakusa avait surmonté cette difficulté en démontrant qu'il était possible d'encoder les labeling cosets par des systèmes d'équations sur des anneaux. C'est ici que notre approche se distingue de la démonstration de Pakusa : notre représentation des labeling cosets reste plus proche des groupes de permutations. Plus précisément, nous définissons un groupe $\mathcal{G}$, définissable dans FPC (cf. Section 4.6), dans lequel se plonge l'ensemble des étiquetages initiaux propre au coloriage abélien considéré. De plus, nous montrons qu'à chaque étape de restriction du labeling coset, le sous-groupe sous-jacent au nouveau labeling coset est définissable par morphisme depuis $\mathcal{G}$. Ici, nous employons une propriété supplémentaire des groupes définissables par morphismes : si $m : G \to K$ définit H, alors les éléments de $\mathrm{im}(m) \leq K$ constituent des représentants des classes à gauche de H dans G ; de plus, si m est définissable, cette représentation des classes de H dans G est invariante par isomorphisme. C'est ainsi que nous représentons les labeling cosets au cours de la simulation de l'algorithme de Babai & Luks.

Enfin, le Chapitre 5 étudie le cas particulier où, bien que le domaine des permutations ne soit pas ordonné, l'ensemble de générateurs du groupe lui-même l'est. Cette situation intervient, par exemple, dans le cas des structures à coloriage abélien. Nous démontrons que $\mathsf{FP}+\mathsf{ord}$ peut, dans ce cas, définir la troisième opération de Schreier-Sims mentionnée plus haut. Pour ce faire, nous montrons que $\mathsf{FP}+\mathsf{ord}$ peut simuler l'algorithme de Schreier-Sims partiellement, en déléguant une partie de l'entretien des structures de données nécessaire à l'opérateur ord. Nous montrons aussi qu'un tel entretien n'est pas nécessaire lorsque le groupe maximal est abélien. Curieusement, ceci

implique que, bien que FPC ne capture pas P sur les structures de CFI, FPC définit leurs groupes d'automorphismes.

Pour conclure, nous avons initié l'étude des groupes de permutations dans un contexte logique, et nous avons montré les divers obstacles que l'invariance par isomorphisme engendre dans une telle entreprise. Pour les surmonter, nous avons dû employer extensivement les propriétés structurelles des groupes étudiées. Ceci suggère qu'une approche logique de la Théorie Computationnelle des Groupes nécessite un fondement structurel, y compris pour concevoir des représentations adéquates des groupes en question. Les groupes nilpotents, résolubles, ainsi que les groupes ne possédant pas de sous-groupes normaux abéliens constituent des pistes potentielles d'extension de ce travail.

À mes grand-parents

Remerciements

Il est de coutume de remercier ses directeurs de thèse en premier lieu. C'est pourquoi il m'est nécessaire d'exprimer à quel point, au-delà des conventions, cette place leur rend justice. Arnaud Durand et Luc Segoufin m'ont soutenu, tout au long de cette thèse, tandis que je formais ce projet de recherche personnel. Je leur suis infiniment reconnaissant de la liberté qu'ils m'ont laissée tout au long de cette élaboration, sans jamais me laisser seul pour autant. Leur accompagnement a constitué une passionnante formation et un encouragement à une perpétuelle curiosité mathématique. Je tiens aussi à remercier Anuj Dawar pour ses conseils et son encadrement. J'ai grande hâte du travail que nous accomplirons ensemble dans les prochaines années.

Bien que leurs contributions à cette thèse sont moins directes, plusieurs professeurs m'ont partagé leurs passions pour les mathématiques au cours de mon parcours scolaire. J'aimerais donc rendre hommage à Mme. Breillot, M. Djellouli, M. Sablé et M. Saint-Germain. J'aimerais encore remercier mon oncle Bernard pour les nombreuses énigmes qu'il me posa enfant et qui ont aussi nourri mon intérêt pour les mathématiques.

Mais ce travail de longue haleine n'a pas été seulement scientifique ; et ce sont aussi mes proches qui l'ont rendu possible. Tout d'abord, mes parents qui m'ont encouragé, durant cette thèse comme tout au long de ma vie, à être curieux de tout, sans m'arrêter aux premiers aprioris. Puis Carl, pour les enseignements tirés des nombreuses aventures vécues ensemble. Pour leur constante attention, je remercie Anne, Camille C, Leïla, Lucas, Martin et Maud. Pour leur écoute, je remercie Aurélien, Bernadette, Capucine et Hannah. Je remercie Florence, pour ses multiples relectures, conseils, encouragements, et son soucis permanent pour l'avancée de mon travail. Je remercie aussi David, pour sa curiosité, son enthousiasme, et les innombrables discussions sur nos sujets de thèse respectifs ; ainsi que Camille G, sans qui ce travail n'aurait pas sa forme finale. Nos multiples débats ont certainement nourri mes réflexions et élargi l'horizon d'application de mon travail. Et pour la joyeuse force qu'ils me donnent, Agathe, Arthur, Enora, Hugo, Ioannis, Manel, Maxime, Nihil, Tommaso, Ricardo, et tous mes amis, merci !

Enfin, cette thèse n'aurait pu voir le jour sans ma compagne, Anna. En plus de sept ans de vie commune, elle a tant participé à mes accomplissements personnels qu'il en devient difficile de séparer mon travail de son aide. Déjà à l'été 2020, c'est avec son soutien et ses relectures que j'ai constitué ma candidature à un contrat doctoral. Et c'est avec une constance remarquable que ce soutien s'est renouvelé tout au long de ma thèse. Elle a su m'apaiser dans les moments de doute, m'épauler face au découragement, me relire et m'écouter malgré la distance qui sépare mon sujet d'étude de son domaine de prédilection. Aujourd'hui, tandis que le mien touche à sa fin, Anna entame sa deuxième année de doctorat. Je ne peux qu'espérer faire pour elle un dixième de ce qu'elle a fait pour moi.

Contents

Introduction

In *Paragone* [Vin49, p. 78], Leonardo da Vinci draws the following comparison of poetry and painting: "There is the same difference between the poet's and the painter's representations of the human figure as there is between dismembered bodies and undivided bodies. Because the poet in describing the beauty or ugliness of any figure can only show it to you consecutively, bit by bit, while the painter will display it all at once."

Unlike poetry, painting allows the artist to transmit, together with the image, the immediacy of its impression. This immediacy is experienced by the viewer as a choice: "the privilege of the image — opposed in that to text, which is linear — is that it enforces no reading order" [Bar72, p. 100][4].

Trying to impose an ordering on our representation of some structure we know is often detrimental to our intuition. For instance, I know by heart the telephone numbers of a few elected loved ones. This set of records is a clear and definite structure in my mind. However, trying to consider those records in any fixed order — e.g. first name alphabetical — only obscures it, investing most of my mental effort not to miss someone.

Yet, the representations of data in computer architectures are ordered, having inherited the ordering that underlies *all* usual models of computation. To make up for this limitation, we have devised *encodings*, functions that map structures of arbitrary shape to integers on which the computers can carry out their operations.

This bears a societal cost: those encodings are complex, and their understanding constitutes a barrier of entry to the design of algorithms. This may partially explain why most users trust large companies with their data, and why software mostly provides "commercial 'user journeys'—efficiently scripted interactions toward game-like objectives that somebody else has defined for you[...]" [Bla24, p. 77].

Most user interfaces invisibilise encodings, and as a result of the gap between the user's and the machine's representation of the data, any operation outside the scope of the interface at hand requires technical knowledge. This state of affairs hinders most users' autonomy.

This prompts the question: can we produce a model of computation acting *directly* on unordered data structures ? While coming up with a formalism which captures all computable properties of structures is not difficult, refining this formalism as to

[4]Our translation.

correspond to complexity classes — and in particular, to the class of tractable problems — has shown quite hard to achieve, and is the subject of Descriptive Complexity.

In a seminal article [Fag74], Fagin showed that the class of NP problems corresponds precisely to those properties of structures that can be *described* in existential second-order logic. This result prompted similar logical characterisations of various complexity classes, a summary of which can be found in [Imm99]. Descriptive Complexity is the study of complexity classes through this angle, looking for formalisms which *capture* complexity classes, in the sense that the problems they are able to express are *exactly* the problems that fall within that complexity class. This approach bears similarities to the work that has been carried out on the Chomsky hierarchy [Cho59]: each type of grammar has been successfully associated with a restricted model of computation which enables the definition of exactly the same languages — for instance, any regular language is defined by a type-3 grammar, and can be computed by finite-state machine [Kle56]. Another similar field is that of *Implicit Computational Complexity*, which aims at defining restricted programming languages corresponding to various complexity classes [Lag22].

The main difference between those two approaches and Descriptive Complexity is that the latter aims to operate a shift of representation of computational problems, from languages — sets of words, ordered sequences of characters, which ultimately can be mapped to integers — to classes of relational structures. As such, it provides a radically different context in which to think about computation, and the fact that some complexity classes translate in this structural approach, while others seem not to, seems to hint at a fundamental discrepancy between those complexity classes. On a more practical level, capture results have implications on the design of Database languages, as discussed in [AHV95, Part E]. This is also how Descriptive Complexity applies to the societal problem presented earlier: if all-purpose computation is an intricate concept, relying on encodings, recursion and control structures, "it is easy to understand the conceptual underpinnings of the relational model, thus making relational databases accessible to a broad audience of end users" [AHV95, p. 29].

The quest to identify a logic that captures P is a central challenge in Descriptive Complexity. This question, which can be traced back to [CH82], was stated in its modern formulation by Gurevich [Gur88]. While fixed-point logic (FP) captures P on *ordered* structures [Imm83; Var82] — which, through usual encoding techniques, reduces to classical computation on words — no logic is currently known to capture P in the unordered case. FP and its natural extensions, such as fixed-point logic with counting (FPC), fail to capture P [CFI92]. This limitation of FPC was demonstrated using the CFI-construction, a class of structures encoding the satisfiability of systems of equations over the finite field $\mathbb{F}_2$ [CFI92].

To address these limitations, extensions of FP incorporating linear-algebraic operations, such as the rank operator (rk), have been proposed [Daw+09; Hol10; Pak10; GP19]. However, even FP + rk falls short of capturing P, as recently demonstrated by Lichter [Lic23] through the use of a generalised class of CFI-structures.

On the other hand, a lot of work has been devoted to *partial* capture results, showing that on restricted classes of structures, extensions of FP are able to define all P queries. For instance, Grohe showed that FPC captures P on any class of structures which excludes a minor [Gro17]. Those results typically rely on the definition within the logic at hand of a canonisation of the structures under consideration. Indeed, for any logic extending FP, the Immerman-Vardi theorem implies that the definability of a canonisation on a class of structures yields the capture of P on that class. This motivates the study of canonisation algorithms, and their definability within candidate logics for P.

Parallel to this investigation, significant progress has been made in the development of efficient algorithms for graph isomorphism and canonisation through a group-theoretic approach. This line of research has yielded polynomial-time isomorphism and canonisation algorithms for various classes of structures [Bab79; Luk82; BL83], as well as Babai's recent breakthrough that general graph isomorphism is solvable in quasi-polynomial time [Bab16]. Notably, an early result in this area demonstrates the polynomial-time canonisability of CFI-structures. This result generalises seamlessly to the broader classes of CFI-constructions used in [GP19], or even in [Lic23] to separate $\mathsf{FP} + \mathsf{rk}$ from P. These findings underscore the potential of integrating group-theoretic operators into FP to extend its expressive power.

This approach relies on the Schreier-Sims algorithm [Sim70; Sim71], which enables, given a set of permutations, to compute the order, and to recognise elements of the group generated by that set. It is arguably the most fundamental polynomial-time permutation group algorithm, as it makes the very representation of groups by generating sets of permutations relevant. However, this procedure relies on stabilising one by one the elements of the domain on which the permutations act. This process thus depends on an ordering of the domain of the permutation group at hand, and cannot be defined in an isomorphism-invariant way, while its output is isomorphism-invariant.

This situation is quite similar to the one which motivated the introduction of the rk operator: Gaussian elimination is inherently dependent on an ordering of the rows and columns of the matrix at hand, yet the rank of the matrix is not. Note that, the fact that $\mathsf{FP} + \mathsf{rk}$ is strictly more expressive than FPC implies that FPC indeed cannot define Gaussian elimination, nor can it define the rank of a matrix by any other means.

In this thesis, we aim to introduce a novel group-theoretic operator, ord, computing the order of a group generated by a definable set of permutations. Because the Schreier-Sims algorithm enables the computation of this operation in polynomial-time, whether a structure satisfies a formula in $\mathsf{FP} + \mathsf{ord}$ can be decided in polynomial-time (in the size of the structure). Thus, like rk, the ord operator defines the isomorphism-invariant result of a polynomial-time algorithm whose computation inherently depends on an ordering of the structure at hand.

The ord operator fills an interesting space within the algebraic extensions of fixed-point logics that have been studied. In [Daw+13], the authors consider various solvability quantifiers — expressing the satisfiability of definable systems of equations over different algebraic structures, as abelian groups, fields, or commutative rings. In this

work, the authors suggest a new *permutation group membership* quantifier, that subsumes all the quantifiers considered in this article. The ord operator defines this quantifier. Actually, the ord operator can be thought of as being to the permutation group membership quantifier what the rk operator is to the field solvability quantifier. This should be contrasted with the apparent absence of such a matrix rank operator in the context of rings. Finally, note that further algebraic generalisations of those operators — say, to monoids — would hit complexity barriers: the membership problem for a monoid defined by a generating set is known to be PSPACE-complete [Koz77].

Even with this operator available, simulating group-theoretic algorithms within FP + ord presents significant challenges due to the reliance of those algorithms on an implicit ordering of the domain. One such challenge is the representation of *cosets* of a subgroup within a larger group, both groups admitting definable generating sets. In the computational context, such a coset could be represented by *any* of its element. However, this constitutes a choice which is hardly isomorphism-invariant. As such, a different representation of cosets is to be used in the logical context.

This is especially true with regards to graph canonisation. In this context, the group-theoretic approach consists in the computation of a canonical *labeling* coset, that is, a coset which contains all relabelings of the graph at hand into its canonical copy. Building a canonical labeling coset rather than a mere encoding of the canonical structure enables exploiting the structure of underlying permutation groups, but depends on the existence of an ordering of the domain. Indeed, in the ordered setting, a labeling is a reordering, and thus a permutation; while in the unordered setting, it is a bijection from the domain to an initial segment of the integers, and those bijections cannot be composed. Schweitzer and Wiebking [SW19] provide such a definition of labeling cosets that accounts for the distinct nature of the structure's domain and its canonical numerical representation. However, their contributions remain algorithmic and still rely on the traditional, order-dependent representation of labeling cosets.

Our contributions We provide a generic representation of sets of permutations in extensions of first-order logic, and define the ord operator accordingly. Our main result is that FP + ord strictly extends the expressive power of FP + rk. Precisely, we show that the rank of a definable matrix is definable in FP + ord, and that FP + ord captures P on the class of structures used by Lichter to separate FP + rk from P. This result has the direct implication that FP + rk cannot define the order of a group given by a generating set, in the same way that FPC $<$ FP + rk implies that FPC cannot define the rank of a matrix.

We obtain this canonisation result by showing that, on CFI-structures, FP + ord can simulate the graph canonisation algorithm defined in [BL83], using an isomorphism-invariant representation of labeling cosets. This representation of labeling cosets relies on a notion of *morphism-definable subgroups*.

A similar approach to the canonisation of CFI-structures was taken in [Pak15], where the same algorithm is simulated in the context of CPT — another candidate logic for P. However, the two results differ in the way labeling cosets are represented. In [Pak15], labeling cosets were represented as systems of equations over a finite ring. In the case

of FP+ord, our representation of labeling cosets remains purely group-theoretic. While this does not seem to directly allow generalisation of the classes of structures canonised, it opens the door to new representation schemes for labeling cosets.

More generally, this thesis studies the adequacy of this representation of groups by generating sets of permutations in extensions of first-order logic. We show some basic operations on those representations to be definable in weaker extensions of first-order logic, and we also show that another operation enabled by the Schreier-Sims algorithm is to remain out of reach of FP+ord (and actually of *any* logic extending FO), as long as we represent groups in this way. The operation in question consists in the computation of a generating set for any subgroup of polynomially-bounded index of a larger group for which a generating set is provided. The ability to derive generating sets for such subgroups is a central part of the group-theoretic framework to graph-isomorphism. This limitation actually underlines a nuance in our representation: a representation of a group can be *ordered*, in the sense that the definable set of permutations generating that group can be indexed on an ordered domain. We show that, in such a case, FP+ord does define this operation of the Schreier-Sims framework.

While we do not expect FP + ord to capture P, our results suggest that FP + ord represents a meaningful advancement in the landscape of logics for polynomial-time computation. Moreover, many other operations on permutation groups are known to be polynomial-time computable, many of them playing an important role in polynomial-time Graph Isomorphism algorithms for broader classes of graphs. This first group-theoretic logic for P sets the stage to study the relationship of those different problems in an isomorphism-invariant context.

Outline of the thesis In Chapter 1, we review the usual notions of Descriptive Complexity and Permutation Group theory which we will use throughout this thesis. This includes the definitions of several extensions of first-order logic which will play a part in our results, the introduction of notations, and the formal definition of a logic capturing P, as set out in [Gur88]. Most of the group-theoretic notions introduced in Section 1.2 follow usual nomenclature and notation. Our definition of accessible subgroups (Definition 1.68), which is central in Chapters 3 to 5, is perhaps not as common.

Because neither first-order logic nor FP provide a formalism for the definition of functions, in order to define the ord operator as set out above, we must provide a natural representation of permutations, and sets thereof within fixed-point logics. Chapter 2 is devoted to the definition of such a representation, and to the introduction of the ord operator.

In Chapter 3, we initiate the study of FP+ord, and more generally, of the definability of permutation group properties in extensions of first-order logic. In particular, we show that FP + ord defines the permutation group membership quantifier mentioned earlier (Lemma 3.2), and introduce our representation of *morphism-defined subgroups*, which will enable — in the setting of Chapter 4 — an isomorphism-invariant representation of labeling cosets. In Section 3.4, we show that FP+ord expresses the rk operator. This draws on the notion of morphism-definability. Finally, in Section 3.2, we show that, in

general, accessible subgroups cannot be represented by generating sets in $\mathsf{FP} + \mathsf{ord}$.

Chapter 4 is devoted to the proof of our central result: the separation between $\mathsf{FP} + \mathsf{rk}$ and $\mathsf{FP} + \mathsf{ord}$. As we rely on Lichter's result [Lic23], we first introduce the class of structures exhibited by Lichter to separate $\mathsf{FP} + \mathsf{rk}$ from P. Then, Sections 4.3 to 4.7 are devoted to the canonisation of those structures within $\mathsf{FP} + \mathsf{ord}$.

Finally, in Chapter 5, we show that, if a group is provided by an ordered generating set of permutations, its accessible subgroups admit a definable generating set, thus showing that under this additional assumption that an ordering of the generating set is provided, the result of Section 3.2 do not hold anymore. This proof relies on a partial simulation of the Schreier-Sims algorithm within $\mathsf{FP} + \mathsf{ord}$. In particular, we exploit the ordering on the generating set of the group to represent *all* cosets appearing within the run of the Schreier-Sims algorithm.

Chapter 1

Preliminaries

In this chapter, we introduce the notations and material needed for the remainder of this thesis. We first review notions from logic and descriptive complexity, and then turn to Group Theory, and in particular algorithms on permutation groups.

In this thesis, we will assume the reader to be knowledgeable in complexity theory. Our model of computation is the Turing Machine, with a constant number of work tapes. Recall that L and P are the classes of problems decidable respectively in logarithmic space and polynomial time on a deterministic Turing Machine. NL and NP are the corresponding classes on non-deterministic Turing Machines.

1.1 Logic and Descriptive Complexity

As we have mentioned in the introduction, computability and logic share a common history. While the two fields have, in a large proportion, grown apart since their fruitful co-construction, numerous links have been drawn between the two in the last decades. In this section, we provide a formal introduction to descriptive complexity, one such connection between computation and logic.

For a function $f : A \to B$ and $X \subseteq A$, we write $f_{\restriction X}$ for the restriction of f to X.

Structures, First-Order logic, Models

A *signature* (or *vocabulary*) Σ is a set of *relation symbols* and *function symbols*, each associated with a non-negative integer, called the *arity* of this symbol. Relation symbols are usually upper-case letters (R, S, E, ...), and lower-case letters (f, g, s, ...) are function symbols. In particular, c is only ever used as a function symbol of arity 0, also called a *constant* symbol.

When presenting a signature, we often provide the arity of each relation in superscript. For instance, $\Sigma := \{R^{(3)}, S^{(4)}\}$ contains a ternary relation symbol R, and a quaternary relation symbol S. Given a symbol R, we also denote its arity $\mathrm{ar}(R)$.

Definition 1.1. A Σ-*structure* is a tuple $\mathfrak{A} = (A, (R^{\mathfrak{A}})_{R \in \mathrm{rel}(\Sigma)}, (f^{\mathfrak{A}})_{f \in \mathrm{fun}(\Sigma)})$, where:

- A is a set, called the *domain* of $\mathfrak{A}$, and denoted $\mathrm{dom}(A)$.
- For any relation symbol R in Σ, of arity k, $R^{\mathfrak{A}}$ is a subset of A^k, that is, a k-ary relation over A
- For any function symbol f in Σ, of arity k, $f^{\mathfrak{A}}$ is an element of A^{A^k}, that is, a k-ary function on A.

$R^{\mathfrak{A}}$ (resp. $f^{\mathfrak{A}}$) is the *interpretation* of R (resp. f) in $\mathfrak{A}$.

The set of all Σ-structures is denoted $\mathrm{STRUC}[\Sigma]$. As we will extensively refer to the domain of structures, we adopt the following notation: all structures are represented by upper-case, Fraktur characters ($\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$,...), and their domain is represented by the corresponding roman character (A, B, C,...). A structure is *finite* if its domain is a finite set. The *size* of a finite structure $\mathfrak{A}$ is the integer $|A|$, and we often denote it $|\mathfrak{A}|$.

Example 1.2. A (finite) *graph* is a (finite) $\{E\}$-structure, where E is a relation symbol of arity 2 (binary). Given a graph $\mathfrak{G}$, an element of G, the domain of $\mathfrak{G}$, is called a *vertex*, and an element of $E^{\mathfrak{G}}$ is called an *edge*.

We aim to provide a general definition of a *logic*. However, it seems necessary to first and foremost introduce first-order logic, as it is the skeleton of all the logics we will consider in this thesis.[1] From now on, we fix a set of first-order variable symbols $\mathcal{X}$.

Definition 1.3 (Syntax of *first-order logic*)**.** The set of first-order *terms* over a signature Σ is the smallest set $\mathcal{T}_\Sigma$ such that $\mathcal{X} \subseteq \mathcal{T}_\Sigma$, and for any $t_1, \ldots, t_k \in \mathcal{T}_\Sigma$ and f a k-ary function symbol in Σ, $f(t_1, \ldots, t_k) \in \mathcal{T}_k$.

The set of first-order *formulae* over a signature Σ is the smallest set $\mathsf{FO}[\Sigma]$ such that:

- For all $t_1, t_2, \ldots, t_k \in \mathcal{T}_\Sigma$, $(t_1 = t_2) \in \mathsf{FO}[\Sigma]$, and for any k-ary relation symbol in Σ, $(R(t_1, \ldots, t_k)) \in \mathsf{FO}[\Sigma]$. Those formulae are the *atomic formulae*.
- For any $\varphi, \psi \in \mathsf{FO}[\Sigma]$, and $x \in \mathcal{X}$, $\mathsf{FO}[\Sigma]$ includes $\varphi \vee \psi$, $\varphi \wedge \psi$, $\neg\varphi$ and $\exists x, \varphi$

We also introduce $\forall x, \varphi$ as a shorthand notation for $\neg\exists x, \neg\varphi$. In a similar recursive manner, one can formally define the set of variables occurring in any formula φ, denoted $\mathrm{Var}(\varphi)$. A variable x is *bound* in φ if x only appears quantified in φ. A variable is *free* if it is not bounded, and we denote $\mathrm{free}(\varphi)$ the set of free variables of φ. A formula without free variables is a *sentence*.

[1] One notable exception to this is Choiceless Polynomial time. However, this logic does not play an important part in this thesis.

Finally, we only consider formulae up to renaming of bound variables, i.e. $\exists x, E(x, y)$ and $\exists z, E(z, y)$ are equal, while $\exists x, E(x, y)$ and $\exists x, E(x, z)$ are not.

We depart from tradition in our use of the notation $\varphi(\vec{x})$. While this notation is often used in the literature to provide information on the free variables of φ, in the context of this thesis, this notation will serve as a substitution mechanism. Given a formula φ with $x \in \text{free}(\varphi)$ and y a variable, we denote $\varphi[x/y]$ the formula φ where each occurrence of x has been substituted by y. When defining a formula φ, if the tuple $\vec{x}$ is exhibited as follows

$$\varphi(\vec{x}) := \dots$$

then, for any tuple of variables $\vec{y}$ with $|\vec{y}| = |\vec{x}| = i$, any occurrence of $\varphi(\vec{y})$ should be read as

$$\varphi[x_1/y_1, \dots, x_i/y_i]$$

This syntax is similar to the specification of arguments during the declaration of a function in most programming languages (C, Java, ...). We will soon introduce a similar formalism for second-order variables.

In order to improve the readability of large formulae, we often write the conjunction $C_1 \wedge \cdots \wedge C_k$ and the disjunction $C_1 \vee \cdots \vee C_k$ as

$$\begin{cases} C_1 \\ \wedge \vdots \\ C_k \end{cases} \quad \text{and} \quad \begin{cases} C_1 \\ \vee \vdots \\ C_k \end{cases}$$

respectively. We will also use this notation in a nested fashion. For instance,

$$\begin{cases} A \\ \wedge \begin{cases} B_1 \\ \vee B_2 \\ B_3 \end{cases} \\ C \\ D \end{cases}$$

denotes $(A \wedge (B_1 \vee B_2 \vee B_3) \wedge C \wedge D)$.

For $X \subseteq \mathcal{X}$ and $\mathfrak{A}$ a structure, a *valuation* of X over $\mathfrak{A}$ is a function $v : X \to A$. Valuations extend naturally from variables to terms:

Definition 1.4. Let $\mathfrak{A}$ be a Σ-structure, $t \in \mathcal{T}_\Sigma$ and v be a valuation of $\text{free}(t)$ over $\mathfrak{A}$. $v^*(t)$ the value defined inductively as:

- $v^*(x) := v(x)$ for $x \in \text{dom}(v)$.
- $v^*(f(t_1, \dots, t_k)) := f^{\mathfrak{A}}(v^*(t_1), \dots, v^*(t_k))$

Over a structure $\mathfrak{A}$, a formula φ will take as value a set of valuations $\varphi(\mathfrak{A}) \subseteq A^{\text{free}(\varphi)}$.

Our definition of the semantics of FO requires a few basic operations on such sets of valuations: given $\mathcal{V} \subseteq A^Y$ and $X \subseteq Y$, the *restriction* of $\mathcal{V}$ to X, denoted $\mathcal{V}_{\downarrow X}$ is the set $\{v_{\restriction X} \mid v \in \mathcal{V}\}$. Conversely, the *extension* of $\mathcal{V}$ to $Z \supseteq Y$ is the set $\mathcal{V}_{\uparrow Z} := \{v \in A^Z \mid v_{\restriction Y} \in \mathcal{V}\}$.

Definition 1.5 (Semantics of *first-order logic*)**.** The value of φ over $\mathfrak{A}$ is a set of valuations of $\text{free}(\varphi)$ over $\mathfrak{A}$ defined inductively as follows:

$$
\begin{aligned}
(t_1 = t_2)(\mathfrak{A}) &:= \{v : \text{free}(t_1) \cup \text{free}(t_2) \to A \mid v^*(t_1) = v^*(t_2)\} \\
(R(t_1, t_2, \ldots, t_k))(\mathfrak{A}) &:= \left\{ v : \bigcup_{i=1}^{k} \text{free}(t_i) \to A \mid (v^*(t_1), \ldots, v^*(t_k)) \in R^{\mathfrak{A}} \right\} \\
(\varphi \wedge \psi)(\mathfrak{A}) &:= (\varphi(\mathfrak{A}))_{\uparrow Z} \cap (\psi(\mathfrak{A}))_{\uparrow Z} \text{ where } Z := \text{free}(\varphi) \cup \text{free}(\psi) \\
(\varphi \vee \psi)(\mathfrak{A}) &:= (\varphi(\mathfrak{A}))_{\uparrow Z} \cup (\psi(\mathfrak{A}))_{\uparrow Z} \text{ where } Z := \text{free}(\varphi) \cup \text{free}(\psi) \\
(\neg\varphi)(\mathfrak{A}) &:= A^{\text{free}(\varphi)} \setminus \varphi(\mathfrak{A}) \\
(\exists x, \varphi)(\mathfrak{A}) &:= (\varphi(\mathfrak{A}))_{\downarrow(\text{free}(\varphi)\setminus\{x\})}
\end{aligned}
$$

Note that, when $\text{free}(\varphi) = \emptyset$, $\varphi(\mathfrak{A})$ is either the empty set, or the set containing the empty valuation. We identify those values with the booleans false and true, respectively ; and write $\mathfrak{A} \models \varphi$ when $\varphi(\mathfrak{A})$ contains the empty valuation. In the presence of a natural linear-order on $\text{free}(\varphi)$, we identify $\varphi(\mathfrak{A})$ with a relation $S \subseteq A^k$, where $k = |\text{free}(\varphi)|$, and $(a_1, \ldots, a_k) \in S$ iff $v_{(a_1,\ldots,a_k)} \in \varphi(\mathfrak{A})$, where $v_{(a_1,\ldots,a_k)}$ is the valuation of $\text{free}(\varphi)$ that maps the i-th free variable of φ to a_i.

We will often need to consider formulae with second-order free variables. In the context of FO, and for a given tuple of second-order variables $\vec{X} = (X_1, \ldots, X_k)$ (each of fixed arity r_i) this entails, for a given signature Σ, to consider formulae in $\mathsf{FO}[\Sigma \cup \{\vec{X}\}]$. The semantics of such a formula φ is then the function

$$
\begin{aligned}
\varphi(\mathfrak{A}) : \prod_{i=1}^{k} \mathcal{P}(A^{r_i}) &\to A^{\text{free}(\varphi)} \\
(R_i)_{i \le k} &\mapsto \varphi(\mathfrak{A}, R_1, \ldots, R_k)
\end{aligned}
\tag{1.1}
$$

Note that this function has a domain of exponential size. As we are concerned with polynomial-time computations, we cannot expect to simply evaluate a $\mathsf{FO}[\Sigma]$-formula with free second-order variables. However, we can introduce a substitution mechanism similar to the one we considered for first-order variables:

Definition 1.6. Given two formulae φ, ψ, R a second-order variable, and $\vec{x}$ a tuple of first-order variables,

$$\psi[(R/\varphi)_{\vec{x}}]$$

is the formula ψ where every occurrence of $R(\vec{y})$ is substituted by the subformula $\varphi[\vec{x}/\vec{y}]$.

A (finite) first-order *theory* $\mathcal{T}$ over a signature Σ is a (finite) set of first-order Σ-sentences. Any theory defines a class of structures, called the *models* of $\mathcal{T}$, and denoted $\mathrm{Mod}(\mathcal{T})$, which are exactly the structures $\mathfrak{A}$ such that, for any $\varphi \in \mathcal{T}$, $\mathfrak{A} \models \varphi$. Note that if a theory $\mathcal{T}$ is finite, it is equivalent to the single formula $\bigwedge_{\varphi \in \mathcal{T}} \varphi$.

Example 1.7. Consider the $\{E\}$-theory composed of the following axioms:

$$\forall x, \neg E(x, x)$$
$$\forall x, y, E(x, y) \iff E(y, x)$$

The models of this theory are exactly the undirected graphs with no loops: *simple graphs*.

Definition 1.8. A class of Σ-structures $\mathcal{K}$ is FO-*definable* if and only there is a first-order theory $\mathcal{T}$ such that $\mathcal{K} = \mathrm{Mod}(\mathcal{T})$.

We will now explain the ties between logic and computation studied in this thesis. In this particular context, we are only interested in finite structures and finite theories. Except when explicitly stated, $\mathrm{Mod}(\mathcal{T})$ now denotes the set of *finite* structures modelling $\mathcal{T}$. Finally, while we have defined structures and first-order logic on arbitrary vocabularies, the presence of function symbols of positive arity induces technical issues — mainly because the inductive definition of terms enables the definition of an infinity of terms using a finite number of variables. We therefore follow the widely accepted usage to consider only signatures containing solely relations and constant symbols in the remaining of this work. Note that, except in very restricted cases[2], this does not alter the expressivity of FO, as any k-ary function can be encoded as a $k+1$-ary relation, interpreted as the graph of the function at hand.

First capture results

Descriptive complexity is the study the relation between the computational complexity of a problem, that is the amount of resources (often time and space) needed by a Turing Machine to solve it, and its logical complexity, that is, how complex a sentence *defining* the problem must be. Indeed, both Turing Machines and sentences are finite objects, that define an infinite object: in the case of a Turing Machine $\mathcal{M}$ over an alphabet Σ, the language accepted by $\mathcal{M}$, $L(\mathcal{M}) \subseteq \Sigma^*$; and in the case of a formula φ over a vocabulary Σ, the set of finite structures $\mathfrak{A} \in \mathrm{STRUC}[\Sigma]$ such that $\mathfrak{A} \models \varphi$, that is, $\mathrm{Mod}(\varphi)$.

There are several motivations to descriptive complexity, and Immerman's book [Imm99] constitutes a thorough introduction to the field. In this section, we aim to present all the results of descriptive complexity which are relevant to the work presented in this thesis. Those results are mainly *capture* theorems, where an extension $\mathcal{L}$ of first-order logic is shown to express a property if and only if the class of all structures satisfying that property belongs to a specific complexity class $\mathcal{C}$. We then say that $\mathcal{L}$ *captures*

[2]For instance, when considering formulae with a constantly bounded number of nested quantifiers

$\mathcal{C}$. Note that, for such a statement to make sense, we need to introduce a translation between words and structures, that works both ways.

Example 1.9. Fix a finite alphabet Σ, and consider the following theory over $\{\leq, (R_u^{(1)})_{u\in\Sigma}\}$:

$$\begin{aligned}
&\forall x, x \leq x \\
&\forall x, y, x \leq y \wedge y \leq x \implies x = y \\
&\forall x, y, z, x \leq y \wedge y \leq z \implies x \leq z \\
&\forall x, y, x \leq y \vee y \leq x \\
&\forall x, \bigwedge_{u,v\in\Sigma} \neg(R_u(x) \wedge R_v(x)) \\
&\forall x, \bigvee_{u\in\Sigma} R_u(x)
\end{aligned}$$

($\leq$ is a linear-order)

the relations R_u, $u \in \Sigma$, partition the domain

One can easily show that there is a bijection between Σ^* and the isomorphism classes of the finite models of this theory.

This allows us to compare the expressivity of FO on words and that of Turing machines, and the contrast is stark: numerous basic problems, which are trivially computable, cannot be defined in FO, for instance *Even*, the class of even-length words, cannot be defined [Imm99, Proposition 6.45].

On the other hand, FO is well-behaved in aspects where sheer computability comes short. First and foremost, its *model-checking* is *decidable* on finite structures. That is, for any FO[Σ]-sentence φ, there is a Turing Machine *that always terminate*, accepting exactly those finite Σ-structures $\mathfrak{A}$ that model φ. This problem (given a structure $\mathfrak{A}$, deciding whether $\mathfrak{A} \models \varphi$ for a fixed formula φ) plays an important role in descriptive complexity.

Actually, much better can be said about the computational status of this problem, and we will now see that its *complexity* is very low. To state this, we need to define an encoding (as words) of Σ-structures. Let Σ be any finite, relational signature, and fix a linear-ordering of $\Sigma = \{R_1, \ldots, R_k, c_1, \ldots, c_l\}$.

Definition 1.10. Let $\mathfrak{A}$ be a finite Σ-structure of size n. An *encoding* of $\mathfrak{A}$ is a word $w \in \{0, 1, \square\}^*$ such that, for some bijection $f : A \to [n]$, w is of the form

$$w = 1^n \square R_1^{\mathfrak{A},f} \square R_2^{\mathfrak{A},f} \square \ldots \square R_k^{\mathfrak{A},f} \square c_1^{\mathfrak{A},f} \square \ldots \square c_l^{\mathfrak{A},f}$$

where, for all $i \leq k$, $R_i^{\mathfrak{A},f}$ is a word over $\{0, 1\}$ encoding the relation $R_i^{\mathfrak{A}}$; and for all $i \leq l$, $c_i^{\mathfrak{A},f}$ is a word encoding $c_i^{\mathfrak{A}}$. $R_i^{\mathfrak{A}}$ is encoded as the concatenation of all the lines of an array of dimension $d_i := \mathrm{ar}(R_i)$, whose rows and columns are ordered according to f, that contains value 1 at position $(\lambda_0, \ldots, \lambda_{d_i-1})$ iff $(f^{-1}(\lambda_0), f^{-1}(\lambda_2), \ldots, f^{-1}(\lambda_{d_i-1})) \in R_i^{\mathfrak{A}}$. This can be defined formally as follows:

$$R_i^{\mathfrak{A},f}\left(\sum_{j=0}^{d_i-1} n^j \lambda_j\right) := \begin{cases} 1 & \text{if } (f^{-1}(\lambda_0), f^{-1}(\lambda_2), \ldots, f^{-1}(\lambda_{d_i-1})) \in R_i^{\mathfrak{A}} \\ 0 & \text{otherwise} \end{cases}$$

$c_i^{\mathfrak{A}}$ is encoded as a word of length n which contains a 1 at position j iff $f(c_i^{\mathfrak{A}}) = j$.

This definition is cumbersome, and many schemes of encoding of structures could be used instead of this particular one. The main takeaway of this definition is that we can encode any structure as a word *of length polynomial in the size of the structure*, the degree of this polynomial bound depending only on the signature of the structures at hand.

Proposition 1.11 ([Imm99, Theorem 3.1])**.** *For any* $\mathsf{FO}[\Sigma]$*-sentence* φ*, the language* L_φ *of all encodings of all models of* φ *is in* L*.*

While the termination of Turing Machines is undecidable, as is whether a Turing machine uses logarithmic space, first-order logic can be used to define problems that all belong to L. Even more so, the proof of Proposition 1.11 is constructive, and any sentence φ yields a logarithmic space algorithm to check, given an encoding of $\mathfrak{A}$, whether $\mathfrak{A} \models \varphi$.

However, Proposition 1.11 leaves a lot of questions unanswered. As we have stated above, FO cannot define the language of all even-length words, which is clearly in L. From a complexity point of view, is there a characterisation, within L, of the languages which are FO-definable? Conversely, is there a logical characterisation of L? What about higher complexity classes?

Surprisingly, all those questions have been answered in the positive, and most natural complexity classes are *captured* by extensions of first-order logic. As we lack a unified definition of all complexity classes, we cannot provide a systematic and formal definition of this notion of *capture*. Intuitively, a logic $\mathcal{L}$ *captures* a complexity class $\mathcal{C}$ if the model-checking problem for $\mathcal{L}$ is in $\mathcal{C}$; and given any class $\mathcal{K}$ of Σ-structures whose encodings form a language in $\mathcal{C}$, there is a formula in $\mathcal{L}$ defining $\mathcal{K}$.

Fagin's theorem

The first such result is Fagin's theorem, yielding a capture of NP. To be perfectly precise, we should define second-order logic formally, while this logic is outside the scope of this thesis. Let us just provide an idea of this formalism: *second-order logic*, denoted SO, is the extension of first-order logic where we allow the quantification of *relations* within a formula.

$\mathsf{SO}\exists$ is the fragment of SO where all occurences of second-order quantifiers are *existential*, and within an even number of negations.

Theorem 1.12 (*Fagin* [Fag74])**.** *A class of structures* $\mathcal{K}$ *is* $\mathsf{SO}\exists$*-definable iff the language* $L_\mathcal{K}$ *of all encodings of structures in* $\mathcal{K}$ *is in* NP*.*

As a side note, all the classes Σ, Π in the polynomial hierarchy (introduced in [Sto76] also discussed in [AB09, chapter 5]) are captured by the fragment of SO with the adequate alternation of second-order quantifiers.

Captures below NP

Many weaker extensions of FO have been considered and shown to capture complexity classes below NP. Often, those extensions of FO are *restrictions* of SO, restrictions that allow to construct new relations, but only through certain operations, which enable the evaluation of those relations under stricter complexity assumptions.

A first example of such operators are transitive closures:

Definition 1.13. A binary relation $X \subseteq A \times A$ is *transitive* if for any $a, b, c \in A$, $X(a, b)$ and $X(b, c)$ implies $X(a, c)$. It is *symmetric* if for any $a, b \in A$, $X(a, b)$ iff $X(b, a)$.

Definition 1.14. Let A be a finite set, and $X \subseteq A \times A$ (i.e. a binary relation on A). The *transitive closure* of X is the smallest binary relation $\mathrm{tc}(X) \subseteq A \times A$, such that $X \subseteq \mathrm{tc}(X)$ and $\mathrm{tc}(X)$ is transitive.

The *symmetric transitive closure* of X, denoted $\mathrm{stc}(X)$ is the smallest such relation that is also symmetric.

This set-theoretic operation can be added to FO in the following way: let $\varphi \in \mathsf{FO}[\Sigma]$ with $\{\vec{x}, \vec{y}\} \subseteq \mathrm{free}(\varphi)$, $\{\vec{x}\} \cap \{\vec{y}\} = \emptyset$ and $k := |\vec{x}| = |\vec{y}|$. For any structure $\mathfrak{A}$, and any valuation v of $\mathrm{free}(\varphi) \setminus \{\vec{x}, \vec{y}\}$, the set $(\varphi(\mathfrak{A}) \cap \{v\}_{\uparrow \mathrm{free}(\varphi)})_{\downarrow \{\vec{x}, \vec{y}\}}$ of valuations of $\{\vec{x}, \vec{y}\}$ on $\mathfrak{A}$ compatible with v that satisfy φ is a subset of $A^{2k} = A^k \times A^k$.[3] We aim to define operators tc and stc, compatible with the syntax of FO, that take as input such a formula φ, together with the variables $\vec{x}, \vec{y}$, and evaluate, on any structure $\mathfrak{A}$ together with a valuation v of $\mathrm{free}(\varphi) \setminus \{\vec{x}, \vec{y}\}$, to $\mathrm{tc}((\varphi(\mathfrak{A}) \cap \{v\}^{\mathrm{free}(\varphi)})_{\{\vec{x}, \vec{y}\}})$ and $\mathrm{stc}((\varphi(\mathfrak{A}) \cap \{v\}^{\mathrm{free}(\varphi)})_{\{\vec{x}, \vec{y}\}})$ respectively. Formally:

Definition 1.15. The set $(\mathsf{FO} + \mathsf{tc})[\Sigma]$ of $\mathsf{FO} + \mathsf{tc}$ formulae over a signature Σ is the smallest set such that:

- $\mathsf{FO}[\Sigma] \subseteq (\mathsf{FO} + \mathsf{tc})[\Sigma]$.
- If $\varphi \in (\mathsf{FO} + \mathsf{tc})[\Sigma]$ with $\mathrm{free}(\varphi) = \{\vec{x}, \vec{y}, \vec{p}\}$, then $(\mathsf{tc}_{\vec{x}, \vec{y}} \varphi) \in (\mathsf{FO} + \mathsf{tc})[\Sigma]$, and has free variables $\vec{x}, \vec{y}, \vec{p}$.

In this last case, for any valuation v of $\vec{p}$,

$$(\mathsf{tc}_{\vec{x}, \vec{y}} \varphi)(\mathfrak{A}, v) := \mathrm{tc}((\varphi(\mathfrak{A}) \cap \{v\}_{\uparrow \mathrm{free}(\varphi)})_{\downarrow \{\vec{x}, \vec{y}\}})$$

Example 1.16. Consider $\Sigma = \{E\}$ the signature of graphs, $\mathfrak{G}$ a Σ-structure, and a, b two vertices of $\mathfrak{G}$. The formula

$$\mathsf{ustcon}(s, t) := (\mathsf{tc}_{x,y} x = y \vee E(x, y) \vee E(y, x))(s, t)$$

holds on $(\mathfrak{G}, a, b)$ iff a and b belong to the same connected component of $\mathfrak{G}$.

[3] Note that, to identify the set of valuations of $\vec{x}, \vec{y}$ over $\mathfrak{A}$ with a $2k$-ary relation, we have used the natural linear-order that the tuples $\vec{x}, \vec{y}$ define on $\{\vec{x}, \vec{y}\}$

The definition of $\mathsf{FO} + \mathsf{stc}$ is identical, replacing each occurrence of tc or tc by stc or stc, respectively. Additionally, a structure $\mathfrak{A}$ is *ordered* if its signature contains the relation symbol $\leq$, and $\mathfrak{A}$ satisfies the first three axioms given in Example 1.9.

The following holds:[4]

Theorem 1.17 ([Imm99, Corollary 9.22])**.** $\mathsf{FO}+\mathsf{tc}$ *captures* NL *over ordered structures.*

Theorem 1.18 ([Imm83, Theorem 3.3 (b)] and [Rei08])**.** $\mathsf{FO} + \mathsf{stc}$ *captures* L *over ordered structures.*

In both cases, we can be a bit more precise. The model-checking of a sentence in $\mathsf{FO} + \mathsf{tc}$ (resp. $\mathsf{FO} + \mathsf{stc}$) is *always* a NL (resp. L) problem. In the other direction, a linear order on the domain is needed to encode and describe the computation within a formula.

Last but not least, another similar capture result is known for P.

Definition 1.19. Let A be a finite set, and let $f : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ be a monotone function (w.r.t. inclusion). Then, there exists a minimal integer T such that $f^{(T+1)} = f^{(T)}$, and T is polynomially bounded by $|A|$. The *least-fixed point* of f is the set $f^{(T)}(\emptyset)$ denote $\mathrm{LFP}(f)$.

Moreover, if f is computable in polynomial time (w.r.t. $|A|$), $\mathrm{LFP}(f)$ is too, as the monotonicity of f yields $T < |A|^k$. Once again, we aim to extend FO with such an operator allowing the definition of fixed-points. While, in the case of tc, the operator takes as input a formula φ to define a $2k$-ary relation, in the context of LFP, we need a formula to define a *higher-order function*, mapping a relation R of arity k to a relation of arity k. This is achieved by considering a formula φ with a second-order free variable R of arity k, and k free variables. Then, for any structure $\mathfrak{A}$,

$$\begin{aligned} f_\varphi^{\mathfrak{A}} : \mathcal{P}(A^k) &\to \mathcal{P}(A^k) \\ X &\mapsto \varphi(\mathfrak{A}, X) \end{aligned}$$

In this definition, we are using the semantics of formulae with second-order variables, as defined in Eq. (1.1). This yields the following logic:

Definition 1.20. Fixed-point logic, denoted FO+lfp or FP, is the logic whose formulae, for a given signature Σ, are the smallest set $\mathsf{FP}[\Sigma]$ such that:

- $\mathsf{FO}[\Sigma] \subseteq \mathsf{FP}[\Sigma]$
- Given a k-ary relation symbol R not in Σ, a formula $\varphi \in \mathsf{FP}[\Sigma \cup \{R\}]$ in which R occurs only positively, and any tuple of variables $\vec{x}$ such that $|\vec{x}| = k$,

  $$(\mathrm{lfp}_{R,\vec{x}}\varphi)(t_1, \ldots, t_k)$$

  is a formula of $\mathsf{FP}[\Sigma]$, for any terms $t_1, \ldots, t_k$.

[4]Note that the second result depends on the fact that $\mathsf{SL} = \mathsf{L}$, as proven by Reingold.

Given a Σ-structure $\mathfrak{A}$, $((\mathrm{lfp}_{R,\vec{x}}\varphi)(\vec{t}))(\mathfrak{A})$ is the set of all valuations v of $\mathrm{free}(\varphi) \setminus \{\vec{x}\} \cup \mathrm{free}(\vec{t})$ such that:

$$v^*(\vec{t}) \in \mathsf{LFP}(f_\varphi^{(\mathfrak{A},v)})$$

Example 1.21. Recall Example 1.16, where we provided a $\mathsf{FO} + \mathsf{tc}$ formula for weak connectivity in graphs. Given a graph $\mathfrak{G}$ and vertices a, b, the same relation can be defined in $\mathsf{FO} + \mathsf{LFP}$ as follows:

$$\mathsf{ustcon}(x, y) := (\mathrm{lfp}_{R,s,t} s = t \vee \exists u, E(s, u) \wedge R(u, t))(x, y)$$

In Definition 1.19, we have assumed the function f to be monotone w.r.t. inclusion, that is, for any X, Y, if $X \subseteq Y$ then $f(X) \subseteq f(Y)$. This is needed to ensure that a fixed-point is reached in a number of iterations bounded (polynomially) by the size of the domain of the structure at hand. Moreover, such a restriction can be enforced syntactically (as in Definition 1.20), by only allowing the $\mathrm{lfp}_{R,\vec{x}}$ operator to apply on formulae φ where R appears only positively (i.e. within an even number of negations). Note that there is another way to ensure a polynomial bound on the number of iterations needed for f to reach a fixed-point, namely, if f is *inflationary*, i.e. for any X, $X \subseteq f(X)$. This allows an alternative fixed-point operator to be considered where, instead of a syntactic restriction to monotone functions, a semantic restriction to inflationary functions is enforced. Formally, we can consider the operator ifp, such that, for any relation symbol R, for any tuple $\vec{x}$ such that $|\vec{x}| = k$ and any formula φ,

$$(\mathrm{ifp}_{R,\vec{x}}\varphi)$$

is a k-ary relation. The semantic of this operator is as follows. Given a Σ-structure $\mathfrak{A}$ and a formula $\varphi \in \mathcal{L}[\Sigma \sqcup \{R\}]$, $((\mathrm{ifp}_{R,\vec{x}}\varphi))(\mathfrak{A})$ is the set of all tuples $\vec{a}$ that belong to the least fixed-point of $\overline{f}_\varphi^{\mathfrak{A}}$, where

$$\begin{aligned} \overline{f}_\varphi^{\mathfrak{A}} : \mathcal{P}(A^k) &\to \mathcal{P}(A^k) \\ X &\mapsto X \cup \varphi(\mathfrak{A}, X) \end{aligned}$$

One can easily see that, as in Definition 1.19, an easy combinatorial argument ensures that iterative applications of $\overline{f}_\varphi^{\mathfrak{A}}$ reach a fixed-point in a polynomially-bounded number of iterations.

Those two different ways to define a fixed-point operator turn out to yield equivalent extensions of first-order logic [GS86], and we will therefore use both operators interchangeably, depending on which is more convenient in the situation at hand. The next theorem plays an important role in the remainder of this thesis:

Theorem 1.22 (*Immerman-Vardi* [Imm86; Var82])**.** FP *captures* P *over ordered structures.*

Simultaneous (inflationary) fixed-point In Chapters 4 and 5, we will use a more general form of the ifp operator, that we define now. Suppose that we now consider

a *family* of relation functions $f_1, \dots f_k$, each taking as input an identical family of relations $R_1, \dots R_k$ over a domain A, and such that the arity of $f_i(R_1, \dots, R_k)$ is equal to the arity of R_i (for all $i \leq k$). Then, mapping $(R_1, \dots, R_k)$ to

$$(R_1 \cup f_1(R_1, \dots, R_k), R_2 \cup f_2(R_1, \dots, R_k), \dots, R_k \cup f_k(R_1, \dots, R_k))$$

is an inflationary function (w.r.t. inclusion component-wise) from $\prod_{i=1}^{k} \mathcal{P}(A^{\mathrm{ar}(f_i)})$ to itself, and as such the least fixed-point of $(f_1, f_2, \dots, f_k)$ is, in this case also, computable in polynomial-time. More importantly, this kind of simultaneous fixed-point are definable using the lfp operator.

We first define this *simultaneous fixed-point* operator. Given $\phi_1, \dots, \phi_k$ formulae, each with second-order variables amongst $\{R_1, \dots, R_k\}$, and $(\vec{x_i})_{i \leq k}$ a family of tuples of first-order variables, such that Let $\lambda := \mathrm{ar}(R_1)$. For any tuple $(t_1, \dots, t_\lambda)$ of terms,

$$\psi := (\text{s-ifp}_{\vec{x_1}, R_1; \vec{x_2}, R_2; \dots; \vec{x_k}, R_k} \phi_1; \phi_2; \dots; \phi_k)(t_1, \dots, t_\lambda)$$

is a formula whose free variables are

$$\mathcal{V} = \bigcup_{i=1}^{k} (\mathrm{free}(\phi_i) \setminus \{\vec{x_i}\}) \cup \bigcup_{i=1}^{\lambda} \mathrm{free}(t_i).$$

For any structure $\mathfrak{A}$, and any valuation v of $\mathcal{V}$, $(\phi_1, \dots, \phi_k)$ defines a tuple of functions as above, that is,

$$\begin{aligned} f_i^v : \prod_{j=1}^{k} \mathcal{P}(A^{\mathrm{ar}(R_j)}) &\to \mathcal{P}(A^{\mathrm{ar}(R_i)}) \\ (X_1, \dots, X_k) &\mapsto X_i \cup \phi_j(\mathfrak{A}, v, X_1, \dots, X_k). \end{aligned}$$

$\mathfrak{A}, v \models \psi$ if $v^*(t_1, \dots, t_\lambda) \in Z_1$, where $(Z_1, \dots, Z_k)$ is the least fixed-point of the simultaneous iteration of the functions $f_1^v, \dots f_k^v$, starting with $X_i = \emptyset$ for each $i \leq k$.

As demonstrated in [EF95, Theorem 8.2.2] (together with the result of [GS86] mentioned above), any formula in FO + s-ifp is equivalent to a formula in FP. As such, we will freely use the s-ifp operator, as it allows for more readable formulae in Chapters 4 and 5.

Partial fixed-points The above definitions of the lfp and ifp operators raises an interesting question. Indeed, both those operators introduce a discrepancy between the definition of FP and the whole of polynomial-time queries on structures: while any fix-point computation that requires a polynomially bounded number of iterations can be computed in polynomial time[5], only those fix-point computations which are either monotone or inflationary can be defined in FP. As such, one can wonder whether this limit factually hinders the expressivity of FP. In order to present the answer to this question, one additional logic has to be presented: partial fixed-point logic.

[5] If, of course, the function to be iteratively applied is itself polynomial-time computable

Definition 1.23. The *partial fixed-point* operator, denoted pfp, associates a k-ary relation to *any* $\Sigma \sqcup \{R\}$ formula φ with $k = \mathrm{ar}(R)$ free variables $x_1, \ldots x_k$ and any Σ-structure $\mathfrak{A}$. This relation is denoted

$$(\mathrm{pfp}_{R,\vec{x}}\varphi)(\mathfrak{A})$$

Its semantics is as follows: consider the function

$$\begin{aligned} f_\varphi^{\mathfrak{A}} : \mathcal{P}(A^k) &\to \mathcal{P}(A^k) \\ X &\mapsto \varphi(\mathfrak{A}, X) \end{aligned}$$

If successive iterations of $f_\varphi^{\mathfrak{A}}$ on $\emptyset$ reach a fix-point Y, $(\mathrm{pfp}_{R,\vec{x}}\varphi)(\mathfrak{A}) = Y$, otherwise, $(\mathrm{pfp}_{R,\vec{x}}\varphi) = \emptyset$.

That is, the pfp operator enables *any* fix-point computation, even those that require a super-polynomial number of iterations. As such, it is quite intuitive that the corresponding logic exceeds the complexity of P:

Definition 1.24. *Partial fixed-point* logic, denoted PFP is the logic $\mathsf{FO} + \mathrm{pfp}$, defined in the same manner as $\mathsf{FP} = \mathsf{FO} + \mathrm{lfp}$ in Definition 1.20.

Theorem 1.25 ([Var82])**.** PFP *captures* PSPACE *over ordered structures.*

While neither of those capture results (Theorems 1.22 and 1.25) extend to the case of unordered structures (as we will see for FP in the following subsection), interestingly, separation results do transfer from the unordered setting to the ordered setting:

Theorem 1.26 (Abiteboul-Vianu theorem [AV91])**.** *Over arbitrary structures,* $\mathsf{FP} = \mathsf{PFP}$ *iff* $\mathsf{P} = \mathsf{PSPACE}$.

This result can be strengthened to answer the exact question we asked at the beginning of this paragraph: denote $\mathsf{PFP} \restriction \mathrm{poly}$ the fragment of PFP where the pfp operator can only be applied to formulae whose fixpoint is reached in a polynomially-bounded number of steps. Note that $\mathsf{PFP} \restriction \mathrm{poly}$ corresponds precisely to the class of queries we mentioned at the beginning of this paragraph. In [AV91], the following is shown about this class:

Theorem 1.27 (Theorem 3.3 in [AV91])**.** *Over arbitrary structures,* $\mathsf{FP} = \mathsf{PFP} \restriction \mathrm{poly}$ *iff* $\mathsf{P} = \mathsf{PSPACE}$.

As such, there are strong reasons to believe that arbitrary fixpoint computations of polynomially bounded iteration length are *not* definable in FP.

The importance of being ordered

There is an important discrepancy between Fagin's theorem and the other capture results we have presented : the latter require a linear order on the structures at hand. More precisely, the logics $\mathsf{FP}, \mathsf{FO} + \mathsf{tc}$ and $\mathsf{FO} + \mathsf{stc}$ *fail* to capture the corresponding complexity classes in the unordered seting, as they are still unable to define the EVEN property [CH82]. While such a small difference might seem innocuous at first, it drastically reduces the impact of capture results, and we now provide a few reasons why.

First, this weakens the abstraction provided by Descriptive Complexity. Beyond computation models, time and space resources, unordered capture results enable a shift in the way we think of a computational problem. While strings and languages form a unifying setting to define all problems, they often shadow the fact that the problems at hand concern structures (graphs, relational tables, ...), and the intuition behind the best algorithms are often structural insights. Being able to reason directly on the structure, without an encoding, is another layer of encoding enabled by Fagin's theorem in the context of NP.

Moreover, the need to provide a linear order on the structure at hand to ensure that FP is expressive enough to capture polynomial time raises the question of *how dependent on this ordering is the value of a given formula.* Consider a formula $\varphi \in \mathcal{L}[\Sigma \sqcup \{\leq\}]$ (for some logic $\mathcal{L}$). We say that φ is *order-invariant* if, given a Σ-structure $\mathfrak{A}$, and any two linear-orders $\leq_1, \leq_2$ over A,

$$(\mathfrak{A}, \leq_1 \models \varphi) \iff (\mathfrak{A}, \leq_2 \models \varphi)$$

Unfortunately (although expectedly), such a question is undecidable, and the class of FO formulae over ordered structures which are order-invariant is not recursive [Gur88].

Finally, another motivation to find logics corresponding to complexity classes in the first place is that reasoning about logical sentences is often easier. For both FO and FP, we have strong inexpressibility results, relying respectively on Ehrenfeucht-Fraïssé games, and pebble games. The presence of a linear order disables those tools. We do not introduce those tools any further, as they are not relevant to our study. However, we refer the reader to [EF95] for an introduction to both.

The quest of a logic for P

Having introduced our motivation to find a logic for P, we now give a quick overview of the research on the topic. Let us first provide a formal definition of what a logic capturing P should be. While we can trace this notion as far as [CH82], it was stated in its modern form by Gurevich. We reproduce this definition quite faithfully:

Definition 1.28 ([Gur88])**.** A *logic* $\mathcal{L}$ is a pair of functions SEN, SAT such that, for any signature Σ:

- SEN(Σ) is a recursive set. Its elements are the $\mathcal{L}$-sentences on Σ.

- $\mathrm{SAT}(\Sigma) \subseteq \{(\mathfrak{A}, \varphi), \mathfrak{A} \in \mathrm{STRUC}[\Sigma], \varphi \in \mathrm{SEN}(\Sigma)\}$ is also recursive. Moreover, it is *isomorphism-invariant*, that is, if $\mathfrak{A} \simeq \mathfrak{B}$, for any $\varphi \in \mathrm{SEN}(\Sigma)$,

$$(\mathfrak{A}, \varphi) \in \mathrm{SAT}(\Sigma) \iff (\mathfrak{B}, \varphi) \in \mathrm{SAT}(\Sigma)$$

For $\varphi \in \mathrm{SEN}(\Sigma)$, let $\mathrm{Mod}(\varphi) := \{\mathfrak{A} \in \mathrm{STRUC}[\Sigma], (\mathfrak{A}, \varphi) \in \mathrm{SAT}(\Sigma)\}$. $\mathcal{L}$ *captures* P if the following conditions hold:

- There is a Turing machine that, on input $\varphi \in \mathrm{SEN}(\Sigma)$, outputs a pair (M, p), where M is itself a Turing machine, and p a polynomial, such that M accepts exactly all the encodings of all structures in $\mathrm{Mod}(\varphi)$. Moreover, M runs in time bounded by p.
- For any class of Σ-structures $\mathcal{K}$ such that, $L_{\mathcal{K}} \in \mathsf{P}$, there is a sentence $\varphi \in \mathrm{SEN}(\Sigma)$ such that $\mathrm{Mod}(\varphi) = \mathcal{K}$.

As we have stated above, FP does not express Even, which, for any Σ, is the class of Σ-structures with even-sized domain (this is easily shown using pebble games, and a proof can be found, for instance, in [Imm99]).

In this context, a quite natural idea to extend the expressivity of FP is to add to its syntax a *counting mechanism*. There are many ways to define this mechanism, with subtle differences. Before introducing FPC formally, let us give an intuition of its definition.

Consider, for a structure $\mathfrak{A}$ with domain A, the *numerical domain* of $\mathfrak{A}$, is the $\{\leq\}$-structure

$$A^{<} := (\{0, \ldots, |A|\}, \leq_{\mathbb{N}})$$

We can add to any Σ-structure $\mathfrak{A}$ its numerical domain, and consider formulae of $\mathsf{FP}[\Sigma \cup \{\leq\}]$. The Immerman-Vardi theorem ensures that all P computable arithmetic functions can be defined on $A^{<}$. Moreover, since, for any isomorphic structures $\mathfrak{A}, \mathfrak{B}$, they have *the same* numerical domain, this does not break the isomorphism-invariance condition in Definition 1.28. At this point, this extension only enables arithmetic on the size of the structure, however, we can easily represent numbers up to $|A|^d$ using k-tuples from the numerical domain: for each k, let

$$(\vec{\mu}) <_{lex} (\vec{\nu}) := \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{i} \mu_j = \nu_j \right) \wedge \mu_i < \nu_i$$

While $<_{lex}$ is defined with a different formula for each different size of tuple, we refer to $<_{lex}$ in the same way to improve readability. Any integer $N \leq |A|^k$ can be represented by the unique k-tuple $\vec{a}$ on $A^{<}$ such that

$$|\{\vec{b} <_{lex} \vec{a}\}| = N.$$

Note that this representation corresponds exactly to the decomposition of N in base $|A| + 1$. Because $<_{lex}$ is a linear-ordering on $(A^{<})^k$ (for any k), the Immerman-Vardi

theorem once again ensures that all polynomial-time computable arithmetic properties can be defined in FP on integers less than $|A|^k$ using this representation.

In order for this new numerical sort to enable the expression of new properties of *structures*, we aim to introduce a generic counting mechanism, which can be done using a *counting quantifier*. For any structure $\mathfrak{A}$ and any formula φ, $(\#\vec{x}, \varphi)$ is a tuple of terms of size $|\vec{x}|$, representing the integer $|\{\vec{a} \mid (\mathfrak{A}, \vec{a}) \models \varphi\}|$

We now provide a formal definition of FPC:

Definition 1.29 (Syntax of FPC)**.** *Fixed-point logic with counting*, denoted FPC, is defined on any signature Σ such that $(\leq) \notin \Sigma$. In that context, $\mathsf{FPC}[\Sigma]$ is the smallest set of formulae such that:

- $\mathsf{FP}[\Sigma \cup \{\leq\}] \subseteq \mathsf{FPC}$.
- For any $\mathsf{FPC}[\Sigma]$ formula φ, and $\vec{x}$ a tuple of variables, $(\#\vec{x}, \varphi)$ is a tuple of FPC-term, with free variables $\mathrm{free}(\varphi) \setminus \{\vec{x}\}$.

Given a Σ-structure $\mathfrak{A}$, let

$$\mathfrak{A}^+ := (A \sqcup [0, \ldots, |A|], (R^{\mathfrak{A}})_{R \in \Sigma}, \leq_{\mathbb{N}})$$

When $\varphi \in \mathsf{FPC}[\Sigma]$, $\varphi(\mathfrak{A})$ is the set of valuations of $\mathrm{free}(\varphi)$ over $\mathfrak{A}^+$ that satisfy φ, in the sense of Definition 1.5, with the following additional rule for the interpretation of terms: if v is a valuation of $\mathrm{free}(\varphi) \setminus \{\vec{x}\}$, let

$$N_v := |(\varphi(\mathfrak{A}) \cap \{v\}_{\uparrow \mathrm{free}(\varphi) \cup \{\vec{x}\}})_{\downarrow \vec{x}}|$$

(that is, N_v is the number of extensions of v to $\vec{x}$ which satisfy φ). Then, $v^*(\#\vec{x}, \varphi)$ is a tuple of elements in $A^<$, the i-th one taking the value of the i-th digit in the decomposition of N_v in base $|A| + 1$.

Example 1.30. Recall the formula ustcon on $\{E\}$-structures defined in Example 1.21. We now provide a FPC-term which evaluates to the number of connected components in a graph $\mathfrak{G}$.

First, the Immerman-Vardi theorem ensures that, for any numerical FPC-term t with $\mathrm{free}(t) = \vec{\mu}$, all numerical variables, FPC defines a term t^* which evaluates to $\sum_{i<n^k} t(\mathfrak{A}, i)$.

For any $i \leq n$,

$$\mathsf{cc\#size}(i) := (\#x, (\#y, \mathsf{ustcon}(x, y)) = i)/i$$

is a FPC-term which counts the number of connected components of size i. As such,

$$\mathsf{cc\#} := \sum_{i=1}^{n} \mathsf{cc\#size}(i)$$

evaluates on any graph $\mathfrak{G}$ to the number of connected components of $\mathfrak{G}$.

This presentation of FPC is a direct reformulation of [Ott17]. There is an alternative definition of FPC, where the numerical sort is defined entirely through the semantics of the logic, rather than by considering evaluations over $\mathfrak{A}^+$. In this case a single numerical variable can store arbitrarily large numerical values, and no encoding scheme is needed to store such a large numerical value within a tuple of numerical variables. On the other hand, this unboundedness of values taken by numerical variables implies that, in order to maintain polynomial-time computability, a polynomial bound on quantification of numerical variables must be syntactically enforced in the traditional presentation of FPC. This alternative presentation is used, for instance, in [Daw15]. All in all, both presentations have merits. However, our need in further chapters to consider permutations over $A^<$ makes our presentation more convenient in our setting.

We will respect usage, and follow further conventions when writing and handling formulae. First and foremost, we separate *domain variables*, which range over A, composed of *domain elements*, and *numerical variables*, which range over $A^<$, the set of *numerical elements*. This distinction constitutes the *type* of a variable.

When reasonable, we keep distinct names for domain variables (for instance x, y, z) and numerical variables (for instance n, k, μ, ν, λ). However, for reasons to become clear in the following chapters, the strong separation of variable symbols can often bear a cost on readability. In particular, we will often consider tuples of variables of the form $\vec{x}$, and it is often convenient to consider such tuples containing different types.

We define the *type* of a tuple $\vec{x}$ of variables, denoted $\mathrm{type}(\vec{x})$ to be the unique word $w \in \{\mathrm{element}, \mathrm{number}\}^*$ such that x_i is a domain variable iff $w_i = \mathrm{element}$. If a type belongs to $\{\mathrm{element}\}^*$, it is called *plain* We often need to consider the set underlying all potential valuations of a tuple $\vec{x}$. We therefore denote $A^{\mathrm{type}(\vec{x})}$ the set $\prod_{i=1}^{|\vec{x}|} A^{\mathrm{type}(\vec{x})_i}$, where $A^{\mathrm{element}} := A$ and $A^{\mathrm{number}} := A^<$.

We now allow types instead of arity in the definition of signatures. For instance, if a relation symbol R has type $(\mathrm{number}, \mathrm{number}, \mathrm{element})$, any interpretation of R on A should be a subset of $A^< \times A^< \times A$. Finally, we overload this notation to relations themselves, so that if X is a relation over $\mathfrak{A}^+$, $\mathrm{type}(X)$ is the unique type-word such that $X \subseteq A^{\mathrm{type}(X)}$.

If a signature Σ contains one or more non-plain relation symbols, it should also contain a binary relation $\leq$, and we expect Σ-structures to interpret $\leq$ as the linear-order over the numerical sort. In other words, to consider structures with numerical relations, we expect the numerical sort to be provided within the structure.

Recall that Theorem 1.27 states that polynomially-bounded iteration of arbitrary formulae (as in the definition of the pfp operator) is unlikely to be definable in FP, as that would imply P = PSPACE. The picture is different in the context of counting logics. Denote PFPC the logic PFP augmented with the counting operator, and PFPC $\restriction$ poly the fragment of PFPC where the pfp operator can only be applied to formulae whose fixpoint is reached in a polynomially-bounded number of steps.

Theorem 1.31 (Theorem 4.21 in [Ott17])**.** FPC = PFPC $\restriction$ poly

That is, FPC can simulate *arbitrary* fixed-point computations, as long as this simulation lasts for a polynomially-bounded number of iterations. Because this result plays a part in the proof of our results in Chapters 4 and 5, we provide a construction of this simulation:

Proof. Let $\varphi(R, \vec{x})$ be a Σ-formula such that, for any Σ-structure $\mathfrak{A}$, the sequence $(X_i^{\mathfrak{A}})$ defined by $X_0^{\mathfrak{A}} = \emptyset, X_{i+1}^{\mathfrak{A}} := \varphi(\mathfrak{A}, X_i^{\mathfrak{A}})$ reaches a fixed point in time bounded by $|A|^d$. Consider the following formula $\psi(T, \vec{\mu}, \vec{x})$, where $\vec{\mu}$ is a d-tuple of numerical variables, and $\text{type}(T) = \text{number}^d \cdot \text{type}(R)$.

$$\psi(T, \vec{\mu}, \vec{x}) := \bigwedge \begin{cases} \vec{\mu} > 0 \\ \forall \vec{y}, \neg T(\vec{\mu}, \vec{y}) \\ \forall \vec{\nu}, 0 <_{lex} \vec{\nu} <_{lex} \vec{\mu} \implies \exists \vec{y}, T(\vec{\nu}, \vec{y}) \\ \exists \vec{m}, \vec{\mu} = \vec{m} + 1 \wedge \varphi(R, \vec{x})[R(\vec{\alpha})/T(\vec{m}, \vec{\alpha})] \end{cases}$$

We claim that $(\text{ifp}_{T,\vec{\mu},\vec{x}}\psi)$ defines the *trace* of the sequence $(X_i^{\mathfrak{A}})$ up to $|A|^d$, that is, the relation $\mathcal{T}_{|A|^d}(X_i^{\mathfrak{A}})$, where

$$\mathcal{T}_k(X_i^{\mathfrak{A}}) := \{(i, \vec{a}), \vec{a} \in X_i^{\mathfrak{A}}, i < k\}$$

(where i is encoded as usual). We show this by induction on the number of times ψ is applied within the ifp definition. That is, let $(Y_i^{\mathfrak{A}})$ be the sequence of relations on $\mathfrak{A}$ defined by $Y_0^{\mathfrak{A}} := \emptyset$ and $Y_{i+1}^{\mathfrak{A}} := Y_i^{\mathfrak{A}} \cup \psi(\mathfrak{A}, Y_i^{\mathfrak{A}})$. We claim that, for any $k < |A|^d$, $Y_k^{\mathfrak{A}} = \mathcal{T}_k(X_i^{\mathfrak{A}})$. This is obviously true for $k = 0$. Now, proving the induction step amounts to show that

$$\mathcal{T}_k(X_i^{\mathfrak{A}}) \cup \psi(\mathcal{T}_k(X_i^{\mathfrak{A}})) = \mathcal{T}_{k+1}(X_i^{\mathfrak{A}}).$$

If we first exclude the case where $X_i^{\mathfrak{A}} = \emptyset$ for some $i > 0$, remark that the only possible value of $\vec{m}$ satisfying all the conjuncts in the definition of ψ evaluated on $\mathcal{T}_k(X_i^{\mathfrak{A}})$ is the unique $\vec{m}$ which encodes k. As such, the only tuples $(\vec{\mu}, \vec{x})$ in $\psi(\mathfrak{A}, \mathcal{T}_k(X_i^{\mathfrak{A}}))$ are those where $\vec{\mu}$ encodes $k+1$, and $\vec{x} \in \varphi(\mathfrak{A}, S)$, where S is the k-component of $\mathcal{T}_k(X_i^{\mathfrak{A}})$. But, by definition of $\mathcal{T}$, $S = X_k^{\mathfrak{A}}$, and thus $(\vec{\mu}, \vec{x}) \in \psi(\mathfrak{A}, \mathcal{T}_k(X_i^{\mathfrak{A}}))$ iff $\vec{\mu}$ encodes $k+1$ and $\vec{x} \in X_{k+1}^{\mathfrak{A}}$, which yields the desired result. Now, if for any $0 < k < |A|^d$, $X_k^{\mathfrak{A}} = \emptyset$, then $\psi(\mathfrak{A}, \mathcal{T}_k(X_i^{\mathfrak{A}})) = \emptyset$ and a fixed-point is reached immediately. This is coherent with the fact that we have just witnessed a loop in the partial fixed-point computation of φ, and as such the pfp operator would evaluate to the empty relation.

Finally, we can retrieve the $|A|^d - 1$ component of $\mathcal{T}_{|A|^d}(X_i^{\mathfrak{A}})$ to obtain the result of $|A|^d - 1$ iterations of φ, starting from $\emptyset$:

$$\mathsf{result}(\vec{y}) := (\text{ifp}_{T,\vec{\mu},\vec{x}}\psi(T, \vec{\mu}, \vec{x}))(|A|^d - 1, \vec{y})$$

□

The *trace argument* presented in this proof will be used several times in the following chapters.

Caï, Furer and Immerman [CFI92] showed that FPC fails to capture P, by introducing structures which can be distinguished in the latter but not in the former. Those structures became known as Cai-Fürer-Immerman structures (or CFI-structures for short), and play an important part in the quest of a logic for P. They will be discussed at length in Section 4.1.

The separation between FPC and P led to the introduction of new candidate logics for P. In [BGS97], the authors define *Choiceless polynomial time* (CPT for short) which, while it conforms to Gurevich's definition, requires explicit bound on the computation time. While CPT is an important object in this field of research, its definition is somewhat tedious, and is not relevant to our work. We therefore refer the interested reader to [DRR08; Ros10] for a formal definition of this logic. An interesting characterisation in terms of sequences of interpretations was given recently in [Gra+15]. Let us mention that CPT is known to extend FPC [BGS97], and to express the CFI-property [DRR08].

Another extension of FPC was introduced by Dawar et al. [Daw+09], motivated by the observation that the CFI problem reduces to deciding the satisfiability of a system of linear equations over the two-elements field. In this article, they introduce the rk operator, a generalisation of the counting quantifier, that enables the definition of the rank of a definable matrix. We assume the reader to be familiar with linear algebra, and only remind that if $M \in \mathbb{K}^{I\times J}$ is a matrix over $\mathbb{K}$, the *rank* of M, denoted $\mathsf{rank}(M)$ is the dimension of the vector space $\{M \cdot v, v \in \mathbb{K}^I\}$. Gaussian elimination enables the computation of the rank of a matrix in polynomial time. For p a prime number, we denote $\mathbb{F}_p$ the prime field with p elements. For $M \in \mathbb{Z}^{I\times J}$, we denote $\mathsf{rank}_p(M)$ the rank of the matrix M_p in $\mathbb{F}_p^I$, where M_p is the matrix whose coefficients are the residuals of M modulo p.

Definition 1.32. A relation $R \subseteq X \times X \times \mathbb{N}$ is said to *define a matrix over* X if, for all $x, y \in X$, there is a unique $m_{x,y} \in \mathbb{N}$ such that $(x, y, m_{x,y}) \in R$. $M_R := (m_{x,y})_{x,y\in X}$ is the *matrix defined by* R.

When such a relation R is definable by a formula within FPC, the rk operator enables to define its rank, over any prime field $\mathbb{F}_p$ (with $p \leq |\mathfrak{A}|$):

Definition 1.33. $(\mathsf{FP}+\mathsf{rk})[\Sigma]$ is extension of FP the smallest set of formulae such that:

- $\mathsf{FP}[\Sigma] \subseteq (\mathsf{FP} + \mathsf{rk})[\Sigma]$
- Given a formula $\varphi \in (\mathsf{FP} + \mathsf{rk})[\Sigma]$, two tuples of variables $\vec{x}, \vec{y}$ of same type, and one additional tuple of numeric variable $\vec{\pi}$,

  $$(\mathsf{rk}_{\vec{x},\vec{y},\vec{\pi}}\varphi)$$

  is a tuple of numerical term in $\mathsf{FP} + \mathsf{rk}$ of size $|\vec{x}|$, and free variables $\mathrm{free}(\varphi) \setminus \{\vec{x}, \vec{y}, \vec{\pi}\}$.

For such a formula φ, a structure $\mathfrak{A}$ and a valuation v of all free variables in φ except $\{\vec{x}, \vec{y}\}$,

$$v^*(\mathsf{rk}_{\vec{x},\vec{y},\vec{\pi}}\varphi) := \mathsf{rank}_{v(\vec{\pi})}(M_R)$$

where $v(\vec{\pi})$ is the integer represented by the tuple of values

$$(v(\pi_1), v(\pi_2), \ldots, v(\pi_\lambda)),$$

and M_R is the matrix defined by $(\varphi(\mathfrak{A}) \cap v_{\uparrow \mathrm{free}(\varphi)})_{\downarrow \vec{x},\vec{y}}$. If $\varphi(\mathfrak{A})$ does not define a matrix in this way, the value of the rk term is 0 by convention. We call rk the *rank operator*.

Note that, if $\vec{x}, \vec{y}$ are composed only of numerical variables, the whole matrix defined by φ is ordered, and the Immerman-Vardi theorem ensures that FPC can define its rank.

Note also that this definition is not exactly the one presented in [Daw+09]. A first innocuous difference is the way in which we encode matrices. In the original presentation, φ was supposed to be a numerical term. A second, more important difference, is that in [Daw+09], the parameter p was not a variable, but rather part of the operator, while here, the size of the field on which the rank computation is to be done is dynamically selected within the logic. We call the former the *non-uniform rank operator*, and the latter the *uniform rank operator*. It was shown in [GP19] that the non-uniform version is strictly weaker than the uniform one.

In [Daw+09], the authors showed that $\mathsf{FP} + \mathsf{rk}$ expresses the CFI-query. Moreover, the counting quantifier can easily be simulated using the rk quantifier. Therefore, $\mathsf{FPC} < \mathsf{FP} + \mathsf{rk}$. Quite recently, Lichter [Lic23] showed that $\mathsf{FP} + \mathsf{rk}$ does not capture P. This separation result will be discussed further in Chapter 4.

Having given a satisfying overview of the research question studied in our work, we complete our introduction of the descriptive complexity notions used in this thesis.

Reductions, interpretations, generalised quantifiers

In Chapter 2, we will introduce another extension of FPC, through another operator. To avoid defining it in the same tedious way as we have defined the counting quantifier, and the rank operator, we now introduce a uniform framework to define further extensions of FO: generalised quantifiers. As we will see, this formalism of generalised quantifiers happens naturally as we spin the metaphor between computational complexity and logical definability.

First and foremost, a fundamental notion of complexity is that of reduction. Let us show how this notion translates in the descriptive complexity setting.

A (many-one) *reduction* between two languages $\mathcal{L} \subseteq \Sigma^*$ and $\mathcal{M} \subseteq \Omega^*$ is a function $f : \Sigma^* \to \Omega^*$ such that, for any word $w \in \Sigma^*$, $w \in \mathcal{L} \iff f(w) \in \mathcal{M}$. f is a P (resp. L)-reduction if it is computable in polynomial time (resp. logarithmic space). We are usually interested in low-complexity reductions. Reductions are pervasive in complexity theory, as they allow the definition of problems complete for a class. While first-order logic seems well-equipped to define decision problems, it is not the case for function

problems. Actually, with our definition of the semantics of FO (Definition 1.5), we can define functions from A to A^k for any fixed k, which is not so far from our objective. However, we would expect our formalism to enable the definition of functions from Σ-structures to Ξ-structures, for any couple of signatures (Σ, Ξ). The notion of FO-interpretation covers exactly that, while also allowing the domain of the structure to change in FO-definable ways.

Definition 1.34. Given a Ξ-structure $\mathfrak{A}$, a *congruence* on $\mathfrak{A}$ is an equivalence relation $\simeq$ over A such that, for any $R \in \Xi$, and any $\vec{a}, \vec{b} \in A^{\mathrm{ar}(R)}$ such that $a_i \simeq b_i$,

$$\vec{a} \in R^{\mathfrak{A}} \iff \vec{b} \in R^{\mathfrak{A}}$$

If $\simeq$ is a congruence on $\mathfrak{A}$, $\mathfrak{A}/\simeq$ is the Ξ-structure $\mathfrak{C}$ defined as follows:

- $C = A/\simeq$
- for all $R \in \Xi$, $R^{\mathfrak{C}} = \iota_{\mathrm{ar}(R)}(R^{\mathfrak{A}})$

where ι_k is the component wise canonical injection from A^k to C^k.

It is useful to state the definition of interpretations in a general way, for any logic $\mathcal{L}$ extending FO. Note that here, our definition of a logic is slightly less general than in Definition 1.28: we expect $\mathcal{L}$ to have a notion of free variables, which enables the definition of relations, and not only boolean queries. All the logics considered in this thesis have this property.

Definition 1.35 ($\mathcal{L}$-interpretation)**.** Let Σ, Ξ be two signatures such that Ξ contains no constant symbol, and $k := \max_{R \in \Xi} \mathrm{ar}(R)$. A $\mathcal{L}[\Sigma, \Xi]$-*interpretation* $\mathcal{I}$ of *type* t with parameters $\vec{p}$ is a family of $\mathcal{L}[\Sigma]$ formulae

$$\left\langle \varphi_{\mathrm{dom}}, \varphi_{\simeq}, (\varphi_R)_{R \in \Xi} \right\rangle$$

such that

- $\mathrm{free}(\varphi_{\mathrm{dom}}) \subseteq \{\vec{x}, \vec{p}\}$
- $\mathrm{free}(\varphi_{\simeq}) \subseteq \{\vec{x}, \vec{y}, \vec{p}\}$
- For all $R \in \Xi$, $\mathrm{free}(\varphi_R) \subseteq \{\vec{z}_1, \ldots, \vec{z}_{\mathrm{ar}(R)}, \vec{p}\}$

for some pairwise-disjoint tuples of variables $\vec{x}, \vec{y}, \vec{z_1} \ldots, \vec{z_k}$, each of the same type t.

For $\mathfrak{A} \in \mathrm{STRUC}[\Sigma]$, and $\vec{a}$ a valuation of $\vec{p}$ on $\mathfrak{A}$, consider the following Ξ-structure $\mathfrak{B}$:

- $B = \varphi_{\mathrm{dom}}(\mathfrak{A}, \vec{a}) \subseteq A^t$

- For any $R \in \Xi$, $R^{\mathfrak{B}} = \varphi_R(\mathfrak{A}, \vec{a}) \subseteq A^{t^{\mathrm{ar}(R)}}$

If $\varphi_{\simeq}(\mathfrak{A}, \vec{a}) \subseteq A^{t^2}$ is a congruence on $\mathfrak{B}$, $\mathcal{I}$ maps $(\mathfrak{A}, \vec{a})$ to $\mathfrak{B}/(\varphi_{\simeq}(\mathfrak{A}, \vec{a}))$, which we denote $\mathcal{I}(\mathfrak{A}, \vec{a})$. Otherwise, $\mathcal{I}(\mathfrak{A}, \vec{a})$ is the empty Ξ-structure.

The definition of interpretations varies widely between different sources. Definition 1.35 is very general, as it allows parameters, multi-dimensional interpretations, filtering of the domain and quotienting by a congruence. This is the flavour of interpretations adopted, amongst others, in [Gra+15; DFS22].

Example 1.36. Consider Σ the signature of graphs.

$$\mathcal{I} := \langle x = x, x = y, \neg E(z_1, z_2) \rangle$$

is a $[\Sigma, \Sigma]$-interpretation of type (element) with no parameters. For any graph $\mathfrak{G}$, $\mathcal{I}(\mathfrak{G})$ is the complement graph of $\mathfrak{G}$.

Given a graph $\mathfrak{G} = (V, E)$, its *line graph* is the graph $L_G = (E, X)$, where

$$X = \{((u, v), (u', v')) \mid u = u' \vee u = v' \vee v = u' \vee v = v'\}$$

that is, the vertices of L_G are the edges of G, and there is an edge in L_G between two edges of G if those edges share a common end-point in G.

$$\begin{aligned}\mathcal{J} := \langle & E(x_1, x_2), \\ & (x_1 = y_1 \wedge x_2 = y_2) \vee (x_1 = y_2 \wedge x_2 = y_1), \\ & (x_1 = y_1 \iff x_2 \neq y_2) \vee (x_1 = y_2 \iff x_2 \neq y_1) \rangle\end{aligned}$$

is a $[\Sigma, \Sigma]$-interpretaion of type (element^2) with no parameters. For any *simple* graph $\mathfrak{G}$, $\mathcal{J}(\mathfrak{G})$ is the line graph of $\mathfrak{G}$.

Note that, if Ξ contains mixed-type relations, a $[\Sigma, \Xi]$-interpretation $\mathcal{I}$ should explicitly define the numerical sort, which can be done when the logic $\mathcal{L}$ at hand extends FPC:

Lemma 1.37. *Let $\mathcal{L}$ be any logic extending* FPC*, and $\mathcal{I} = \langle \varphi_{dom}, \varphi_{\simeq}, (\varphi_R)_{R \in \Xi} \rangle$ be a $[\Sigma, \Xi]$-interpretation in $\mathcal{L}$. There is a $\mathcal{L}$ interpretation $\mathcal{I}' = \langle \psi_{dom}, \psi_{\simeq}, (\psi_R)_{R \in \Xi} \rangle$, such that for any Σ-structure $\mathfrak{A}$, $\mathcal{I}'(\mathfrak{A}) \simeq \mathcal{I}(\mathfrak{A})^+$.*

Proof. Suppose $\mathcal{I}$ has type t. We will consider an interpretation $\mathcal{I}'$ of type t', where $t' = (\mathrm{number} \cdot t \cdot \mathrm{number}^{|t|})$. For ease of writing, we represent tuples of type t' as $\vec{x} = b_x \vec{z}_x \vec{\mu}_x$, where $\mathrm{type}(\vec{z}_x) = t$ and $\mathrm{type}(\vec{\mu}_x) = \mathrm{number}^{|t|}$. Intuitively, b_x is used to disambiguate between the numerical and domain sorts of $\mathcal{I}'$. If $b_x = 0$, $\vec{x}$ represents an element of the domain sort, provided by $\vec{z}_x$, and if $b_x = 1$, $\vec{x}$ represents an element

of the numerical sort, provided by $\vec{\mu}_x$. This already enables the definition of ψ_{dom} and $\psi_{\simeq}$:

$$\psi_{dom}(b_x\vec{z}_x\vec{\mu}_x,\vec{p}) := \bigvee\begin{cases} b_x = 0 \wedge \varphi_{dom}(\vec{z}_x,\vec{p}) \\ b_x = 1 \wedge \vec{\mu}_x \leq_{lex} \mathsf{cc}\mathcal{I}(\vec{p}) \end{cases}$$

$$\psi_{\simeq}(b_x\vec{z}_x\vec{\mu}_x, b_y\vec{z}_y\vec{\mu}_y,\vec{p}) := \bigvee\begin{cases} b_x = b_y = 0 \wedge \varphi_{\simeq}(\vec{z}_x,\vec{z}_y,\vec{p}) \\ b_x = b_y = 1 \wedge \vec{\mu}_x = \vec{\mu}_y \end{cases}$$

where $\mathsf{cc}\mathcal{I}(\vec{p})$ is a numerical term which evaluates on $\mathfrak{A},\vec{a}$ to the number of elements in the structure $\mathcal{I}(\mathfrak{A},\vec{a})$. In FPC, $\mathsf{cc}\mathcal{I}$ can be defined by counting the number of equivalence classes of $\varphi_{\simeq}$ amongst the tuples of type t that satisfy φ_{dom}. We have already seen in Example 1.30 that FPC can count the number of connected components of a graph, and $\mathsf{cc}\mathcal{I}(\vec{p})$ can be defined in $\mathcal{L}$ as a direct adaptation of $\mathsf{cc}\#$ (as defined in Example 1.30), by replacing each occurrence of $E(s,t)$ by $\varphi_{\simeq}(\vec{z}_x,\vec{z}_y,\vec{p})$, and renaming variables accordingly. Care should be taken to ensure that only equivalence classes of tuples satisfying φ_{dom} are counted, and that if $\varphi_{\simeq}$ does not define a congruence on $(\mathfrak{A},\vec{a})$, $\mathsf{cc}\mathcal{I}(\mathfrak{A},\vec{a})$ evaluates to 0.

The linear order on the newly defined numerical sort is easily definable:

$$\psi_{\leq}(b_x\vec{z}_x\vec{\mu}_x, b_y\vec{z}_y\vec{\mu}_y,\vec{p}) := b_x = b_y = 1 \wedge \vec{\mu}_x \leq_{lex} \vec{\mu}_y$$

and the interpretation of relations from $\mathcal{I}$ is easily transferred to $\mathcal{I}'$:

$$\psi_R(b_1\vec{z}_1\vec{\mu}_1, b_2\vec{z}_2\vec{\mu}_2,\ldots,b_k\vec{z}_k\vec{\mu}_k,\vec{p}) := \varphi_R(\vec{z}_1,\vec{z}_2,\ldots,\vec{z}_k,\vec{p}) \wedge \bigwedge_{i=1}^{k} b_i = 0 \qquad \square$$

Therefore, to define an interpretation of a mixed-type relation R, we first construct such an interpretation $\mathcal{I}'$, and add an interpretation of R using such tuples of type $(\text{number}\cdot t\cdot \text{number}^{|t|})$ to encode both types in the target structure.

The notion of reduction allows us to compare the complexity of two problems with respect to a given complexity class. If there is a P-reduction f from $\mathcal{L}$ to $\mathcal{M}$, any polynomial-time algorithm for $\mathcal{M}$ yields an algorithm for $\mathcal{L}$: on input w, compute $f(w)$ in polynomial time, then run the algorithm for $\mathcal{M}$ on $f(w)$[6]. Interpretations enable the same reasoning with structures and FO:

Proposition 1.38. *Let φ be a $\mathsf{FO}[\Xi]$ sentence defining a class of structures $\mathcal{K}$, and $\mathcal{I}$ a $[\Sigma,\Xi]$-interpretation. There is a Σ-formula $\varphi^{\mathcal{I}}$ such that, for any $\mathfrak{A}\in \mathrm{STRUC}[\Sigma]$, and any valuation $\vec{a}$ of the parameters of $\mathcal{I}$,*

$$\mathfrak{A},\vec{a} \models \varphi^{\mathcal{I}} \iff \mathcal{I}(\mathfrak{A},\vec{a}) \in \mathcal{K}$$

[6]Note that $|f(w)|$ is necessarily polynomially bounded by $|w|$ since f is computable in polynomial-time

Sketch of proof. The intuition is simple: it suffices to rewrite φ, replacing each reference to the Ξ-structure to its interpretation through $\mathcal{I}$. Care should be taken as to treat the case where $\mathcal{I}(\mathfrak{A})$ is the empty structure. □

In complexity theory, a second, weaker reduction relation is used: Turing reductions. A *Turing reduction* from $\mathcal{L}$ to $\mathcal{M}$ is a Turing machine that decides $\mathcal{L}$, but is allowed to query an arbitrary number of times an *oracle* (a black box within the Turing machine) for the language $\mathcal{M}$. Once again, Turing reductions may be considered for various complexity classes, the resources bounds from that complexity class applying to the reduction Turing machine. A formal definition of oracle Turing machines can be found in [AB09, Definition 3.4] Once again, we introduce a formalism to represent such oracle machines. Note that, now, we are not merely defining functions, but *extending* what a computation can do, and this will naturally correspond to an *extension* of first-order logic.

Definition 1.39 (*Generalised quantifier*)**.** Let $\mathcal{K}$ be a class of Ξ-structures. We denote $\mathsf{FO}{+}\mathcal{K}$ (resp $\mathsf{FP}{+}\mathcal{K}$) the logic whose sentences are defined inductively as first-order logic (resp. fixed-point logic), with the additional rule that, if $\mathcal{I}$ is a $[\Sigma, \Xi]$-interpretation,

$$(Q_{\mathcal{K}}\mathcal{I})$$

is a formula of $(\mathsf{FO} + \mathcal{K})[\Sigma]$ (resp. $(\mathsf{FP} + \mathcal{K})[\Sigma]$). $(Q_{\mathcal{K}}\mathcal{I})(\mathfrak{A})$ is the set of valuations $\vec{a}$ of the parameters of $\mathcal{I}$ such that

$$\mathcal{I}(\mathfrak{A}, \vec{a}) \in \mathcal{K}$$

Like interpretations, the definition of generalised quantifiers admit many variations, and the one given here is very general. It is the same as in [DFS22].

In Chapters 4 and 5, we will define many complex formulae. In that context, having a denotational way to represent procedural rewritings in a formula is invaluable.[7] To that effect, we will use the formalism of generalised quantifiers in the following way:

Definition 1.40. Given a class of structures $\mathcal{K}$, and a logic $\mathcal{L}$ extending FO, the generalised quantifier $Q_{\mathcal{K}}$ is *definable within* $\mathcal{L}$ if, for any $[\Sigma, \Xi]$-interpretation $\mathcal{I}$ definable in $\mathcal{L}$, there is a formula $\varphi_{\mathcal{K}(\mathcal{I})}$ such that, for any structure $\mathfrak{A}$,

$$\varphi_{\mathcal{K}(\mathcal{I})}(\mathfrak{A}) = (Q_{\mathcal{K}}\mathcal{I})(\mathfrak{A})$$

Clearly, if $Q_{\mathcal{K}}$ is definable within $\mathcal{L}$, we do not alter the expressivity of $\mathcal{L}$ by using $Q_{\mathcal{K}}$ within $\mathcal{L}$ (this is a direct corollary of Proposition 1.38). As such, we will sometimes show that such a quantifier is definable in some logic $\mathcal{L}$ in order to use it as a rewriting mechanism in $\mathcal{L}$-formulae.

[7] Seeing formulae as programs, it allows us to factor programs into smaller and simpler modular programs.

Capture through canonisation

After this digression, let us delve back into the quest of a logic for P. While whether a logic captures P in general has remained open for more than thirty years, a great amount of progress has been made on some restricted classes of structures. The best instantiation of this effort is probably Grohe's result, that FPC captures P on any class of structures which exclude a minor. A book [Gro17] was devoted to the proof of this result, and the following definitions and examples can be found formally stated in its third chapter. In this section, we only aim to give a solid intuition of this initiative, and will provide references to [Gro17].

Intuitively, the definition of a logic $\mathcal{L}$ capturing P on a class of structures $\mathcal{C}$ is a relaxation of that of Definition 1.28, where the two translations (between P-properties and $\mathcal{L}$-sentences) are only required to be correct on structures (and encoding thereof) in $\mathcal{C}$ [Gro17, Definition 3.1.9].

This allows us to formalize a natural intuition about the Immerman-Vardi theorem: we can extend its reach from the class of *linearly-ordered* structures to the class of structures which can be *ordered* within FP (or any logic $\mathcal{L}$ extending FP). A structure $\mathfrak{A}$ can be ordered within FP if there is a formula $\varphi \in \mathsf{FP}$ such that $\varphi(\mathfrak{A}) \subseteq A \times A$ is a linear-order relation. We can relax this definition so that φ has additional free variables, such that for some valuation $\vec{v}$ of those free variables, $\varphi(\mathfrak{A}, \vec{v})$ is a linear-order on A. Finally, we can extend this definition to classes of structures [Gro17, Definition 3.2.2].

Let us fix a logic $\mathcal{L}$ that extends FP. If $\mathcal{C}$ is a class of Σ-structures on which $\varphi \in \mathcal{L}[\Sigma]$ defines linear-orders, then the Immerman-Vardi theorem naturally implies that $\mathcal{L}$ captures P on $\mathcal{C}$: given a P-property $\mathcal{P}$ on $\mathcal{C}$,[8] the formula Ψ, obtained by replacing each occurrence of $\leq$ by φ in ψ is *exactly* a $\mathcal{L}[\Sigma]$-sentence defining $\mathcal{P}$. While this presentation does not take into account the parameters of φ, those can be dealt with, as formalised in [Gro17, Lemma 3.2.6]. An example of such a class $\mathcal{C}$ is given in Grohe's book:

Example 1.41 ([Gro17, Example 3.2.15])**.** The class $\mathrm{CGraphs}_2$ of connected graphs with maximum degree 2 admits FP definable orders, and as such, FP captures P on $\mathrm{CGraphs}_2$.

While this indeed extends the reach of Immerman-Vardi theorem, there are strong limitations to the definition of a linear-ordering on a structure, and this is due to the fact that any such defined relation must be *canonical*. By canonical, we mean that it is isomorphism-invariant, in the following sense:

Definition 1.42. Let $\mathfrak{A}, \mathfrak{B}$ be two isomorphic structures, and let $f : \mathfrak{A} \to \mathfrak{B}$ be an isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$. Fix a type $w \in \{\text{element}, \text{number}\}^*$.

[8] According to [Gro17, Definition 3.1.9], $\mathcal{P}$ outside of $\mathcal{C}$ is undefined. It must only be so that there is a Turing machine that always terminates in polynomial-time that agrees with $\mathcal{P}$ over $\mathcal{C}$.

- For any $\vec{a} \in A^w$, the *image* of $\vec{a}$ through f is the tuple $\vec{b}$, where $b_i = f(a_i)$ if a_i is a domain element (that is, $w_i =$ element), and $b_i := a_i$ if a_i is a numerical element (that is, $w_i =$ number). We denote it $\vec{a}^f$.
- Given a relation $R \subseteq A^w$, the *image* of R through f is

$$R^f := \{\vec{a}^f, \vec{a} \in R\}$$

A relation $R \subseteq A^w$ is *canonical* w.r.t. $\mathfrak{A}$, if, for any structure $\mathfrak{B}$ and any two isomorphisms $f, g : \mathfrak{A} \to \mathfrak{B}$, $R^f = R^g$. We sometimes call such a relation *isomorphism-invariant*.

It is easy to see that, for any formula φ and any structure $\mathfrak{A}$, $\varphi(\mathfrak{A})$ is canonical. The notion of canonicity is a natural extension of the isomorphism-invariance requirement of Definition 1.28 from sentences to formulae.

While the canonicity of relations defined by a formula is a valuable advantage of logics, it dramatically hinders their ability to define linear-orderings. Indeed, fix a structure $\mathfrak{A}$, and suppose that it has N pairwise distinct isomorphisms onto itself. For any fixed linear-order $\prec$ on $\mathfrak{A}$, each of those isomorphisms yield a *distinct* image of $\prec$. As such, we cannot expect a formula to define a *single* linear-ordering on $\mathfrak{A}$. The inclusion of parameters in the definition of definable linear-orders gives hope, as, if φ has k parameters, it can define a *collection* of as many as $|A|^k$ different linear-orders, which might be canonical.

However, many classes of structures have a number of such isomorphisms that is not bounded by any polynomial. Consider, for instance, the class of all complete undirected graphs $(K_n)_{n\in\mathbb{N}}$. For each n, each bijection $\tau : [n] \to [n]$ is an isomorphism of K_n onto itself, and they are $n!$ such bijections. As such, we cannot expect a formula to define a linear-ordering of all complete graphs, even with parameters.

To overcome this limitation we aim, instead of defining a linear-order on the relational part of the structure, to define an isomorphic copy of the relational part of the structure on the numerical sort, that is, on the ordered domain.

Definition 1.43 (Canonisation)**.** Given a logic $\mathcal{L}$ and $\mathcal{C}$ a class of structures, $\mathcal{L}$ *canonises* structures in $\mathcal{C}$, if for any signature Σ, there is a $\mathcal{L}$-definable $[\Sigma, \Sigma \cup \{\leq\}]$-interpretation $\mathcal{I}$ such that, for any Σ-structure $\mathfrak{A} \in \mathcal{C}$

- $\leq^{\mathcal{I}(\mathfrak{A})}$ is a linear-order on the domain of $\mathcal{I}(\mathfrak{A})$
- $\mathfrak{A} \cong \mathcal{I}(\mathfrak{A})_{\restriction\Sigma}$, where $\mathcal{I}(\mathfrak{A})_{\restriction\Sigma}$ is the structure $\mathcal{I}(\mathfrak{A})$ where we have forgotten $\leq$.

Graph Canonisation is also a computational problem, and we will talk in depth about its connection with the definability of a canonical encoding in the next subsection. For now, let us just remark that, in the computational context, we also expect two

isomorphic graphs G, H to yield the same output. Here, this restriction is not needed since $\mathcal{I}(A)$ is necessarily canonical, as stated under Definition 1.42.

Like definable orderings, canonisation of $\mathcal{C}$ within $\mathcal{L} \geq \mathsf{FP}$ implies that $\mathcal{L}$ captures[9] P on $\mathcal{C}$ [Ott17]. This follows from the same line of reasoning as sketched in the case of definable orderings.

It is in this context that Grohe proved in [Gro17] that FPC captures P on any class of structures that excludes a minor. This is a very general result, encompassing, by instance, the class of planar graphs, or the class of graphs embeddable on a torus. The techniques used to show that FPC canonises those structures are deeply linked to the computational problems of Graph Canonisation and Graph Isomorphism.

First, it is well-known [Hod93, Definition 5.5.1] that FO defines an isomorphism-preserving, bijective interpretation from Σ-structures to simple graphs, for any signature Σ. As such (relying once again on Proposition 1.38), we can reduce the search of a canonisation interpretation for any signature Σ to the search of a graph canonisation. At that point, the canonisation of graphs in the sense of Definition 1.43 is precisely the logical equivalent of the graph canonisation problem (hereafter GC):

Definition 1.44. A *graph canonisation function* is a function $\mathsf{can} : \{0,1\}^* \to \{0,1\}^*$ such that:

- For any $w \in \{0,1\}^*$ encoding a graph $\mathfrak{G}$, $\mathsf{can}(w)$ is also an encoding of $\mathfrak{G}$
- For any $w, w' \in \{0,1\}^*$ two encodings of one graph $\mathfrak{G}$, $\mathsf{can}(w) = \mathsf{can}(w')$.

Whether any such function is computable in polynomial-time is unknown. This problem is known to be itself tightly linked to the *Graph Isomorphism problem*:

Graph Isomorphism problem	
Input:	w, w', the encodings of two graphs G, G'
Question:	Do $G \cong G'$?

It is not known either whether the Graph Isomorphism problem (thereafter GI) is in P. Obviously, computing a graph canonisation is at least as hard as GI: if we can compute such a canonisation function can, it is only left, on input w, w', to check whether $\mathsf{can}(w) = \mathsf{can}(w')$. While no reduction from graph canonisation to GI is known, (almost) all advances w.r.t. the complexity of GI have translated to a similar complexity result for GC.

As mentioned earlier, Grohe's capture result is tightly connected to Graph Isomorphism in that it relies on the definition, within FPC, of a well-known algorithm for Graph Isomorphism and Graph Canonisation: the *Weisfeler-Leman algorithm*.

[9] Obviously, we still expect $\mathcal{L}$ to have polynomial-time model checking.

This algorithm, in its simplest form, amounts to iteratively refine a colouring of the vertices of the graph(s) at hand. Initially, we colour each vertex according to its number of neighbours. At the next step, we can now refine this colouring, by considering not only the size of each neighbourhood, but their distribution amongst the colours obtained in the previous step. This can be iterated until the colouring reaches a fix-point (in $O(n)$ iterations), and the resulting colouring is a canonical pre-ordering of the vertices of the graph.

If two graphs do not yield the same pre-ordering, we know they are not isomorphic. However, there are many non-isomorphic graphs which yield the same pre-ordering (regular graphs for instance).

This algorithm can be extended to colour k-tuples of vertices, incurring a multiplicative cost (a fixed-point is now reached in $O(n^k)$ steps), in exchange for a finer distinguishing power. This generalised algorithm is called *k-dimensional Weisfeler-Leman*, and denoted k-WL.

There is a strong connection between FPC and k-WL in that FPC can canonise a class of structures $\mathcal{K}$ if and only if there is some k such that k-WL distinguishes structures (or rather, encodings of those structures as graphs) in $\mathcal{K}$ [Ott17]

Whether for some k, k-WL defines a complete isomorphism test remained open until the introduction of the CFI-construction in [CFI92], which showed that there are pairs of non-isomorphic graphs which cannot be distinguished by k-WL with a constant dimension k. As such, it seems that, if we are to find logics capturing P on larger classes of structures, we need to bring new isomorphism-testing techniques within their reach. We aim to show now that, in that prospect, Computational Group Theory is a promising candidate.

1.2 Graph Isomorphism and Computational Group Theory

The connection between the Graph Isomorphism problem and group theory have been studied for a long time. It can be traced at least to [Bab79], and led to results out of reach of k-WL as soon as 1983, with Luks' isomorphism test for graphs of bounded degree [Luk82]. Quite recently, they led to Babai's groundbreaking quasipolynomial-time (i.e. $O(n^{\log^{O(1)} n})$) algorithm for Graph Isomorphism on all graphs. However, those two results involve highly-complex results in finite group theory, and remain outside of the scope of this thesis. However, they are based on a set of group-theoretic techniques that have seldom been brought within the scope of Descriptive Complexity and Finite Model theory. As such, our aim in this section is to introduce those group theoretic notions as well as their use in [Bab79] to solve Graph Isomorphism on Bounded colour-class graphs.

Computational Group Theory

Abstract groups, subgroups and morphisms

We begin this section with basic definitions concerning groups:

Definition 1.45.

- A *group* is a couple $(G, \cdot)$ where G is a set and $\cdot$ an operation $G \times G \to G$ that is associative, admits a neutral element, and such that each element admits an inverse:

$$\forall x, y, z, (x \cdot y) \cdot z = x \cdot (y \cdot z)$$
$$\exists e, \begin{cases} \forall x, x \cdot e = e \cdot x = x \\ \forall x, \exists x^{-1}, x \cdot x^{-1} = x^{-1} \cdot x = e \end{cases}$$

- Such a group $(G, \cdot)$ is *finite* if its underlying set is.
- The *order* of a group $(G, \cdot)$ is the cardinality of G, and is denoted $|G|$.
- A *subgroup* of G is a group H whose underlying set is a subset of G, and whose operation agrees with that of G. This relation between H and G is denoted $H \leq G$.
- Given $x \in G$ a group, the *order* of x is the order of the smallest subgroup of G which contains x.
- Given two groups $(G, \cdot_G), (H, \cdot_H)$, a morphism from G to H is a function $f : G \to H$ that respects the group operation, that is,

$$f(x \cdot_G y) = f(x) \cdot_H f(y)$$

- A group is *Abelian* if $\cdot$ is commutative, that is

$$\forall x, y, x \cdot y = y \cdot x$$

When unambiguous, we omit the group operation, and refer to $(G, \cdot)$ as G.

Example 1.46.

- $(\mathbb{Z}, +) \leq (\mathbb{Q}, +) \leq (\mathbb{R}, +)$ are groups.
- $(\mathbb{Q} \setminus \{0\}, \times) \leq (\mathbb{R} \setminus \{0\}, \times)$ are groups
- For any field $\mathbb{K}$, $(\mathrm{GL}_n(\mathbb{K}), \times)$, the set of all $n \times n$ invertible matrices over $\mathbb{K}$ is a group.

- For any n, the set $[n]$ together with addition modulo n is a group. We denote it $\mathbb{Z}_n$.
- We denote 1 the *trivial group* with 1 element.

The axioms of groups have very strong implications on the structural relationship between a group and its subgroups. An elementary illustration of this is Lagrange's theorem:

Theorem 1.47 (Lagrange)**.** *For any finite group G and $H \leq G$, $|H|$ divides $|G|$.*

This fact relies on the fact that G can be partitioned into cosets:

Definition 1.48. Let G be a group.

- For $X, Y \subseteq G$, we denote $X \cdot Y$, or even XY the set $\{x \cdot y \mid x \in X, y \in Y\}$. If X is of the form $\{x\}$, XY is denoted xY.
- Given $H \leq G$, a *left coset* (resp. *right coset*) of H is G is any set of the form gH (resp. Hg) for some $g \in G$.

Proposition 1.49. *If G is a group and $H \leq G$, the left cosets of H in G form a partition of G, or equivalently, for any $g, g' \in G$, $gH = g'H$ or $gH \cap g'H = \emptyset$. The same holds for right cosets.*

The number of such cosets of H in G is called the *index* of H in G, and denoted $|G : H|$. From there, it is not difficult to obtain

$$|G : H| = \frac{|G|}{|H|} \tag{1.2}$$

Definition 1.50. Let G be a group and $H \leq G$. A left (resp. right) *transversal* for H in G is a set $S \subseteq G$ such that, for any $g \in G$, $|S \cap gG| = 1$ (resp. $|S \cap Gg| = 1$)

Note that, if S is a left transversal of H in G, then $G = \bigcup_{s \in S} sH$.

Definition 1.51. Let $H \leq G$. H is a *normal subgroup* of G, denoted $H \trianglelefteq G$, if for all $g, g' \in G$, $gHg' = H$.

When $H \trianglelefteq G$, $(gH, g'H) \mapsto gg'H$ defines a group operation on the cosets of H in G. The resulting group is the *quotient group* G/H, and its neutral element is H.

We now turn to morphisms:

Definition 1.52. Let G, H be two groups, and $m : G \to H$ a group morphism.

- $\ker(m) := \{g \in G, m(g) = e_H\}$ is the *kernel* of m. It is a normal subgroup of G.
- $\mathrm{im}(m) := \{h \in H, \exists g \in G, m(g) = h\}$ is the *image* of m. It is a subgroup of H.

When $H \trianglelefteq G$, there is a canonical surjective morphism from G to G/H, whose kernel is H:

$$\begin{aligned} \rho_H : G &\to G/H \\ g &\mapsto gH \end{aligned}$$

This enables the statement of the following theorem, which is pervasive in group theory:

Theorem 1.53 (*First Isomorphism Theorem*)**.** *Let G, H be two groups and $m : G \to H$. The group $G/\ker(m)$ is isomorphic to $\mathrm{im}(m)$.*

Let us mention briefly a theorem that will be used in Chapter 2:

Theorem 1.54 (*Cauchy*)**.** *Let G be a finite group, and p a prime number. If p divides $|G|$, then G contains an element of order p*

Permutation groups and their encoding

Before we can delve into group problems, and the first results of Computational Group theory, we must discuss how groups are given as input.

It might seem natural to represent a group as a model of the theory given in Definition 1.45, and thus encoding it in a manner comparable to Definition 1.10. We refer to this representation of groups as the *Cayley table representation*.

The Cayley table representation does not take advantage of the intricate structure that the groups axiom induce. For any group G, this naive representation has size $O(|G|^3)$ (if one sees $\cdot$ as a ternary relation) or $O(|G|^2 \log |G|)$ (if one sees $\cdot$ as a binary function, and encodes each value the function takes in binary), and it happens that most groups admit a drastically more compact representation as permutation groups, as we will now show.

Definition 1.55. Fix a set Ω.

- A *permutation* of Ω is a bijection $\sigma : \Omega \to \Omega$.
- The set of all permutations of Ω, equipped with functions composition, is a group, the *symmetric group* on Ω, denoted $\mathrm{Sym}(\Omega)$.
- A *permutation group* on Ω is any subgroup $G \leq \mathrm{Sym}(\Omega)$.
- For G any group, an *action* of G on Ω is a morphism $m : G \to \mathrm{Sym}(\Omega)$. This action is *faithful* if $\ker(m) = 1$.

When G is a permutation group over Ω, g an element of G and $X \subseteq \Omega$, we denote X^g the set of all images of elements of X through g

$$X^g := \{g(x), x \in X\}$$

Let us add that, when $\Omega = [n]$, its symmetric group is denoted S_n. In the case of S_n, we often use a compact presentation of permutations, by decomposing them into disjoint cycles:

Definition 1.56. Let σ be a permutation in S_n. We write

$$\sigma = (c_1^1\ c_2^1 \dots c_{k_1}^1)(c_1^2\ c_2^2 \dots c_{k_2}^2) \dots (c_1^\lambda\ c_2^\lambda \dots c_{k_\lambda}^\lambda)$$

- for any $i \leq \lambda$, $j \leq k_i$, $c_j^i \in [n]$
- for any $i \leq \lambda$, $j < k_i$, $\sigma(c_j^i) = c_{j+1}^i$
- for any $i \leq \lambda$, $\sigma(c_{k_i}^i) = c_1^i$
- for any $c \notin \bigcup_{i \leq \lambda} \{c_1^i, \dots, c_{k_i}^i\}$, $\sigma(c) = c$

Definition 1.57. Fix a set Ω, a group $G \leq \mathrm{Sym}(\Omega)$, and $B_1, \dots, B_k \subseteq \Omega$.

- The *point-wise stabiliser* of $(B_1, \dots, B_k)$ is the subgroup

$$\mathrm{Stab}_G(B_1, \dots, B_k) := \{g \in G, \forall \lambda \leq k, B_\lambda^g = B_\lambda\}$$

- The *set-wise stabiliser* of $(B_1, \dots, B_k)$ is the subgroup

$$\mathrm{Stab}_G\{B_1, \dots, B_k\} := \{g \in G, \forall \lambda \leq k, \exists \mu, B_\lambda^g = B_\mu\}$$

- When each B_i contains exactly one element ω_i, we omit the braces, and denote the point-wise (resp. set-wise) stabiliser $\mathrm{Stab}_G(\omega_1, \dots, \omega_k) := \mathrm{Stab}_G(\{\omega_1\}, \dots, \{\omega_k\})$.
- If $G = \mathrm{Stab}_G(B_1, \dots, B_k)$ (resp. $G = \mathrm{Stab}_G\{B_1, \dots, B_k\}$), G is said to *stabilise* $(B_1, \dots, B_k)$ point-wise (resp. set-wise).
- G is *transitive* if the only partitions of Ω that G stabilises point-wise are the trivial partitions (Ω) and $(\{\omega\})_{\omega \in \Omega}$.
- G is *primitive* if the only partitions of Ω that G stabilises set-wise are the trivial partitions.

Obviously, a permutation group is a group, by the properties of bijections and functions composition. More interestingly, all groups can be seen as permutation groups:

Theorem 1.58 (*Cayley*)**.** *Any group G is isomorphic to a permutation group on G.*

We now state a well-known fact about abelian transitive groups, that will play a part in Chapter 4:

Proposition 1.59. *Let G be an abelian group acting faithfully and transitively on a set X. Then, the action of G on X is* regular*, i.e. for any $x \in X$, $\mathrm{Stab}_G(\{x\}) = \{1\}$.*

Proof. Consider $g \in G$ such that $g \cdot x = x$. We will show that $g = 1_G$. Because the action of G on X is faithful, this amounts to show that, for any y, $g \cdot y = y$. Consider such a $y \in X$, and by transitivity, let $h \in G$ such that $h \cdot x = y$. Then,

$$\begin{aligned} g \cdot y &= (hh^{-1}g) \cdot y \\ \textit{Because } G \textit{ is abelian,} \quad &= (hgh^{-1}) \cdot y \\ &= (hg) \cdot x \\ &= h \cdot x \\ &= y \end{aligned}$$

□

Notice that any permutation in S_n can be represented using $n \log n$ bits. Thus, we could represent any group $G \leq S_n$ by a list of the encodings of all elements of G, which yields a representation of the group of length $|G| \cdot n \cdot \log n$. Compared to the Cayley table representation, we no longer need to encode the product of elements, as this is given implicitely by the composition of functions. Moreover, the dependence of the size of the encoding on $|G|$ can still be reduced, because not *all* elements of G must be provided to define G uniquely;

Definition 1.60. Let G be a group, and $S \subseteq G$. The *subgroup of G generated by S* is the smallest subgroup $\langle S \rangle \leq G$ such that $S \subseteq \langle S \rangle$. If $\langle S \rangle = G$, S is a *generating set* of G.

Theorem 1.61 (Laplace)**.** *Any group G admits a generating set of size $O(\log |G|)$.*

Therefore, any group $G \leq S_n$ can be encoded by a set of $\log |G|$ permutations, each of which can be encoded using $n \log n$ bits, for a total of $O(\log |G| n \log n)$ bits.

Let us now compare this to the Cayley table encoding. Even in the worst case, the permutation encoding is shorter than the Cayley table encoding, while staying comparable:

Example 1.62. According to Cayley's theorem, $\mathbb{Z}_n$ is isomorphic to a permutation group on its underlying set $[n]$:

$$\begin{aligned} \varphi : \mathbb{Z}_n &\to S_n \\ k &\mapsto (1\ 2 \dots n)^k \end{aligned}$$

However, $\mathbb{Z}_n$ cannot act faithfully on any set of size $< n$. Moreover, $\mathbb{Z}_n$ is generated by $\{1\}$, and therefore $\varphi(\mathbb{Z}_n)$ is generated by $\{\varphi(1)\} = \{(1\ 2 \dots n)\}$. As such, $\mathbb{Z}_n$ can be encoded as a permutation group using $O(n \log n)$ bits, which is to be contrasted with the $O(n^2 \log n)$ bits needed for the Cayley table representation.

In other cases, the permutation representation incurs a drastic reduction of the size of the group:

Example 1.63. S_n is a group of order $n!$. As such, its Cayley table representation has length $O((n!)^2 \log(n!)) = O((n!)^2 n \log n)$ bits.

On the other hand, it is well known that S_n is generated by all *transpositions* over $[n]$, that is, all permutations of the form $(i\ j)$ for $i, j \in [n]$. This yields a permutation representation of S_n of length $O(n^3 \log n)$.

It seems that various variables impact the length of this new encoding :the size of the group, the size of a small generating set for this group, or the domain of the permutations. Let us remark that any group $G \leq \mathrm{Sym}(\Omega)$ admits a representation of length $\ell \leq |\Omega|^4$. As such, the size of the domain of the permutations is, of itself, a suitable metric for the size of the group, when focused on polynomial-time computations.

Elementary computations over permutation groups: the Schreier-Sims algorithm

However, this more efficient encoding of groups has a cost, and new computational problems emerge that did not necessarily have significance in the table representation.

As a matter of fact, one of the most natural such problems is the *membership problem*: fix a domain Ω ; given $S \subseteq \mathrm{Sym}(\Omega)$ and $\sigma \in \mathrm{Sym}(\Omega)$, does it hold that $\sigma \in \langle S \rangle$? Another such problem is the *order problem*: given $S \subseteq \mathrm{Sym}(\Omega)$, compute $\mathsf{ord}\langle S \rangle$.

We now recall the *Schreier-Sims algorithm* introduced in [Sim70; Sim71] to answer these problems (although the fact that this algorithm runs in polynomial time[10] is due to Furst, Hopcroft and Luks [FHL80]). It relies on the fundamental structure of strong generating sets, and two subprocedures: a sifting procedure, which allows one to compute membership tests given a strong generating set, and a construction procedure, which allows the construction of a strong generating set. We will present those objects in that order.

Strong generating sets The notion of strong generating set allows to structure the set of generators of the group at hand. This structure relies on a chain of subgroups for the group. We begin the introduction of strong generating sets with an auxiliary definition.

[10] That is, in time $p(|S| + |\Omega|)$

Definition 1.64. Let $H \leq G \leq \mathrm{Sym}(\Omega)$. A *conditional membership test* for H w.r.t. G is a Turing machine M that takes as input the encoding of a permutation $\sigma \in \mathrm{Sym}(\Omega)$, such that, under the condition that $\sigma \in G$, M accepts if and only if $\sigma \in H$.

Definition 1.65. Let G be a permutation group over a domain Ω, and

$$G = H_0 \geq \cdots \geq H_k = 1$$

a decreasing chain of subgroups, together with *conditional membership tests* for H_{i+1} w.r.t. H_i for all i. A *strong generating set* (hereafter *SGS*) for G along (H_i) is a collection $(\sigma_{i,j})_{i \leq k, j \leq c_i}$ of permutations in G such that, for each i, the permutations $\sigma_{i,0}, \sigma_{i,1} \ldots \sigma_{i,c_i}$ form a left transversal of H_{i+1} in H_i, for each i.

It is easily shown that, for any $i \leq k$, $\{\sigma_{i',j} \mid i' > i, j \leq c_{i'}\}$ generates H_{i+1}. As such, each $\sigma_{i,j}$ can be seen as *encoding* a coset of H_{i+1} in H_i. When clear from context, we omit the chain of subgroups a strong generating set is constructed upon. The main insight about strong generating sets is that they allow for a unique decomposition of elements in G:

Lemma 1.66. *Let G be a group and $(\sigma_{i,j})$ be a SGS for G. Then, for any $g \in G$, there is a* unique *sequence $(j_i) \in \prod_{i=1}^{k} [c_i]$ such that $g = \sigma_{1,j_1}\sigma_{2,j_2} \ldots \sigma_{k,j_k}$.*

A direct application of this lemma is that, given a SGS for G, we can compute the order of G in polynomial time: it is exactly $\prod_{i=1}^{k} c_i$.

Also note that we can assume the trivial coset $1H_{i+1}$ to be represented first, and be represented by the identity. That is, we can assume that any strong generating set $(\sigma_{i,j})$ is such that for all i, $\sigma_{i,0} = 1$.

Sifting and membership test Suppose that $G \leq \mathrm{Sym}(\Omega)$ is a group, (H_i) is a decreasing chain of subgroups with $H_0 = G$ and $H_m = 1$. A SGS $(\sigma_{i,j})$ on G along (H_i) allows us to reduce the membership test on G to iterated membership tests on the groups H_i. Let $\tau \in Sym(\Omega)$ be the permutation we wish to test membership for. By induction, we aim define a sequence of permutations ρ_i such that $\rho_i \in H_i \iff \tau \in G$. The initialisation is easy: set $\rho_0 := \tau$. Now, suppose the property holds for ρ_i. For each $\sigma_{i,j}$ in the SGS, we use the conditional membership test on $\sigma_{i,j}^{-1}\rho_i$. If there is a unique j such that the membership test succeeds, then we set $\rho_{i+1} := \sigma_{i,j}^{-1}\rho_i$, and the properties of SGS and conditional membership tests ensure that $\rho_{i+1} \in H_{i+1} \iff \tau \in G$. If no such j exists (or if several such j exist), then $\rho_i \notin H_i$, and we therefore interrupt the procedure, rejecting the input τ. For reasons to become clear in the following, in such a case, we also output the last value of i, and of ρ_i upon rejection. This algorithm is called the *sifting procedure*, and is depicted in Algorithm 1.

The time-complexity of this algorithm depends on three factors: the length k of the chain (H_i); the size, for any given i, of the coset transversal $(\sigma_{i,0}, \sigma_{i,1} \ldots)$ (which is exactly $|H_{i+1} : H_i| = c_i$), and the time-complexity of the conditional membership tests. If we assume those values to be bounded by $p(|\Omega|)$ for a fixed polynomial p, Algorithm 1 clearly runs in polynomial-time. This motivates the following definition:

Algorithme 1 : Sifting Procedure

Input : $(\sigma_{i,j})$, a SGS for G along $(H_i)_{i=0}^k$, a subgroup chain with conditional membership tests $(P_i)_{i=1}^k$; and $\tau \in \mathrm{Sym}(\Omega)$

Result : Does $\tau \in G$?

```
1  Function sift(g, λ) is
       // Invariant :  g ∈ H_λ ⟺ τ ∈ G
2      if λ = k then
3          if g = Id then
4              return true
5          else
6              return false outputting (g, λ)
7      else
8          if ∃!j s.t. P_{λ+1}(σ_{λ,j}^{-1} · g) then
9              return sift((σ_{λ,j}^{-1} · g), λ + 1)
10         else
11             return false outputting (g, λ)
12 return sift(τ, 0)
```

Definition 1.67. A subgroup chain $H_0 \geq H_1 \geq \cdots \geq H_k$ is called *d-adequate* for G if:

- $H_0 = G$ and $H_k = 1$
- $k \leq |\Omega|^d$
- for all $i < k$, $|H_i : H_{i+1}| \leq |\Omega|^d$
- Given $\sigma \in H_i$, we can check whether $\sigma \in H_{i+1}$ in time $O(|\Omega|^d)$.

Constructing a SGS in polynomial time We now show how to construct a strong generating set for G, given a regular generating set S, and an adequate subgroup chain (H_i). This second algorithm constitutes the second part of the Schreier-Sims framework, introduced in [Sim70; Sim71].

In Algorithm 1, a couple (σ, λ) is outputted upon rejection on input τ. Intuitively, this couple locates a missing element from the SGS for τ to be accepted by the sifting procedure. We call σ the *residual* of τ through the SGS. This is the gist of the construction procedure: we iteratively add the elements from S to a strong generating set structure for the adequate subgroup chain H_i, starting with the family consisting solely of $\sigma_{i,0} := 1$ for each i. [11] However, while one can easily check that the resulting underlying set of permutations generates G, it is not necessarily the case that this corresponds to the permutations accepted by the sifting procedure. This is because so

[11] Note that, now that we consider an adequate subgroup chain for G, we can encode this family as an array of permutation of dimension $|\Omega|^d \times |\Omega|^d$ for some d

far, no care has been given to the *saturation* of the strong generating set. To overcome this issue, each time an element σ is inserted in the SGS, we must sift all elements of the form $\sigma\sigma_{i,j}$ or $\sigma_{i,j}\sigma$ and insert their residuals, if any. This is the *construction procedure* as presented in Algorithm 2.

Algorithme 2 : Constructing a SGS

Input : S, a generating set for G, $(H_i)_{i\leq k}$, an adequate subgroup chain for G, with conditional membership tests $(P_i)_{i=1}^k$.

Output : $(\sigma_{i,j})_{i<k}$, a SGS for (H_i)

1 Initialize an array $(\sigma_{i,j})$ with empty cells, except $\sigma_{i,0} := \mathrm{Id}$ for all $i \leq k$
2 $\ell := S$
3 **while** $\ell \neq \emptyset$ **do**
4 Let τ be the first element of ℓ and remove it from ℓ
5 **if** $\mathrm{sift}(\tau, 0)$ *rejects on* $(\sigma_{i,j})$, *outputting* (g, i) **then**
6 $j := \min\{\lambda, \sigma_{i,\lambda} = \emptyset\}$
7 $\sigma_{i,j} := g$
8 $\ell := \{g \cdot \sigma_{a,b}, a \leq i, \sigma_{a,b} \neq \emptyset\} \oplus \{\sigma_{a,b} \cdot g, a \geq i, \sigma_{a,b} \neq \emptyset\} \oplus \ell$
9 **return** $(\sigma_{i,j})$

In line 8, we add new elements to process in order to ensure that our SGS is composition-closed (here, $\oplus$ denotes the concatenation of lists).

Let us note that, (H_i) being d-adequate for some d, at most $|\Omega|^{2d}$ permutations will be added to the array. Each such insertion incurs at most $|\Omega|^{2d}$ elements to sift, for a total of $|\Omega|^{4d} + |S|$ iterations of the while loop.

Therefore, under the condition that G admits an adequate subgroup chain, we can, given a set S such that $\langle S\rangle = G$, compute a SGS for G using Algorithm 2, and then check membership to G using Algorithm 1, or compute the size of G using Lemma 1.66.

A uniform construction of an adequate subgroup chain It happens that, for any group $G \leq \mathrm{Sym}(\Omega)$, there is an adequate subgroup chain that can be used in Algorithms 1 and 2. Fix a linear-ordering $<$ of Ω, and let $\omega_1, \ldots, \omega_n$ be the corresponding enumeration of Ω, that is,

$$\omega_1 < \omega_2 < \cdots < \omega_n$$

Then, consider the chain $(H_i)_{i=0}^n$ where $H_i := \mathrm{Stab}_G(\omega_1, \ldots, \omega_i)$ (and $H_0 := G$). If $\sigma \in H_i$, checking whether $\sigma \in H_{i+1}$ amounts to check that $\sigma(\omega_{i+1}) = \omega_{i+1}$, which requires linear time[12]. Moreover, the length of the chain is bounded by $|\Omega|$, so that it is only left to bound its *width* to show the adequacy of this chain. That is, we must show that $|H_i : H_{i+1}|$ is polynomially bounded by $|\Omega|$. For $\sigma, \tau \in H_i$,

$$\begin{aligned} \sigma H_{i+1} = \tau H_{i+1} &\iff \sigma^{-1}\tau \in H_{i+1} \\ &\iff \sigma(\omega_{i+1}) = \tau(\omega_{i+1}) \end{aligned}$$

[12]This depends on our computation model and the encoding of permutations, but we will not delve into such considerations here.

As such, the index of H_{i+1} in H_i is bounded by $|\Omega|$ for all i, and the chain (H_i) is 1-adequate.

Remark that the definition of this adequate subgroup chain relies critically on a linear-ordering on Ω, while the Schreier-Sims algorithm answers a question that does not depend on such an ordering. This observation is the starting point for the definition of the ord-operator in the next chapter. Notice also that we have a somehow weaker ordering assumption in Algorithm 2: iterating over S requires that an ordering of S is provided. This assumption is weaker in the sense that it may happen that a canonical adequate subgroup tower is available, and that the set S is ordered, without yielding an ordering of the domain Ω. Such a situation constitutes the setting of our results in Chapter 5, and to a lesser extent, of Chapter 4.

Subgroup computations We conclude this subsection with a final observation on the Schreier-Sims framework: in addition to membership tests and order computation, it also allows to find generators for subgroups of G in some cases.

Suppose $\langle S\rangle = G \leq \mathrm{Sym}(\Omega)$, and $H \leq G$ is a subgroup such that:

- Given $g \in G$, we can check whether $g \in H$ in polynomial time (in $|\Omega|$)
- $|G : H|$ is polynomially-bounded.

Under those assumptions, we can find a generating set for H in the following way: we use Algorithm 2 on the chain of subgroups

$$G \geq H \geq \mathrm{Stab}_H(\omega_1) \geq \cdots \geq \mathrm{Stab}_H(\omega_1, \ldots, \omega_{n-1}) = 1$$

This results in a SGS $(\sigma_{i,j})_{i=0}^{n}$ for G, and it is not difficult to see that the subfamily $(\sigma_{i,j})_{i=1}^{n}$ is a SGS for H.

We now formalise this result with a convenient iteration:

Definition 1.68. Let $G \leq \mathrm{Sym}(\Omega)$ and $H \leq G$. H is *k-accessible* from G if there exists a decreasing sequence of subgroups

$$G = H_0 \geq H_1 \geq \cdots \geq H_m = H$$

such that the chain of subgroups

$$H_0 \geq H_1 \cdots \geq H_m = H \geq \mathrm{Stab}_H(\omega_1) \ldots \mathrm{Stab}_H(\omega_1, \ldots, \omega_{n-1}) = 1$$

is k-adequate for G. If $m = 1$, we say that H is *directly k-accessible* from G.

Proposition 1.69. *If $G \leq \mathrm{Sym}(\Omega)$ and $H \leq G$ is k-accessible from G, one can compute in polynomial-time a SGS for H when given as input a generating set for G.*

The Graph Automorphism problem

Having introduced the Schreier-Sims framework, we are now ready to exhibit the strong connection between Group computations and the Graph Isomorphism problem.

First and foremost, the *automorphisms* of any givens structure form a group:

Definition 1.70. Given any structure $\mathfrak{A}$, its *automorphism group* is the group of all isomorphisms from $\mathfrak{A}$ onto itself, with composition as operation. It is denoted $\mathrm{Aut}(\mathfrak{A})$.

Intuitively, $\mathrm{Aut}(\mathfrak{A})$ is the set of all symmetries of $\mathfrak{A}$.

Example 1.71. Consider

$$\mathfrak{G} = (\{a, b, c, d\}, \{\{a, b\}, \{a, d\}, \{b, c\}, \{c, d\}\})$$

That is, $\mathfrak{G}$ is an undirected cycle of length 4. It is easy to see that $\mathrm{Aut}(\mathfrak{G})$ is generated by $(a\ b\ c\ d)$ and $(a\ b)(c\ d)$.

$$\begin{array}{ccc} a & — & b \\ | & & | \\ c & — & d \end{array}$$

Consider a square $S \subseteq \mathbb{R}^2$. The set of all isometries of $\mathbb{R}^2$ that stabilise S set-wise form a group G. The corners of S form a set C of four elements of $\mathbb{R}^2$, and G acts naturally (and faithfully) on C. The action of G on C is isomorphic to $\mathrm{Aut}(\mathfrak{G})$. $\mathrm{Aut}(\mathfrak{G})$ is the *dihedral group* of order 8.

Now, a fundamental observation is that checking GI between two graphs X, Y reduces to finding generatings sets for $\mathrm{Aut}(X), \mathrm{Aut}(Y)$, and $\mathrm{Aut}(X \sqcup Y)$. We now formalise this.

The *Graph Automorphism problem* (or *GA*) entails to find a generating set for the automorphism group of a graph given as input:

Graph Automorphism Problem	
Input:	$\mathfrak{G}$, a graph with domain Ω
Output:	A set $S \subseteq \mathrm{Sym}(\Omega)$ such that $\langle S \rangle = \mathrm{Aut}(\mathfrak{G})$

Definition 1.72. If $\mathfrak{A}, \mathfrak{B}$ are two σ-structures, the *union* of $\mathfrak{A}$ and $\mathfrak{B}$, denoted $\mathfrak{A} \sqcup B$ is the structure with domain $A \sqcup B$, and for any $R \in \sigma$, $R^{\mathfrak{A} \sqcup \mathfrak{B}} := R^{\mathfrak{A}} \sqcup R^{\mathfrak{B}}$.

Let $\mathcal{C}$ be any class of structures. We denote by $\mathrm{GI}_{\mathcal{C}}$ the Graph Isomorphism problem restricted to input graphs within $\mathcal{C}$, and do similarly for $\mathrm{GA}_{\mathcal{C}}$.

While the proof of the following property is quite straight-forward, [Bab79] is, to our knowledge, the first reference to give it some computational significance.

Proposition 1.73. *Suppose that $\mathcal{C}$ is union-closed. There is a (Turing) polynomial-time reduction from $GI_{\mathcal{C}}$ to $GA_{\mathcal{C}}$.*

Proof. On input X, Y two graphs in $\mathcal{C}$, we use our oracle to $\mathrm{GA}_{\mathcal{C}}$ to compute R, S, T, three sets of permutations that generate $\mathrm{Aut}(X), \mathrm{Aut}(Y)$ and $\mathrm{Aut}(X \sqcup Y)$ respectively. Using the Schreier-Sims algorithm, we can compute the orders of those three groups. It happens that

$$X \cong Y \iff |\mathrm{Aut}(X \sqcup Y)| = 2 \cdot |\mathrm{Aut}(X)| \cdot |\mathrm{Aut}(Y)|$$

which can be checked in polynomial-time. □

Great effort has been put into solving GA, leading to Babai's breakthrough [Bab16] that GA and GI can be solved in quasi-polynomial time ($O(n^{\log^{O(1)} n})$) on the class of all graphs.

As a side-note, let us remark that GA reduces to the general problem of finding a generating set for the intersection of two groups.

Group Intersection	
Input:	S, T two sets of permutations over a domain Ω
Output:	$R \subseteq \mathrm{Sym}(\Omega)$ such that $\langle R \rangle = \langle S \rangle \cap \langle T \rangle$

Indeed, if $\mathfrak{G}$ is a directed graph with domain Ω, consider the two following groups $G_1, G_2 \leq \mathrm{Sym}(\Omega^2)$:

- G_1 is the group defined by the natural action of $\mathrm{Sym}(\Omega)$ on Ω^2, i.e. for any $\sigma \in \mathrm{Sym}(\Omega)$, G_1 contains the permutation of Ω^2 mapping (i, j) to $(\sigma(i), \sigma(j))$.
- G_2 is the group of permutation of edges $\mathrm{Sym}(E^{\mathfrak{G}})$.

Consider S any generating set of $\mathrm{Sym}(\Omega)$ (a small such set exists, as already mentioned in Example 1.63). Then, G_1 is generated by the set $f(S)$, where

$$\begin{aligned} f : \mathrm{Sym}(\Omega) &\to \mathrm{Sym}(\Omega^2) \\ \sigma &\mapsto ((i, j) \mapsto (\sigma(i), \sigma(j))) \end{aligned}$$

At the same time, G_2 is generated by $\{((a, b)\ (c, d)), (a, b), (c, d) \in E^{\mathfrak{G}}\}$. Moreover, $f^{-1}(G_1 \cap G_2)$ is exactly $\mathrm{Aut}(\mathfrak{G})$.

While this last remark does not play a part in the results presented in this thesis, it does show that, in the general case (opposed to the restricted case discussed in Proposition 1.69), finding generators for subgroups of a given group is hard (precisely, GI-hard), even when a membership test for the subgroup is present.

A worked example: BCGI

We now present as an example the class of structures studied in [Bab79], and show how the Schreier-Sims framework enables the decision on the isomorphism problem on that class.

Definition 1.74. A *coloured graph* is a graph $X = (V, E)$ equipped with a *colouring* $c : V \to \mathbb{N}$. Each integer in $c(V)$ is a *colour*, and a *colour-class* is a set of vertices of the form $c^{-1}(i)$ for some colour i.

An isomorphism between two coloured graphs $X = (V, E, c)$ and $X' = (V', E', c')$ is a bijection $f : V \to V'$ such that $f(E) = E'$, and for any two vertices $x, y \in V$, $c(x) \leq c(y) \iff c'(f(x)) \leq c'(f(y))$. Automorphisms of coloured graphs are defined accordingly.

X has *k-bounded colour-classes* if, for any colour i, the set of vertices of colour i, $c^{-1}(i)$ has at most k elements. Let *k-BCG* denote the class of all k-bounded colour-classes graphs.

We denote *k-BCGI* (resp. *k-BCGA*) the problem $\mathrm{GI}_{k\text{-BCG}}$ (resp. $\mathrm{GA}_{k\text{-BCG}}$). We will now present Babai's result [Bab79] that the Schreier-Sims algorithm can be used to solve k-BCGA. The careful reader will notice that, in this context, Proposition 1.73 cannot be used directly to conclude that k-BCGI is in P. However, the proof of Proposition 1.73 can still be applied, and yields a reduction from k-BCGI to $2k$-BCGA. Therefore, proving that for any k, k-BCGA is polynomial-time computable yields k-BCGI $\in \mathsf{P}$.

Fix a coloured graph $X = (V, E, c)$, and let m be the maximum colour of X. In the context of bounded colour-graphs, the colouring greatly restrict the space of potential automorphisms. Indeed, any such automorphism must be within the group

$$G := \prod_{i=1}^{m} \mathrm{Sym}(c^{-1}(i)) \tag{1.3}$$

The key observation is that there is a decreasing sequence $(G_i)_{i \leq m^2}$ of subgroups of G, that witnesses the fact that $\mathrm{Aut}(X)$ is 2-accessible from G. Namely, having fixed a bijection s between $[m^2]$ and pairs of colours, we set $G_{\lambda+1}$ to be the set of all permutations in G_λ that stabilise (set-wise) the edges between the two colours in $s(\lambda)$. That is, if $s(\lambda) = \{i, j\}$,

$$G_{\lambda+1} := \mathrm{Stab}_{G_\lambda}\{E \cap (V_i \times V_j)\} \tag{1.4}$$

where $V_i := c^{-1}(i)$. The fact that conditional membership for $G_{\lambda+1}$ in G_λ is decidable in polynomial time is straight-forward, and it is only left to show that $|G_\lambda : G_{\lambda+1}|$ is polynomially bounded. Consider $\sigma, \tau \in G_\lambda$. Obviously, if σ and τ agree on $V_i \cup V_j$, they belong to the same $G_{\lambda+1}$ coset. Yet, there are only $|V_i|!|V_j|! \leq k!^2$ permutations of $V_i \cup V_j$ (that stabilise both V_i and V_j). Therefore, $|G_\lambda : G_{\lambda+1}| \leq k!^2 = O(1)$. Since the chain of subgroup is of length m^2 and m is proportional to n, we obtain that $\mathrm{Aut}(X)$ is 2-accessible from G. Finally, G is generated by all transpositions of two elements in a single colour class. Therefore, we can apply Proposition 1.73.

This example will play an important role in Chapter 4. In the logical context, a coloring on a structure $\mathfrak{A}$ is given by a total preorder $\preceq$ over A. As providing a definition of a pre-order as a binary relation over A can be quite notation-heavy, we will often abuse notation, and define a pre-order by *the partition and the linear-order over that partition it induces*, that is, if $(\preceq) \subseteq \mathcal{X}^2$ is a linear-ordering of $\mathcal{X}$ which is a partition of A, we view $\preceq$ as a pre-order over A.

Chapter 2

Groups in Fixed-point logics

We have depicted the link between descriptive complexity and the Graph Isomorphism problem, as well as the Permutation group framework to Graph Isomorphism. These connections are a motivation to use groups, and in particular permutation groups more extensively in the quest of a logic for P.

To our knowledge, the main use of groups in this context has been to obtain lower bounds, or inexpressibility results. Intuitively, the symmetric nature of the primitive operations within reach of the logic at hand greatly restricts the auxiliary structures that can be defined from the input, as they have to respect its automorphism group. This line of reasoning led to the proof that CPT does not express all P function problems [Ros10]. The case of decision problems remains open.

Our aim in this thesis is to exhibit and study a way to *extend* the expressive power of a logic using a group-theoretic operator. This is in line with the study of $\mathsf{FP} + \mathsf{rk}$, initiated in [Daw+09]: having identified an order-invariant algebraic operation whose computation seems to require a linear-order, we add this operation as a new primitive of the logic at hand. Note that this reasoning also applies to the counting operator. This motivates us to restrict our study to Fixed-point logics, in the continuity of rank logic.

We first review the rk operator, and how it defines an unordered setting to the study of linear maps. Then, we introduce two different such settings for the study of groups: Cayley tables, and sets of permutations. Finally, we introduce the ord operator to allow FP to handle the latter representation of groups.

2.1 The unordered setting of rank logic

Let us first review the definition of the rk operator. We do so for two reasons: it illustrates the kind of unordered setting we want to introduce for group computations ; and it provides an example of the use of generalised quantifiers as a mean to define algebraic extensions of FO (and FP). Note that we will use the extended definition of generalized quantifiers introduced in Definition 1.39.

Fix a field $\mathbb{K}$ and consider two $\mathbb{K}$-vector spaces U, V. The rank of a linear map

$f : U \to V$ is invariant under automorphisms of both U and V. In particular, given bases $(u_i)_{i\in I}, (v_j)_{j\in J}$ of U and V respectively, the rank of f is invariant under the automorphisms of U generated by any permutation of $\{u_i, i \in I\}$ and the automorphisms of V generated by any permutation of $\{v_j, j \in J\}$.[1] In the computational case, U can be identified with $\mathbb{K}^{\dim U}$, V with $\mathbb{K}^{\dim V}$ and f can then be encoded by a matrix $M_f \in \mathbb{K}^{\dim U \times \dim V}$ over the canonical bases of $\mathbb{K}^{\dim U}$ and $\mathbb{K}^{\dim V}$ (the linear-order on $\mathbb{N}$ then allowing us to apply Gaussian elimination to obtain the rank of the underlying linear map).

The rk operator definition in [Daw+09] is based on an unordered setting to the study of linear maps: while $\mathbb{K}$ remains linearly-ordered and therefore lies in the numerical domain, the vector spaces U, V do not. More precisely, we consider $U = \mathbb{K}^I$ and $V = \mathbb{K}^J$, with I and J unordered. In this context, for any $\vec{u}$ in U (resp. V) and i in I (resp. J), we denote $\vec{u}(i)$ the value of $\vec{u}$ at the coordinate indexed by i. A linear map $f : U \to V$ can then be encoded as a structure with domain $D := I \sqcup J$ and a numerical function M such that

$$M(x,y) := \begin{cases} f(\vec{c_x})(y) & \text{if } (x,y) \in I \times J \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{c_x} \in \mathbb{K}^I$ is the x-th vector in the canonical basis of U, that is,

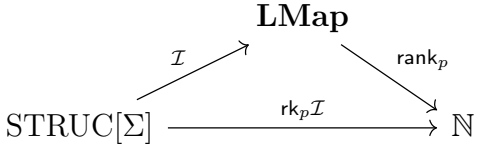
$$\vec{c_x}(i) := \begin{cases} 1 & \text{if } x = i \\ 0 & \text{otherwise} \end{cases}$$

Let **LMap** denote the class of such structures encoding linear maps, that is, the class of $\langle I, J, M\rangle$-structures, where I, J are monadic relations, and M a binary numerical function that fulfil the following axiom :

$$\forall x, y, (I(x) \vee J(y)) \implies M(y,x) = 0$$

For $\mathfrak{A} \in \mathbf{LMap}$, and p a prime, the residuals modulo p of $M^{\mathfrak{A}}$ define a linear map $f_p(\mathfrak{A}) : \mathbb{F}_p^{I(\mathfrak{A})} \to \mathbb{F}_p^{J(\mathfrak{A})}$ in the way described above. Now, we have a function $\mathsf{rank}_p : \mathbf{LMap} \to \mathbb{N}$ which associates to $\mathfrak{A}$ the rank of the linear-map $f_p^{\mathfrak{A}}$. Clearly, if $\mathfrak{A} \simeq \mathfrak{B}$, $\mathsf{rank}_p(\mathfrak{A}) = \mathsf{rank}_p(\mathfrak{B})$, and $\mathsf{rank}_p(\mathfrak{A})$ is polynomially-bounded by $|\mathfrak{A}|$, and is computable in polynomial-time

Therefore, if such a structure $\mathfrak{A}$ was definable *within* a Σ-structure $\mathfrak{B}$, it would be reasonable to define a logic $\mathcal{L}$ in which one could define, in $\mathfrak{B}$, the numeric value $\mathsf{rank}_p(\mathfrak{A})$. This is the gist of the rk operator, where the inner definability of $\mathfrak{A}$ is witnessed by a $[\Sigma, \{I, J, M\}]$-interpretation. Note that $\mathcal{I}$ could have parameters.



[1] Note that this is not the case of the determinant, whose value depends on the choice of a basis for both U and V.

However, the syntax of interpretations being quite cumbersome, the rk operator only takes as input the formula φ_M, as we can circumvent the use of φ_{dom} (since, instead of removing the x-th row and column from a matrix, setting all the values on those row and column leaves the rank unchanged) and $\varphi_{\simeq}$ (since, instead of identifying the x-th and y-th rows and columns, we can assign the same values to them, leaving once again the rank unchanged). Only two pieces of information must be conveyed: the domains I, J, and the matrix encoded by φ_M. By specifying which free variables of φ_M pertain to the interpretation, and using different variables for I and J separated by a comma, we obtain the syntax of the rk operator from Definition 1.33. In that sense, the rk operator is a (family of) generalised quantifier(s), as defined in Definition 1.39. Whether allowing such a hybrid index set of I, J — with an ordered and an unordered part — increases the expressibility of the operator is still, to our knowledge, an open question.

Note that this point of view adapts well to to the uniform setting of the rank operator: let $\mathbf{LMap}^*$ be the class of all structures in $\mathbf{LMap}$ equipped with an additional numerical constant p, which evaluates to a prime number. For $\mathfrak{A} \in \mathbf{LMap}^*$, we define $\mathsf{rank}^*(\mathfrak{A})$ as $\mathsf{rank}_{p^{\mathfrak{A}}}(\mathfrak{A}_{\restriction I,J,M})$. From this definition, we can define rk^* in the same way as we defined rk_p from rank_p.

Our aim in this chapter is to introduce such an unordered setting to encode groups in first-order syntax, and we will start by finding a satisfying group-theoretic equivalent to $\mathbf{LMap}$. Each of the representations of groups introduced in Section 1.2 yields such a class of structures. As expected, only the class of structures corresponding to the permutation group representation is a satisfying setting in which to study the logical use of the tools of Computational Group theory introduced in Section 1.2. For the sake of completeness, we nevertheless begin our study with Groups encoded as Cayley tables.

2.2 Encoding Groups as Cayley tables

As we have seen in Section 1.2, groups in their formal presentation form a first-order theory. As such, it is quite natural that this representation of groups translates seamlessly to the context of Finite Model Theory: a finite group is a $\{\cdot, e\}$-structure $\mathfrak{A}$ — where $(\cdot)$ is a binary function symbol (that we will use in infix notation) and e a constant symbol — such that $(\cdot)^{\mathfrak{A}}$ fulfills the group axioms and $e^{\mathfrak{A}}$ is the neutral element of that group.

As we have mentioned before, it is often convenient in Finite Model Theory to consider only relational structures[2]. This can be easily achieved by replacing the binary operation $(\cdot)$ by a ternary relation T, such that $T(x, y, z)$ holds iff $x \cdot y = z$. We denote this class of finite relational structures $\mathbf{CGrp}$, as this corresponds to encoding groups by their Cayley tables with unordered rows and columns.

Lemma 2.1. CGrp *is a* FO*-definable class of* $\{T, e\}$*-structures*

[2]Let us remind the reader that relational signatures may contain constant symbols

Proof. Consider the following first-order formulae:

$$\begin{aligned}
\varphi_{\text{op}} &:= \forall x, y, \exists! z, T(x, y, z) \\
\varphi_{\text{assoc}} &:= \forall x, y, z, \exists s_1, s_2, s_3, (T(x, y, s_1) \wedge T(s_1, z, s_3)) \wedge (T(y, z, s_2) \wedge T(x, s_2, s_3)) \\
\varphi_{\text{neutral}} &:= \forall x, T(x, e, x) \wedge T(e, x, x) \\
\varphi_{\text{inverse}} &:= \forall x, \exists y, T(x, y, e) \wedge T(y, x, e)
\end{aligned}$$

Given a $\{T, e\}$-structure $\mathfrak{A} = \langle A, T^{\mathfrak{A}} \rangle$, one can easily see that $\mathfrak{A} \models \varphi_{\text{op}}$ if and only if $T^{\mathfrak{A}}$ is the graph of a function $A \times A \to A$. Let us now suppose indeed that $\mathfrak{A} \models \varphi_{\text{op}}$, and denote this function by $(\cdot)$, using the infix notation. Then:

- $\mathfrak{A} \models \varphi_{\text{assoc}}$ iff $(\cdot)$ is an associative operation.
- $\mathfrak{A} \models \varphi_{\text{neutral}}$ iff $e^{\mathfrak{A}}$ is a neutral element w.r.t. $(\cdot)$.
- Therefore, $\mathfrak{A} \models \varphi_{\text{inverse}}$ iff each element x admits an inverse w.r.t. $(\cdot)$.

Thus, $\mathfrak{A} \models \varphi_{\text{op}} \wedge \varphi_{\text{assoc}} \wedge \varphi_{\text{neutral}} \wedge \varphi_{\text{inverse}}$ if and only if it belongs to **CGrp**. □

As we have seen in Section 1.2, the Cayley table representation of groups is not the most concise one we could use, and this undermines the relevance of expressivity results on **CGrp** — in the same way that we are not usually interested in measuring the complexity of arithmetic problems w.r.t. the unary encoding of integers. Still, there are some facts worth mentioning concerning the expressive power of FO and its extensions on **CGrp**.

Lemma 2.2. *Over* **CGrp***,* Even *is* FO*-definable*

Proof. By Cauchy's theorem, a group G has even order if and only if it has an element of order two, which is an FO-definable property:

$$\varphi_{even} := \exists y, e \neq y \wedge T(y, y, e)$$

□

This contrasts with the fact that even FP does not express Even over the class of all structures (see Section 1.1).

In [Bar+01; BM91], Barrington et al. study the computational complexity of one particular problem over groups encoded as Cayley tables:

Cayley Group Membership (CGM)	
Input:	A group G given by its Cayley table, a set of elements $X \subseteq G$ and $g \in G$
Question:	Does $g \in \langle X \rangle$?

They remark that CGM is reducible to USTCON, which implies that CGM $\in \mathsf{L}$, using Reingold's result that $\mathsf{USTCON} \in \mathsf{L}$ [Rei08]. We show that this reduction still works in the unordered case:

Lemma 2.3. *There is an* FO $\langle\{T, e, X, g\}, \{E, s, t\}\rangle$*-interpretation* I *such that, given a group* $(G, T, e) \in \mathbf{CGrp}$*,* $X \subseteq G$ *and* $g \in G$*,* $\mathcal{I}(G, T, e, X, g)$ *is a graph* (V, E, s, t) *such that* s *is connected to* t *if and only if* $g \in \langle X \rangle$

Proof. We build $\mathcal{I}(G, T, e, X, g)$ to be the graph whose vertices are elements of G, with edges corresponding to the action of elements in X. Setting s as e and t as g yields the desired result : a path in this graph from e to g is exactly a sequence $x_1, \ldots, x_n$ of elements of X such that $x_n \ldots x_2 x_1 = g$. We define $\mathcal{I}$ as the unary interpretation $(\varphi_{\mathrm{dom}}(x), \varphi_{\simeq}(x, y), \varphi_E(x, y), \varphi_s(x), \varphi_t(x))$, where:

$$\begin{aligned}
\varphi_{\mathrm{dom}}(x) &:= \top \\
\varphi_{\simeq}(x, y) &:= x = y \\
\varphi_E(x, y) &:= \exists z, X(z) \wedge T(x, z, y) \\
\varphi_s(x) &:= (x = e) \\
\varphi_t(x) &:= (x = g)
\end{aligned}$$

□

Corollary 2.4. CGM *is definable in* $\mathsf{FO} + \mathsf{stc}$ *and* $\mathsf{FO} + \mathsf{tc}$

However, Reingold's algorithm is not definable in the unordered setting, as witnessed by [GM95]. Thus, whether $\mathsf{FO} + \mathsf{dtc}$ expresses CGM remains unknown.

Example 2.5. Consider $\mathfrak{A} = (V, E, \leq)$ a k-bounded colour-class graph (where the colouring is encoded by the pre-order relation $\leq$). For any i, the i-th colour has an induced automorphism group

$$\Gamma_i := \{\sigma \in \mathrm{Sym}(V_i), \forall s, t \in V_i, E(s, t) \iff E(\sigma(s), \sigma(t))\}$$

(where V_i is the set of vertices of colour i).

FPC defines a $2k$-dimensional $[\{E, \leq\}, \{T, e\}]$-interpretation $\mathcal{I}$ with parameter i

such that $\mathcal{I}(\mathfrak{A}, i) \in \mathbf{CGrp}$ is isomorphic to Γ_i: consider $\mathcal{I} = (\varphi_{\mathrm{dom}}, \varphi_{\simeq}, \varphi_T, \varphi_e)$ where:

$$\varphi_{\mathrm{dom}}(\vec{x}\vec{y}, i) := \begin{cases} \bigwedge_{\lambda=1}^{k} V_i(x_\lambda) \wedge V_i(y_\lambda) \\ \circledwedge \forall z, V_i(z) \implies \left(\bigvee_{\lambda=1}^{k} x_\lambda = z\right) \wedge \left(\bigvee_{\lambda=1}^{k} y_\lambda = z\right) \\ \bigwedge_{\lambda=1}^{k} \bigwedge_{\mu=1}^{k} E(x_\lambda, x_\mu) \iff E(y_\lambda, y_\mu) \end{cases}$$

$$\varphi_{\simeq}(\vec{x}\vec{y}, \vec{x'}\vec{y'}, i) := \bigwedge_{\lambda=1}^{k} \bigwedge_{\mu=1}^{k} x_\lambda = x'_\mu \implies y_\lambda = y'_\mu$$

$$\varphi_T(\vec{x}\vec{y}, \vec{x'}\vec{y'}, \vec{x''}\vec{y''}, i) := \bigwedge_{\lambda=1}^{k} \bigwedge_{\mu=1}^{k} x_\lambda = x''_\mu \implies \bigvee_{\nu=1}^{k} x'_\nu = y_\lambda \wedge y''_\mu = y'_\nu$$

$$\varphi_e(\vec{x}\vec{y}, i) := \bigwedge_{\lambda=1}^{k} x_\lambda = y_\lambda$$

where $V_i(x)$ is a predicate stating that x belongs to the i-th colour-class. This can be defined in FPC as:

$$V_i(x) := \mathrm{ifp}_{R,y,\mu} \left(\begin{cases} \forall z, y \leq z \wedge \mu = 0 \\ \circledvee \exists z, \nu, \circledwedge \begin{cases} R(z, \nu) \\ \mu = \nu + 1 \\ z < y \\ \forall w, \neg(z < w \wedge w < y) \end{cases} \end{cases} \right)(x, i) \tag{2.1}$$

$\varphi_{dom}(\vec{x}\vec{y}, i)$ states that $x_\lambda \mapsto y_\lambda$ constitutes an automorphism of V_i, $\varphi_{\simeq}(\vec{x}\vec{y}, \vec{x'}\vec{y'}, i)$ that $x_\lambda \mapsto y_\lambda$ and $x'_\lambda \mapsto y'_\lambda$ define the same permutations. φ_T then defines the composition of such permutations, and φ_e holds on $\vec{x}\vec{y}$ iff $x_\lambda \mapsto y_\lambda$ is the identity permutation.

However, the automorphism group construction algorithm presented in Section 1.2 requires as input a generating set for $\prod_{i=1}^{m} \Gamma_i$. Such a group has, in the worst case, order $(k!)^{n/k}$, which is super-polynomial. As such, we cannot hope to construct an interpretation which builds this entire group in **CGrp**. As expected, **CGrp** does not seem sufficient to capture the class of group computations described in Section 1.2.

2.3 Encoding permutation groups

We now depart from the study of **CGrp**, and aim to define a class of structures encoding permutation groups, in the manner exhibited at the beginning of the chapter for linear maps.

We wish to avoid representing permutations as functions, as we lack a proper way to define functions within the formalism of first-order logic (and its extensions). Instead, we will represent a function as its graph:

Definition 2.6. Let X, Y be finite sets, and $f : X \to Y$. The *graph* of f is the relation

$$\text{graph}(f) := \{(a, b) \in X \times Y \mid f(a) = b\}$$

If $R \subseteq X \times Y$ is the graph of a function, this function is uniquely determined by R, and we define $\text{func}(R)$ as the unique (surjective) f such that $\text{graph}(f) = R$.

Because permutations play an important role in this thesis, in the case where R is the graph of a permutation, we denote $\text{func}(R)$ as $\text{perm}(R)$.

Definition 2.7. Let $\mathcal{L}$ be a logic extending FO, $\mathfrak{A}$ be a Σ-structure, t_1, t_2 two types, and $f : X \to A^{t_2}$ with $X \subseteq A^{t_1}$. We say that f is *definable* in $\mathcal{L}$ if there is a $\mathcal{L}$-formula $\varphi(\vec{p}, \vec{\alpha}, \vec{\beta})$ with $\text{type}(\vec{\alpha}) = t_1$, $\text{type}(\vec{\beta}) = t_2$, such that, for some $\vec{a} \in A^{\text{type}(\vec{p})}$,

$$\text{graph}(f) = \varphi(\mathfrak{A}, \vec{a})$$

For $X \subseteq \text{Sym}(D)$, we can encode (D, X) as a structure $\mathfrak{A}$ whose domain is $D \sqcup X$, that is, both the points on which any $\sigma \in X$ acts, and the permutations that are part of X. The action of X over D is given by a ternary relation $R^{\mathfrak{A}} \subseteq X \times D \times D$, defined as follows:

$$R^{\mathfrak{A}} := \bigcup_{\sigma \in X} \{\sigma\} \times \text{graph}(\sigma)$$

that is, $(\sigma, x, y) \in R^{\mathfrak{A}}$ iff $\sigma x = y$. We also add X as a unary relation to distinguish points from permutations. Let us denote **PGrp** the class of such $\{X, R\}$ structures.

Lemma 2.8. PGrp *is a* FO*-definable class of* $\{X, R\}$*-structures*

Proof. First, we ensure that R is *well-typed*:

$$\varphi_{\text{type}} := \forall x, y, z, R(x, y, z) \implies X(x) \wedge \neg X(y) \wedge \neg X(z)$$

Now, we make sure that, the action of X on D described by A is indeed a set of permutations:

$$\varphi_{\text{action}} := \forall x, X(x) \implies \bigwedge \begin{cases} \forall y, \neg X(y) \implies \exists! z, R(x, y, z) \\ \forall z, \neg X(z) \implies \exists! y, R(x, y, z) \end{cases}$$

$\{\varphi_{\text{type}}, \varphi_{\text{action}}\}$ constitutes an axiomatisation of **PGrp**. □

We introduce a lighter notation to define structures in **PGrp**: given a set D and $X \subseteq \text{Sym}(D)$, we denote $\mathbf{G}(D, X)$ the structure

$$(D \sqcup X, X, \{(x, d_1, d_2) \mid xd_1 = d_2\})$$

Example 2.9. Consider $\mathfrak{A}_n = \mathbf{G}([n], X)$ where $X = \{(i\ j), i, j \leq n\}$. $\langle X \rangle = S_n$ has order $n!$, and $|\mathfrak{A}_n| = O(n^2)$.

Lemma 2.10. *There is an* FO*-interpretation from* **CGrp** *to* **PGrp** *that maps any group G to a permutation group isomorphic to G.*

Proof.

$$\begin{aligned}
\varphi_{\mathrm{dom}}(x,b) &:= b = 0 \vee b = 1 \\
\varphi_{\simeq}(x,b,y,b') &:= x = y \wedge b = b' \\
\varphi_X(x,b) &:= b = 1 \\
\varphi_R(x,b,y,b',z,b'') &:= \textcircled{\wedge} \begin{cases} b = 1 \wedge b' = 0 \wedge b'' = 0 \\ T(x,y,z) \end{cases}
\end{aligned}$$

$\mathcal{I} = (\varphi_{\mathrm{dom}}, \varphi_{\simeq}, \varphi_X, \varphi_R)$ defines the action of G over itself by left multiplication. By Cayley's theorem, the underlying permutation group is isomorphic to G. □

Together with Example 2.9, Lemma 2.10 implies that, if we consider any finite structure $\mathfrak{A}$, strictly more groups can be defined by interpretation into **PGrp** than **CGrp**, be it only for the fact that groups of super-polynomial order can only be built within **PGrp**. As a partial converse result, we have the following:

Lemma 2.11. *There are* FO*-formula $\varphi_\times(x,y,z)$ and $\varphi_{-1}(x,y)$ such that, for any $\mathfrak{A} = \mathbf{G}(D,X) \in$* **PGrp** *and $\sigma, \tau, \rho \in X$,*

- $(\mathfrak{A}, \sigma, \tau, \rho) \models \varphi_\times$ *iff* $\sigma \cdot \tau = \rho$
- $(\mathfrak{A}, \sigma, \tau) \models \varphi_{-1}$ *iff* $\sigma^{-1} = \rho$.

Proof.

$$\begin{aligned}
\varphi_\times(x,y,z) &:= \textcircled{\wedge} \begin{cases} X(x) \wedge X(y) \wedge X(z) \\ \forall s,t,\exists u, R(z,s,t) \implies (R(y,s,u) \wedge R(x,u,t)) \end{cases} \\
\varphi_{-1}(x,y) &:= \textcircled{\wedge} \begin{cases} X(x) \wedge X(y) \\ \forall s,t, R(x,s,t) \implies R(y,t,s) \end{cases}
\end{aligned}$$

□

However, while the first-order theory of groups is quite natural, it is not quite clear which properties of those groups can be defined in FO or FP. As a corollary of our main result in Chapter 4 we will see that the expressive power of FPC over **PGrp** is quite limited. However, there are two fundamental properties of permutation groups that are definable even in FO + tc: transitivity and, to a lesser extent, primitivity.

Lemma 2.12. *There is a* FO + tc *formula $\varphi(x,y)$ such that, for any $\mathfrak{A} = \mathbf{G}(D,X) \in$* **PGrp**, *$a, b \in D$, $(\mathfrak{A}, a, b) \models \varphi$ if and only if a and b belong to the same $\langle X \rangle$-orbit.*

Proof.

$$\varphi(x,y) := (\mathsf{tc}_{u,v}\exists g, R(g,u,v))(x,y) \qquad \square$$

The proof of the following statement is a direct translation to the logical framework of [Luk82, Lemma 1.3]

Lemma 2.13. *There is a* $\mathsf{FO}+\mathsf{tc}$ *sentence* ψ *such that, for any* $\mathfrak{A} = \mathbf{G}(D,X) \in \mathbf{PGrp}$ *with* $\langle X\rangle$ *transitive,* $\mathfrak{A} \models \psi$ *if and only if* $\langle X\rangle$ *is a primitive group.*

Proof. Consider the following formula:

$$\theta(u,v,x,y) := \exists g, (R(g,u,x) \wedge R(g,v,y)) \vee (R(g,u,y) \wedge R(g,v,x))$$

For any $a,b \in A$, $(\mathsf{tc}_{x,y}\theta)(\mathfrak{A},a,b)$ defines the graph with vertex-set D, and edges $\{\{a,b\}\}^{\langle X\rangle} = \{\{a^\sigma, b^\sigma\} \mid \sigma \in \langle X\rangle\}$. As remarked in [Luk82] (where the author cites [Sim67]) the connected component of a in this graph is the smallest set $X \subseteq \Omega$ such that $\{a,b\} \subseteq X$ and X is stabilised set-wise by $\langle X\rangle$. Therefore, the group is imprimitive if and only if, for some couple $(a,b) \in D^2$, this set is not equal to D. As such, the following sentence has the intended behaviour:

$$\psi := \forall u,v,w, (\mathsf{tc}_{x,y}\theta(u,v,x,y))(u,w)) \qquad \square$$

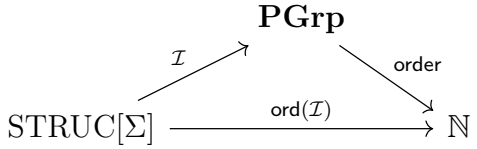
In [Luk82], this technique is used to compute a *maximal* block of imprimitivity of the group at hand. This seems to remain out of reach of any logic considered in this thesis (including $\mathsf{FP}+\mathsf{ord}$ which will be introduced in the next subsection, and also CPT). Indeed, as there might be different maximal blocks of imprimitivity for a given $\mathfrak{A} \in \mathbf{PGrp}$, it is quite unclear how we could enforce the definition of one such block without choice. Yet, the computation of such a block is a very important step in Luks's algorithm for graphs of bounded degree, and this limitation is the first obstacle towards the translation of this method to a choiceless framework.

We conclude this section with a small remark on this translation of the permutation groups framework to logic. Our definition of **PGrp** introduces two different levels of "unorderedness". For a given structure $\mathbf{G}(D,X)$, a linear-order of D yields a canonical (and FP-definable) total linear-order over X. However, the converse does not necessarily hold for X, and we can construct cases where X is ordered without yielding a canonical ordering of D: to get an intuition of this, consider the case where $D = \{x_1,\dots,x_n\}$ and $X = \{(x_1 \dots x_n)\}$. $x_i \mapsto x_{i+1}$ is a non-trivial automorphism of $\mathbf{G}(D,X)$, which renders the definability of an ordering of D, the domain of $\mathbf{G}(D,X)$, impossible (without parameters).

We denote $\mathbf{PGrp}^<$ the class of $\{X,R,<\}$-structures whose restriction to $\{X,R\}$ is in **PGrp**, and such that $<$ is a strict linear order on X.

2.4 The ord operator

Consider the function order : **PGrp** $\to \mathbb{N}$, which maps $\mathfrak{A} = (D \cup X, X, R)$ to $|\langle X \rangle|$. This function is obviously isomorphism-invariant, and we have introduced in Section 1.2 a polynomial-time algorithm that computes order. We aim to introduce an operator able to define order in the syntax of FO, in the same way that rk enables the definition of rank, that is, given a $[\Sigma, \{X, A\}]$-interpretation $\mathcal{I}$, ord($\mathcal{I}$) should make the following diagram commute:



The first obstacle is that order($\mathfrak{A}$) is not polynomially-bounded by $|\mathfrak{A}|$, as seen in Example 2.9.

However, this example is *maximal* in the sense that, for any structure $\mathfrak{A} = \mathbf{G}(D, X)$, order($\mathfrak{A}$) $\leq |D|!$; and $\log(|D|!) \leq |D| \log |D| \leq |D|^2$. As such, order($\mathfrak{A}$) can be represented as a binary numeric predicate over D (that is, a subset of $(D^<)^2$). Therefore, while we can't expect the ord operator to output a numeric value, it could output a numerical term: if $\mathcal{I}$ is a k-dimensional $[\Sigma, \{X, R\}]$-interpretation, and $\mathfrak{B}$ is a Σ-structure such that $\mathcal{I}(\mathfrak{B}) = \mathbf{G}(D, X) \in \mathbf{PGrp}$, $|D| \leq |\mathfrak{B}|^k$, and

$$\log_2(\mathsf{order}(\mathcal{I}(\mathfrak{B}))) \leq \log_2(|\mathfrak{B}|^k!) \leq |\mathfrak{B}|^k \log_2(|\mathfrak{B}|^k) \leq |\mathfrak{B}|^{2k}$$

Therefore order($\mathcal{I}(\mathfrak{B})$) can be encoded in binary as a $2k$-ary numerical predicate. We could thus introduce, as in our discussion on the rk-operator at the beginning of this chapter, a family of operators $\{\mathsf{order}, n \in \mathbb{N}\}$, such that order takes as input a k-dimensional interpretation, and outputs a $2k$-ary numeric predicate such that order($\mathcal{I}(\mathfrak{A})$)($\vec{m}$) holds if the $\vec{m}$-th bit of order($\mathcal{I}(\mathfrak{A})$) is a 1, that is:

$$(\mathfrak{A}, \vec{m}) \models \mathsf{order}(\mathcal{I}) \iff \mathrm{bit}\left(\mathsf{order}(\mathcal{I}(\mathfrak{A})), \sum_{i=0}^{2k} m_i |\mathfrak{A}|^i\right) = 1 \tag{2.2}$$

While this definition of ord would be satisfying from an expressivity perspective, it is quite impractical. It is very heavy in notation as it requires us to define a full-blown interpretation; and it also doesn't provide a distinction between points and permutations, which will often be useful as the number of points usually provide a better measure of the size of the group at hand. Since we will use the ord operator extensively in the following chapters, a simplification of the operator is therefore in order. Unlike the rk operator, it seems that disallowing quotienting (that is, cases where the formula $\varphi_\sim(\vec{x}, \vec{y})$ is not equivalent to $\vec{x} = \vec{y}$) could decrease the expressiveness of the operator. However, our expressiveness results on the ord operator do not make use of quotienting, and as such, we will restrict ourselves to cases where $\varphi_\sim(\vec{x}, \vec{y}) \equiv \vec{x} = \vec{y}$. In this subset of interpretations, we can introduce a much simpler syntax, quite close to the syntax of the rk operator:

Definition 2.14. For $\vec{p}, \vec{s}, \vec{t}$ four tuples of variables such that type($\vec{s}$) = type($\vec{t}$), $\vec{\mu}$ a tuple of $2|\vec{s}|$ numerical variables, and a formula $\varphi(\vec{p}, \vec{s}, \vec{t})$ over some signature Σ,

$\psi := (\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\ \varphi)(\vec{\mu})$ is a formula with $\mathrm{free}(\psi) = \{\vec{\mu}\} \cup (\mathrm{free}(\varphi) \setminus \{\vec{p}, \vec{s}, \vec{t}\})$. Given a Σ-structure $\mathfrak{A}$ and $\vec{a} \in A^{\mathrm{type}(\vec{r})}$, $(\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\varphi)(\mathfrak{A}, \vec{a}) = \mathsf{order}(\mathcal{I})$ in the sense of Eq. (2.2), where $\mathcal{I}$ is a $[\Sigma, \{X, R\}]$-interpretation of type $T = (\mathrm{number} \cdot \mathrm{type}(\vec{p}) \cdot \mathrm{type}(\vec{s}))$ provided by the formulae:

$$\begin{aligned}
\varphi_{dom}(\vec{x}) &:= b_x = 0 \vee b_x = 1 \\
\varphi_{\simeq}(\vec{x}, \vec{y}) &:= \vec{x} = \vec{y} \\
\varphi_X(\vec{x}) &:= b_x = 1 \\
\varphi_R(\vec{x}, \vec{y}, \vec{z}) &:= \bigwedge \begin{cases} b_x = 1 \wedge b_y = 0 \wedge b_z = 0 \\ \varphi(\vec{p}_x, \vec{e}_y, \vec{e}_z) \\ \vec{p}_y = \vec{p}_z \end{cases}
\end{aligned}$$

where, to ease readability, we have split any tuple $\vec{x}$ of type T into $(b_x, \vec{p}_x, \vec{e}_x)$, where $\mathrm{type}(\vec{p}_x) = \mathrm{type}(\vec{p})$ and $\mathrm{type}(\vec{e}_x) = \mathrm{type}(\vec{s})$.

In the definition of φ_R, we require $\vec{p}_y = \vec{p}_z$ so that for any value $\vec{a}$ of $\vec{x}$, $\varphi_R(\mathfrak{A}, \vec{a})$ defines the graph of a permutation (as such, for any $\vec{y}$, there must be a *unique* $\vec{z}$ such that φ_R holds). That is, we expect φ to define a family of permutations over A^t (where t is the type of $\vec{s}$), this family being indexed over $A^{t'}$ (where t' is the type of $\vec{p}$); in which case $\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\varphi$ is a numerical relation encoding the integer

$$\left|\left\langle \left\{ \sigma \mid \exists \vec{a} \in A^{t'}, \mathrm{graph}(\sigma) = \varphi(\mathfrak{A}, \vec{a}) \right\} \right\rangle\right|$$

Note that this definition uses the numerical domain (in the variables b_x, b_y, b_z) to distinguish between points and permutations. This could have been circumvented using extra variables and equality types, although this would have reduced readability. We now introduce some additional notations that will ease reading in the following chapters.

Definition 2.15. Given a formula $\varphi(\vec{p}, \vec{s}, \vec{t})$ where $\mathrm{type}(\vec{s}) = \mathrm{type}(\vec{t})$, a structure $\mathfrak{A}$ and $\vec{a} \in A^{\mathrm{type}(\vec{p})}$, we say that φ *defines a permutation* $\sigma \in \mathrm{Sym}(A^{\mathrm{type}(\vec{s})})$ on $(\mathfrak{A}, \vec{a})$ if $\varphi(\mathfrak{A}, \vec{a}) = \mathrm{graph}(\sigma)$.

We denote by $\langle\varphi\rangle_{\vec{p}.\vec{s}\vec{t}}(\mathfrak{A})$ the group generated by the set of permutations defined by φ in $\mathfrak{A}$ for some assignment of $\vec{p}$. When clear from context, we omit the $\vec{s}\vec{t}$ part of the subscript when using this notation.

With these notations, $(\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\varphi)(\mathfrak{A})$ is a numerical predicate encoding $|\langle\varphi\rangle_{\vec{p}}(\mathfrak{A})|$.

In the following chapters, we will often introduce formulae such as $\varphi(\vec{p}, \vec{s}, \vec{t})$, which define a permutation over $(\mathfrak{A}, \vec{a})$ for all values of $\vec{a} \in A^{\mathrm{type}(\vec{p})}$. In such a case, we call $\vec{p}$ the *enumeration parameters* of φ, and $\vec{s}, \vec{t}$ the *permutation variables* of φ. We usually provide a hint for the separation between the enumeration parameters and the two component of the permutation variables by omitting the commas within the three

tuples $\vec{p}$, $\vec{s}$ and $\vec{t}$. Thus, if $\vec{p} = (p_1, \ldots, p_l)$, $\vec{s} = (s_1, \ldots, s_k)$ and $\vec{t} = (t_1, \ldots, t_k)$, we would denote the definition of φ as

$$\varphi(p_1 p_2 \ldots p_l, s_1 s_2 \ldots s_k, t_1 t_2 \ldots t_k) := \ldots$$

Example 2.16. Consider, as in Example 2.5, a k-bounded colour-class graph $\mathfrak{A} = (V, E, \leq)$. We write $x \sim y$ for $x \leq y \wedge y \leq x$, and consider

$$\varphi(\vec{x}\vec{y}, s, t) := \begin{cases} \bigwedge_{\lambda=1}^{k} x_\lambda \sim y_\lambda \sim x_1 \\ \forall z, z \sim x_1 \implies (\bigvee_{\lambda=1}^{k} x_\lambda = z) \wedge (\bigvee_{\lambda=1}^{k} y_\lambda = z) \\ \bigwedge_{\lambda=1}^{k} \bigwedge_{\lambda=1}^{k} E(x_\lambda, x_\mu) \iff E(y_\lambda, y_\mu) \\ \begin{cases} \bigvee_{\lambda=1}^{k} s = x_\lambda \wedge t = y_\lambda \\ \left(\bigwedge_{\lambda=1}^{k} s \neq x_\lambda\right) \wedge s = t \end{cases} \end{cases}$$

This formula shares many similarities with the interpretation defined in Example 2.5: using a vector of length $2k$, we parameterise automorphisms of a single colour class. Here however, we do not describe the group of automorphisms by providing its multiplication table, but by specifying its action on the set of vertices. Moreover, this new formula is not restricted to an individual colour-class: in Example 2.5, $\mathcal{I}(\mathfrak{A}, i)$ was isomorphic to Γ_i, while here, $\langle\varphi\rangle_{\vec{x}\vec{y}}(\mathfrak{A}) = \prod_i \Gamma_i$. As such, $\mathsf{ord}_{\vec{x}\vec{y}.st}\varphi$ is a binary numeric predicate encoding (in the sense of Eq. (2.2)) $|\prod_i \Gamma_i| = \prod_i |\Gamma_i|$.

We are now ready to study the logic $\mathsf{FP} + \mathsf{ord}$ in depth in the following chapter.

Chapter 3

First study of FP + ord

Having introduced the ord operator, we now initiate the study of its expressivity. In the first section, we start using the ord operator to define basic group-related operations and problems within FP + ord. This section also serves as an example for the reader to get acquainted with the logic at hand in the remainder of this thesis.

In the second section, we will discuss the limits of the representation of groups by generating sets of permutations. In particular, we will show that isomorphism-invariance prevents the definition of a fundamental operation enabled in the computational context by Schreier-Sims framework: the computation of generating sets for accessible subgroups.

This limitation motivates the definition of another representation of groups, which we introduce in Section 3.3. This representation is a central tool in our results of Chapter 4.

Finally, we show that FP + ord is a satisfying new candidate logic for P in the strand of rank logic: first, we show that FP + ord has polynomial-time model-checking, and that FP + rk $\leq$ FP + ord.

3.1 A group-theoretic framework within FP + ord

In this section, we introduce some fundamental operations on group that can be defined within FP + ord. This introduction serves two purposes: first, it showcases the use of the ord operator, and gives first instances of FP + ord formulae ; but it also provides definition, within FP + ord, of operations pervasive to Computational Group Theory. As we expect to use Computational Group Theory in Chapter 4, in a manner quite similar to Section 1.2, it seems natural to first introduce basic operations that we will then be able to use seamlessly in the following chapters.

A first result is that, given a structure encoding a group $G \leq \mathrm{Sym}(A)$ and a permutation $\sigma \in \mathrm{Sym}(A)$, there is a FP + ord sentence that holds iff $\sigma \in G$. We start by defining this encoding, in a manner quite similar to our definition of **PGrp**:

Definition 3.1. A *membership problem instance structure* is a $\{X^{(1)}, R^{(3)}, \mathsf{target}^{(2)}\}$-structure $\mathfrak{A}$ such that:

- $D^{\mathfrak{A}} := A \setminus X^{\mathfrak{A}}$
- $R^{\mathfrak{A}} \subseteq X^{\mathfrak{A}} \times D^{\mathfrak{A}} \times D^{\mathfrak{A}}$
- For any $x \in X^{\mathfrak{A}}$, $\{(s,t) \in D^{\mathfrak{A}} \times D^{\mathfrak{A}}, (x,s,t) \in R^{\mathfrak{A}}\}$ defines the graph of a permutation τ_x.
- $\mathsf{target}^{\mathfrak{A}} \subseteq D^{\mathfrak{A}} \times D^{\mathfrak{A}}$ defines the graph of a permutation σ

We denote the class of all such structures **MembProb**. Such an instance $\mathfrak{A}$ is *positive* if $\sigma \in \langle \tau_x \mid x \in X^{\mathfrak{A}} \rangle$. We denote the class of positive instances $\mathbf{MembProb}^+$

Lemma 3.2. *There is an* $\mathsf{FP} + \mathsf{ord}$ *sentence on signature* $\{X, R, \mathsf{target}\}$ *that holds on a membership problem instance structure iff it is a positive instance.*

The proof of this lemma relies on the fact that

$$\sigma \in \langle S \rangle \iff |\langle S \rangle| = |\langle S \cup \sigma \rangle|$$

Therefore, the proof of Lemma 3.2 follows directly from the following:

Lemma 3.3. *Let* $\varphi(\vec{p}, \vec{s}, \vec{t}), \psi(\vec{q}, \vec{s}, \vec{t})$ *be two* $\mathcal{L}[\Sigma]$*-formulas, where* $\mathcal{L} \geq \mathsf{FP}$*. There is a* $\mathcal{L}$ *formula* $\eta(\vec{r}, \vec{s}, \vec{t})$ *such that, for any* Σ*-structure* $\mathfrak{A}$*,*

$$\langle \eta \rangle_{\vec{r}}(\mathfrak{A}) = \langle \langle \varphi \rangle_{\vec{p}}(\mathfrak{A}) \cup \langle \psi \rangle_{\vec{q}}(\mathfrak{A}) \rangle$$

Proof. Let $\vec{r} = (\vec{p}, \vec{q}, b)$, where b is a fresh numerical variable. We use $\vec{r}$ to range over both generating sets, and define η accordingly:

$$\eta(\vec{r}, \vec{s}, \vec{t}) := \begin{cases} b = 0 \wedge \varphi(\vec{p}, \vec{s}, \vec{t}) \\ b = 1 \wedge \psi(\vec{q}, \vec{s}, \vec{t}) \end{cases}$$

The result follows immediately. Note that if $\mathcal{L}$ does not implement the numerical sort, we could use two domain variables and equality types in place of variable b. □

Proof of Lemma 3.2. Considering target as a formula defininining a permutation over D with 0 parameters, applying Lemma 3.3 to R and target, we obtain

$$\eta(b, x, s, t) := \begin{cases} b = 0 \wedge R(x, s, t) \\ b = 1 \wedge \mathsf{target}(s, t) \end{cases}$$

Now, it is only left to compare the size of $\langle R \rangle_x(\mathfrak{A})$ and $\langle \eta \rangle_{b,x}(\mathfrak{A})$:

$$\mathsf{memb} := \forall \vec{\mu}, (\mathsf{ord}_{x,st} R(x,s,t))(\vec{\mu}) \iff (\mathsf{ord}_{xb,st} \eta)(\vec{\mu})$$

It follows from the observation below the statement of Lemma 3.2 that, when $\mathfrak{A} \in$ **MembProb**, $\mathfrak{A} \models \mathsf{memb}$ if and only if $\mathrm{perm}(\mathsf{target}^{\mathfrak{A}}) \in \langle R \rangle_x(\mathfrak{A})$, concluding the proof. □

This shows that the "abstract" membership problem for **PGrp** is within reach of FP+ord. However, we aim to use membership tests seamlessly within FP+ord formulae definitions. In general, we expect to have already defined Σ-formulae $\varphi(\vec{p}, \vec{s}, \vec{t})$ and $\psi(\vec{s}, \vec{t})$ such that, for any structure $\mathfrak{A}$, they define respectively a generating set $S^{\mathfrak{A}}$ for a group $G^{\mathfrak{A}}$, and a permutation $\mathrm{perm}(\psi^{\mathfrak{A}})$. Can we use φ and ψ as building blocks to define a sentence that holds on $\mathfrak{A}$ iff $\mathrm{perm}(\psi^{\mathfrak{A}}) \in \langle S^{\mathfrak{A}} \rangle$? Having shown Lemma 3.2, this problem does not bear any fundamental complexity. However, any solution should be a function that maps the *formulae* φ and ψ to a *sentence*, and this would be notationally heavy to define.

This is where our definition of definable quantifiers in Definition 1.40 shows useful: here, $\mathcal{K}$ is the class of positives instance in **MembProb**, and as we have already shown $\mathcal{K}$ to be defined, it is only left to show how such a couple of formulae (φ, ψ) can be seen as an interpretation from Σ-structures to **MembProb**:

Definition 3.4. Given Σ-formulae $\varphi(\vec{p}, \vec{s}, \vec{t})$ and $\psi(\vec{s}, \vec{t})$, we write $(\psi \in \langle \varphi \rangle)_{\vec{p},\vec{s}\vec{t}}$ for the FP+ord definable generalised quantifier $\mathbf{MembProb}^+(\mathcal{I})$, where $\mathcal{I}$ is the interpretation of type $T = \mathrm{number} \cdot \mathrm{type}(\vec{p}) \cdot \mathrm{type}(\vec{s})$ from Σ-structures to $\{X, R, \mathsf{target}\}$-structures defined as follows:

$$
\begin{aligned}
\varphi_{dom}(\vec{x}) &:= b_x = 0 \vee b_x = 1 \\
\varphi_{\simeq}(\vec{x}, \vec{y}) &:= \vec{x} = \vec{y} \\
\varphi_X(\vec{x}) &:= b_x = 1 \\
\varphi_R(\vec{x}, \vec{y}, \vec{z}) &:= \begin{cases} b_x = 1 \wedge b_y = 0 \wedge b_z = 0 \\ \circledwedge\, \varphi(\vec{p}_x, \vec{e}_y, \vec{e}_z) \\ \vec{p}_y = \vec{p}_z \end{cases} \\
\varphi_{\mathsf{target}}(\vec{x}, \vec{y}) &:= \begin{cases} b_x = 0 \wedge b_y = 0 \\ \circledwedge\, \psi(\vec{e}_x, \vec{e}_y) \\ \vec{p}_x = \vec{p}_y \end{cases}
\end{aligned}
$$

where any tuple $\vec{x}$ of type T is split into $\vec{x} = (b_x, \vec{p}_x, \vec{e}_x)$ with $\mathrm{type}(\vec{p}_x) = \mathrm{type}(\vec{p})$ and $\mathrm{type}(\vec{e}_x) = \mathrm{type}(\vec{s})$.

Together with Proposition 1.38, Lemma 3.2 implies that this generalised quantifier is definable in FP + ord, and as such, we will use it pervasively in FP + ord formulae. As a corollary, we can also define a *subgroup* predicate within FP + ord:

Corollary 3.5. *Given $\varphi(\vec{p}, \vec{s}, \vec{t})$ and $\psi(\vec{q}, \vec{s}, \vec{t})$, there is a* FP + ord *sentence $(\langle \varphi \rangle \leq \langle \psi \rangle)_{\vec{p}\vec{q},\vec{s}\vec{t}}$ that holds on $\mathfrak{A}$ iff $\langle \varphi \rangle_{\vec{p},\vec{s}\vec{t}}(\mathfrak{A}) \leq \langle \psi \rangle_{\vec{q},\vec{s}\vec{t}}(\mathfrak{A})$*

Proof.

$$(\langle\varphi\rangle \leq \langle\psi\rangle)_{\vec{p}\vec{q},\vec{s}\vec{t}} := \forall\vec{p}, (\varphi(\vec{p}) \in \langle\psi\rangle)_{\vec{q},\vec{s},\vec{t}}$$ □

3.2 Limits to the expressive power of the ord operator

We have shown that the ord operator enables the expression of quite a few natural permutation group properties. However, we have presented one polynomial-time computable operation on permutation groups which is missing from this list: obtaining generating sets for accessible subgroups of an input group, itself represented by a generating set. We showed in Proposition 1.69 how this operation can be computed in polynomial time.

It happens, as we will show in this section, that the representation of groups as generating sets of permutations is not well-suited for this task, in the isomorphism-invariant computation context.

Below Definition 1.42, we have mentioned the fact that any relation defined by a formula in a logic must be canonical. In Theorem 3.9, we will exhibit a class of structures on which FPC witnesses the accessibility of some group $\mathbf{H}$ (in a sense to be made precise in the statement of the theorem), while any canonical set of permutations that generates $\mathbf{H}$ has exponential size.

This result does not exhibit a weakness of FP + ord, but rather a shortcoming of the representation of groups by generating sets of permutations. Indeed, the proofs of Theorems 3.6 and 3.9 do not rely on the actual expressive power of FP+ord, but merely on the fact that it is isomorphism-invariant. Nevertheless, to improve readability, we will still state and prove those results in the context of FP + ord.

As a first step, we show that there are some groups for which a generating set can be constructed in polynomial time, and which cannot be defined within FP + ord:

Theorem 3.6. *There is a class of graphs $\mathcal{K}$ such that:*

- *There is a polynomial-time Turing Machine M which computes, on input* $\mathrm{enc}(\mathfrak{A})$ *for $\mathfrak{A} \in \mathcal{K}$, a generating set for* $\mathrm{Aut}(\mathfrak{A})$.
- *For any* FP+ord *formula φ on graphs, there is some graph $\mathfrak{A} \in \mathcal{K}$ such that $\varphi(\mathfrak{A})$ does not define a generating set of* $\mathrm{Aut}(\mathfrak{A})$.

Our proof relies on the two following lemmas:

Lemma 3.7. *Let $\varphi \in (\mathsf{FP} + \mathsf{ord})[\sigma]$ be a formula with $k+2$ variables, and for $\mathfrak{A} \in$* $\mathrm{STRUC}[\sigma]$, *let $S^{\mathfrak{A}} := \{\mathrm{perm}(\varphi(\mathfrak{A}, \vec{a})) \mid \vec{a} \in A^k\}$ be the set of permutations defined by φ on $\mathfrak{A}$ binding the first k variables. Then:*

- $|S| < |A|^k$

- *For any* $\tau \in \mathrm{Aut}(\mathfrak{A})$*,*
$$S = \{\tau\rho\tau^{-1}, \rho \in S\} =: S^\tau$$

Proof. First, we argue that FP + ord is isomorphism-invariant. The isomorphism-invariance of FP + ord is a direct consequence of the isomorphism-invariance of FP together with the fact that order is an isomorphism-invariant query.

In particular, when $\tau \in \mathrm{Aut}(\mathfrak{A})$, the isomorphism-invariance of FP + ord yields:

$$\psi(\mathfrak{A})^\tau = \psi(\mathfrak{A})$$

Consider now a formula φ as in the hypothesis of the lemma. We argue that $\varphi(\mathfrak{A})^\tau$ defines S^τ in the same way $\varphi(\mathfrak{A})$ defined S, i.e.

$$S^\tau := \{\tau\rho\tau^{-1}, \rho \in S\} = \{\mathrm{perm}(\varphi(\mathfrak{A}, \vec{a})) \mid \vec{a} \in \varphi(\mathfrak{A})^\tau\}$$

Indeed, fix some parameters $\vec{a}$, and let $\rho := \mathrm{perm}(\varphi(\mathfrak{A}, \vec{a}))$. Then $\varphi(\mathfrak{A}, \vec{a})^\tau$ is also the graph of a permutation ρ', which maps $\tau(s)$ to $\tau(t)$, for any s, t such that $\rho(s) = t$, and thus

$$\rho'(s) = \tau(\rho(\tau^{-1}(s)))$$

which concludes the proof that $S = S^\tau$. The fact that $|S|$ is bounded by $|A|^k$ is straight-forward. □

A set of permutations S over $\mathfrak{A}$ (and more generally, any relation over $\mathfrak{A}$) is *canonical* when, for any $\tau \in \mathrm{Aut}(\mathfrak{A})$, $S^\tau = S$. Notice that Lemma 3.7 does not depend on the expressive power of FP + ord but merely on the fact that, for any couple $(\mathfrak{A}, \varphi)$, $\varphi(\mathfrak{A})$ is canonical w.r.t. $\mathfrak{A}$. Now, this fact alone suffices to show Theorem 3.6:

Lemma 3.8. *There is a class of graphs* $\mathcal{K}$ *such that:*

- *There is a polynomial-time Turing Machine that defines* $\mathrm{Aut}(\mathfrak{A})$ *on input* $\mathrm{enc}(\mathfrak{A})$ *for* $\mathfrak{A} \in \mathcal{K}$*.*
- *For any* $\mathfrak{A} \in \mathcal{K}$*, any generating set* S *of* $\mathrm{Aut}(\mathfrak{A})$ *that is canonical has size* $\geq (|A|/2)!$*.*

Recall that, according to Definition 1.42,

Proof. Consider $\mathcal{K} := \{K_{n,n}, n \in \mathbb{N}\}$, where $K_{n,n}$ is the complete bipartite graph with n vertices. Formally, $K_{n,n}$ is a graph with $2n$ vertices $[n] \sqcup [n]'$, where $[n]' := \{1', 2', \ldots, n'\}$ and such that there is an edge between u and v iff $u \in [n] \wedge v \in [n]'$ or $u \in [n]' \wedge v \in [n]$.

For any n, $\mathrm{Aut}(K_{n,n})$ is the set of all permutations of $[n] \cup [n]'$ that stabilise the partition $([n], [n]')$ setwise. It is generated by the set of $n(n-1)/2 + 1$ permutations

$$S := \{(i\ j), i, j \in [n]\} \cup \{(1\ 1')(2\ 2') \ldots (n\ n')\}$$

From there, the fact that S can be defined in polynomial-time when given $\mathrm{enc}(\mathfrak{A})$ as input is straight-forward.

Now, suppose that S is a canonical (w.r.t. $\mathfrak{A}$) set of permutations that generates $\mathrm{Aut}(K_{n,n})$. We show that, for any permutation τ of $\mathrm{Sym}([n])$, S must contain the permutation $\mathrm{Swap}(\tau)$ defined as

$$\mathrm{Swap}(\tau) := (\tau(1)\ 1')(\tau(2)\ 2')\dots(\tau(n)\ n')$$

which then concludes the proof, as this implies that $|S| \geq n!$. First, it must happen that S contains *at least one* $\mathrm{Swap}(\tau_0)$ for some permutation $\tau_0 \in \mathrm{Sym}([n])$, for if it does not, S cannot generate $K_{n,n}$ (either it stabilises $([n], [n]')$ point-wise, or it does not stabilise it at all).

Now, consider any permutation $\tau \in \mathrm{Sym}[n]$, and let τ' be the permutation of $[n]'$ which maps i' to $(\tau(i))'$. $\tau_0\tau'$ stabilises $([n], [n]')$, and is therefore an automorphism of $K_{n,n}$. For $i \in [n]$,

$$\begin{aligned}
(\mathrm{Swap}(\tau_0))^{(\tau_0\tau')^{-1}}(\tau(i)) &= (\tau_0\tau')^{-1}\mathrm{Swap}(\tau_0)(\tau_0\tau')(\tau(i)) \\
&= (\tau_0\tau')^{-1}\mathrm{Swap}(\tau_0)(\tau_0(\tau(i))) \\
&= (\tau_0\tau')^{-1}((\tau(i))') \\
&= (\tau_0\tau')^{-1}(\tau'(i')) \\
&= i'
\end{aligned}$$

Conversely,

$$\begin{aligned}
(\mathrm{Swap}(\tau_0))^{(\tau_0\tau')^{-1}}(i') &= (\tau_0\tau')^{-1}\mathrm{Swap}(\tau_0)(\tau_0\tau')(i') \\
&= (\tau_0\tau')^{-1}\mathrm{Swap}(\tau_0)(\tau'(i')) \\
&= (\tau_0\tau')^{-1}\mathrm{Swap}(\tau_0)((\tau(i))') \\
&= (\tau_0\tau')^{-1}(\tau_0(\tau(i))) \\
&= \tau(i)
\end{aligned}$$

Therefore, $(\mathrm{Swap}(\tau_0))^{(\tau_0\tau')^{-1}} = \mathrm{Swap}(\tau)$. We have thus shown that, for any τ, $\mathrm{Swap}(\tau)$ is a conjugate of $\mathrm{Swap}(\tau_0)$ under the action of $\mathrm{Aut}(\mathfrak{A})$. Together with our assumption that S is closed under conjugation by elements of $\mathrm{Aut}(\mathfrak{A})$, this concludes our proof. □

It happens that the same argument can be adapted to an accessible subgroup, therefore demonstrating that $\mathsf{FP} + \mathsf{ord}$ is not able to express the third primitive on permutation groups provided by the Schreier-Sims algorithm. Formally,

Theorem 3.9. *There is a class of structures $\mathcal{K}$ and a constant l such that:*

- *There is a polynomial-time Turing machine that, on input* $\mathrm{enc}(\mathfrak{A})$ *for* $\mathfrak{A} \in \mathcal{K}$*, computes a generating set for* $\mathrm{Aut}(\mathfrak{A})$.

- *There is a* FP+ord *formula* $\varphi_{\mathbf{G}}(\vec{p}, s, t)$ *such that, on any structure* $\mathfrak{A} \in \mathcal{K}$ $\mathbf{G}(\mathfrak{A}) := \langle \varphi_{\mathbf{G}} \rangle_{\vec{p}}(\mathfrak{A}) \geq \mathrm{Aut}(\mathfrak{A})$ *and* $\mathrm{Aut}(\mathfrak{A})$ *is* l*-accessible from* $\mathbf{G}(\mathfrak{A})$

- *The* l*-accessibility of* $\mathrm{Aut}(\mathfrak{A})$ *from* $\mathbf{G}(\mathfrak{A})$ *is witnessed in* FP + ord*, that is:*

  - *There is a formula* $\varphi_{\in}$ *with* $\mathrm{free}(\varphi_{\in}) = \{\vec{\mu}, X\}$ *(where* X *is a second-order variable of type* $2l$ *and* $\vec{\mu}$ *a* l*-tuple of numerical variables) such that, iterating through all* $\vec{\lambda}$ *encoding an integer* $m \leq |A|^l$*, and starting with* $\mathbf{G}_0(\mathfrak{A}) := \mathbf{G}(\mathfrak{A})$*, the set of permutations* τ *in* $\mathbf{G}_m(\mathfrak{A})$ *such that* $\mathfrak{A}, \vec{\lambda}, \mathrm{graph}(\tau) \models \varphi_{\in}$ *is a subgroup* $\mathbf{G}_{m+1}(\mathfrak{A}) \leq \mathbf{G}_m(\mathfrak{A})$*, and* $\mathbf{G}_{n^l}(\mathfrak{A}) = \mathrm{Aut}(\mathfrak{A})$
  - *For any* $m < |A|^l$, $|\mathbf{G}_m(\mathfrak{A}) : \mathbf{G}_{m+1}(\mathfrak{A})| < |A|^l$.

- *All canonical generating sets for* $\mathrm{Aut}(\mathfrak{A})$ *have size* $\geq 2^{|A|/4}$.

Proof. We construct, for each $n \in \mathbb{N}$, a structure $\mathfrak{A}_n$ such that the class $\mathcal{K} := \{\mathfrak{A}_n \mid n \in \mathbb{N}\}$ satisfies the conditions of the theorem. Fix n an integer. For any integer i, let $C_i := \{(i,0),(i,1)\}$ and $D_i := \{(i,2),(i,3)\}$. Let $C := \bigcup_{i=1}^{n} C_i$ and $D := \bigcup_{i=1}^{n} D_i$. Consider the $\{\preceq, E\}$-structure $\mathfrak{A}_n$ such that:

- $A = C \cup D$

- $a \preceq^{\mathfrak{A}_n} b$ iff there are some $i \leq j$ such that $a \in C_i \cup D_i$ and $b \in C_j \cup D_j$.

- $E^{\mathfrak{A}_n} = (C \times D) \cup (D \times C)$

In words, $\mathfrak{A}_n$ is a coloured simple graph with $4n$ vertices such that each colour class induces a subgraph with 4 vertices, isomorphic to two disconnected edges. Finally, the whole graph is isomorphic to $K_{2n,2n}$, and each colour-class intersects both of those cliques at two distinct vertices. That is, we consider precisely the same structure as in the previous theorem, except a colouring of colour-class size 4 has been added.

We will now show that $\mathcal{K} := \{\mathfrak{A}_n, n \in \mathbb{N}\}$ has the desired properties. First, for any n, $\mathfrak{A}_n$ is a 4-bounded colour-class graph, and as such the automorphism groups of structures in $\mathcal{K}$ is computable in polynomial-time, as we have seen in Section 1.2.

We now show that the l-accessibility of $\mathrm{Aut}(\mathfrak{A}_n)$ is definable in FP + ord (uniformly on n): Consider $\mathbf{G}(\mathfrak{A}_n) := \prod_{i=1}^{n} \mathrm{Sym}(C_i \cup D_i)$. Obviously, $\mathrm{Aut}(\mathfrak{A}_n) \leq \mathbf{G}(\mathfrak{A}_n)$. Moreover, $\mathbf{G}(\mathfrak{A}_n)$ is generated by

$$\{((i,\lambda)\ (i,\mu)), i \leq n, \lambda, \mu \in \{0,1,2,3\}\}$$

We now define $\varphi_{\mathbf{G}}$, such that, for any n, $\varphi_{\mathbf{G}}(\mathfrak{A}_n)$ defines a generating set of $\mathbf{G}(\mathfrak{A}_n)$:

$$\varphi_{\mathbf{G}}(p,q,s,t) := \begin{cases} p \preceq q \\ q \preceq p \\ \bigwedge \begin{cases} s = p \wedge t = q \\ \bigvee \begin{cases} s = q \wedge t = p \\ s \neq p \wedge s \neq q \wedge s = t \end{cases} \end{cases} \end{cases}$$

Let us now show that $\mathrm{Aut}(\mathfrak{A}_n)$ is accessible from $\mathbf{G}(\mathfrak{A}_n)$, and that this accessibility is *definable* in $\mathsf{FP} + \mathsf{ord}$. Consider $\mathbf{G}_0(\mathfrak{A}_n) := \mathbf{G}(\mathfrak{A}_n)$, and for any $k < n$,

$$\mathbf{G}_{k+1}(\mathfrak{A}_n) := \mathrm{Stab}_{\mathbf{G}_k(\mathfrak{A}_n)} \left\{ \bigcup_{i=1}^{k+1} C_i, \bigcup_{i=1}^{k+1} D_i \right\}$$

By induction, any $\sigma \in \mathbf{G}_k(\mathfrak{A}_n)$ fixes $\left\{ \bigcup_{i=1}^{k} C_i, \bigcup_{i=1}^{k} D_i \right\}$ setwise. As such, any two $\sigma, \tau \in \mathbf{G}_k(\mathfrak{A}_n)$ belong to the same $\mathbf{G}_{k+1}(\mathfrak{A}_n)$-coset if they act in the same way *on the partition* $\left\{ \bigcup_{i=1}^{k} C_i, \bigcup_{i=1}^{k} D_i \right\}$, and they act in the same way on $C_k \cup D_k$. As there are two possible ways to act on the former, and $|S_4| = 24$ ways to act on the latter, there are at most 48 cosets of $\mathbf{G}_{k+1}(\mathfrak{A}_n)$ in $\mathbf{G}_k(\mathfrak{A}_n)$, for any n. It only remains to show that a formula $\varphi_{\in}$ that defines conditional membership to $\mathbf{G}_{k+1}$ from $\mathbf{G}_k$ can be constructed in $\mathsf{FP} + \mathsf{ord}$, which can be done as follows:

$$\varphi_{\in}(k, X) := \forall x, y \in \bigcup_{i=1}^{k} C_i \cup D_i, \exists x', y', \bigwedge \begin{cases} X(x, x') \\ X(y, y') \\ E(x,y) \iff E(x', y') \end{cases}$$

The fact that membership to the first k colour-classes (used here in the quantification of x and y) can be defined in FPC is straight-forward. The correctness of the formula $\varphi_{\in}$ relies on the fact that the edge relation, by assumption, defines the equivalence relation of the partition we are aiming to stabilise.

Finally, it is left to show that any canonical generating set for $\mathrm{Aut}(\mathfrak{A}_n)$ has size at least 2^n. The idea is similar to that of Lemma 3.8: such a canonical generating set S must contain all possible swaps between C and D.

Here, the set of swaps is in bijection with the set of functions $f : n \to \{0, 1\}$. For such a function f, we denote $\mathrm{Swap}(f)$ the following automorphism of $\mathfrak{A}_n$:

$$\mathrm{Swap}(f)(i,b) := \begin{cases} (i, 2 + b) & \text{if } b < 2 \wedge f(i) = 0 \\ (i, 2 + (1 - b)) & \text{if } b < 2 \wedge f(i) = 1 \\ (i, b - 2) & \text{if } b \geq 2 \wedge f(i) = 0 \\ (i, (1 - (b - 2))) & \text{if } b \geq 2 \wedge f(i) = 1 \end{cases}$$

Clearly, $(f \mapsto \mathrm{Swap}(f))$ is injective, and as such, the set of all swaps has size 2^n. Moreover, any $\sigma \in \mathrm{Aut}(\mathfrak{A}_n)$ that permutes C and D is of the form $\mathrm{Swap}(f)$ for some f. As such, any generating set of $\mathrm{Aut}(\mathfrak{A}_n)$ must contain such a swap.

Now, consider S such a generating set, and suppose that S is isomorphism-invariant. Let f_0 be so that $\mathrm{Swap}_{f_0} \in S$. Consider any $f : n \to \{0,1\}$. Let $\sigma_f \in \prod \mathrm{Sym}(C_i)$ be the permutation that swaps $(i,0)$ and $(i,1)$ for all i such that $f(i) \neq f_0(i)$. It is easy to see that $\sigma_f \in \mathrm{Aut}(\mathfrak{A})$, and, if $b < 2$ (note that $\sigma_f^{-1} = \sigma_f$). Now, suppose that i is such that $f(i) = f_0(i)$. Then, σ_f fixes (i,b), and $\mathrm{Swap}(f)$ and $\mathrm{Swap}(f_0)$ agree on (i,b) for all b. As such,

$$\mathrm{Swap}(f_0)^{\sigma_f}(i,b) = \sigma_f \cdot \mathrm{Swap}(f_0) \cdot \sigma_f(i,b) = \mathrm{Swap}(f)(i,b)$$

for all b. Otherwise, if $f(i) \neq f_0(i)$, for $b < 2$:

$$\begin{aligned}
\mathrm{Swap}(f_0)^{\sigma_f}(i,b) &= \sigma_f \cdot \mathrm{Swap}(f_0) \cdot \sigma_f(i,b) \\
&= \sigma_f \cdot \mathrm{Swap}(f_0)(i, 1-b) \\
&= \begin{cases} \sigma_f(i, 2 + (1-b)) & \text{if } f_0(i) = 0 \\ \sigma_f(i, 2 + (1-(1-b))) & \text{if } f_0(i) = 1 \end{cases} \\
\text{Since } f(i) \neq f_0(i), \ &= \begin{cases} \sigma_f(i, 2 + (1-b)) & \text{if } f(i) = 1 \\ \sigma_f(i, 2 + b) & \text{if } f(i) = 0 \end{cases} \\
&= \sigma_f \cdot \mathrm{Swap}(f)(i,b) \\
\text{Since } \mathrm{Swap}(f)(i,b) \in D, \ &= \mathrm{Swap}(f)(i,b)
\end{aligned}$$

Finally, the same argument as in Lemma 3.8 can be used: $\mathrm{Swap}(f_0)^{\sigma_f}$ is the conjugate of a product of disjoint transpositions, and as such is also a product of disjoint transpositions. This implies that $\mathrm{Swap}(f_0)^{\sigma_f}(i,b) = \mathrm{Swap}(f)(i,b)$ for $b \geq 2$, and concludes our proof that $\mathrm{Swap}(f_0)^{\sigma_f} = \mathrm{Swap}(f)$.

Thus, all swaps are conjugate w.r.t. $\mathrm{Aut}(\mathfrak{A}_n)$, and must therefore be contained in S, which must have size $\geq 2^n$, concluding our proof. □

The fact that our proof remains true for any logic extending FO that preserves isomorphism-invariance hints at the fact that yet another abstraction on group computations is needed to capture the power of polynomial-time permutation group computations. Said differently, our way to encode permutation groups in extensions of FO forbids the definition of groups which should be definable.

3.3 The morphism representation of subgroups

We have just shown that isomorphism invariance has a cost when trying to represent groups by generating sets of permutations. This shortcoming motivates the search

of new representations for permutation groups; representations that would allow the definition of groups which admit no canonical generating sets of polynomial size.

In this section, we provide a partial solution to this question. Precisely, we show that if G is a group which admits a definable generating set, and $H \trianglelefteq G$, the graph of a morphism $m : G \to \mathrm{Sym}(\Omega)$ such that $\ker(m) = H$ constitutes a satisfying representation of H, that we call a *morphism-definition of H from G* In particular, we will show that the basic operations enabled by the ord operator on groups provided by generating sets can also be defined on groups provided by such a morphism.

Additionally, we will show that, in this restricted setting of morphism-definable subgroups, we can actually define arbitrary intersections of groups. The ability to construct representations of subgroups is central to the group-theoretic framework to graph isomorphism and canonisation. Indeed, these algorithms rely on gradually refining a large group until a generating set for the automorphism group of the graph in question has been computed.

Another important feature of morphism-definability is that it enables the representation of *cosets* of a subgroup in an isomorphism-invariant way. The representation of cosets constitutes another limitation of the presentation of groups by generating sets of permutations in an isomorphism-invariant context: in the computational context, one can represent a coset σH of a subgroup $H \leq G$ by a generating set for H and *any choice* of $\tau \in G$ such that $\tau H = \sigma H$. In general, there is no canonical such choice of representative τ of the coset, which prohibits such a representation scheme for cosets in $\mathsf{FP} + \mathsf{ord}$ (or any logic for P). On the other hand, when H is provided through a morphism $m : G \to \mathrm{Sym}(\Omega)$ such that $\ker(m) = H$, there is a bijection between $\mathrm{im}(m)$ and the cosets of H in G. Assuming that m is definable (in the sense of Definition 3.10 below), this bijection yields a canonical representation of cosets of H in G. As we have briefly mentioned in the introduction of this thesis, and as we will see in Section 4.3, the ability to represent cosets is particularly important in the group-theoretic framework to Graph Canonisation. The techniques developped in this section will thus play an important part in our canonisation procedure in Chapter 4.

Our notion of definability of a morphism $m : G \to H$ is quite natural: a formula φ_m should define, given $g \in G$, the graph of $m(g)$ (as a permutation). However, as G is assumed to be a permutation group, φ_m defines a function from second-order objects to second-order objects. Thus, to represent a morphism in this way, a second-order variable is necessary:[1]

Definition 3.10. For T, T' two types, and $\varphi_m(R, \vec{x}, \vec{y})$ a formula with R a relational variable of type $T \cdot T$, and $\mathrm{type}(\vec{x}) = \mathrm{type}(\vec{y}) = T'$, φ_m is said to *define a morphism* $m : G \to \mathrm{Sym}(A^{T'})$ *on* $\mathfrak{A}$, where $G \leq \mathrm{Sym}(A^T)$ if, for all $\sigma \in G$,

$$\varphi_m(\mathfrak{A}, \mathrm{graph}(\sigma)) = \mathrm{graph}(m(\sigma))$$

[1] Actually, if G admits a definable generating set, m can be represented by the images of each element in this generating set. To improve clarity, we present morphism-definability with second-order variables

$H \trianglelefteq G$ is *$\mathcal{L}$-morphism-definable* from G in $\mathfrak{A}$ if $\mathcal{L}$ defines a generating set for G in $\mathfrak{A}$, and there is a $\mathcal{L}$-formula φ_m which defines a morphism $m : G \to \mathrm{Sym}(A^{T'})$ on $\mathfrak{A}$ such that $\ker(m) = H$.

In the remainder of this section, we will show that if we restrict our attention to morphism-definable subgroups (rather than accessible subgroups), many natural group operations are definable in FP + ord. Before doing so, let us study the relationship between morphism-definable subgroups and accessible subgroups. First, it is a consequence of the Correspondence Theorem (see e.g. [Isa08, Theorem X.21]) that any subgroup morphism-definable from G in $\mathfrak{A}$ is accessible from G in $\mathfrak{A}$:

Theorem 3.11 (Correspondence Theorem)**.** *Given a group G and $N \trianglelefteq G$, $\phi : H \mapsto H/N$ is a bijection between the subgroups of G containing N and the subgroups of G/N. Moreover, ϕ is monotone w.r.t. inclusion. In particular, $\phi(N) = 1$ and $\phi(G) = G/N$. Finally, if $G \geq A \geq B \geq N$, $|A : B| = |A/N : B/N|$.*

Lemma 3.12. *Consider a structure $\mathfrak{A}$ and two formulae $\varphi_G(\vec{p}, \vec{s}, \vec{t}), \varphi_m(R, \vec{x}, \vec{y})$ such that φ_m defines a morphism m from $G := \langle \varphi_G \rangle_{\vec{p}}(\mathfrak{A})$ to $\mathrm{Sym}(A^{\mathrm{type}(\vec{x})})$. Then, $H := \ker(m)$ is $|\vec{x}|$-accessible from G.*

Proof. Let $T = \mathrm{type}(\vec{s}), T' = \mathrm{type}(\vec{x}), \lambda = |A|^{|T'|}$, and fix an enumeration $(\omega_i)_{i<\lambda}$ of $A^{T'}$. This enumeration allows us to build an adequate chain of subgroups for $m(G) \leq \mathrm{Sym}(A^{T'})$ as described on Page 42, i.e. we consider the chain of subgroups

$$m(G) = K_0 \geq K_1 \cdots \geq K_\lambda = 1$$

where

$$K_{i+1} := \{\sigma \in K_i, \sigma(\omega_i) = \omega_i\}$$

and note that the length of this chain, as well as the index of K_{i+1} in K_i (for each i) are bounded by λ. Since each K_i is a subgroup of $m(G) \simeq G/H$ (by the First Isomorphism Theorem), the Correspondence Theorem implies that there is a unique subgroup H_i such that $K_i \simeq H_i/H$. Because this correspondence is monotone, the groups H_i form a subgroup chain

$$G = H_0 \geq H_1 \geq \cdots \geq H_\lambda = H.$$

It is left to show that this chain witnesses the accessibility of H from $G \leq \mathrm{Sym}(A^T)$. The fact that H_{i+1} admits a conditional membership test in H_i relies on a fact which we haven't proven at this point, namely, that FP + ord has polynomial-time model-checking. This will be shown in Theorem 3.21, and the proof of this fact does not depend on the present lemma. Given $\sigma \in H_i$, σ belongs to H_{i+1} iff $m(\sigma) \in \mathrm{Sym}(A^{T'})$ stabilises ω_{i+1}, which can be defined by the following FP + ord formula:

$$\varphi_{\mathrm{Stab}}(R_\sigma, \vec{x}) := \varphi_m(R_\sigma, \vec{x}, \vec{x})$$

For $\sigma \in H_i$, we therefore have

$$\sigma \in H_{i+1} \iff (\mathfrak{A}, \mathrm{graph}(\sigma), \omega_{i+1}) \models \varphi_{\mathrm{Stab}}$$

which is polynomial-time computable according to Theorem 3.21. Finally, by the Correspondence Theorem, $|H_i : H_{i+1}| = |K_i : K_{i+1}| \leq N = |A|^{|T'|}$. □

We have just shown that if $H \trianglelefteq G$ is morphism-definable from G, it is then accessible from G. Interestingly, the reverse implication does not hold:

Lemma 3.13. *There is a family of couples of groups $(K_n, G_n)_{n\in\mathbb{N}}$ such that $K_n \trianglelefteq G_n \leq S_{4n}$, with K_n 1-accessible from G_n, while for each $\mathsf{FP} + \mathsf{ord}$-formula $\varphi_m(R, \vec{x}, \vec{y})$, there is an n such that φ_m does not morphism-define K_n over G_n on the structure $\mathbf{G}([4n], X_n)$, where X_n is any small generator of the faithful action of G_n on $[4n]$.*

Proof. This stems from the fact that some quotient groups only admit faithful permutation representations over exponentially larger sets. Such a sequence of couples $(K_n, G_n)_{n\in\mathbb{N}}$ is exhibited in [Neu87, Example 1.1]: consider $\sigma_i := (4i+1\ 4i+2\ 4i+3\ 4i{+}4)$, $\tau_i := (4i{+}2\ 4i{+}4)$, and let $G_n := \langle \sigma_i, \tau_i \mid i < n\rangle$, and $K_n := \langle \sigma_0^2\sigma_i^2 \mid 0 < i < n\rangle$.

It is shown in [Neu87] that any set Γ such that G_n/K_n is isomorphic to a subgroup of $\mathrm{Sym}(\Gamma)$ is such that $|\Gamma| > 2^{n+1}$. As such, for any formula $\varphi_m(R, \vec{x}, \vec{y})$, it suffices to pick n such that $(4n)^{|\mathrm{type}(\vec{x})|} < 2^{n+1}$. Because $|A^{\mathrm{type}(\vec{x})}| < 2^{n+1}$, φ_m cannot morphism-define K_n: otherwise, we would have $G_n/K_n \simeq m(G_n) \leq \mathrm{Sym}(A^{\mathrm{type}(\vec{x})})$.

To show that K_n is 1-accessible from G_n, a quick analysis of the structure of G_n is necessary: G_n is the direct product of n instances of D_8, the dihedral group of order 8. K_n is a subgroup of $Z_n := \langle \sigma_i^2 \mid i < n\rangle$ (which is the center of G_n, i.e. the group). K_n has order 2 in Z_n, so it only remains to show that Z_n is 1-accessible from G_n. To do so, it suffices to consider the chain of subgroups

$$G_n = H_{0,n} \geq H_{1,n} \geq \cdots \geq H_{n,n} = Z_n$$

where

$$H_{i,n} := \langle \{\sigma_j^2 \mid j < i\} \cup \{\sigma_j \mid j \geq i\} \cup \{\tau_j \mid j \geq i\}\rangle$$

that is, $H_{i,n}$ is the direct product of i copies of $\mathbb{Z}_2$, and $n - i$ copies of D_8. It is easy to see that $|H_{i,n} : H_{i,n+1}| = 4$, and as such this chain of subgroups indeed witnesses the accessibility of Z_n (and hence K_n) from G_n. □

Thus, morphism-definable subgroups of G constitute, in general, a strict subset of accessible normal subgroups of G. Having delimited the range of our definitions, we now proceed with our main results concerning morphism-definable groups. First, we show that if H is morphism-definable from G in $\mathfrak{A}$, a generating set for a faithful representation of G/H is definable in $\mathfrak{A}$:

Lemma 3.14. *Let $\mathcal{L} \geq \mathsf{FPC}$ and $\mathfrak{A}$ be a structure. Let $\varphi(\vec{p}, \vec{s}, \vec{t})$ be a $\mathcal{L}$-formula that defines a permutation on $(\mathfrak{A}, \vec{a})$ for all $\vec{a} \in A^{\mathrm{type}(\vec{p})}$, and let $\varphi_m(R, \vec{x}, \vec{y})$ be a formula defining a morphism $m : G \to \mathrm{Sym}(A^{\mathrm{type}(\vec{x})})$, where $G := \langle\varphi\rangle_{\vec{p}.\vec{s}\vec{t}}(\mathfrak{A})$. Then, $\mathrm{im}(m)$ admits a $\mathcal{L}$-definable generating set.*

Proof.

$$\varphi_{\mathrm{im}(m)}(\vec{p}, \vec{x}, \vec{y}) := \varphi_m(R, \vec{x}, \vec{y})[R(\vec{s}, \vec{t})/\varphi(\vec{p}, \vec{s}, \vec{t})]$$ □

This first result will be quite important in Section 3.4. Moreover, given such a $v \in \mathrm{im}(m)$,

$$m^{-1}(v) = \{\sigma \in G, m(\sigma) = v\}$$

is a *coset* of $\ker(m)$, equal to $\sigma \ker(m)$ for any $\sigma \in m^{-1}(v)$. Recall that we have argued at the beginning of this section that, unlike in the computational framework, the isomorphism-invariance requirement of a logic prohibits the representation of cosets through the choice of witnesses. In the case of a subgroup H defined by a morphism m, the unique v such that $m(\sigma H) = \{v\}$ constitutes a canonical representative of σH. This fact will play a major role in Chapter 4.

Now, we show that the first two operations of the Schreier-Sims framework are definable in the context of morphism-defined subgroups:

Lemma 3.15. *Consider two formulae $\varphi_G(\vec{p}, \vec{s}, \vec{t}), \varphi_m(R, \vec{x}, \vec{y})$. There are $\mathsf{FP} + \mathsf{ord}$ formulae $\varphi_\in(R), \varphi_{\mathsf{ord}}(\vec{\mu})$ such that, on any structure $\mathfrak{A}$ on which φ_m defines a morphism m from $G := \langle\varphi_G\rangle_{\vec{p}}(\mathfrak{A})$ to $\mathrm{Sym}(A^{\mathrm{type}(\vec{x})})$,*

- *for any $\tau \in \mathrm{Sym}(A^{\mathrm{type}(\vec{s})})$, $(\mathfrak{A}, \mathrm{graph}(\tau)) \models \varphi_\in$ iff $\tau \in \ker(m)$*
- *$\varphi_{\mathsf{ord}}(\mathfrak{A})$ is a $2|\vec{s}|$-ary numerical predicate encoding $|\ker(m)|$.*

Proof. Given $\tau \in \mathrm{Sym}(A^{\mathrm{type}(\vec{s})})$, it is in $\ker(m)$ iff it is in G and $m(\tau) = \mathrm{Id}$. Using Lemma 3.2, this is easily definable in $\mathsf{FP} + \mathsf{ord}$:

$$\varphi_\in(R) := (R(\vec{s}, \vec{t}) \in \langle\varphi_G\rangle)_{\vec{p}.\vec{s}\vec{t}} \wedge \forall \vec{x}, \varphi_m(R, \vec{x}, \vec{x})$$

To compute the order of $\ker(m)$, we use the fact that $|\ker(m)| = \frac{|G|}{|\mathrm{im}(m)|}$:

$$\varphi_{\mathsf{ord}}(\vec{\mu}) := \left(\frac{(\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\varphi_G)}{(\mathsf{ord}_{\vec{p}.\vec{s}\vec{t}}\varphi_{\mathrm{im}(m)})}\right)(\vec{\mu})$$

where $(\frac{P}{Q})$ denotes the result of the euclidean division of P by Q, where P and Q, and $(\frac{P}{Q})$ are $2|\vec{s}|$-ary numerical relations encoding integers bounded by $2^{(n^{2|\vec{s}|})}$. Note that, since euclidean division of integers (encoded in binary) is a polynomial-time computable arithmetic function, the Immerman-Vardi theorem ensures that the above expression is definable in $\mathsf{FP} + \mathsf{ord}$. □

Finally, we show how we can construct a representation for the intersection of two groups morphism-defined from G.

Definition 3.16. For $m_1 : G \to X$ and $m_2 : G \to Y$ two morphisms, let

$$\begin{aligned} m_1 \otimes m_2 : G &\to X \times Y \\ g &\mapsto (m_1(g), m_2(g)) \end{aligned}$$

It is quite clear that $\ker(m_1 \otimes m_2) = \ker(m_1) \cap \ker(m_2)$, and as such, $m_1 \otimes m_2$ defines (over G) the intersection of the subgroups defined by m_1 and m_2 over G. It remains to show that, under a suitable permutation representation of direct products of groups, the $\otimes$ operation on morphisms is definable in FPC. Definition 3.16 relies on direct product of groups, and let us thus first provide a representation of direct products of groups in fixed-point logics. In the following, the *family of groups defined by* $\varphi_{\mathcal{G}}(\vec{p}, \vec{q}, \vec{s}, \vec{t})$ *over* $\mathfrak{A}$ *binding* $\vec{p}$ is the family of groups $(\mathcal{G}_{\vec{a}})$ where, for $\vec{a} \in A^{\mathrm{type}(\vec{p})}$, $\mathcal{G}_{\vec{a}} := \langle \varphi_{\mathcal{G}} \rangle_{\vec{q}.\vec{s}\vec{t}}(\mathfrak{A}, \vec{a})$.

Lemma 3.17. *Let* $\mathcal{L} \geq$ FPC*, and* $\varphi_{\mathcal{G}}(\vec{p}, \vec{q}, \vec{s}, \vec{t})$ *be a* $\mathcal{L}[\Sigma]$*-formula defining a* family *of groups binding* $\vec{p}$*, and let* $\Omega := A^{\mathrm{type}(\vec{p})} \times A^{\mathrm{type}(\vec{s})}$*. Then,* $\prod_{\vec{a} \in A^{\mathrm{type}(\vec{p})}} \mathcal{G}_{\vec{a}}$ *is isomorphic to a subgroup of* $\mathrm{Sym}(\Omega)$*. Moreover, there is a* $\mathcal{L}[\Sigma]$ *formula* $\varphi_{\Pi\mathcal{G}}$ *that defines a generating set for this permutation group isomorphic to* $\prod \mathcal{G}_{\vec{n}}$*.*

Proof. Consider the following action of $\prod_{\vec{a}} \mathcal{G}_{\vec{a}}$ on Ω:

$$(\vec{g}) \cdot (\vec{\lambda}, \vec{a}) := (\vec{\lambda}, g_{\vec{\lambda}}(\vec{a}))$$

It is quite straight-forward that this indeed defines an action, and that it is faithful. As such, we have described a permutation group representing $\prod \mathcal{G}_{\vec{a}}$

Let us now show that it is definable in FPC. As $\prod \mathcal{G}_{\vec{a}}$ is generated by $\bigcup S_{\vec{a}}$, where $S_{\vec{a}}$ is a generating set for $\mathcal{G}_{\vec{a}}$, it is sufficient to show that, for any permutation defined by $\varphi_{\mathcal{G}}$, we can define its action on Ω in FPC:

$$\varphi_{\Pi\mathcal{G}}(\vec{p}, \vec{q}, \vec{x}_s, \vec{s}, \vec{x}_t, \vec{t}) := \begin{cases} \vec{x}_s = \vec{x}_t \\ \bigwedge \begin{cases} \vec{p} = \vec{x}_s \wedge \varphi_{\mathcal{G}}(\vec{p}, \vec{q}, \vec{s}, \vec{t}) \\ \bigvee \begin{cases} \vec{p} \neq \vec{x}_s \wedge \vec{s} = \vec{t} \end{cases} \end{cases} \end{cases}$$

For any $(\vec{a}, \vec{c}) \in A^{\mathrm{type}(\vec{p})} \times A^{\mathrm{type}(\vec{q})}$, $\varphi_{\Pi\mathcal{G}}(\mathfrak{A}, \vec{a}, \vec{c})$ defines the action of the permutation in $\mathcal{G}_{\vec{a}}$ whose graph is $\varphi_{\mathcal{G}}(\mathfrak{A}, \vec{a}, \vec{c})$ on Ω. As such, $\langle \varphi_{\Pi\mathcal{G}} \rangle_{\vec{p},\vec{q}}(\mathfrak{A}) \simeq \prod_{\vec{a} \in A^{\mathrm{type}(\vec{p})}} \mathcal{G}_{\vec{a}}$. □

Lemma 3.18. *Consider two formulae* $\varphi_G(\vec{q}, \vec{s}, \vec{t})$ *and* $\varphi_m(\vec{p}, R, \vec{x}, \vec{y})$ *such that, for all structure* $\mathfrak{A}$ *and* $\vec{a} \in \mathcal{A}^{\mathrm{type}(\vec{p})}$*,* $\varphi_m(\mathfrak{A}, \vec{a})$ *defines a morphism* $m_{\vec{a}} : \langle \varphi_G \rangle_{\vec{q}}(\mathfrak{A}) \to \mathrm{Sym}(A^{\mathrm{type}(\vec{s})})$*. There is a formula* $\varphi_{\otimes m}(R, \vec{p}_x \vec{x}, \vec{p}_y, \vec{y})$ *that defines the morphism*

$$\otimes m := \bigotimes_{\vec{a} \in A^{\mathrm{type}(\vec{p})}} m_{\vec{a}} : G \mapsto \prod_{\vec{a} \in A^{\mathrm{type}(\vec{p})}} \mathrm{im}(m_{\vec{a}})$$

Proof.

$$\varphi_{\otimes m}(R, \vec{p}_s, \vec{s}, \vec{p}_t, \vec{t}) := \bigwedge \begin{cases} \vec{p}_s = \vec{p}_t \\ \varphi_m(\vec{p}_s, R, \vec{s}, \vec{t}) \end{cases}$$

□

We have already seen that one feature of morphism-defined groups is that they allow for canonical representation of their cosets: if $m : G \to \mathrm{Sym}(\Omega)$ is such that $\ker(m) = H$, any $v \in \mathrm{im}(m)$ defines the coset $m^{-1}(v) = \sigma \ker(m)$ for any $\sigma \in G$ such that $m(\sigma) = v$. Now, if m is of the form $\bigotimes m_{\vec{n}}$ for some family of morphisms $m_{\vec{a}} : G \to X_{\vec{a}}$, any coset of m is an intersection of cosets of the groups defined by the morphisms $m_{\vec{a}}$. In order to represent cosets of the intersection of such a morphism-defined subgroup, we show how to encode values in the image of the morphism $\bigotimes m_{\vec{a}}$, when given a representation of a family of values, one for each $m_{\vec{a}}$: as we use an encoding for the product of the $\mathrm{im}(m_{\vec{a}})$, we need to make sure that, given values in each of the groups $m_{\vec{a}}(G)$, we can construct the corresponding value in the encoding of $\prod_{\vec{a}} \mathrm{im}(m_{\vec{a}})$, in order to represent cosets of the intersection of those morphism-defined subgroups.

Lemma 3.19. *Let $\varphi_{\mathcal{G}}$ be as in Lemma 3.17. Suppose that $\varphi_v(\vec{p}, \vec{s}, \vec{t})$ is a formula such that for all $\vec{a}$, $\varphi_v(\mathfrak{A}, \vec{a})$ is the graph of some element of $\langle \varphi_{\mathcal{G}} \rangle_{\vec{q}.\vec{s}\vec{t}}(\mathfrak{A}, \vec{a})$. Then, there is a formula $\varphi_{\oplus v}(\vec{p}_s, \vec{s}, \vec{p}_t, \vec{t})$ that defines the image of $(v_{\vec{n}})$ through the isomorphism between $\prod \mathcal{G}_{\vec{a}}$ and $\mathrm{Sym}(\Omega)$ defined in Lemma 3.17.*

Proof.

$$\varphi_{\oplus v}(\vec{p}_s, \vec{s}, \vec{p}_t, \vec{t}) := \bigwedge \begin{cases} \vec{p}_s = \vec{p}_t \\ \varphi_v(\vec{p}_s, \vec{s}, \vec{t}) \end{cases}$$

□

To conclude this section, we illustrate the use of morphism-definability by showing that the groups shown in Theorem 3.9 not to admit polynomial-size canonical generating sets are morphism-definable.

Example 3.20. Let $\mathcal{K}$ be the class of structures considered in the proof of Theorem 3.9. That is, any structure $\mathfrak{A} \in \mathcal{K}$ is a complete bipartite graph $K_{2n,2n}$ for some n, with a colouring of colour-class size 4, such that each colour-class contains exactly two vertices of each part of the graph. We use the same notations as in Theorem 3.9: the two parts of the bipartite graph are denoted C and D, and the colour-classes are denoted A_i, $i \leq m$, with $C_i := A_i \cap C$ and $D_i := A_i \cap D$. Obviously, $x \in A_\mu$ is a relation definable in FPC.

It is easy to see that FPC defines a generating set for the automorphism group of each colour-class, as each of those groups has constant size: first, we can define a formula $\varphi_{\mathrm{aut}}(\mu, \vec{p}, \vec{q})$ which holds iff $p_i \mapsto q_i$ is an automorphism of the μ-th colour-class:

$$\varphi_{\mathrm{aut}}(\mu, \vec{p}, \vec{q}) := \bigwedge \begin{cases} \bigwedge_{i=1}^{4} (p_i \in A_\mu \wedge q_i \in A_\mu) \\ \forall s \in A_i, \bigvee_{i=1}^{4} p_i = s \wedge \bigvee_{i=1}^{4} q_i = s \\ \bigwedge_{i=1}^{4} \bigwedge_{j=1}^{4} (E(p_i, p_j) \iff E(q_i, q_j)) \end{cases}$$

and one can easily define the action of such an automorphism on the whole graph (making it act trivially on all other colour-classes):

$$\varphi_{\text{gen}}(\mu,\vec{p},\vec{q},s,t) := \varphi_{\text{aut}}(\mu,\vec{p},\vec{q}) \wedge \bigvee \begin{cases} s \notin A_\mu \wedge s = t \\ s \in A_\mu \wedge \bigvee_{i=1}^{4}(s = p_i \wedge t = q_i) \end{cases}$$

Thus, $\langle\varphi_{\text{gen}}\rangle_{\mu,\vec{p},\vec{q};s,t}(\mathfrak{A}) = \mathbf{H}(\mathfrak{A}) := \prod_{i=1}^{n} \text{Aut}(\mathfrak{A}_i)$ where $\mathfrak{A}_i$ is the subgroup induced by the i-th colour-class. The difference between $\mathbf{H}(\mathfrak{A})$ and the group $\mathbf{G}(\mathfrak{A})$ defined in Theorem 3.9 is that on any colour-class A_i, $\sigma \in \mathbf{H}(\mathfrak{A})$ stabilises $\{C_i, D_i\}$ (setwise). We claim that $\text{Aut}(\mathfrak{A})$ is morphism-definable in $\mathbf{H}(\mathfrak{A})$.

To show this, first remark that $\sigma \in \mathbf{H}(\mathfrak{A})$ belongs to $\text{Aut}(\mathfrak{A})$ if and only if σ stabilises $\{C, D\}$. We can rephrase this last condition by restricting our attention to two colour-classes at a time. Indeed, it is easy to see that σ stabilises the partition $\{C, D\}$ iff for any $i, j \leq n$,

$$C_i^\sigma = C_i \iff C_j^\sigma = C_j$$

Consider $m_i : \mathbf{H}(\mathfrak{A}) \to \mathbb{Z}_2$ defined by

$$m_i(\sigma) := \begin{cases} 1 & \text{if } C_i^\sigma = C_i \\ 0 & \text{otherwise} \end{cases}$$

and $M_{i,j} : \mathbf{H}(\mathfrak{A}) \to \mathbb{Z}_2$ defined by

$$M_{i,j}(\sigma) := m_j^{-1}(\sigma) m_i(\sigma)$$

It is easy to see that m_i is a morphism[2], and that $M_{i,j}(\sigma) = 0$ if and only if $(C_i^\sigma = C_i \iff C_j^\sigma = C_j)$. The fact that $M_{i,j}$ is a morphism is a direct consequence of the fact that $\mathbb{Z}_2$ is abelian:

$$\begin{aligned} M_{i,j}(\sigma\tau) &= m_i(\sigma\tau) m_j^{-1}(\sigma\tau) \\ &= m_i(\sigma) m_i(\tau) m_j^{-1}(\tau) m_j^{-1}(\sigma) \\ &= m_i(\sigma) m_j^{-1}(\sigma) m_i(\tau) m_j^{-1}(\tau) \\ &= M_{i,j}(\sigma) M_{i,j}(\tau) \end{aligned}$$

Now, let us show that m_i is definable in FPC:

$$\varphi_m(\mu, R, b_s, b_t) := \bigwedge \begin{cases} b_s = 0 \vee b_s = 1 \\ b_t = 0 \vee b_t = 1 \\ b_s \neq b_t \iff (\exists x, y \in A_\mu, R(x,y) \wedge E(x,y)) \end{cases}$$

From there, one can easily define $M_{i,j}$:

$$\varphi_M(\mu,\nu,R,b_s,b_t) := \exists b' \leq 1, \bigwedge \begin{cases} \varphi_m(\mu, R, b_s, b') \\ \varphi_m(\nu, R, b_t, b') \end{cases}$$

[2]When some group G stabilises a partition Σ, G always acts on Σ (see [DM96, below Exercise 1.5.2]). The morphism m_i under consideration here is exactly the morphism defining this action.

Using Lemma 3.18, we can define $\bigotimes_{i,j\leq n} M_{i,j}$, and, as argued above, its kernel is exactly $\mathrm{Aut}(\mathfrak{A})$.

This shows that the notion of morphism-definability effectively pushes the limits of the definability of groups exhibited in Theorem 3.9. However, the scope of this technique is quite restrained, and we can easily build a subgroup which seems to resist such an approach. Indeed, the morphism-definability of $\mathrm{Aut}(\mathfrak{A})$ depicted above depends crucially on the fact that $M_{i,j}$ is a morphism; and $M_{i,j}$ is a morphism because the action of $\mathbf{H}(\mathfrak{A})$ on $\{C, D\}$ is isomorphic to an abelian group. Therefore, if we generalise the approach from Theorem 3.9 to use $K_{2n,2n,2n}$, the complete tripartite graph (with colour-classes of size 6), we can once again define $\mathrm{Aut}(\mathfrak{A}_i)$ for each i, but the partition stabilised by $\mathrm{Aut}(\mathfrak{A}_i)$ is now made of three sets, and the action of $\mathrm{Aut}(\mathfrak{A}_i)$ is now isomorphic to S_3, which is not abelian. As such, the argument above is not applicable, and we do not know of a morphism-definition of $\mathrm{Aut}(\mathfrak{A})$ for $K_{2n,2n,2n}$.

3.4 FP + ord as a candidate logic for P

We conclude this chapter by proving that FP + ord is at least as strong a candidate logic for P as FP + rk. First, as was the case for FP + rk, FP + ord can be model-checked in polynomial-time:

Theorem 3.21. *Fix a signature Σ, and a $(\mathsf{FP} + \mathsf{ord})[\Sigma]$ formula φ. The function that maps any encoding of a structure $\mathfrak{A}$ to its set of satisfying valuations $\varphi(\mathfrak{A})$ is a polynomial-time computable function.*

Proof. We prove this by induction on φ. Fix a Σ-structure $\mathfrak{A}$. Note that, if φ is of any of the forms $\psi \vee \xi$, $\neg\psi$, $\exists x, \psi$, or $(\mathrm{ifp}_{R,\vec{x}}\psi)$ — that is, any other case than the use of the ord operator — this is a direct application of the fact that FP has polynomial-time model checking: let X, Y be two fresh relation symbols, and let $\mathfrak{A}^+$ be the $\Sigma \sqcup \{X, Y\}$-structure $(\mathfrak{A}, \psi(\mathfrak{A}), \xi(\mathfrak{A}))$. Then, $\varphi(\mathfrak{A}) = \varphi[\psi \leftarrow X, \xi \leftarrow Y](\mathfrak{A}^+)$, which is a FP-formula (if ξ does not appear in φ, ξ can just be set to $\top$ in this reduction).

Let us remark that, while the fact that FP *sentences* have polynomial-time model-checking is often proved in textbooks (see [Imm99, Theorem 4.10] or [Lib04, Theorem 10.14 and Proposition 6.6]), the fact that the set of satisfying valuations of *formulae* is polynomial-time computable given $\mathfrak{A}$ for any fixed φ requires a few observations. Consider a formula φ with free variables $x_1, \ldots, x_k$. The set of all potential assignments of those variables on $\mathfrak{A}$ is $A^{\mathrm{type}(\vec{x})}$, which has size $|A|^k$, and constructing an encoding of $\varphi(\mathfrak{A})$ reduces to check, for each valuation v, whether $v \in \varphi(\mathfrak{A})$, which can be reduced to the model-checking over sentences easily: introduce fresh constant symbols $c_1, \ldots, c_k$. The following holds:

$$v \in \varphi(\mathfrak{A}) \iff \mathfrak{A} \models \varphi[x_i/c_i, i \leq k](\mathfrak{A}, v(x_1), \ldots, v(x_k)) \tag{3.1}$$

Thus it is only left to treat the case where $\varphi = (\mathsf{ord}_{\vec{p},\vec{s}\vec{t}}\psi(\vec{p},\vec{s},\vec{t}))(\vec{\mu})$. By induction hypothesis, $\psi(\mathfrak{A})$ can be computed, and let

$$\tau_{\vec{a}} := \mathrm{perm}(\{(\vec{u},\vec{v}) \mid (\vec{a},\vec{u},\vec{v}) \in \psi(\mathfrak{A})\})$$

for each $\vec{a} \in A^{\mathrm{type}(\vec{p})}$. The list of all $\tau_{\vec{a}}$ can be computed in polynomial-time, and they form a polynomially-large set of permutations on a polynomially-bounded domain ($A^{\mathrm{type}(\vec{x})}$). (Here, the polynomials are with respect to $|A|$.) Under those conditions, we have seen in Section 1.2 that the order of $\langle \tau_{\vec{a}}, \vec{a} \in A^{\mathrm{type}(\vec{p})}\rangle$ can be computed in polynomial-time. Therefore, the valuations of $\vec{\mu}$ which correspond to the 1-valued bits of the binary representation of $|\langle \tau_{\vec{a}} \mid \vec{a} \in A^{\mathrm{type}(\vec{p})}\rangle|$ can be found in polynomial time, and the set of those valuations form $\varphi(\mathfrak{A})$. The case where free variables beyond $\vec{p},\vec{s},\vec{t}$ occur within ψ can be dealt with using the remark above (Eq. (3.1)). □

Finally, we show that $\mathsf{FP}+\mathsf{ord}$ can express the rk operator in its uniform definition:

Theorem 3.22. *The generalised quantifier Q_{rank^*} is $\mathsf{FP}+\mathsf{ord}$ definable.*

Proof. Using Proposition 1.38, it is only left to show that $\mathsf{FP}+\mathsf{ord}$ expresses the rank^* query on $\mathbf{LMap}^*$. Consider a structure $\mathfrak{A} \in \mathbf{LMap}^*$. Recall that such a structure consists of two relations $I^{\mathfrak{A}}, J^{\mathfrak{A}} \subseteq A$ and a binary numeric function $M^{\mathfrak{A}} : A \times A \to A^<$, together with a numerical value $p^{\mathfrak{A}}$. For any $(a,b) \in I^{\mathfrak{A}} \times J^{\mathfrak{A}}$, $M^{\mathfrak{A}}(a,b) \mod p^{\mathfrak{A}}$ is a coefficient of the matrix $\mathcal{M} \in \mathbb{F}_{p^{\mathfrak{A}}}^{I^{\mathfrak{A}}\times J^{\mathfrak{A}}}$, and our aim is to define the rank of $\mathcal{M}$.

From now, we fix the structure $\mathfrak{A}$ and omit superscripts when mentioning I, J, M and p. Recall that $\mathsf{rank}_p(\mathcal{M})$ is the dimension of the image of $\mathcal{M}$, that is, $\mathrm{Im}_{\mathbb{F}_p}(\mathcal{M}) = \{\mathcal{M}\cdot X \mid X \in \mathbb{F}_p^I\}$ and

$$\mathsf{rank}_p(\mathcal{M}) = \log_p \left|\mathrm{Im}_{\mathbb{F}_p}(\mathcal{M})\right|$$

Since the base p logarithm is a P-computable arithmetic function, it can be defined in FPC, and it is only left to show that $|\mathrm{Im}_{\mathbb{F}_p}(\mathcal{M})|$ can be defined in $\mathsf{FP}+\mathsf{ord}$. As $\mathrm{Im}_{\mathbb{F}_p}(\mathcal{M})$ is a group (for the addition of vectors), we can use the ord operator to compute its order given a generating set, and we now show how to define such a generating set.

Consider the family $(E_x^v)_{x\in I, v\in\mathbb{F}_p}$ of vectors of $\mathbb{F}_p^I$ defined by:

$$E_x^v(x') := \begin{cases} 0_{\mathbb{F}_p} & \text{if } x \neq x' \\ v & \text{otherwise} \end{cases}$$

Then, $\mathrm{Im}_{\mathbb{F}_p}(\mathcal{M})$ is generated by $\{\mathcal{M}\cdot E_x^v, x \in I, v \in \mathbb{F}_p\}$. Indeed, consider $Y \in \mathrm{Im}_{\mathbb{F}_p}(M)$. Then, there is an $X \in \mathbb{F}_p^I$ such that $\mathcal{M}\cdot X = Y$. Since $X = \sum_{x\in I} E_i^{X(x)}$:

$$Y = \sum_{x\in I} \mathcal{M}\cdot E_x^{X(x)}$$

We must now show that this group can be represented as a permutation group in FPC. First, we represent $v \in \mathbb{F}_p$ as the numerical permutation

$$f_v(w) := \begin{cases} w & \text{if } w \geq p \\ w + v \mod p & \text{otherwise} \end{cases}$$

$v \mapsto f_v$ defines a morphism from $\mathbb{F}_p$ to $\mathrm{Sym}(A^<)$, and can obviously be defined in FPC. In the same way, we define a morphism from $\mathbb{F}_p^J$ to $\mathrm{Sym}(J \times A^<)$:

$$f_J(B)(y, w) := (y, f_{B(y)}(w))$$

While elements $\mathbb{F}_p^J$ cannot be represented as first-order entities, we can still define, for $x \in I$ and $v \in \mathbb{F}_p$, the f_J image of $\mathcal{M} \cdot E_x^v$:

$$\psi(x, v, y_s, w_s, y_t, w_t) := \begin{cases} w_t \geq p \wedge w_s = w_t \wedge y_s = y_t \\ \vee \begin{cases} w_s < p \wedge w_t < p \\ \wedge \begin{cases} w_s + (M(x, y_s) \cdot v) = w_t \mod p \\ y_s = y_t \end{cases} \end{cases} \end{cases}$$

$\{\mathrm{perm}(\psi(\mathfrak{A}, a, \lambda)), a \in A, \lambda \in A^<\}$ is a generating set for a group isomorphic to $\mathrm{Im}_{\mathbb{A}}(M)$, which concludes our proof. □

Remark 3.23. Let us mention that this proof actually generalises to broader algebraic structures than fields: for any ring $\mathcal{R}$, a set of equations over this ring can be seen as a matrix $\mathcal{M} \subseteq \mathcal{R}^{I \times J}$ for some index-sets I, J. As long as $|\mathcal{R}| \leq |A^k|$ for some k, and the addition and multiplication in $\mathcal{R}$ are provided[3], we can construct a generating set for the additive group $\mathrm{Im}_{\mathcal{R}}(\mathcal{M}) := \{\mathcal{M} \cdot X, X \in \mathcal{R}^I\}$.

This bears some importance as a further candidate logic for P has been introduced, which considers an operator allowing to check, given such a matrix $\mathcal{M}$ and a tuple $Y \in \mathcal{R}^J$, whether there is a tuple $X \in \mathcal{R}^I$ such that $\mathcal{M} \cdot X = Y$. This is the *Ring Equation Satisfiability* operator (or *RES* for short).

Since a generating set for $\mathrm{Im}_{\mathcal{R}}(\mathcal{M})$ can, under the assumptions given above, be defined in FP+ord, we can check whether $Y \in \mathrm{Im}_{\mathcal{R}}(\mathcal{M})$ using the membership operator defined in Lemma 3.2, which shows that FP + ord is also at least as expressive as FP + RES.

Having shown that FP + ord $\geq$ FP + rk, the next chapter is devoted to the proof that this inequality is strict.

[3]That is, the graph of those operations are definable

Chapter 4

$\mathsf{FP} + \mathsf{rk} < \mathsf{FP} + \mathsf{ord}$

This chapter is devoted to the proof that the ord operator is strictly more expressive than the rk operator. We have shown in Chapter 3 that $\mathsf{FP} + \mathsf{rk} \leq \mathsf{FP} + \mathsf{ord}$, so that it is only left to exhibit a property inexpressible in $\mathsf{FP} + \mathsf{rk}$ that is definable in $\mathsf{FP} + \mathsf{ord}$.

Recently, Lichter [Lic23] provided such a property separating $\mathsf{FP} + \mathsf{rk}$ from CPT (although whether $\mathsf{FP} + \mathsf{rk} \leq \mathsf{CPT}$ remains unknown). More precisely, Lichter exhibits a class of structures $\mathcal{K}$ and a property $\mathcal{P} \subseteq \mathcal{K}$ such that no $\mathsf{FP} + \mathsf{rk}$ formula expresses $\mathcal{P}$, while $\mathcal{P}$ is CPT-definable.

The CPT-definability of $\mathcal{P}$ stems from the fact that $\mathcal{P}$ is P-computable, and that CPT canonises structures in $\mathcal{K}$, as discussed under Definition 1.43. The ability of CPT to canonise structures in $\mathcal{K}$ is a direct consequence of the fact that those structures have *definable abelian colours*, a notion that we will make precise in the second section of this chapter. The fact that structures with abelian colours can be canonised in CPT was proved by Pakusa in his PhD dissertation [Pak15].

The study of structures with abelian colours was motivated by the Cai-Fürer-Immerman construction (CFI for short), first used in [CFI92] to separate FPC from P. Since their introduction, CFI-structures have been generalised on several occasions [Hol10; Tor04], and those generalised versions were used in [GP19] to separate the first version of the rank operator from P, motivating the introduction of the uniform rk operator under study here. It turns out that all extensions of CFI-structures have definable abelian colours, and Lichter's separation structures also constitute such an extension.

In this chapter, we will show that $\mathsf{FP} + \mathsf{ord}$ also canonises structures in $\mathcal{K}$, which implies that $\mathsf{FP} + \mathsf{ord}$ captures P over $\mathcal{K}$, and thus expresses the property $\mathcal{P}$ exhibited by Lichter which is not expressible in $\mathsf{FP} + \mathsf{rk}$. In the first section of this chapter, we formalise the CFI-construction. In the second section, we show that, like CPT, $\mathsf{FP}+\mathsf{ord}$ also defines abelian colours over CFI-structures. The remainder of this chapter is then devoted to the proof that $\mathsf{FP} + \mathsf{ord}$ canonises structures with abelian colours, which yields the desired result that $\mathsf{FP}+\mathsf{rk} < \mathsf{FP}+\mathsf{ord}$. Section 4.3 introduces the canonisation algorithm, due to Babai and Luks [BL83], that we simulate withinin $\mathsf{FP}+\mathsf{ord}$. The main obstacle to this simulation is the representation of *labelling cosets*. Sections 4.4 to 4.6

gradually introduce our representation scheme for labelling cosets, and Section 4.7 concludes the proof of the canonisation of structures with abelian colours in $\mathsf{FP} + \mathsf{ord}$.

4.1 CFI structures and Abelian colour-class graphs

The simplest context in which to apply the group-theoretic framework to Graph Isomorphism is probably the case of Bounded colour-class graphs (hereafter denoted BCG), that we introduced in Section 1.2. Let us recall the definition of a k-bounded colour-class graph (k-BCG): a structure $\mathfrak{A}$ is coloured if it is equipped with a function $c : A \to \mathbb{N}$. Such a function c is called a colouring, and any $i \in \mathbb{N}$ such that $c^{-1}(i) \neq \emptyset$ is a colour. $\mathfrak{A}$ has k-bounded colour-classes if, for any colour i, the set of vertices of colour i, $c^{-1}(i)$, contains at most k elements. A *class* of structures $\mathcal{K}$ has bounded colour-classes if there exists a k, such that any $\mathfrak{A} \in \mathcal{K}$ has k-bounded colour-classes.

The isomorphism problem for k-BCG (denoted k-BCGI) plays an important role in Descriptive Complexity, as it separates FPC from P: on the one hand, we have seen in Section 1.2 that for each k, k-BCGI can be decided in polynomial time ; while there is a family of CFI-graphs — which cannot be distinguished in FPC [CFI92] — that are bounded colour-class graphs (as we will see, the original CFI-graphs have colour-class size 4).

The introduction of CFI-structures initiated a "game of cat and mouse" between stronger and stronger candidate logics for P [BGS97; Daw+09; GP19; GH98], and generalisations of the CFI-constructions witnessing the inability of those logics to capture P [Tor04; Hol10; GP19; Lic23]. To the best of our knowledge, none of those logics canonise, or even define the isomorphism of k-bounded colour-class graphs, for $k \geq 5$. This hints at the fact that the aforementioned generalisations of the CFI-construction increase the difficulty of the isomorphism problem in a somehow independent fashion.

Indeed, we will see in the following subsection that those new CFI-constructions do not have bounded colour-classes, and that the P-computability of the associated query relies on a reduction to the solvability of systems of equations. In the second subsection, we introduce the notion of abelian colours, which was used to show that $\mathsf{CPT} \not\leq \mathsf{FP} + \mathsf{rk}$, or, more precisely, that the property separating $\mathsf{FP} + \mathsf{rk}$ from P is expressible in CPT.

The evolution of the Cai-Fürer-Immerman construction

We now provide a formal definition of *CFI*-graphs. We start by presenting the original CFI construction from [CFI92]. However, we will describe it in a fashion that will ease the introduction of the generalisations. Our presentation is thus inspired from [GP19].

Given a simple graph $G = (V, E)$, and $v \in V$, we denote $E(v)$ the set of neighbours of v. Fix an *ordered* simple connected graph $\mathcal{G} = (V, \leq, E)$. For each function $d : V \to \{0, 1\}$, we define a graph $\mathsf{CFI}(\mathcal{G}, d)$ by replacing each vertex $v \in V$ by a *vertex gadget* $\mathrm{Gadget}(v)$, each composed of:

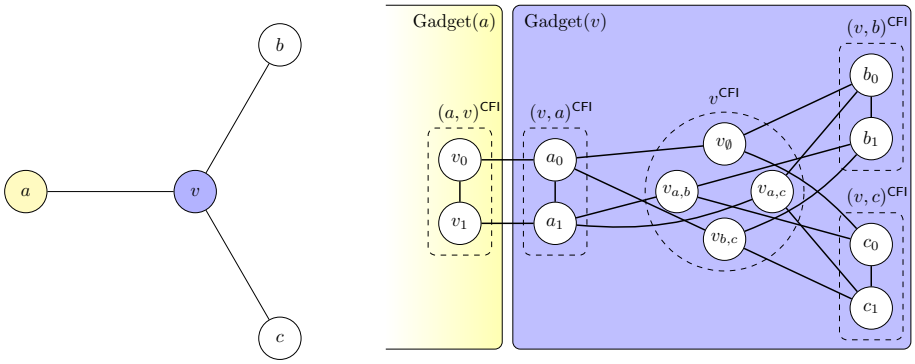


Figure 4.1: The CFI-construction depicted on a vertex v with three neighbours a, b, c. The neighbourhood of v in the base graph is on the left, the right-hand side depicts the corresponding part of $\mathsf{CFI}(\mathcal{G}, d)$, for any d with $d(v) = 0$.

- an *inner sub-gadget* $v_d^{\mathsf{CFI}} := \{(v, f), f : E(v) \to \{0,1\}, \sum_{w \in E(v)} f(w) = d(v) \mod 2\}$. That is, v_d^{CFI} contains a vertex for each subset of the neighbourhood of v whose cardinality parity is equal to $d(v)$
- for each neighbour w of v, an *outer sub-gadget* $(v, w)^{\mathsf{CFI}}$, made of two adjacent vertices $(v, w)_0$ and $(v, w)_1$,
- for each $(v, f) \in v_d^{\mathsf{CFI}}$, and $w \in E(v)$, (v, f) is adjacent to $(v, w)_{f(w)}$.

Finally, for any edge $(v, w) \in E$ (recall that $\mathcal{G}$ is a simple graph), there are edges in $\mathsf{CFI}(\mathcal{G}, d)$ between $(v, w)_0 \in (v, w)^{\mathsf{CFI}} \subseteq \mathrm{Gadget}(v)$ and $(w, v)_0 \in (w, v)^{\mathsf{CFI}} \subseteq \mathrm{Gadget}(w)$ and between $(v, w)_1$ and $(w, v)_1$.

Moreover, $\mathsf{CFI}(\mathcal{G}, d)$ is equipped with a total pre-order derived from the linear order on $\mathcal{G}$: given x, y two vertices of $\mathsf{CFI}(\mathcal{G}, d)$, if $x \in \mathrm{Gadget}(u), y \in \mathrm{Gadget}(v)$ and $u \leq v$, $x \preceq y$. Note that this pre-order's equivalence classes are exactly the gadgets of $\mathsf{CFI}(\mathcal{G}, d)$, and that it can be refined into a pre-order $\preceq_f$ whose equivalence classes are the *subgadgets* of $\mathsf{CFI}(\mathcal{G}, d)$: given $x, y \in \mathrm{Gadget}(u)$, set $x \preceq_f y$ if $x \in u^{\mathsf{CFI}}$, or if $x \in (u, a)^{\mathsf{CFI}}$ and $y \in (u, b)^{\mathsf{CFI}}$ with $a \leq b$. On any CFI-structure, the finer ordering is clearly FO-definable from the coarser one, so we will assume both to be provided interchangeably.

An illustration of this construction is given in Fig. 4.1. For any two functions $d, d' : V \to \{0,1\}$, $\mathsf{CFI}(\mathcal{G}, d) \simeq \mathsf{CFI}(\mathcal{G}, d')$ if and only if $\sum_{v \in V} d(v) = \sum_{v \in V} d'(v) \mod 2$. In other words, for any graph $\mathcal{G}$, up to isomorphism, there are only two structures of the form $\mathsf{CFI}(\mathcal{G}, d)$, which we denote $\mathsf{CFI}(\mathcal{G}, \mathbf{0})$ and $\mathsf{CFI}(\mathcal{G}, \mathbf{1})$.

For $\mathbb{G}$ a class of ordered, simple graphs, let $\mathcal{K}_{\mathbb{G}}$ be the class of CFI-graphs over $\mathbb{G}$, i.e.

$$\mathcal{K}_{\mathbb{G}} := \{\mathsf{CFI}(\mathcal{G}, d) \mid \mathcal{G} \in \mathbb{G}, d : V_{\mathcal{G}} \to \{0,1\}\}$$

and $\mathcal{P}_{\mathbb{G}} \subseteq \mathcal{K}_{\mathbb{G}}$ be the class of *even* CFI-graphs over $\mathbb{G}$, i.e.

$$\mathcal{P}_{\mathbb{G}} := \left\{ \mathsf{CFI}(\mathcal{G}, d) \;\middle|\; \mathcal{G} \in \mathbb{G}, d : V_{\mathcal{G}} \to \{0,1\}, \sum_{v \in V_{\mathcal{G}}} d(v) = 0 \right\}$$

Note that, given $\mathfrak{A} \in \mathcal{K}_{\mathbb{G}}$, the unique ordered graph $\mathcal{G} \in \mathbb{G}$ such that $\mathfrak{A} \simeq \mathsf{CFI}(\mathcal{G}, d)$ for some $d : V_{\mathcal{G}} \to \{0, 1\}$ is easily FO-definable from $\mathfrak{A}$. Moreover, given an ordered graph $\mathcal{G}$, there is a Turing machine that outputs an encoding of $\mathsf{CFI}(\mathcal{G}, \mathbf{0})$ in time polynomially bounded by $|\mathsf{CFI}(\mathcal{G}, \mathbf{0})|$. Therefore, deciding $\mathcal{P}_{\mathbb{G}}$ reduces to isomorphism testing over $\mathcal{K}_{\mathbb{G}}$.

Cai, Fürer and Immerman showed that, for some suitable class of graphs $\mathbb{G}$, no FPC-formula defines the property $\mathcal{P}_{\mathbb{G}}$, and that such a class $\mathbb{G}$ can be chosen to contain only graphs of degree three. In such a case, $\mathcal{K}_{\mathbb{G}}$ is a class of graphs of 4-bounded colour-class graphs (for the colouring induced by $\preceq_f$). As we have seen in Section 1.2, isomorphism of bounded colour-class graphs can be decided in polynomial time, which, together with the aforementioned reduction from $\mathcal{P}_{\mathbb{G}}$ to isomorphism testing over $\mathcal{K}_{\mathbb{G}}$, yields a polynomial-time algorithm to decide $\mathcal{P}_{\mathbb{G}}$, thus separating FPC from P.

While the above is an accurate depiction of Cai, Fürer and Immerman's proof, further study of those structures have shown that the polynomial-time computability of $\mathcal{P}_{\mathbb{G}}$ holds for any class of graphs $\mathbb{G}$, as it reduces to the solvability of a system of equations over $\mathbb{Z}_2$. This fact, already hinted at in [ABD07], is proved in [Daw+09, Theorem III.8], where it motivated the introduction of $\mathsf{FP} + \mathsf{rk}$ as a new candidate logic for P. Intuitively, $\mathsf{CFI}(\mathcal{G}, d)$ replaces each edge in $\mathcal{G}$ by a connection of outer gadgets that can either be "straight" or "twisted", in such a way that the number of twisted connections of v has parity $d(v)$. It is well-known that Gaussian elimination — the method enabling the decision of the solvability of systems of equations over fields — can be generalised to decide the solvability of systems of equation over any abelian finite group in polynomial time. While this result is folklore, a proof is given in [GR02], where it is also shown that over any non-abelian finite groups, this solvability problem is NP-complete. Interestingly, the CFI construction can also be generalised as to encode system of equations over any finite abelian group. Here, we only depict the encoding of equations over cyclic groups (of the form $\mathbb{Z}_m$ for some $m \geq 1$), as they are sufficient to the separation result we aim to present:

- Let $d : V \to \mathbb{Z}_m$
- The inner gadget of x is now

$$x_d^{\mathsf{CFI}} := \left\{ (x, f) \;\middle|\; f : E(x) \to \mathbb{Z}_m, \sum_{y \in E(x)} f(y) = d(x) \right\}$$

- The outer gadget of (x, y) is now

$$(x, y)^{\mathsf{CFI}} := \{(x, y)_\lambda \mid \lambda \in \mathbb{Z}_m\}$$

  with directed edges from $(x, y)_\lambda$ to $(x, y)_{\lambda+1}$, for $\lambda \in \mathbb{Z}_m$.

- There is an edge between (x, f) and $(x, y)_{f(y)}$ (in both directions)
- There is an edge between $(x, y)_\lambda$ and $(y, x)_{-\lambda}$ (in both directions).

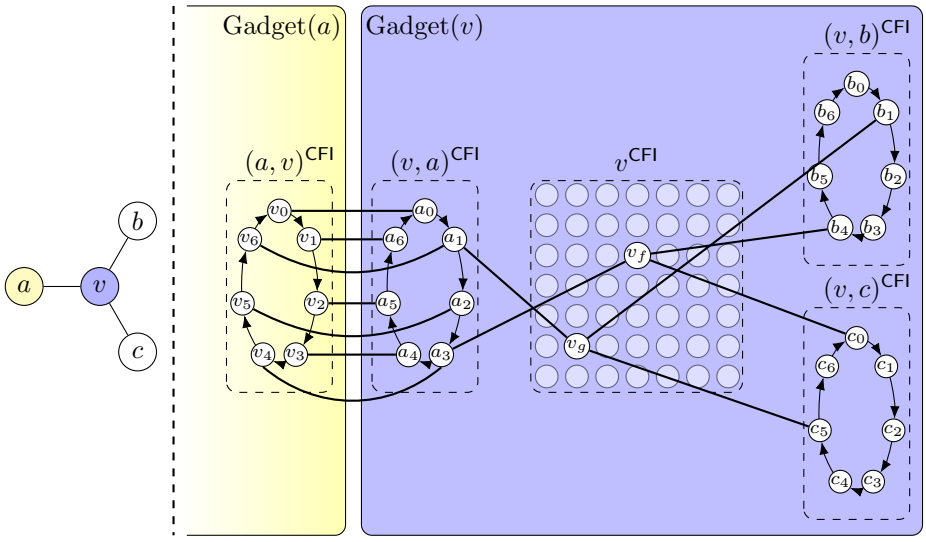


Figure 4.2: The $\mathsf{CFI}_{\mathbb{Z}_7}$ construction depicted on a vertex v with three neighbours a, b, c (assuming $d(v) = 0$). For clarity, not all edges are represented: we have omitted all edges between inner-nodes and outer-nodes, except for two inner-nodes v_f and v_g, where $f(a) = 3$, $f(b) = 4$, $f(c) = 0$, $g(a) = g(b) = 1$ and $g(c) = 5$.

The resulting structure is denoted $\mathsf{CFI}_{\mathbb{Z}_m}(\mathcal{G}, d)$. Note that, when $m > 2$, $\mathsf{CFI}_{\mathbb{Z}_m}(\mathcal{G}, d)$ is a directed graph. This construction is depicted in Fig. 4.2. Note that in this example, v^{CFI} contains $49 = 7^2$ vertices.

This generalisation of CFI-structures can be traced back to Toran [Tor04] and Holm [Hol10], and was used in [GP19] to show that the non-uniform version of rank logic does not capture P. Note that once again, if $\mathbb{G}$ is a class of graphs of degree bounded by k, each structure in

$$\mathcal{K}_{\mathbb{Z}_m,\mathbb{G}} := \{\mathsf{CFI}_{\mathbb{Z}_m}(\mathcal{G}, d) \mid \mathcal{G} \in \mathbb{G}, d : V_{\mathcal{G}} \to \mathbb{Z}_m\}$$

has colour-class size bounded by m^{k-1} (for the colouring induced by $\preceq_f$). In Fig. 4.2, this bound is reached for v^{CFI}, which has $49 = 7^2$ vertices.

Those two separation results (for FPC [CFI92] and the non-uniform version of rank logic [GP19]) follow the same line of reasoning: exhibit a class of structures $\mathcal{K}$ such that, on the one hand, $\mathsf{GI}_{\mathcal{K}} \in \mathsf{P}$ (because $\mathcal{K}$ has bounded colour-class size), while no sentence of the logic at hand can express isomorphism over $\mathcal{K}$.

Recently, Lichter introduced another class of CFI-structures $\mathcal{K}$ over finite rings $\mathbb{Z}_{2^i}$, such that CPT captures P over $\mathcal{K}$, while $\mathsf{FP} + \mathsf{rk}$ does not. The class of structures defined by Lichter is in line with the general evolution of CFI constructions, in that it involves more general algebraic structures in the construction — namely rings instead of fields. However, the colouring on those structures are no longer bounded, and for two reasons: first, the class of base graphs $\mathbb{G}$ no longer has bounded degree; and also because the CFI structures under consideration are built on arbitrarily large rings. That is, for some suitable family of base graphs $\mathbb{G}$, Lichter considers [Lic23, Definition 5.19]

$$\mathcal{K} := \{\mathsf{CFI}_{\mathbb{Z}_{2^i}}(\mathcal{G}, d) \mid \mathcal{G} \in \mathbb{G}, i \in \mathbb{N}, d : V_{\mathcal{G}} \to 2^i\} \tag{4.1}$$

that is, the class of structures contains *all* CFI constructions over $\mathbb{G}$, for *each* ring of the form $\mathbb{Z}_{2^i}$. Lichter shows that the CFI-query (i.e. to identify the graphs $\mathsf{CFI}_{\mathbb{Z}_{2^i}}(\mathcal{G}, d)$ in $\mathcal{K}$ such that $\sum_{v \in V_\mathcal{G}} d(v) = 0$) is not definable in $\mathsf{FP} + \mathsf{rk}$ [Lic23, Theorem 10.4]. The proof of this difficult result is outside of the scope of this thesis, and so are the combinatorial properties on the class of base graphs $\mathbb{G}$ needed to obtain such a result.

On the other hand, we will now show how this result actually separates $\mathsf{FP}+\mathsf{rk}$, not only from P, but also from CPT, another candidate logic for P. Before delving into this, we point out that our presentation of the CFI-construction differs from the one used in Lichter's article, which does not use both inner and outer gadgets. However, this is merely a presentation difference, and it is easily shown that there are FO-definable interpretations between both definitions of the CFI-construction.

Abelian colour-class graphs

The pertinence of the CFI construction relies on the fact that the resulting structures can be distinguished in polynomial time (and in the case of Lichter's result, also in *Choiceless* polynomial time), which in turn implies that the CFI query can be decided in polynomial time.

We have just mentioned that, for all the CFI-constructions, the CFI-query is polynomial-time computable because it reduces to the solvability of systems of equations over some finite abelian group: $\mathbb{Z}_2$ in the case of the original CFI-construction from [CFI92], $\mathbb{Z}_p$ for some prime p in the case of its generalisation in [GP19], and finally $\mathbb{Z}_{2^i}$ in [Lic23]. Unlike the original proof of Cai-Fürer-Immerman, which revolved around bounded colour-class graphs, this way to prove the polynomial-time computability of the CFI-query seems only remotely connected to the group-theoretic framework to Graph Isomorphism.

To show that the CFI-query as considered by Lichter is definable in CPT, a different notion has been used: *abelian colours*. This notion was introduced and studied in [Pak15], to define a general class of structures containing all the queries known to separate FPC from P. In this work, Pakusa shows that CPT canonises structures with abelian colours. In [Lic23], Lichter demonstrated that the class of structures $\mathcal{K}$ used to separate $\mathsf{FP} + \mathsf{rk}$ from P has abelian colours, and thus $\mathcal{K}$ separates $\mathsf{FP} + \mathsf{rk}$ from CPT. The original definition of structures with abelian colours was tailored for CPT, so we now provide an adaptation of this definition in the context of fixed-point logics:

Definition 4.1 (adaptation of [Pak15, Definition 6.1])**.** A Σ-*structure with Abelian colours* is a $\Sigma \cup \{\prec^{(2)}, \Phi^{(4)}\}$-structure $\mathfrak{A}$, where $\text{type}(\Phi) = \text{number}^2\text{element}^2$, such that:

- $\prec$ is a total pre-order on A. From now on, let m be the number of equivalence classes of $\prec$; and A_i be the i-th equivalence class.
- for any $i < m$, $j < |A_i|$, $\Phi\ (\mathfrak{A}, i, j)$ is the graph of a permutation over A_i;
- for any $i < m$, $\Gamma_i := \{\sigma, \exists \nu, \Phi\ (\mathfrak{A}, i, \nu) = \text{graph}(\sigma)\}$ is an abelian, transitive permutation group over A_i.

That is, $\mathfrak{A}$ has abelian colours if it is equipped with a total pre-order $\prec$ *and* a relation Φ that explicitly enumerates a family of abelian transitive groups acting on each of the colour-classes defined by $\prec$. Even more so, the type of Φ implies that each of those groups is *linearly ordered*. This ordering of the groups acting on each colour-class is crucial to the canonisation of those structures in FP + ord. Given a Σ-structure $\mathfrak{A}$, we call such a couple of relations $(\prec^{\mathfrak{A}}, \Phi^{\mathfrak{A}})$ an *abelian colouring* of $\mathfrak{A}$. Before studying how this notion relates to CFI-structures, let us make a few remarks on this definition.

First, notice that for each i, we expect $j \mapsto \Phi(\mathfrak{A}, i, j)$ to enumerate the whole group Γ_i. Because j can only take $|A|$ values, this requires that $|\Gamma_i| < |A|$. This always holds for abelian transitive permutation groups. Indeed, we have shown in Proposition 1.59 that in such a case Γ_i is regular, and the following lemma shows that this implies $|\Gamma_i| = |A_i|$.

Lemma 4.2. *Let Γ be an abelian group acting faithfully and transitively on some set X. Then, there is a bijection between Γ and X.*

Proof. Fix $x \in X$, and let $\phi : \Gamma \to X$ be the function that maps γ to $\gamma(x)$. Let us show that ϕ is bijective. The surjectivity of ϕ is a direct consequence of the transitivity of the action of Γ on X. Now, suppose that for some γ, γ', $\gamma(x) = \gamma'(x)$. We have shown in Proposition 1.59 that the action of G on X is regular, which implies that $\gamma = \gamma'$. □

Let us also remark that the connection between the group theoretic framework to graph isomorphism and the notion of abelian colours seems quite tenuous, compared to the case of bounded colour-class graphs. In particular, notice that Definition 4.1 does not explicitely require any relationship between the group Γ_i acting on A_i defined by Φ, and the automorphism group of the induced substructure of $\mathfrak{A}$ on A_i. Recall that, in the case of BCGI, the fact that each colour-class was bounded allowed the definition of a group G (Eq. (1.3)), containing $\mathrm{Aut}(\mathfrak{A})$ as a subgroup, and with sufficiently restricted structure to allow the definition of a chain of subgroups witnessing the accessibility of $\mathrm{Aut}(\mathfrak{A})$ from G (Eq. (1.4)). We will see in the following sections that abelian colours enable the same procedure, with the following difference: while in the case of bounded colour-class graphs, this group was defined as the direct product of the automorphism groups of the substructures induced by each colour-class, here, we will use the group $\Gamma := \prod_{i=1}^{m} \Gamma_i$:

Lemma 4.3. *Let $\mathfrak{A}$ be a structure with abelian colours, and let (Γ_i) be the family of permutation groups defined by the abelian colouring on $\mathfrak{A}$. Then,* $\mathrm{Aut}(\mathfrak{A}) \leq \prod_{i=1}^{m} \Gamma_i$.

Proof. Let $\tau \in \mathrm{Aut}(\mathfrak{A})$. Because τ must preserve the colouring c defined by $\prec$, it must stabilise setwise each colour-class. Therefore, we must only show that, for any i, the restriction σ of τ to the i-th colour-class is in Γ_i. Let $x \in c^{-1}(i)$. By transitivity of Γ_i, there is an element $g \in \Gamma_i$ such that $g(x) = \sigma(x)$. We now show that $\sigma = g$, or, equivalently, that $\sigma^{-1} \cdot g = \mathrm{Id}$.

Let $y \in c^{-1}(i)$. By transitivity of Γ_i, there is an element $h \in \Gamma_i$ such that $h(y) = x$. Because Γ_i is ordered, h must be fixed by conjugation by any automorphism of $\mathfrak{A}$: indeed, there is some j such that $h = \mathrm{perm}(\Phi\ (\mathfrak{A}, i, j))$, and for all $s, t \in c^{-1}(i)$ and $\xi \in \mathrm{Aut}(\mathfrak{A})$,

$$\begin{aligned} h(s) = t &\iff \mathfrak{A} \models \Phi\ (i, j, s, t) \\ &\iff \mathfrak{A} \models \Phi\ (\xi(i), \xi(j), \xi(s), \xi(t)) \\ &\iff \mathfrak{A} \models \Phi\ (i, j, \xi(s), \xi(t)) \\ &\iff h \cdot \xi(s) = \xi(t) \\ &\iff h^{\xi}(s) = t \end{aligned}$$

In particular, $h^{\sigma^{-1}} = h$. Because Γ_i is abelian, we also have $h^g = h$. Thus $x = h(y) = h^{(\sigma^{-1}g)}$, which yields:

$$\begin{aligned} & x = h^{\sigma^{-1}g}(y) \\ \implies & x = g^{-1}\sigma h \sigma^{-1} g(y) \\ \implies & \sigma^{-1}g(x) = h\sigma^{-1}g(y) \\ \implies & x = h\sigma^{-1}g(y) \\ \implies & y = \sigma^{-1}g(y) \end{aligned}$$

Therefore, $\sigma^{-1}g = \mathrm{Id}$, which concludes the proof. □

Actually, we have shown something stronger than Lemma 4.3:

Lemma 4.4. *Let $\mathfrak{A}$ be a structure with abelian colours, m be the number of colours of $\mathfrak{A}$, and for $i \leq m$, let A_i be the i-th colour class of A, $\mathfrak{A}_i$ the substructure induced by $\mathfrak{A}$ on A_i, and Γ_i be the permutation group defined by Φ on A_i. For any $i \leq m$, $\mathrm{Aut}(\mathfrak{A}_i) \leq \Gamma_i$.*

We will see in Lemma 4.20 that, in FPC, we can actually expect Γ_i and $\mathrm{Aut}(\mathfrak{A}_i)$ to be equal.

In other words, in the case of BCG, the colouring implicitly defined a group containing $\mathrm{Aut}(\mathfrak{A})$ sufficiently restricted to apply group-theoretic arguments, while in the case of abelian colours, the definition of such a group is "deferred" to the definition of Φ.

Definition 4.5. Let $\mathcal{K}$ be a class of Σ-structures. $\mathcal{L} \geq \mathsf{FPC}$ *defines abelian colours on* $\mathcal{K}$ if there are $\mathcal{L}[\Sigma]$ formulae $\varphi_{\prec}, \varphi_{\Phi}$ such that for any $\mathfrak{A} \in \mathcal{K}$, $(\mathfrak{A}, \varphi_{\prec}(\mathfrak{A}), \varphi_{\Phi}(\mathfrak{A}))$ is a Σ-structure with abelian colours.

Pakusa showed in his PhD dissertation [Pak15] that CPT canonises structures with abelian colours. This has the immediate implication that CPT canonises all structures on which it can define abelian colours:

Lemma 4.6. *Let $\mathcal{L} \geq \mathsf{FPC}$ be a logic that canonises structures with abelian colours. Let $\mathcal{K}$ be a class of Σ-structures, and suppose that $\mathcal{L}$ defines abelian colours on $\mathcal{K}$. Then, $\mathcal{L}$ canonises structures in $\mathcal{K}$ and thus captures P over $\mathcal{K}$.*

Proof. Consider $\mathcal{I}$, the $\mathcal{L}$-definable interpretation that witnesses the canonisability (in the sense of Definition 1.43) of Σ-structures with abelian colours, replace each occurence of $\prec$ and Φ by formulae $\varphi_\prec$ and φ_Φ respectively, where $\varphi_\prec$ and φ_Φ are the $\mathcal{L}$-formulae witnessing the $\mathcal{L}$-definability on abelian colours on $\mathcal{K}$. □

Let $\mathcal{K}$ be the class of structures separating $\mathsf{FP}+\mathsf{rk}$ from P introduced above. The facts that CPT defines abelian colours on $\mathcal{K}$ [Lic23, Theorem 10.4], and that CPT canonises structures with abelian colours [Pak15], imply that CPT captures P over $\mathcal{K}$, which concludes the proof that $\mathsf{CPT} \not\leq \mathsf{FP}+\mathsf{rk}$. We aim to translate this proof in the context of $\mathsf{FP}+\mathsf{ord}$, thus proving, together with Theorem 3.22, that $\mathsf{FP}+\mathsf{rk} < \mathsf{FP}+\mathsf{ord}$. In the rest of this section, we show that FPC defines abelian colourings on all CFI-structures, the remaining sections of this chapter being devoted to the proof that $\mathsf{FP}+\mathsf{ord}$ canonises structures with abelian colours.

4.2 FPC defines abelian colours on CFI-structures

Recall that, according to Lemma 4.3, an abelian colouring provides a family of abelian groups, each acting transitively on a subset of the domain of the structure, such that the product of those groups contains the automorphism group of the structure. The existence of such a family of groups (and thus of an abelian colouring) heavily restricts the automorphism group of the structure. Let us thus start by studying the automorphims of CFI-structures.

In the following, we fix a base graph $\mathcal{G} = (V, \leq, E)$ and $d : V \to \mathbb{Z}_k$. First, remark that there are actually two ways to induce a colouring[1] on $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$ from the ordering of $\mathcal{G}$: first, there is the pre-order provided in $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$, where $x \prec y$ iff $x \in \mathrm{Gadget}(v), y \in \mathrm{Gadget}(w)$ and $v < w$, i.e. each gadget constitutes a colour-class. A finer colouring can be obtained by using different colours for all subgadgets, by additionally setting $v^{\mathsf{CFI}} \prec (v,w)^{\mathsf{CFI}}$, $w^{\mathsf{CFI}} \prec (w,v)^{\mathsf{CFI}}$ and $(v,w)^{\mathsf{CFI}} \prec (v,w')^{\mathsf{CFI}}$ for all suitable $v, w, w' \in V_{\mathcal{G}}$ with $v < w < w'$. We refer to those respectively as *the coarse* and *fine colouring of* $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$ (for any $d : V \to \mathbb{Z}_d$). Note that the finer colouring is easily definable in FPC on any structure $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$. This implies that any automorphism of $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$ must stabilise the *fine* colouring. Moreover, any outer gadget $(v,w)^{\mathsf{CFI}_{\mathbb{Z}_k}}$ has an abelian, transitive group of automorphisms:

Lemma 4.7. *Consider C_k, the directed cyclic graph with k vertices.* $\mathrm{Aut}(C_k)$ *is isomorphic to $\mathbb{Z}_k$ and thus abelian, and acts transitively on the vertices of C_k.*

Proof. Let $x_0, \dots x_{k-1}$ be the vertices of C_k, indexed in such a way that $(x_i, x_{i+1}) \in E_{C_k}$ for all $i < k$ and $(x_{k-1}, x_0) \in E_{C_k}$. It is easy to see that $\sigma := (x_0\ x_1\ \dots\ x_{k-1})$

[1] Recall that, in the logical context, a colouring is given by a pre-order.

is an automorphism of C_k, and it suffices to show that $\mathrm{Aut}(C_k) = \langle\sigma\rangle$. Consider $\tau \in \mathrm{Aut}(C_k)$, and let λ be such that $\tau(x_0) = x_\lambda$. Because x_i is the unique vertex at distance i from x_0, we must have $\tau(x_i) = x_{\lambda+i \mod k}$, and thus, $\tau = \sigma^\lambda$. □

A final key property of CFI gadgets is that the action of an automorphism on an inner sub-gadget is entirely determined by the action of that automorphism on the related outer sub-gadgets:

Lemma 4.8. *Let (σ_w) be a family of permutations such that, for all $w \in E(v)$, $\sigma_w \in \mathrm{Aut}((v,w)^{\mathsf{CFI}})$. There is at most one automorphism ρ of $\mathrm{Gadget}(v)$ such that, for all w, $\rho_{\restriction (v,w)^{\mathsf{CFI}}} = \sigma_w$.*

Proof. Consider such a family of permutations (σ_w). Then, for any $x \in v^{\mathsf{CFI}}$, there is a unique tuple $(x_w) \in \prod_{w\in E(v)}(v,w)^{\mathsf{CFI}}$ such that, for all $w \in E(v)$, x and x_w are adjacent in $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$. Therefore, any $\rho \in \mathrm{Aut}(\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d))$ such that $\rho_{\restriction (v,w)^{\mathsf{CFI}}} = \sigma_w$ must be such that, for all $x \in v^{\mathsf{CFI}}$, ρ maps x to the unique y (if such a y exists) adjacent to all $\sigma_w(x_w)$. □

Together, those two lemmas imply that $\mathrm{Aut}(\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d))$ is abelian. We can actually consider a family of abelian transitive groups acting on each subgadget. Denote $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ the group of all automorphisms of $\mathrm{Gadget}(v)$ which stabilise each subgadget, i.e.

$$\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v)) := \mathrm{Stab}_{\mathrm{Aut}(\mathrm{Gadget}(v))}\left(v^{\mathsf{CFI}}, ((v,w)^{\mathsf{CFI}})_{w\in E(v)}\right)$$

Lemma 4.9. *For any v, $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ is isomorphic to a subgroup of $\mathbb{Z}_k^{\deg_{\mathcal{G}}(v)}$, and is therefore abelian. Moreover, for any subgadget X of $\mathrm{Gadget}(v)$ (i.e. $X = v^{\mathsf{CFI}}$ or $X \in \{(v,w)^{\mathsf{CFI}}, w \in E(v)\}$), $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ acts transitively on X.*

Proof. Let $G_{v,w}$ be the projection on $(v,w)^{\mathsf{CFI}}$ of $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$.

$$\begin{aligned}\pi_{\mathrm{Outer}(v)} : \mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v)) &\to \prod_{w\in E(v)} G_{v,w}\\ \rho &\mapsto \prod_{w\in E(v)} \rho_{\restriction (v,w)^{\mathsf{CFI}}}\end{aligned}$$

is a well-defined group morphism, since $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ stabilises each subgadget. Moreover, Lemma 4.8 implies that $\pi_{\mathrm{Outer}(v)}$ is injective. Therefore (by the first isomorphism theorem), $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ is isomorphic to a subgroup of $\prod_{w\in E(v)} G_{v,w}$. By Lemma 4.7, each $G_{v,w}$ is isomorphic to $\mathbb{Z}_k$, which yields the first part of the lemma.

Concerning the second part of the lemma, note that we have already proved this in Lemma 4.7 for any $X \in \{(v,w)^{\mathsf{CFI}} \mid w \in E(v)\}$. Therefore, it remains to prove it

in the case where $X = v^{\mathsf{CFI}}$. Let $x, y \in v^{\mathsf{CFI}}$. For each $w \in E(v)$, let σ_w be the unique automorphism of $(v,w)^{\mathsf{CFI}}$ which maps the unique vertex in $(v,w)^{\mathsf{CFI}}$ adjacent to x to the unique vertex in $(v,w)^{\mathsf{CFI}}$ adjacent to y. We claim that $\sigma := \prod_{w \in E(v)} \sigma_w$ lies in the image of $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$, which indeed yields the transitivity of the action of $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ on v^{CFI}.

We now show how to extend σ to v^{CFI}. For $w \in E(v)$, let $\lambda_w \in \mathbb{Z}_k$ be the image of σ_w along the isomorphism depicted in Lemma 4.7. Note that, with those definitions, if $x = (v, f)$ where $f : E(v) \to \mathbb{Z}_k$ (as in the definition of CFI-structures), $y = (v, f')$ where $f'(w) = f(w) + \lambda_w$, and thus, $\sum_{w \in E(v)} \lambda_w = 0$. From this, it follows that for any $y = (v, g)$, there is a unique z such that the image under $\prod_{w \in E(v)} \sigma_w$ of the neighbourhood of y is the neighbourhood of z, namely $z := (v, g')$, where $g'(w) = g(w) + \lambda_w$. □

This yields the desired abelian colouring: $\varphi_\prec$ will define the fine pre-ordering of $\mathsf{CFI}_{\mathbb{Z}_k}(\mathcal{G}, d)$, and φ_Φ will define:

- On $(v,w)^{\mathsf{CFI}}$, the automorphism group of the substructure induced by $(v,w)^{\mathsf{CFI}}$
- On v^{CFI}, the projection of $\mathrm{Aut}_{\prec_f}(\mathrm{Gadget}(v))$ on v^{CFI}, i.e. the group of all automorphisms of the substructure induced by v^{CFI}, which can be extended to an automorphism of the substructure induced by $\mathrm{Gadget}(v)$.

Before we formally prove the FPC-definability of this abelian colouring, notice that this also provides an intuition as to why the definition of abelian colours does not simply require — as was the case for BCG — the groups of automorphisms of the substructures induced by each colour class to be restricted: if we consider the fine colouring, v^{CFI} is an independant set and constitutes a colour-class, hence $\mathrm{Aut}(v^{\mathsf{CFI}})$ is not abelian. In the same way with the coarse colouring, the action of $\mathrm{Aut}(\mathrm{Gadget}(v))$ on v^{CFI} is once again not abelian. One way to circumvent this is to only consider automorphisms that stabilise the partition of $\mathrm{Gadget}(v)$ into subgadgets, however the resulting action is not transitive.

We now show that FPC can define an abelian colouring on CFI-structures:

Theorem 4.10. *Let $\mathcal{K}$ be the class of structures defined in Eq.* (4.1)*.* FPC *defines abelian colours on $\mathcal{K}$.*

Proof. To ease the presentation, we fix a structure $\mathfrak{A} = (A, E, \lhd) \in \mathcal{K}$ (where E is the edge relation of the CFI-structure, and $\lhd$ the strict pre-order that induces the colouring of $\mathsf{CFI}(\mathcal{G}, d)$ into gadgets i.e. the coarse pre-order). First $\varphi_\prec$ must define the fine

colouring:

$$\varphi_{\prec}(x,y) := x \lhd y \vee \begin{cases} \neg x \lhd y \\ \neg y \lhd x \\ \varphi_{\text{outer}}(y) \\ \begin{cases} \varphi_{\text{inner}}(x) \\ \varphi_{\text{outer}}(x) \wedge \exists x', y', E(x,x') \wedge E(y,y') \wedge x' \lhd y' \end{cases} \end{cases}$$

where φ_{inner} defines the set of inner vertices:

$$\varphi_{\text{inner}}(x) := \forall z, E(x,z) \implies (\neg x \lhd z \wedge \neg z \lhd x),$$

and φ_{outer} the set of outer vertices:

$$\varphi_{\text{outer}}(x) := \neg \varphi_{\text{inner}}(x).$$

From now on, we write $x \prec y$ for $\varphi_{\prec}(x,y)$, and $x \sim y$ for $\neg x \prec y \wedge \neg y \prec x$. We now turn to the definition of φ_Φ. Recall from Definition 4.1 that Φ should be of type number^{2}element^2 and thus for all i $\varphi_\Phi(\mathfrak{A}, i)$ should enumerate an *ordered* group on the i-th equivalence class w.r.t. $\prec$. We have already defined in the previous lemma the group Γ_i that we aim to define through φ_Φ, but it remains to describe the linear-order on Γ_i. We treat the cases where the i-th colour-class corresponds to an inner subgadget or an outer subgadget separately. In the case of an outer subgadget $(v,w)^{\mathsf{CFI}}$, we have already described a canonical isomorphism between $\mathrm{Aut}((v,w)^{\mathsf{CFI}})$ and $\mathbb{Z}_k$ in Lemma 4.7, and $\mathbb{Z}_k$ can easily be ordered, thus it is only left to translate this isomorphism in FPC:

$$\varphi_\Phi^{\text{Outer}}(i,\lambda,s,t) := \begin{cases} \forall x \in A_i, \varphi_{\text{outer}}(x) \\ \lambda < k \\ s \in A_i \wedge t \in A_i \wedge \mathrm{dist}_{A_i}(s,t) = \lambda \end{cases}$$

where $(s \in A_i)$ states that s belongs to the i-th colour-class w.r.t. the colouring induced by $\varphi_\prec$. This is a relation of type (element $\cdot$ number) which is definable as follows:

$$(s \in A_i) := \left(\mathrm{ifp}_{R,x,\mu} \begin{cases} \mu = 0 \wedge \forall y, \neg y \prec x \\ \exists y, \nu, \begin{cases} R(y,\nu) \\ \mu = \nu + 1 \\ x \prec y \\ \forall z, \neg(x \prec z \wedge z \prec y) \end{cases} \end{cases} \right)(s,i)$$

and $\mathrm{dist}_{A_i}(s,t) = j$ states that there is a path of length j from s to t within A_i. This is a relation of type (element^2 $\cdot$ number^2) definable as follows:

$$(\mathrm{dist}(s,t)_{A_i} = \lambda) := \left(\mathrm{ifp}_{R,x,y,\mu} \begin{cases} \mu = 0 \wedge x = y \\ \exists z, \nu, \begin{cases} \mu = \nu + 1 \\ R(x,z,\nu) \wedge E(z,y) \\ x \in A_i \wedge y \in A_i \wedge z \in A_i \end{cases} \end{cases} \right)(s,t,i,\lambda)$$

It is now left to define the ordering on Γ_i when i is the index of the colour-class corresponding to an inner subgadget v^{CFI}. Denote $\mathcal{N}(i)$ the set of indices of all $\prec$-colour-classes adjacent to A_i (i.e. the set of indices of all classes $(v,w)^{\mathsf{CFI}}$, for w a neighbour of v in the base graph over which $\mathfrak{A}$ is constructed). $\mathcal{N}(i)$ is FPC-definable:

$$(j \in \mathcal{N}(i)) := \exists x \in A_i, \exists y \in A_j, \varphi_{\text{inner}}(x) \wedge E(x,y)$$

We have seen in Lemma 4.9 that Γ_i is isomorphic to $\{\vec{\alpha} \in \mathbb{Z}_k^{\mathcal{N}(i)} \mid \sum_{j\in\mathcal{N}(i)} \alpha_j = 0\}$, and we have already implicitly defined this isomorphism in the proof of Lemma 4.8. Since $\mathbb{Z}_k$ and $\mathcal{N}(i)$ are ordered, we can define, for any index i of an inner colour-class, a bijection between $[k^{|\mathcal{N}(i)|}]$ and $\mathbb{Z}_k^{\mathcal{N}(i)}$:

$$\varphi_{\text{enum}}(i,\lambda,j,\mu) := \exists \nu, \circledwedge \begin{cases} j \in \mathcal{N}(i) \\ \nu = (\#j' < j, j' \in \mathcal{N}(i)) \\ \mu = \lfloor \frac{\lambda}{k^\nu} \rfloor \mod k \end{cases}$$

For j the index of a colour-class adjacent to A_i, let ν_j be the index of j in $\mathcal{N}(i)$ (i.e. $\nu_j := |\{j' < j \mid j' \in \mathcal{N}(i)\}|$). $\varphi_{\text{enum}}(i,\lambda,j,\mu)$ holds if:

- There is a vector $\vec{\alpha} \in \mathbb{Z}_k^{\mathcal{N}(i)}$ such that $\lambda = \sum_{j'\in\mathcal{N}(i)} k^{\nu_{j'}}\alpha_{j'}$.
- $\alpha_j = \mu$

Given such a λ, we can easily verify that it encodes a tuple $\vec{\alpha}$ such that $\sum_{j\in\mathcal{N}(i)} \alpha_j = 0$:

$$\varphi_{\text{correct}}(i,\lambda) := \exists \kappa, \kappa\lambda = \sum_{j\in\mathcal{N}(i)} \varphi_{\text{enum}}(i,\lambda,j)$$

where $\varphi_{\text{enum}}(i,\lambda,j)$ denotes the unique μ such that $\varphi_{\text{enum}}(i,\lambda,j,\mu)$ holds. While this notation is abusive, this sum is obviously FPC definable. We are now ready to present the FPC-definition of the ordering of Γ_i:

$$\varphi_\Phi^{\text{Inner}}(i,\lambda,s,t) := \forall x, E(s,x) \implies \exists j,y,\mu,\lambda, \circledwedge \begin{cases} s \in A_i \wedge t \in A_i \wedge \varphi_{\text{inner}}(s) \\ x \in A_j \wedge y \in A_j \wedge \varphi_{\text{outer}}(x) \\ \varphi_{\text{correct}}(i,\lambda) \\ \varphi_{\text{enum}}(i,\lambda,j,\mu) \\ \varphi_\Phi^{\text{Outer}}(j,\mu,x,y) \\ E(t,y) \end{cases}$$

$\varphi_\Phi^{\text{Inner}}(i,\lambda,s,t)$ holds if λ encodes (through φ_{enum}) a tuple $\vec{\alpha} \in \mathbb{Z}_k^{\mathcal{N}(i)}$ such that:

- $\sum_{j\in\mathcal{N}(i)} \alpha_j = 0$

- for each $j \in \mathcal{N}(i)$, the (unique) neighbour x of s in A_j is mapped to y, the (unique) neighbour of t in A_j by $\gamma^j_{\alpha_w}$, where γ^j_μ is the μ-th element of the group defined by $\varphi^{\text{Outer}}_\Phi$ on A_j.

Finally,

$$\varphi_\Phi(i,\lambda,s,t) := \begin{cases} \neg s \in A_i \wedge s = t \\ \circledvee \varphi^{\text{Inner}}_\Phi(i,\lambda,s,t) \\ \varphi^{\text{Outer}}_\Phi(i,\lambda,s,t) \end{cases}$$

□

It is now left to show that $\mathsf{FP}+\mathsf{ord}$ canonises graphs with abelian colours. Before doing so, let us note that this implies that $\mathsf{FP}+\mathsf{ord}$ canonises arbitrary structures with abelian colours. In the context of CPT, obtaining such a result bore the additional complexity of dealing with unbounded relations. In the context of FPC, this is a direct adaptation of the traditional result that "Everything is a graph", a proof of which can be found in Hodges [Hod93, Theorem 5.5.1]

Lemma 4.11. *For any fixed relational signature σ, there is an FPC definable bi-interpretation between σ-structures with abelian colours and graphs with abelian colours.*

Therefore, in what follows, we only study the canonisation of *graphs* with abelian colours, which implies the canonisation of structures with abelian colours over any (fixed) signature.

4.3 Canonisation of structures with Abelian colours

Now that we have circumscribed the problem that we need to tackle, let us review its definition. As we recall the definition of graphs with abelian colours, we introduce some additional notations. A graph with abelian colours is a $\{E, \preceq, \Phi\}$-structure $\mathfrak{A}$, such that:

- $\preceq^{\mathfrak{A}}$ is a total pre-order on A. Let m be the number of equivalence classes of $\preceq^{\mathfrak{A}}$ (or colours), and denote A_i, $i \in [m]$ the i-th colour-class.
- For each $i \leq m$, and for all $\nu \leq |A_i|$, $\Phi\ (\mathfrak{A}, i, \nu)$ is the graph of a permutation of A_i.
- For each $i \leq m$, $\Gamma_i := \{\sigma \in \mathrm{Sym}(A_i), \exists \nu \leq |A_i|, \mathrm{graph}(\sigma) =_\Phi\ (\mathfrak{A}, i, \nu)\}$ is an abelian, transitive group.[2]

[2]Recall that, under the assumption that Γ_i is abelian transitive, $|\Gamma_i| = |A_i|$. As such, it is reasonable for Φ to have type number$^2 \cdot$ element^2

We now show how to canonise graphs with abelian colours. In the logical context, this entails to build a relation $E^<$ over an ordered set — here, the numeric domain, that we denote $A^<$ — isomorphic to $E^{\mathfrak{A}}$. From now on, we identify linear-orders on A with bijective functions $f : A \to A^<$.

The group-theoretic framework to graph canonisation

Fundamentally, our technique is an adaptation of Babai & Luks canonisation algorithm [BL83] which we summarise quite tersely here: in the algorithmic context, a graph X over n vertices is given as a *string* w over $\{0,1\}$, of length n^2 (that is, $w : [n^2-1] \to \{0,1\}$), where $w(n \cdot i + j) = 1$ if there is an edge between the i-th and the j-th vertices. S_n acts on those strings, mapping w to

$$\begin{aligned} w^\sigma : [n^2-1] &\to \{0,1\} \\ n \cdot i + j &\mapsto w(n \cdot \sigma^{-1}(i) + \sigma^{-1}(j)) \end{aligned}$$

Two graphs are isomorphic iff their string encodings belong to the same orbit w.r.t. this action of S_n. For $G \leq S_n$, denote $w^G := \{w^\sigma, \sigma \in G\}$. Canonising graphs amounts to find a polynomial-time computable function which maps any $w \in \{0,1\}^{n^2}$ to $\tilde{w} \in \{0,1\}^{n^2}$ (for any n), and such that, for any $w \in \{0,1\}^{n^2}$, it maps all string in w^{S_n} to the same $\tilde{w} \in w^{S_n}$. To obtain such a function, the algorithm defined in [BL83] upkeeps and gradually restricts canonically[3] a coset $\sigma H \subseteq S_n$ of permutations, until $w^\sigma = w^\tau$ for each $\tau \in \sigma H$, at which point $H \leq \mathrm{Aut}(X)$ and $\tilde{w} := w^\sigma$ is a canonical encoding of the graph encoded by w.

Two recursion mechanisms allow us such a restriction of the coset σH (those were already introduced in the context of graph isomorphism in [Luk82]):

- if H acts intransitively on $[n^2-1]$, we can treat each orbit iteratively: if $\mathcal{O}_1 \dots \mathcal{O}_m$ are the orbits of the action of H on $[n^2-1]$, we first find a canonical coset $\sigma_1 H_1$ canonising the substring of w induced on $\mathcal{O}_1$, to then find a subcoset of $\sigma_1 H_1$ which canonises w over $\mathcal{O}_1 \cup \mathcal{O}_2$, yielding a canonical coset $\sigma_2 H_2$ canonising $\mathcal{O}_2$. Iterating this process on all orbits yields a coset canonising the whole string, and thus of X.
- if H acts transitively and imprimitively on $[n^2-1]$, we can find a minimal block system, and treat each permutation of this block system as a subcase (for each permutation of the block-system, we recursively compute a canonisation of the graph, then choose the lexicographical leader in this family of encodings).

When H is transitive primitive, we use brute-force and pick the set $\tau K \subseteq \sigma H$ of permutations which yield the smallest encoding.

Computing the orbits or a minimal block system of a group action are polynomial-time computable functions. Since each recursive call incurs a division of the size of the instance, the whole procedure runs in polynomial time as long as : in the imprimitive

[3] In this context, this should be read as *in an isomorphism-invariant manner*

case, the size of the group acting on the block system is polynomially-bounded ; and in the brute-force case, the size of the group is polynomially bounded.

We now discuss the initial value of the coset. Starting with $\sigma H := S_n$ correctly yields a canonical copy of X. However, none of the two recursion mechanisms apply (as the action of S_n on $[n^2 - 1]$ described above is both transitive and primitive on the orbit $[n^2 - 1] \setminus \{n \cdot i + i, i < n\}$), and the brute force search of the smallest encoding of X (w.r.t. the lexicographical order) incurs an exponential timing of the algorithm.

This is where a combinatorial study of the class of structures at hand comes into play. If X belongs to a restricted class of graphs $\mathcal{C}$, we can look for a polynomial-time computable invariant of X, that is, a function $f : \mathcal{C} \to \mathcal{D}$ where $\mathcal{D}$ is any class of finite structures, such that $X \simeq Y \implies f(X) \simeq f(Y)$; and therefore restrict our search to the set of reorderings that minimize the encoding of $f(X)$.

To illustrate this, consider a coloured graph. The colouring is an invariant of the graph that should be maintained, and instead of starting the procedure with $\sigma H = S_n$, we can set

$$H := \prod_{i=1}^{m} \mathrm{Sym}(\{n \cdot x + y \mid x, y \in c^{-1}(i)\}) \tag{4.2}$$

in which case, choosing a coset of H amounts to choose, for each colour i, the set $\mathcal{I}_i \in \binom{n}{|c^{-1}(i)|}$ of placements of all vertices of colour i. One natural (and canonical) such choice being to map the colour 1 to the indices $[0, |c^{-1}(1)| - 1]$, the colour 2 to the indices $[|c^{-1}(1)|, |c^{-1}(2)| - 1]$, and so on; that is, to pick σH for any σ that is monotone w.r.t. the pre-ordering

$$v \preceq w \iff c(v) \leq c(w)$$

When each colour has bounded size, the structure of this group ensures that the recursive procedure yields a result in polynomial time. More specifically, this group has bounded composition width. Note that this also applies when the colouring is not given explicitly, but only definable. For instance, we know that, for any k, we can order the k-tuples of a structure according to their L^k-types in FP [AV95], and to their C^k-types in FPC [Ott17], where L^k and C^k are the k variables fragments of infinitary logic and counting infinitary logic respectively (while those logics do not have recursive syntax, they play an important role in the study of FP and FPC, for their formal definition, see [Ott17]). This yield a canonical pre-ordering, i.e. a colouring which is trivially isomorphism-invariant.

With this general view of the algorithm in mind, we can delve into its translation to the logical context.

From the algorithmic to the logical context

As we have stated at the beginning of this section, given an unordered graph $\mathfrak{A} = (V, E)$, we aim to build an ordered structure $\mathfrak{A}^< = (A^<, E^<)$ isomorphic to $\mathfrak{A}$. This underlines a subtle change in our objective: we are not looking for a canonical set of permutations of the structure, but of *orderings* of the structure. Indeed, while $E^<$, being a numerical

relation, merely differs from a string in its presentation, this is not true of E which is not ordered.

As such, in the unordered setting, our recursion variable, the coset σH, is not technically a coset: we usually expect a coset of H to lie within a larger group G. To avoid any misunderstanding, we adopt the terminology of [SW19]:

Definition 4.12. A *labelling coset over* A is any set of the form fH for some $f : A \to A^<$ and $H \leq \mathrm{Sym}(A)$.

With a labelling coset fH, f is not a permutation, and it is quite unclear what such a group $G \geq fH$ should be. There are two obstacles to a logical (or choiceless) translation of Babai & Luks algorithm.

First, as we mentioned in the discussion of Lemma 2.13, finding a maximal block of an imprimitive action seems — to our knowledge — out of reach of FP + ord or CPT. Fortunately, in the case of abelian colours[4] , this issue is bypassed: in the presence of such a colouring, we can use a labelling coset of the natural action of the group $\Gamma = \prod_{i=1}^{m} \Gamma_i$ as the initial coset, and the intransitivity recursion scheme alone then reduces the size of the labelling cosets to a point where brute-force is sufficient. More specifically, each orbit (for the action of the group at hand over $A \times A$) corresponds to a couple of colour-classes $A_i \times A_j$, and the ordering on the colour-classes $\prec$ induces a linear-order on those orbits. Because the action of the underlying group of our labelling coset is a subgroup of $\Gamma_i \Gamma_j$, its size is bounded by $|A_i||A_j|$, and a sub-labelling coset which minimizes the encoding of $E \cap (A_i \times A_j)$ can be found in polynomial-time by going through all elements of the current labelling coset. Factoring this simplification, we obtain Algorithm 3.

Algorithme 3 : canonisation procedure

Input : $\mathfrak{A} = (A, E, \prec, \Phi)$ a structure with Abelian colours
Output : A numerical relation $E^<$ isomorphic to E

1 Find, for each $i \leq m$, a canonical set $\mathcal{O}(A_i)$ of orderings of A_i;
2 $\mathcal{C} := \prod_{i=1}^{m} \mathcal{O}(A_i)$;
3 **for** $(i, j) \in [m]^2$ **do**
4 find $E^<_{i,j}$, the smallest lexicographical encoding of $E \cap (A_i \times A_j)$ compatible with $\mathcal{C}$, i.e. $\exists \sigma \in \mathcal{C}, \mathrm{enc}(E, \sigma)_{\restriction A_i \times A_j} = E^<_{i,j}$;
5 $\mathcal{C} \leftarrow \{\sigma \in \mathcal{C}, \mathrm{enc}(E, \sigma)_{\restriction A_i \times A_j} = E^<_{i,j}\}$;
6 **return** $E^< := \bigcup_{i,j} E^<_{i,j}$

An ordering of A_i is a function $f : A_i \to A_i^<$, $A_i^<$ being the set of numerical values associated with A_i in any ordering of A compatible with $\prec$, that is, the interval $[a+1, b]$, where $a = \sum_{j=1}^{i-1} |A_j|$ and $b = a + |A_i|$; and for any $\sigma : A \to A^<$, $\mathrm{enc}(E, \sigma) := \{(\sigma(a), \sigma(b)) \mid (a, b) \in E\}$.

[4]The same argument holds for bounded colour-class graphs in general. In that case, the second obstacle does not admit such an overcoming as will be described in the remainder of this chapter

The for-loop on line 3 implements precisely the iteration of the intransitivity-case recursion mechanism of Babai & Luks framework described above. Within this for-loop, we only use brute-force (on Line 4) to obtain a labelling coset canonising $E_{i,j}$. Note that this incurs a cost of $O(|A_i| \times |A_j|)$ steps of computation, as $|\Gamma_i| = |A_i|$ and $|\Gamma_j| = |A_j|$ as was proved in Lemma 4.2.

While presented a bit differently, Algorithm 3 is precisely the *canonisation procedure* in [Pak15, Figure 6.2]. Seeing $E^<$ as a binary adjacency matrix, entering the for-loop with (i, j) entails to determine the value of $E^<$ on a rectangle of dimensions $(|A_i|, |A_j|)$, encoding the edges from A_i to A_j, while restricting the labelling coset C to those orderings that yield this encoding of $E \cap A_i \times A_j$.

We depart from [Pak15] in how we deal with the *second* obstacle, which is the *representation of labelling cosets*. Consider a group $G \leq \mathrm{Sym}(A)$. As we have hinted at below Definition 4.12, in the algorithmic context, cosets and labelling cosets are objects of the same nature, and a labelling coset fG can be represented by one such function f, together with a generating set for G (enabling the whole group-theoretic framework introduced in Section 1.2). In the logical context, we cannot expect to *choose* such a function f to represent the whole coset, as such a choice would break the symmetry of the structure (we would have in effect chosen a linear order $f : A \to A^<$). In [Pak15], labelling cosets are encoded as systems of equations over some finite ring. Here, we will encode labelling cosets as the preimage, for definable morphisms from a definable base group $\mathcal{G}$ to an ordered group Ω, of a point in Ω.

We will now delve into the definition of Algorithm 3 in $\mathsf{FP} + \mathsf{ord}$. To make our discourse clearer, we give names to the different levels of locality we are dealing with. Indeed, Algorithm 3 handles encodings, orderings, and labelling cosets at three different scales.

First, there is the *local scale*, where we are only interested with one colour-class and its inner edges. This scale is used on Line 1 when we construct local labelling cosets. In the next section, we will show how these sets, and thus the initial labelling coset of the structure, can be defined in $\mathsf{FP} + \mathsf{ord}$.

Then, there is the *semi-local scale*, where we look at up to two colour-classes simultaneously, and at the edges between them. We will see in Section 4.5 that for each i, j, given a suitable encoding of a labelling coset over $A_i \cup A_j$, we can find its smallest lexicographical encoding of $E \cap (A_i \times A_j)$, enabling Line 4 of the canonisation procedure. Moreover, we will show that it is possible to define a *group morphism* $m_{i,j}$ whose domain contains $\mathcal{O}(A_i) \times \mathcal{O}(A_j)$ such that $m_{i,j}(\sigma)$ determines the value of $\mathrm{enc}(E, \sigma)_{\restriction A_i \times A_j}$. This is the main building block of our encoding of labelling cosets, where we make use of the morphism framework defined in Section 3.3 to specify semi-local labelling cosets inside $\mathsf{FP} + \mathsf{ord}$.

Finally, at the *global scale* of the whole domain A, we need to define a general encoding of labelling cosets that enable Line 5 of the canonisation procedure. Building on the morphisms $m_{i,j}$, we do so in Section 4.6, concluding our proof.

4.4 The local scale

With regards to canonisation, the unordered setting is a double-edged sword. The separation between the unordered and ordered domain — the former being the domain of the input structure, and the latter of the output structure — ensures that any such construction is, by design, canonical; but as we have already stated, it also makes the group-theoretic framework quite hard to use, and it is not quite clear how to even start, and define any useful set of orderings. In this section, we will see how, at the local scale, the abelian, ordered group defined by $\Phi\ (\mathfrak{A}, i)$ acting transitively on A_i allows us to define a linear-sized set of local orderings. These sets $\mathcal{O}(A_i)$ enable the translation in FPC of Line 2 of Algorithm 3. We defer the encoding of $\mathcal{C}$ on line 2, the initial global labelling coset, to Section 4.6.

Note that this set of local orderings is canonical w.r.t. $\mathfrak{A}$, but depends on Φ, and in particular, on the ordering of the abelian groups. This means that, if we were to equip the coloured graph underlying $\mathfrak{A}$ with a different family of ordered abelian groups, we may end up with a different output. While this is a limitation of our algorithm, it does not affect the fact that it allows a canonisation of Lichter's construction, as in that case, the ordered abelian groups can be (canonically) defined within FPC, as we have seen in Theorem 4.10.

In the first subsection of this section, we introduce our representation of linear orderings over subsets of A, together with a generalised quantifier that, given such a representation of a linear ordering and a relation on A, produces the corresponding relation on $A^<$. In a second time, we provide a FPC-definition of $\mathcal{O}(A_i)$. Finally, we show how this enables the canonisation of the abelian colours, i.e. to define relations $\prec^<$ and $\Phi^<$ such that $(A, \prec, \Phi) \simeq (A^<, \prec^<, \Phi^<)$.

Partial orderings and partial encodings

Let us summarise the notations we will use in this section, and in the remainder of this chapter. We assume we are given a graph $\mathfrak{A}$ with m Abelian colours. As we are working with a fixed structure, we will omit the superscript $\mathfrak{A}$ when referring to $E, \prec$ or Φ, unless necessary to disambiguate between the actual relation $E^{\mathfrak{A}}$ and the symbol E of the signature of graphs with Abelian colours.

The domain of $\mathfrak{A}$ is A, and we denote A_i the set of vertices of colour i, that is:

$$\forall i, j \leq m, \forall a \in A_i, \forall b \in A_j, a \preceq b \iff i \leq j$$

Moreover, let $A^< := \{\lambda \in \mathbb{N}, \lambda < |A|\}$. A partition of $A^<$ into $A_1^<, \ldots, A_m^<$ was provided during the presentation of Algorithm 3. As another definition of this partition, it is the image of the partition $(A_1, \ldots, A_m)$ through any monotone function $f : (A, \prec^{\mathfrak{A}}) \to (A^<, <_{\mathbb{N}})$.

Finally, let γ_j^i be the j-th permutation of A_i defined by Φ, i.e.

$$\text{graph}(\gamma_j^i) = \Phi\ (\mathfrak{A}, i, j)$$

and Γ_i be the transitive abelian group acting on A_i defined by $\Phi\ (\mathfrak{A}, i)$:

$$\Gamma_i := \{\gamma^i_j, j \leq |A_i|\}$$

We denote Γ the direct product of those groups.

Our aim in this subsection is to detail how a (partial) ordering yields a (partial) encoding of the relations in our structure, and how this encoding is definable within FPC. But first, we need to express how we expect such (partial) orderings to be accessible in extensions of first-order logic with counting:

Definition 4.13. A formula φ *defines an ordering* $f : A \to A^<$ in $\mathfrak{A}$ if

$$\varphi(\mathfrak{A}) = \{(a, f(a)), a \in A\}$$

This definition is coherent with Definition 2.15: a function is definable if its graph is. A *partial ordering* is a bijective function $g : B \to B^<$, where $B = \bigcup_{i\in S} A_i$ and $B^< = \bigcup_{i\in S} A_i^<$, for some $S \subseteq [m]$. As we will detail later, when $|S| = 1$ we say that g is a *local* ordering, when $|S| = 2$, that g is a *semi-local* ordering, and when $S = [m]$, that it is a *global* ordering. We use the same definition as above:

Definition 4.14. A formula φ *defines a partial ordering* $g : B \to B^<$ over B in $\mathfrak{A}$ if

$$\varphi(\mathfrak{A}) = \{(a, g(a)), a \in B\}$$

Note that a partial ordering f is always the restriction of some global ordering to the set of colours on which f is defined.

Given an ordering $f : A \to A^<$, and any relation R on $\mathfrak{A}$, we can define the *encoding of R relative to f*:

$$R^f := \{(\overline{f}(a_1), \overline{f}(a_2), \ldots, \overline{f}(a_k)) \mid \vec{a} \in R^{\mathfrak{A}}\}$$

where

$$\overline{f}(a) = \begin{cases} f(a) & \text{if } a \in A \\ a & \text{if } a \in A^< \end{cases} \tag{4.3}$$

If g is a partial ordering over B, the *partial encoding of R relative to g* is

$$R^g := \{(\overline{g}(a_1), \ldots \overline{g}(a_k)) \mid \vec{a} \in R^{\mathfrak{A}} \cap B^{\mathrm{type}(R)}\}.$$

where, $B^{\mathrm{element}} = B$ and $B^{\mathrm{number}} = A^<$ (so that numerical values exceeding $|B|$ can still be assigned through partial encodings).

Now that all this terminology is defined, we prove a small lemma that will be useful throughout the chapter:

Lemma 4.15. *Let $\phi(\vec{p}, \vec{x})$ be a formula defining, for each valuation $\vec{b}$ of $\vec{p}$, some relation $R_{\vec{b}}$ over $\mathfrak{A}$. Let $\psi(\vec{q}, y, \nu)$ be a formula defining, for any valuation $\vec{a}$ of $\vec{q}$, a partial ordering $g_{\vec{a}}$ over B, a union of colour-classes of $\mathfrak{A}$.*
There is a FPC*-definable quantifier $(\mathrm{enc}_{\vec{x};y,\nu}\phi; \psi)(\vec{p}, \vec{q}, \vec{\lambda})$ that defines $\vec{a}, \vec{b} \mapsto (\phi(\mathfrak{A}, \vec{b}))^{g_{\vec{a}}}$.*

Proof. We first define, for any type $t \in \{\text{element}, \text{number}\}^*$, a formula $\psi^t(\vec{q}, \vec{\alpha}, \vec{\lambda})$ which defines, for each valuation $\vec{a}$ of $\vec{q}$ the graph of the map:

$$\begin{array}{rccc} g^t_{\vec{a}} : & A^t & \to & (A^<)^{|t|} \\ & (u_1, \dots, u_{|t|}) & \mapsto & (\overline{g}_{\vec{a}}(u_1), \dots, \overline{g}_{\vec{a}}(u_{|t|})) \end{array}$$

where $\overline{g}_{\vec{a}}$ is defined as in Eq. (4.3). In $\psi^t(\vec{q}, \vec{\alpha}, \vec{\lambda})$, $\vec{\alpha}$ has type t and $\vec{\lambda}$ has type number$^{|t|}$:

$$\psi^t(\vec{q}, \vec{\alpha}, \vec{\lambda}) := \bigwedge_{\substack{i=1 \\ t_i=\text{element}}}^{|t|} \psi(\vec{q}, \alpha_i, \lambda_i) \wedge \bigwedge_{\substack{i=1 \\ t_i=\text{number}}}^{|t|} \alpha_i = \lambda_i$$

This allows us to define the quantifier in FPC:

$$(\text{enc}_{\vec{x};y,\nu}\phi; \psi)(\vec{p}, \vec{q}, \vec{\lambda}) := \exists \vec{z}, \psi^{\text{type}(\vec{x})}(\vec{q}, \vec{z}, \vec{\lambda}) \wedge \phi(\vec{p}, \vec{z}) \qquad \square$$

To improve readability, and when the variables bound by the enc quantifier can easily be deduced, we denote enc^{ψ}_{ϕ} the relation defined by $(\text{enc } \phi; \psi)$.

A canonical set of orderings...

In this subsection, we show that there is an FPC-definable map from A_i to the set of orderings of A_i. This construction harnesses the linear-orderings on Γ_i provided by Φ. Lemma 4.16 corresponds to Lemma 6.9 in [Pak15].

Lemma 4.16. *There is an* FPC*-formula* $\mathsf{map}(\lambda, x, y, \mu)$ *such that, for all* $i \leq m$ *and* $a \in A_i$, $\mathsf{map}(\mathfrak{A}, i, a)$ *defines a local ordering of* A_i.

Proof. First, we remind the reader that, under our assumptions, the action of Γ_i on A_i is regular, as was shown in Proposition 1.59. This implies that, for any fixed $a \in A_i$,

$$\gamma^i_1(a) < \gamma^i_2(a) < \cdots < \gamma^i_{|A_i|}(a) \tag{4.4}$$

defines a linear ordering on A_i. This ordering corresponds to the following FPC definable bijection between A_i and $[A_i]$:

$$\Psi(\lambda, x, y, \mu) :=\Phi\ (\lambda, \mu, x, y)$$

$(\mathfrak{A}, i, a, b, k) \models \Psi$ iff b is the k-th element in the ordering of A_i defined by Eq. (4.4). In order to build a bijection whose image is $A_i^<$ instead of $[A_i]$, we add an adequate offset:

$$\mathsf{map}(\lambda, x, y, \mu) := \exists \mu', \left(\mu = \mu' + \sum_{\nu<\lambda} |A_\nu|\right) \wedge \Phi\ (\lambda, \mu', x, y)$$

The definability of those arithmetic operations in FPC is straight-forward. We have successfully built a formula map such that, for all $i \leq m$ and $a \in A_i$, $\mathsf{map}(\mathfrak{A}, i, a)$ is the graph of a bijection between A_i and $A_i^<$, and thus defines a partial ordering over A_i. $\square$

To ease reading, outside of formulae, we denote map^i_a the ordering defined by $\mathsf{map}(\mathfrak{A}, i, a)$. By definition of map, with the notations introduced at the beginning of this section, we have, for any $a, b \in A_i$ and $\mu \in A_i^<$,

$$\mathsf{map}^i_a \cdot b = \mu \iff \gamma^i_\mu \cdot a = b. \tag{4.5}$$

We follow Pakusa's notation and denote $\mathcal{O}(A_i)$ the set of all map^i_a, for $a \in A_i$. We now show that for all $i \leq m$, the set $\mathcal{O}(A_i)$ is a labelling coset over A_i:

Lemma 4.17. *For any $i \leq m$ and $a \in A_i$, $\mathcal{O}(A_i) = \mathsf{map}^i_a \Gamma_i$.*

Proof. Let $a \in A_i$, and $\gamma^i_j \in \Gamma_i$. Then, for any $b \in A_i$,

$$\begin{aligned}
\mathsf{map}^i_a \gamma^i_j \cdot b = \mu &\iff \gamma^i_\mu \cdot a = \gamma^i_j \cdot b && \text{by Eq. (4.5)} \\
&\iff ((\gamma^i_j)^{-1}\gamma^i_\mu) \cdot a = b \\
&\iff \gamma^i_\mu \cdot ((\gamma^i_j)^{-1} \cdot a) = b && \text{since } \Gamma_i \text{ is abelian} \\
&\iff \mathsf{map}^i_{(\gamma^i_j)^{-1} \cdot a} \cdot b = \mu
\end{aligned}$$

Therefore, $\mathcal{O}(A_i)$ is closed by multiplication on the right (i.e. precomposition) by elements of Γ_i, or, said differently, $\mathsf{map}^i_a \Gamma_i \subseteq \mathcal{O}(A_i)$. The transitivity of Γ_i yields the other inclusion: for $a, b \in A_i$, let γ^i_j be the element[5] such that $\gamma^i_j \cdot b = a$. Then, $\mathsf{map}^i_a \gamma^i_j = \mathsf{map}^i_{(\gamma^i_j)^{-1} \cdot a} = \mathsf{map}^i_b$. □

This lemma has the following corollary that will become important when studying the global scale:

Corollary 4.18. *For any $\pi \in \prod_{i=1}^m \mathcal{O}(A_i)$, we have $\prod_{i=1}^m \mathcal{O}(A_i) = \pi\Gamma$.*

...that canonises the Abelian groups

As we have expressed at the beginning of this section, we are not really canonising a graph, but rather a graph with the additional structure of abelian colours. As such, we *should* also provide a copy $\Gamma_i^<$ of the ordered groups Γ_i (that acts on the unordered domain) acting on the ordered domain.

We will now show that this is exactly what we have built so far: for each i, $\mathcal{O}(A_i)$ canonises $\Phi\ (\mathfrak{A}, i)$, in the sense that all the orderings in $\mathcal{O}(A_i)$ yield the same encoding of $\Phi\ (\mathfrak{A}, i)$.

Lemma 4.19. *For any $a, b \in A_i$, the encodings of $\Phi\ (\mathfrak{A}, i)$ relative to map^i_a and map^i_b are equal.*

[5] Recall that Γ_i is regular, and thus this element is unique

Proof. Fix $i \leq m, j \leq [A_i], a \in A_i$. Then, the following holds:

$$\begin{aligned}\Phi^{\mathsf{map}^i_a}(i,j) &= \{(\mathsf{map}^i_a \cdot b, \mathsf{map}^i_a \gamma^i_j \cdot b), b \in A_i\} \\ &= \{(\alpha, \mathsf{map}^i_a \gamma^i_j (\mathsf{map}^i_a)^{-1} \cdot \alpha), \alpha \in A_i^<\}\end{aligned}$$

In words, encoding Γ_i relative to map^i_a entails to enumerate the elements of Γ_i conjugated by map^i_a. It happens that the conjugation actions of map^i_a and map^i_b on Γ_i coincide:

$$\begin{aligned}(\gamma^i_j)^{\mathsf{map}^i_a} &= \mathsf{map}^i_a \gamma^i_j (\mathsf{map}^i_a)^{-1} \\ &= \mathsf{map}^i_a \gamma^i_j (\mathsf{map}^i_a)^{-1} \mathsf{map}^i_b (\mathsf{map}^i_b)^{-1} \\ \text{Since } \Gamma_i \text{ is abelian,} \quad &= \mathsf{map}^i_a (\mathsf{map}^i_a)^{-1} \mathsf{map}^i_b \gamma^i_j (\mathsf{map}^i_b)^{-1} \\ &= \mathsf{map}^i_b \gamma^i_j (\mathsf{map}^i_b)^{-1} \\ &= (\gamma^i_j)^{\mathsf{map}^i_b}\end{aligned}$$

Note that we have used the fact that $(\mathsf{map}^i_a)^{-1}\mathsf{map}^i_b \in \Gamma_i$, which is a direct consequence of Lemma 4.17. □

From now on, we denote $\Gamma_i^<$ the ordered group $\Gamma_i^{\mathsf{map}^i_a}$ for any $a \in A_i$. Looking at our proof of Lemma 4.19, we have actually shown something stronger: for any $a \in A_i$, $\gamma^i_j \mapsto \mathsf{map}^i_a \gamma^i_j (\mathsf{map}^i_a)^{-1}$ defines an isomorphism from Γ_i to $\Gamma_i^<$, and *those isomorphisms coincide* for any $a, b \in A_i$.

A last consequence of the existence and definability of the labellings map^i_a is that we can assume $E_i := E \cap (A_i \times A_i)$ to be canonised by $\mathcal{O}(A_i)$:

Lemma 4.20. *There are* FPC*-formulae $\zeta_\prec$, ζ_Φ such that, if $\mathfrak{A}$ is a graph with abelian colours,*

- $\mathfrak{B} := (A, E^{\mathfrak{A}}, \zeta_\prec(\mathfrak{A}), \zeta_\Phi(\mathfrak{A}))$ *is a graph with abelian colours,*
- $\zeta_\prec(\mathfrak{A})$ *is a refinement of* $\prec^{\mathfrak{A}}$*,*
- *For any i, b, b' such that b, b' belong to B_i, the i-th colour class of $\mathfrak{B}$,* $\mathsf{map}(\mathfrak{B}, i, b)$ *and* $\mathsf{map}(\mathfrak{B}, i, b')$ *define two orderings σ, τ of B_i such that $(E \cap (B_i \times B_i))^\sigma = (E \cap (B_i \times B_i))^\tau$.*

Note that, on any such structure $\mathfrak{B}$, the formula

$$\mathsf{loc}E(\nu_s, \nu_t) := \exists \mu, \textcircled{\wedge} \begin{cases} \nu_s \in A_\mu^< \wedge \nu_t \in A_\mu^< \\ \exists a, s, t \in A_\mu, E(s,t) \wedge \mathsf{map}(\mu, a, s, \nu_s) \wedge \mathsf{map}(\mu, a, t, \nu_t) \end{cases}$$

defines a binary numerical relation isomorphic to $\bigcup_{i \leq m} E_i$.

Proof. We show how to refine $\prec$, and redefine Φ on a single colour-class A_i. The resulting procedure can then be iterated using a fixed-point computation over all colour-classes. Consider

$$\chi_\prec(x,y) := \begin{cases} x \in A_i \\ \textcircled{\wedge}\, y \in A_i \\ \mathrm{enc}_{E_i}^{\mathsf{map}^i_x} <_{lex} \mathrm{enc}_{E_i}^{\mathsf{map}^i_y} \end{cases}$$

Suppose $\chi_\prec$ defines a non-trivial refinement of $\prec$ on A_i, and denote $B_1, \ldots, B_k$ the $\chi_\prec$-equivalence classes of A_i. We show how to define an ordered abelian group of permutations acting transitively on each B_j. More precisely, we aim to show that $\Delta_i := \mathrm{Aut}(\mathfrak{A}_i) \leq \Gamma_i$ is such a permutation group for each B_j. In that prospect, we must first show that Δ_i fixes each B_j setwise. Let $b \in B_j$ and $\delta \in \Delta_i$. For any μ, ν,

$$\begin{aligned} (\mu,\nu) \in E_i^{\mathsf{map}^i_{\delta(b)}} &\iff E_i(\gamma^i_\mu(\delta(b)), \gamma^i_\nu(\delta(b))) && \text{by Eq. (4.5)} \\ &\iff E_i(\delta\gamma^i_\mu(b), \delta\gamma^i_\nu(b)) && \text{since } \Gamma_i \text{ is abelian} \\ &\iff E_i(\gamma^i_\mu(b), \gamma^i_\nu(b)) && \text{since } \delta \in \mathrm{Aut}(\mathfrak{A}_i) \\ &\iff (\mu,\nu) \in E_i^{\mathsf{map}^i_b} \end{aligned}$$

That is, the encodings of E_i relative to map^i_b and $\mathsf{map}^i_{\delta(b)}$ are equal, and thus b and $\delta(b)$ are $\chi_\prec(\mathfrak{A})$-equivalent. Let us now show that this action of Δ_i on B_j is transitive. Consider $b, b' \in B_j$. Because map^i_b and $\mathsf{map}^i_{b'}$ yield the same encoding of E_i, we know that $\delta := (\mathsf{map}^i_{b'})^{-1}\mathsf{map}^i_b \in \Delta_i = \mathrm{Aut}(\mathfrak{A}_i)$ (the fact that it stabilises $\Phi^{\mathfrak{A}}$ is a consequence of Lemma 4.19). Moreover, $\mathsf{map}^i_b(b) = \lambda$, where λ is the unique index such that $\gamma^i_\lambda = \mathrm{Id}$. Thus, $(\mathsf{map}^i_{b'})^{-1}(\lambda) = b'$, i.e. $\delta(b) = b'$, which yields the transitivity of Δ_i on B_j. Now we provide the FPC-definition of χ_Φ. We assume that the predicate $x \in B_i$ is FPC-definable.

$$\chi_\Phi(i,j,s,t) := \exists i_0, j_0, \textcircled{\wedge} \begin{cases} \forall x \in B_i, x \in A_{i_0} \\ \mathrm{Aut}(i_0, j_0) \\ j = (\#\lambda.(\lambda < j_0 \wedge \mathrm{Aut}(i_0, \lambda))) \\ \Phi(i_0, j_0, s, t) \end{cases}$$

where $\mathrm{Aut}(i_0, j_0)$ states that $\gamma^{i_0}_{j_0}$ stabilises E_i, and can be defined in FPC as follows:

$$\mathrm{Aut}(i_0, j_0) := \forall x, y, \exists x', y', \textcircled{\wedge} \begin{cases} \Phi(i_0, j_0, x, x') \\ \Phi(i_0, j_0, y, y') \\ E(x,y) \iff E(x', y') \end{cases}$$

If $\chi_\prec$ is trivial on A_i (i.e. $\chi_\prec$ defines the empty relation on A_i), then for any $a, a' \in A_i$, $E_i^{\mathsf{map}^i_a} = E_i^{\mathsf{map}^i_{a'}}$, and the class already satisfies the conditions of the lemma, and no further refinement is needed. Therefore, the desired relations $\prec', \Phi'$ over $\mathfrak{A}$ are

the fix-point of the following sequence:

$$\prec_0 := \prec^{\mathfrak{A}} \qquad \Phi_0 := \Phi^{\mathfrak{A}}$$
$$\prec_{k+1} := \chi_{\prec}(A, E, \prec_k, \Phi_k) \qquad \Phi_{k+1} := \chi_{\Phi}(A, E, \prec_k, \Phi_k)$$

Such an iteration procedure requires the use of the simultaneous fixed-point operator. Note however that the function iterated in this fixed-point computation is not inflationary. Yet, because each iteration strictly refines $\prec$ on A, a fixed-point of this process is reached in at most $|A|$ steps. As such, the *trace argument* presented in the proof of Theorem 1.31 enables the definition of the fixpoint of this sequence in FPC. An additional fix-point definition is then needed to treat every colour-class, as the treatment of each colour-class induces a reindexing (into potentially many more colours) of the colour-classes of the structure. While the combination of simulatenous fixed-point and this trace argument does not present any difficulty, it is quite tedious to write, so we do not present the full formulae $\zeta_{\prec}, \zeta_{\Phi}$.

□

4.5 The semi-local scale

At this point, we have defined bijections between A_i and $\mathcal{O}(A_i)$, a local labelling coset, for each i. Together, those yield a bijection between tuples $\vec{a} \in \prod_{i=1}^m A_i$ and the initial labelling coset $\mathcal{C} := \prod_{i=1}^m \mathcal{O}(A_i)$, mapping $\vec{a}$ to $\prod_{i=1}^m \mathsf{map}^i_{a_i}$. We have already mentioned that the representation with FPC of global labelling cosets is the major challenge of the canonisation of structures with abelian colours. Yet, at first sight, this bijection Ξ between $\prod_{i=1}^m A_i$ and $\prod_{i=1}^m \mathcal{O}(A_i)$ seems to enable the representation of the successive values of $\mathcal{C}$ along Algorithm 3 by subsets of $\prod_{i=1}^m A_i$, more precisely by their inverse image through Ξ: the initial value of $\mathcal{C}$ is the image of $\prod_{i=1}^m A_i$ after Line 2 ; and at each iteration of the for-loop, if the current value of $\mathcal{C}$ (before Lines 4 and 5) is represented by C, consider $C_{i,j}$ the projection on $A_i \times A_j$ of the set C, that is

$$C_{i,j} := \{(x, y) \in A_i \times A_j \mid \exists \vec{a} \in C, a_i = x \wedge a_j = y\}.$$

For each $(a_i, a_j) \in C_{i,j}$, $\mathrm{enc}_E^{\mathsf{map}(i,a_i)\vee\mathsf{map}(j,a_j)}$ is an encoding of $E \cap (A_i \times A_j)$, and we can pick the minimal lexicographical such encoding. Let $D_{i,j} \subseteq C_{i,j}$ be the set of couples (a_i, a_j) that yield this minimal encoding. To complete the passage in the for-loop, it only remains to update C to only contain the tuples that are coherent with $D_{i,j}$, that is

$$C \leftarrow \{\vec{a} \in C, \pi_{A_i \times A_j}(\vec{a}) \in D_{i,j}\}.$$

Those observations imply that the representation of labelling cosets reduces to the representation of subsets of $\prod_{i=1}^m A_i$, however we are *not* able to encode arbitrary elements, let alone subsets of $\prod_{i=1}^m A_i$ in FP + ord: there are, in the worst case (when $m = \sqrt{n}$ and $|A_i| = \sqrt{n}$ for each $i \leq m$) $O\left(2^{\sqrt{n}^{\sqrt{n}}}\right)$ such subsets, which prohibits any general encoding scheme of $\mathcal{P}(\prod_{i=1}^m A_i)$.

Nevertheless, what precedes serves as a quite accurate sketch of the remainder of this proof: we need such a representation scheme for global labelling cosets, which enables

- the projection of these cosets to any pair of colour-classes ($C \mapsto C_{i,j}$ above)
- the restriction of such a global coset to one which agrees with a semi-local one ($C \mapsto C \cap \pi^{-1}_{A_i \times A_j}(D_{i,j})$ above)

Rather than a combinatorial approach, we will use the algebraic structure of global labelling cosets extensively to encode them. In the last section, we initiated this approach at the local scale, proving that for each colour i, the canonical set of local orderings $\mathcal{O}(A_i)$ forms a local labelling coset of the input abelian group Γ_i (Lemma 4.17), and that we can assume those labelling cosets to canonise the local edges (Lemma 4.20).

In this section, we aim to extend this work to the semi-local scale, taking the edge relation into account. For $i < j \leq m$, we denote $\mathfrak{A}_{i,j}$ the substructure induced by $A_i \cup A_j$ on $\mathfrak{A}$, and $E_{i,j}$ the edge relation of $\mathfrak{A}_{i,j}$.

While we need to define a way to restrict a set of semi-local orderings $\mathcal{X} \subseteq \mathcal{O}(A_i) \times \mathcal{O}(A_j)$ to those that yield the minimal encoding of $E_{i,j}$ — which corresponds to the computation, given $C_{i,j}$, of $D_{i,j}$ in our scheme above — it is crucial that this restriction respect the algebraic structure of $\mathcal{O}(A_i)$ that we have exhibited in the last section. Indeed, this algebraic structure will enable the ord operator to define an encoding of *global* labelling cosets in the next section, which is necessary to implement Algorithm 3.

To do so, we will exhibit, for any $i < j$, a definable morphism $m_{i,j} : \Gamma_i\Gamma_j \to \mathrm{Sym}(\Omega_{i,j})$, where $\Omega_{i,j}$ is an ordered set, such that the kernel of $m_{i,j}$ is equal to $\Gamma_i\Gamma_j \cap \mathrm{Aut}(\mathfrak{A}_{i,j})$, that is, the set of elements of $\Gamma_i\Gamma_j$ that preserve the edge relation between the two colour-classes. For the remainder of this section, we fix (i, j) a couple of colours with $i < j$.

Let $\Delta_{i,j} := \Gamma_i\Gamma_j \cap \mathrm{Aut}(\mathfrak{A}_{i,j})$. As $\Delta_{i,j} \leq \Gamma_i\Gamma_j$ and by assumption, Γ_i and Γ_j are abelian, we obtain $\Delta_{i,j} \trianglelefteq \Gamma_i\Gamma_j$. As such, $\Delta_{i,j}$ is the kernel of the morphism

$$m : \Gamma_i\Gamma_j \to \Gamma_i\Gamma_j/\Delta_{i,j}$$
$$\gamma \mapsto \gamma\Delta_{i,j}$$

which satisfies the requirements we have set above. However, it is not clear how to represent $\Gamma_i\Gamma_j/\Delta_{i,j}$ as a permutation group with FPC. Computationally, we could represent each coset of $\Delta_{i,j}$ in $\Gamma_i\Gamma_j$ by some representative, and represent the quotient group by its action on the set of cosets in this way[6]. However, choosing such a representative of each coset is usually out of reach for FPC.

In this instance, because the groups Γ_i are ordered, for each $\gamma\Delta_{i,j}$, we can actually use the lexicographically minimal element of $\Gamma_i\Gamma_j$ that belongs to $\gamma\Delta_{i,j}$ as a represen-

[6]This procedure can be carried out in polynomial-time because $|\Gamma_i\Gamma_j : \Delta_{i,j}|$ is polynomially-bounded by $|A|$

tative of this coset. Denoting $\Omega_{i,j}$ this set of representatives, we obtain a morphism $m_{i,j} : \Gamma_i\Gamma_j \to \mathrm{Sym}(\Omega_{i,j})$.

To summarise, for $\gamma \in \Gamma_i\Gamma_j$ and $\omega \in \Omega_{i,j}$, $m_{i,j}(\gamma)(\omega)$ is the unique $\omega' \in \Omega_{i,j}$ such that $\gamma\omega\Delta_{i,j} = \omega'\Delta_{i,j}$, i.e. ω' is the unique element of $\Omega_{i,j}$ such that $\omega^{-1}\gamma^{-1}\omega' \in \mathrm{Aut}(\mathfrak{A}_{i,j})$. The rest of this section is devoted to the proof that this morphism is definable in FPC.

Theorem 4.21. *For each $i < j \leq m$, there is a* FPC *definable ordered set $\Omega_{i,j}$, and a* FPC *definable morphism*

$$m_{i,j} : \Gamma_i\Gamma_j \to \mathrm{Sym}(\Omega_{i,j})$$

such that $\ker(m_{i,j}) = \Delta_{i,j} = \Gamma_i\Gamma_j \cap \mathrm{Aut}(\mathfrak{A}_{i,j})$.

Before proving Theorem 4.21, we introduce two small lemmas. First, we show that the membership to $\Delta_{i,j}$ is definable in FPC:

Lemma 4.22. *There is a* FPC *formula* $\mathsf{aut}(i,j,\mu,\nu)$ *such that* $\mathfrak{A} \models \mathsf{aut}(i,j,\alpha,\beta)$ *iff* $\gamma^i_\alpha\gamma^j_\beta \in \Delta_{i,j}$.

Proof. :

$$\mathsf{aut}(i,j,\mu,\nu) := \forall a_i \in A_i, a_j \in A_j, \exists b_i, b_j, \circledwedge \begin{cases} \Phi\ (i,\mu,a_i,b_i) \\ \Phi\ (j,\nu,a_j,b_j) \\ (E(a_i,a_j) \iff E(b_i,b_j)) \end{cases}$$ □

This in turn enables us to check if two elements of $\Gamma_i\Gamma_j$ belong to the same coset of $\Delta_{i,j}$:

Lemma 4.23. *There is a* FPC *formula* $\mathsf{coset}(i,j,\mu,\nu,\mu',\nu')$ *such that*

$$\mathfrak{A} \models \mathsf{coset}(i,j,\alpha,\beta,\alpha',\beta') \iff \gamma^i_\alpha\gamma^j_\beta\Delta_{i,j} = \gamma^i_{\alpha'}\gamma^j_{\beta'}\Delta_{i,j}$$

Proof. For $\alpha,\beta,\alpha',\beta'$, $\gamma^i_\alpha\gamma^j_\beta\Delta_{i,j} = \gamma^i_{\alpha'}\gamma^j_{\beta'}\Delta_{i,j}$ iff $\chi := (\gamma^i_{\alpha'}\gamma^j_{\beta'})^{-1}\gamma^i_\alpha\gamma^j_\beta \in \Delta_{i,j}$. Because Γ_i and Γ_j act on disjoint sets, they commute, thus $\chi \in \Gamma_i\Gamma_j$ and there is a couple α'',β'' such that $\chi = \gamma^i_{\alpha''}\gamma^j_{\beta''}$. Because $\chi \in \Gamma_i\Gamma_j$, $\chi \in \Delta_{i,j}$ is equivalent to $\chi \in \mathrm{Aut}(\mathfrak{A}_{i,j})$, which yields the following formula:

$$\mathsf{coset}(i,j,\mu,\nu,\mu',\nu') := \exists\mu'',\nu'', \circledwedge \begin{cases} \gamma^i_{\mu''} = (\gamma^i_{\mu'})^{-1}\gamma^i_\mu \\ \gamma^j_{\nu''} = (\gamma^j_{\nu'})^{-1}\gamma^j_\nu \\ \mathsf{aut}(i,j,\mu'',\nu'') \end{cases}$$

To ease reading, we have included two clauses of the form $\sigma = \tau^{-1}\rho$ which are not FPC formulae per se. However, when the graphs of σ, τ and ρ are defined by R_σ, R_τ and R_ρ, respectively, this equality can be defined in FPC as in Lemma 2.11. □

We are now ready to prove Theorem 4.21:

Proof of Theorem 4.21. We can choose as a representative of the coset $\sigma\Delta_{i,j}$ the lexicographically smallest couple (α, β) such that $\gamma^i_\alpha\gamma^j_\beta \in \sigma\Delta_{i,j}$. The following formula holds for (i, j, α, β) iff (α, β) is such a couple for some coset:

$$\mathsf{witness}(i, j, \mu, \nu) := \forall \mu', \nu', (\mu'\nu' <_{lex} \mu\nu) \implies \neg\mathsf{coset}(i, j, \mu, \nu, \mu', \nu')$$

This shows that $\Omega_{i,j} := \mathsf{witness}(\mathfrak{A}, i, j)$ is definable in FPC.

We aim to define $m_{i,j}(\gamma)(\alpha, \beta)$ as the unique $(\alpha', \beta') \in \Omega_{i,j}$ such that $\gamma\gamma^i_\alpha\gamma^j_\beta \in (\gamma^i_{\alpha'}\gamma^j_{\beta'})\Delta_{i,j}$, that is, such that $(\gamma^i_{\alpha'}\gamma^j_{\beta'})^{-1}\gamma\gamma^i_\alpha\gamma^j_\beta \in \Delta_{i,j}$. Note that, rather than keeping $m_{i,j}$ undefined on $(A^<)^2 \setminus \Omega_{i,j}$, we simply set it to act trivially. This indeed defines a morphism, as the action of $m_{i,j}(\gamma)$ on $\Omega_{i,j}$ is, by construction, isomorphic to the action of γ on the set of cosets of $\Delta_{i,j}$ in $\Gamma_i\Gamma_j$ by left multiplication. For the same reason, $\ker(m_{i,j}) = \Delta_{i,j}$.

$$\mathsf{slMorph}(i, j, R_\gamma, \mu_s\nu_s, \mu_t\nu_t) := \begin{cases} \neg\mathsf{witness}(i, j, \mu_s, \nu_s) \wedge \mu_s = \mu_t \wedge \nu_s = \nu_t \\ \mathsf{witness}(i, j, \mu_s, \nu_s) \wedge \mathsf{witness}(i, j, \mu_t, \nu_t) \\ \exists \mu', \nu', \begin{cases} \gamma^i_{\mu'}\gamma^j_{\nu'} = (\gamma^i_{\mu_t}\gamma^j_{\nu_t})^{-1}\gamma\gamma^i_\mu\gamma^j_\nu \\ \mathsf{aut}(i, j, \mu', \nu') \end{cases} \end{cases}$$

This corresponds exactly to our definition of a definable morphism, in the sense that for any $\gamma \in \Gamma_i\Gamma_j$,

$$\mathsf{slMorph}(\mathfrak{A}, i, j, \mathrm{graph}(\gamma)) = \mathrm{graph}(m_{i,j}(\gamma))$$

Note that, while we usually prefer to keep second order variables on the left-most side, we did not do so this time to underline the fact that $\mathsf{slMorph}$ actually defines a family of morphisms, one for each couple of colours.

Finally, let us justify our use of permutations equalities in the definition of $\mathsf{slMorph}$. As the graph R_γ of γ is a parameter of $\mathsf{slMorph}$, and for all i, μ, the graph of γ^i_μ is given by $\Phi\ (i, \mu)$, the subformula $\gamma^i_{\mu'}\gamma^j_{\nu'} = (\gamma^i_{\mu_t}\gamma^j_{\nu_t})^{-1}$ is in FPC, following the same idea as in Lemma 2.11. □

Theorem 4.21 is an important step in our reasoning, as it implies that the automorphism group of $\mathfrak{A}$ is morphism-definable in $\mathsf{FP} + \mathsf{ord}$. Indeed, the morphisms $m_{i,j}$ can easily be combined into a morphism

$$m^* : \Gamma \to \prod_{i<j} \mathrm{Sym}(\Omega_{i,j})$$
$$\gamma \mapsto \prod_{i<j} m_{i,j}(\gamma_{\restriction A_i \cup A_j})$$

Interestingly, $\ker(m^*) = \mathrm{Aut}(\mathfrak{A})$. To see this, remark that m^* is equal to the $\otimes$-product (as defined in Definition 3.16) of the morphisms $m_{i,j}$, each extended to the whole of Γ in the trivial way (that is, by setting, for $\gamma \in \Gamma$, $m^*_{i,j}(\gamma) := m_{i,j}(\gamma_{\restriction A_i \cup A_j})$, and $m^* := \bigotimes_{i<j} m^*_{i,j}$). In particular, m^* induces a bijection between cosets of $\mathrm{Aut}(\mathfrak{A})$ and

elements of $\prod_{i,j} \mathrm{Sym}(\Omega_{i,j})$. Our objective in the following section is to transfer this correspondance to the context of labelling cosets.

Indeed, as we aim to canonise structures, we need to handle labelling cosets, which, as argued before, in the logical context, are not group cosets of $\mathrm{Aut}(\mathfrak{A})$. In the next section, we show that we can build a larger group in which labelling cosets are actual cosets. In this context, it is the morphism-definability of $\Delta_{i,j}$ which will enable the representation of those labelling cosets.

4.6 The global scale

As we have mentioned earlier, labelling cosets are not cosets per se, as they do not lie in some larger (permutation) group. It happens, however, that our initial (and thus largest) global labelling coset $\prod_i \mathcal{O}(A_i)$ can be embedded into a permutation group $\mathcal{G}$. In this section, we show that the labelling cosets at hand in Algorithm 3 are all cosets of morphism-definable subgroups of $\mathcal{G}$, i.e. we aim to construct an injective function φ mapping orderings of A to elements of $\mathcal{G}$ and, for any labelling coset C computed during the run of Algorithm 3, a morphism $m_C : \mathcal{G} \to \mathrm{Sym}(A^T)$ (for some type T) and a value $v_C \in \mathrm{Sym}(A^T)$ such that

$$C = \left\{ \varphi^{-1}(g) \mid g \in \mathcal{G}, m_C(g) = v_C \right\}$$

We first exhibit our representation φ of orderings[7] $f : A \to A^<$ into a subgroup $\mathcal{G}$ of $\mathrm{Sym}(A^2)$. However, this introduces an issue: $\mathcal{G}$ — which is actually the closure of $\varphi\left(\prod_i \mathcal{O}(A_i)\right)$ under composition of permutations — contains extraneous elements which do not correspond to any ordering of A.

In the second subsection, we show that $\varphi\left(\prod_i \mathcal{O}(A_i)\right)$ is a coset of a morphism-definable subgroup of $\mathcal{G}$. That is, we introduce a morphism $m_{init} : \mathcal{G} \to \mathrm{Sym}(A^3)$ and a value $v_{init} \in \mathrm{Sym}(A^3)$ such that

$$\prod \mathcal{O}(A_i) = \varphi^{-1}\left(m_{init}^{-1}(v_{init})\right).$$

Finally, we show how to transfer the morphism-definability of $\Delta_{i,j}$ to the space of labeling cosets: building on $m_{i,j}$, we define morphisms $\vartheta_{i,j}$ (for all i, j) such that $\vartheta_{i,j}(\varphi(\sigma)) = \vartheta_{i,j}(\varphi(\tau))$ iff σ and τ encode $E_{i,j}$ in the same way. This enables the restriction of a global coset to one which agrees with a semi-local one, as described in the introduction of Section 4.5.

Orderings as permutations

Fix some $\pi \in \prod_i \mathcal{O}(A_i)$, and let $\pi_i := \pi_{\restriction A_i}$. Recall that, by Corollary 4.18, $\pi\Gamma = \prod_i \mathcal{O}(A_i)$.

[7] Actually, φ does not enable the representation of *all* orderings of A, but only of those which are in $\prod \mathcal{O}(A_i)$

Let us first give an intuition of our construction: suppose we were given a "base" ordering $f : A \to A^<$. Then, there is a bijection mapping any $g : A \to A^<$ to the permutation of A that maps f to g through composition, that is, $f^{-1}g$. Here, while we obviously do not have access to such a fixed *single* ordering, for each colour class A_i, we have a canonical *family* $\pi_i\Gamma_i$ of $|A_i|$ "base" orderings[8], and therefore any $\sigma \in \pi_i\Gamma_i$ can be mapped injectively to ${\Gamma_i}^{A_i}$:

$$\begin{aligned}\varphi_i : \pi_i\Gamma_i &\to \Gamma^A \\ \sigma &\mapsto \left(b \mapsto \begin{cases} (\mathsf{map}_b^i)^{-1}\sigma & \text{if } b \in A_i \\ \mathrm{Id} & \text{otherwise} \end{cases} \right)\end{aligned}$$

This encoding is compatible with the morphism

$$\begin{aligned}\psi_i : \Gamma_i &\to \Gamma^A \\ \gamma &\mapsto \left(b \mapsto \begin{cases} \gamma & \text{if } b \in A_i \\ \mathrm{Id} & \text{otherwise} \end{cases} \right)\end{aligned}$$

in the sense that $\forall \sigma \in \pi_i\Gamma_i, \gamma \in \Gamma_i, \varphi_i(\sigma\gamma) = \varphi_i(\sigma)\psi_i(\gamma)$. Note that this implies that, for any $\sigma, \tau \in \pi_i\Gamma_i$,

$$\varphi_i(\sigma)^{-1}\varphi_i(\tau) = \psi_i(\sigma^{-1}\tau) \tag{4.6}$$

For any $\sigma \in \pi_i\Gamma_i$ and $\gamma \in \Gamma_i$, $\varphi_i(\sigma)$ and $\psi_i(\gamma)$ are families of elements of Γ indexed by A, and given $a \in A$, we denote the a-component of $\varphi_i(\sigma)$ (resp. $\psi_i(\gamma)$) by $\varphi_i(\sigma)_a$ (resp. $\psi_i(\gamma)_a$). Note that, while we have defined φ_i and ψ_i to range over Γ^A, their image is actually quite restricted: first, for any $a \in A$, $\varphi_i(\sigma)_a$ and $\psi_i(\gamma)_a$ are in Γ_i. Moreover, all the non-trivial values of $\varphi_i(\sigma)_a$ and $\psi_i(\gamma)_a$ are reached for $a \in A_i$. That is, morally, φ_i and ψ_i take values in ${\Gamma_i}^{A_i}$. However, providing a uniform codomain to all those functions is convenient, as it allows us to combine them easily into a global representation of $\pi\Gamma$ within Γ^A:

$$\begin{aligned}\varphi : \pi\Gamma &\to \Gamma^A \\ \sigma &\mapsto \prod_{i=1}^{m} \varphi_i(\sigma_{\restriction A_i}) \\ \psi : \Gamma &\to \Gamma^A \\ \gamma &\mapsto \prod_{i=1}^{m} \psi_i(\gamma_{\restriction A_i})\end{aligned}$$

φ is compatible with ψ, which implies that $\varphi(\pi\Gamma) = \varphi(\pi)\psi(\Gamma)$ is a coset of $\psi(\Gamma)$. Moreover, as was the case for φ_i and ψ_i:

$$\varphi(\sigma)^{-1}\varphi(\tau) = \psi(\sigma^{-1}\tau) \tag{4.7}$$

One can also easily check that φ_i, ψ_i are injective for all $i \leq m$, and thus so are φ and ψ.

[8]Recall that Lemma 4.17 implies that $\pi_i\Gamma_i = \mathcal{O}(A_i)$

As a side note, Γ^A is not *exactly* a permutation group, and as is, its definition is not compatible with the use of the ord operator. However, since $\Gamma \leq \mathrm{Sym}(A)$, we can identify Γ^A with a subgroup of $\mathrm{Sym}(A \times A)$ through the injective group morphism

$$\begin{aligned} \iota : \mathrm{Sym}(A)^A &\to \mathrm{Sym}(A \times A) \\ (\sigma_a)_{a \in A} &\mapsto ((a,b) \mapsto (a, \sigma_a(b))) \end{aligned}$$

More precisely, ι is an isomorphism between $\mathrm{Sym}(A)^A$ and the group of all permutations of $\mathrm{Sym}(A \times A)$ which act trivially on their first component. As such, we will usually keep the presence of ι implicit, and discuss the underlying groups, cosets, morphisms and elements of $\mathrm{Sym}(A)^A$, until we actually exhibit fixed-point definitions of said objects, where we rely on ι.

$\varphi(\pi\Gamma)$ is a coset of $\psi(\Gamma)$, and with ι we can expect to represent their elements within fixed-point logic, using the tools developped in Chapters 2 and 3. Precisely, we will show that $\psi(\Gamma)$ is morphism-definable, which will enable the representation of its coset $\varphi(\pi\Gamma)$ by defining, in FPC:

- A generating set for a group $\mathcal{G}$ that contains both $\varphi(\pi\Gamma)$ and $\psi(\Gamma)$ (as subsets)
- A morphism $m_{init} : \mathcal{G} \to \mathrm{Sym}(\Omega)$ such that $\ker(m_{init}) = \psi(\Gamma)$
- A value $v_{init} \in \mathrm{Sym}(\Omega)$ such that $m_{init}^{-1}(v_{init}) = \varphi(\pi\Gamma)$.

In this subsection, we deal with the first item on this list, and define such a group $\mathcal{G}$. Because $\varphi(\sigma) = \prod_{i=1}^m \varphi_i(\sigma_{\restriction A_i})$, and for each $\sigma \in \pi\Gamma$, $\sigma_{\restriction A_i} \in \Gamma_i$, we have

$$\varphi(\pi\Gamma) \subseteq \prod_{i=1}^m \varphi_i(\pi_i\Gamma_i) \leq \langle \bigcup_{i=1}^m \varphi_i(\pi_i\Gamma_i) \rangle$$

Therefore, $\mathcal{G} := \langle \bigcup_{i=1}^m \varphi_i(\pi_i\Gamma_i) \rangle$ satisfies the requirement above. Note in particular that $\mathcal{G}$ contains $\psi_i(\Gamma_i)$, as $\varphi_i(\Gamma_i)$ is a coset of this group, and $\mathcal{G}$ is closed under composition. We now show that a generating set for $\mathcal{G}$ can be defined in FPC:

Lemma 4.24. *There is a* FPC *formula* $\mathsf{gen}\mathcal{G}$ *which defines a generating set for* $\mathcal{G}$*.*

Proof. We remind the reader that we actually define a generating set for the group $\iota(\mathcal{G})$. By definition of $\mathcal{G}$, it is enough to build a formula $\mathsf{gen}\mathcal{G}$ such that, for any i, a,

$$\mathsf{gen}\mathcal{G}(\mathfrak{A}, i, a) = \mathrm{graph}(\iota(\varphi_i(\mathsf{map}_a^i))).$$

Consider the following formula:

$$\mathsf{gen}\mathcal{G}(p_1p_2, b_sx_s, b_tx_t) := (b_s = b_t) \wedge \textcircled{\vee} \begin{cases} i(x_s) = p_1 \wedge x_t = (\mathsf{map}_{b_s}^{p_1})^{-1}\mathsf{map}_{p_2}^{p_1}(x_s) \\ i(x_s) \neq p_1 \wedge x_s = x_t \end{cases}$$

In this formula, p_1, p_2 are the enumeration parameters of this generating set; and we denote $i(x)$ the unique j such that $x \in A_j$. Note that p_1 is numerical (and ranges over

the indices of the colour classes), while p_2 is a domain variable. $b_s x_s$ and $b_t x_t$ are the two pairs of permutation variables used to represent a permutation in $\mathrm{Sym}(A \times A)$ as in the definition of the ord operator.

For any $i, a \in A^< \times A$, $\mathfrak{A} \models \mathsf{gen}\mathcal{G}(i, a, \vec{s}, \vec{t})$ if $s_1 = t_1$ and $t_2 = (\mathsf{map}^i_{s_1})^{-1} \cdot \mathsf{map}^i_a(s_2)$, i.e., if $\vec{t} = \iota(\varphi_i(\mathsf{map}^i_a))(\vec{s})$, which yields the desired result. □

Representation of the initial labelling coset

Now, we show that $\psi(\pi\Gamma)$ is morphism-definable from $\mathcal{G}$ in FPC. Indeed, $\mathcal{G}$ contains many permutations which are not elements of $\varphi(\pi\Gamma)$. First, there are the elements of $\varphi_i(\pi_i\Gamma_i)$, but also the composition of elements in $\iota(\varphi(\pi\Gamma))$: for any two such elements $\sigma, \tau \in \pi\Gamma$,

$$(\varphi(\sigma)\varphi(\tau))_b = (\mathsf{map}^i_b)^{-1}\sigma(\mathsf{map}^i_b)^{-1}\tau$$

while for any ρ and $b \in A_i$, $\varphi(\rho)_b$ is of the form $(\mathsf{map}^i_b)^{-1}\gamma$, for some fixed $\gamma \in \Gamma_i$ (that does not depend on b), which implies that $\varphi(\sigma)\varphi(\tau) \notin \varphi(\pi\Gamma)$.

This section is devoted to the proof of the following:

Theorem 4.25. *There is an* FPC*-definable morphism* $m_{init} : \mathcal{G} \to \Gamma^{A\times A}$ *and an* FPC *definable value* $v_{init} \in \Gamma^{A\times A}$*, such that* $\varphi(\pi\Gamma) = \varphi(\pi)\psi(\Gamma) = \{\lambda \in \mathcal{G}, m_{init}(\lambda) = v_{init}\}$.

That is, we prove that $\psi(\Gamma)$ is morphism-definable from $\mathcal{G}$ in FPC (by the morphism m_{init}), and provide a FPC-definable value (v_{init}) which represent its coset $\varphi(\pi\Gamma)$ (w.r.t. the morphism m_{init}).

Proof. We give the definitions of m_{init} and v_{init} right away:

$$m_{init}(\lambda)_{a,b} := \begin{cases} \lambda_a\lambda_b^{-1} & \text{if } \exists i, \{a,b\} \subseteq A_i \\ \mathrm{Id} & \text{otherwise} \end{cases}$$

$$(v_{init})_{a,b} := \begin{cases} (\mathsf{map}^i_a)^{-1}\mathsf{map}^i_b & \text{if } \exists i, \{a,b\} \subseteq A_i \\ \mathrm{Id} & \text{otherwise} \end{cases}$$

The following Lemmas 4.26 to 4.28 show, respectively, that m_{init} is indeed a morphism, that $m^{-1}_{init}(v_{init}) = \varphi(\pi\Gamma)$, and that m_{init} and v_{init} are FPC-definable, which altogether proves the theorem. □

Lemma 4.26. $m_{init} : \mathcal{G} \to \Gamma^{A\times A}$ *is a morphism*

Proof. Consider $\lambda, \lambda' \in \mathcal{G}$. For any $i \neq j$, $a \in A_i$ and $b \in A_j$, $m_{init}(\lambda\lambda')_{a,b} = \mathrm{Id} =$

$(m_{init}(\lambda)_{a,b})(m_{init}(\lambda')_{a,b})$. For $a, b \in A_i$,

$$\begin{aligned} m_{init}(\lambda\lambda')_{a,b} &= (\lambda\lambda')_a(\lambda\lambda')_b^{-1} \\ &= \lambda_a\lambda'_a{\lambda'_b}^{-1}\lambda_b^{-1} \\ \text{Since } \Gamma_i \text{ is abelian, } &= \lambda_a\lambda_b^{-1}\lambda'_a{\lambda'_b}^{-1} \\ &= m_{init}(\lambda)_{a,b} \cdot m_{init}(\lambda')_{a,b} \end{aligned}$$

□

Lemma 4.27. $\varphi(\pi\Gamma) = \{\lambda \in \mathcal{G}, m_{init}(\lambda) = v_{init}\}$

Proof. For $\sigma \in \pi\Gamma$ and $a, b \in A_i$, we have:

$$\begin{aligned} m_{init}(\varphi(\sigma))_{a,b} &= \varphi(\sigma)_a\varphi(\sigma)_b^{-1} \\ &= (\mathsf{map}_a^i)^{-1}\sigma\sigma^{-1}\mathsf{map}_b^i \\ &= (v_{init})_{a,b} \end{aligned}$$

We now show the other inclusion. Let $\lambda \in \mathcal{G}$ such that $m_{init}(\lambda) = v_{init}$. Fix, for all i, some $a_i \in A_i$, and let $\sigma_i := \mathsf{map}_{a_i}^i\lambda_{a_i}$. Then, for any $b \in A_i$,

$$\begin{aligned} \mathsf{map}_b^i\lambda_b &= \mathsf{map}_b^i\lambda_b\lambda_{a_i}^{-1}\lambda_{a_i} \\ &= \mathsf{map}_b^i m_{init}(\lambda)_{b,a_i}\lambda_{a_i} \\ &= \mathsf{map}_b^i(\mathsf{map}_b^i)^{-1}\mathsf{map}_{a_i}^i\lambda_{a_i} \\ &= \sigma_i \end{aligned}$$

Thus, for any b, $\lambda_b = (\mathsf{map}_b^{i(b)})^{-1}\sigma_{i(b)} = \varphi_{i(b)}(\sigma_{i(b)})_b$, which in turn implies that $\lambda = \varphi(\sigma_1 \dots \sigma_m)$. □

Finally, we show that m_{init} is definable in FPC. Recall that the domain of m_{init} is Γ^A, which, in the context of fixed-point logic, is represented as a subgroup of $\mathrm{Sym}(A \times A)$ through ι. A similar shift of representation is needed for the codomain $\Gamma^{A\times A}$ of m_{init}, which can be embedded in $\mathrm{Sym}(A \times A \times A)$ through

$$\begin{aligned} \iota_2 : \mathrm{Sym}(A)^{A\times A} &\to \mathrm{Sym}(A \times A \times A) \\ (\sigma_{(a,b)})_{(a,b)\in A\times A} &\mapsto \big((a,b,c) \mapsto (a,b,\sigma_{(a,b)}(c))\big) \end{aligned}$$

Lemma 4.28.

- *There is a formula* $\mathsf{initMorph}(R, a_s, b_s, x_s, a_t, b_t, x_t)$*, such that, for any* $\lambda \in \mathcal{G}$*,* $\mathsf{initMorph}(\mathfrak{A}, \mathrm{graph}(\iota(\lambda))) = \mathrm{graph}(\iota_2(m_{init}(\lambda)))$*.*
- *There is a formula* $\mathsf{initValue}(a_s, b_s, x_s, a_t, b_t, x_t)$ *that defines in* $\mathfrak{A}$ *the graph of* $\iota_2(v_{init})$*.*

Proof.

$$\mathsf{initMorph}(R, a_s, b_s, x_s, a_t, b_t, x_t) := \begin{cases} a_s = a_t \\ \textcircled{\wedge} b_s = b_t \\ \exists y, R(b_s, x_t, b_s, y) \wedge R(a_s, x_s, a_s, y) \end{cases}$$

$$\mathsf{initValue}(a_s, b_s, x_s, a_t, b_t, x_t) := \begin{cases} a_s = a_t \\ \textcircled{\wedge} b_s = b_t \\ \exists i, \mu, \mathsf{map}(i, a_s, x_t, \mu) \wedge \mathsf{map}(i, b_s, x_s, \mu) \end{cases}$$

□

Extending the semi-local morphisms

In the previous section, we have introduced morphisms $m_{i,j} : \Gamma \to \mathrm{Sym}(\Omega_{i,j})$ such that $\ker(m_{i,j}) = \Gamma \cap \mathrm{Aut}(\mathfrak{A}_{i,j})$. We motivated the introduction of those morphisms by the need to upkeep the algebraic structure of a labelling coset together with the canonical encoding of the graph at hand. However, managing labelling cosets through morphism-definability prompted a shift of representation of those labelling cosets. Now that we have defined our FPC-definable representation of labelling cosets — as cosets of morphism-definable subgroups of $\mathcal{G}$ — and building upon the morphisms $m_{i,j}$, we define morphisms from $\mathcal{G}$ to $\Omega_{i,j}$ that will allow us to enforce an encoding of $E_{i,j}$, while maintaining the algebraic structure of the underlying labelling coset.

Precisely, we aim to construct a morphism $\vartheta_{i,j}$ on $\mathcal{G}$ and a value $v_{i,j}(a,b) \in \mathrm{Im}(\vartheta_{i,j})$ for each colours $i < j$ and $(a,b) \in A_i \times A_j$ such that, for all $(a,b) \in A_i \times A_j$, $\vartheta_{i,j}^{-1}(v_{i,j}(a,b)) \cap \varphi(\pi\Gamma)$ is exactly the set of orderings that yield the same encoding of $E_{i,j}$ as $\mathsf{map}_a^i \mathsf{map}_b^j$.

Theorem 4.29. *For each $i < j \leq m$, there is a* FPC *definable morphism*

$$\vartheta_{i,j} : \mathcal{G} \to \mathrm{Sym}(A \times A \times \Omega_{i,j})$$

and a FPC *definable function $v_{i,j} : A_i \times A_j \to \mathrm{Sym}(A \times A \times \Omega_{i,j})$ such that, for any $(a,b) \in A_i \times A_j$ and $\sigma \in \pi\Gamma$,*

$$\vartheta_{i,j}(\varphi(\sigma)) = v_{i,j}(a,b) \iff E_{i,j}^{\sigma} = E_{i,j}^{\mathsf{map}_a^i \mathsf{map}_b^j}$$

(where R^f is the encoding of R relative to f, as defined below Definition 4.14)

Proof. We provide the definition of $\vartheta_{i,j}$ and $v_{i,j}$. For readability purposes, we present $\vartheta_{i,j}(\sigma)$ and $v_{i,j}(a,b)$ as elements of $\mathrm{Sym}(\Omega_{i,j})^{A \times A}$ rather than $\mathrm{Sym}(A \times A \times \Omega_{i,j})$, relying on the injective morphism

$$\begin{aligned} \iota_3 : \mathrm{Sym}(\Omega_{i,j})^{A \times A} &\to \mathrm{Sym}(A \times A \times \Omega_{i,j}) \\ (\sigma_{(a,b)})_{(a,b) \in A \times A} &\mapsto ((a,b,\omega) \mapsto (a,b,\sigma_{(a,b)}(\omega))) \end{aligned}$$

as in the previous subsection.

$$\vartheta_{i,j} : \mathcal{G} \to \mathrm{Sym}(\Omega_{i,j})^{A\times A}$$
$$\lambda \mapsto \left((a,b) \mapsto \begin{cases} m_{i,j}(\lambda_a \lambda_b) & \text{if } a \in A_i, b \in A_j \\ \mathrm{Id} & \text{otherwise} \end{cases} \right)$$

For $(a_i, a_j) \in A_i \times A_j$, let $v_{i,j}(a_i, a_j) \in \mathrm{Sym}(\Omega_{i,j})^{A\times A}$ be defined, for any $(a,b) \in A \times A$, as

$$v_{i,j}(a_i,a_j)(a,b) := \begin{cases} m_{i,j}((\mathsf{map}^i_a \mathsf{map}^j_b)^{-1} \mathsf{map}^i_{a_i} \mathsf{map}^j_{a_j}) & \text{if } a \in A_i, b \in A_j \\ \mathrm{Id} & \text{otherwise} \end{cases}$$

In Lemmas 4.30, 4.31 and 4.33, we show, respectively, that $\vartheta_{i,j}$ is a morphism, that it behaves as expected regarding to the encoding of $E_{i,j}$, and that both $\vartheta_{i,j}$ and $v_{i,j}$ are FPC-definable, which altogether yield the theorem. □

Lemma 4.30. *For all $i < j \leq m$, $\vartheta_{i,j}$ is a morphism.*

Proof. Consider $\lambda, \lambda' \in \mathcal{G}$ and $(a,b) \in A \times A$. We aim to show that

$$(\vartheta_{i,j}(\lambda)\vartheta_{i,j}(\lambda'))_{(a,b)} = \vartheta_{i,j}(\lambda\lambda')_{(a,b)}$$

If $(a,b) \notin A_i \times A_j$, both sides of this equation evaluate to Id. Otherwise,

$$\begin{aligned} (\vartheta_{i,j}(\lambda)\vartheta_{i,j}(\lambda'))_{(a,b)} &= m_{i,j}(\lambda_a\lambda_b) m_{i,j}(\lambda'_a\lambda'_b) \\ &= m_{i,j}(\lambda_a\lambda_b\lambda'_a\lambda'_b) && \text{since } m_{i,j} \text{ is a morphism} \\ &= m_{i,j}(\lambda_a\lambda'_a\lambda_b\lambda'_b) && \text{since } \Gamma_i\Gamma_j \text{ is abelian} \\ &= \vartheta_{i,j}(\lambda\lambda')_{(a,b)} && \square \end{aligned}$$

Lemma 4.31. *For any $i < j$, $(a_i, a_j) \in A_i \times A_j$, and $\sigma \in \pi\Gamma$,*

$$\vartheta_{i,j}(\varphi(\sigma)) = v_{i,j}(a_i,a_j) \iff E^\sigma_{i,j} = E^{\mathsf{map}^i_{a_i}\mathsf{map}^j_{a_j}}_{i,j}$$

Proof. First, notice that, for any $i < j$ and $a_i \in A_i, a_j \in A_j$, $v_{i,j}(a_i,a_j) = \vartheta_{i,j}(\varphi(\sigma))$, where σ is any element of $\pi\Gamma$ such that $\sigma_{\restriction A_i \cup A_j} = \mathsf{map}^i_{a_i}\mathsf{map}^j_{a_j}$.

It is thus only left to prove that, for two labellings $\sigma, \tau \in \pi\Gamma$, $\vartheta_{i,j}(\varphi(\sigma)) = \vartheta_{i,j}(\varphi(\tau))$ iff σ and τ yield the same encoding of $E_{i,j}$. Equation (4.7) implies that

$$\vartheta_{i,j}(\varphi(\sigma)) = \vartheta_{i,j}(\varphi(\tau)) \iff \vartheta_{i,j}(\psi(\sigma^{-1}\tau)) = 1$$

so that it is only left to show that, $\psi(\gamma) \in \ker(\vartheta_{i,j}) \iff \gamma \in \mathrm{Aut}(\mathfrak{A}_{i,j})$. And indeed:

$$\begin{aligned} \vartheta_{i,j}(\psi(\gamma)) = 1 &\iff \forall a \in A_i, b \in A_j, m_{i,j}(\psi(\gamma)_a \psi(\gamma)_b) = 1 \\ &\iff \forall a \in A_i, b \in A_j, m_{i,j}(\gamma_{\restriction A_i}\gamma_{\restriction A_j}) = 1 && \text{by definition of } \psi \\ &\iff \gamma_{\restriction A_i \cup A_j} \in \ker(m_{i,j}) \\ &\iff \gamma_{\restriction A_i \cup A_j} \in \mathrm{Aut}(m_{i,j}) && \text{by definition of } m_{i,j} \end{aligned}$$

□

Note that, in particular, this implies the following:

Corollary 4.32. *For any two couples* $(a, b), (a', b') \in A_i \times A_j$,

$$E_{i,j}^{\mathsf{map}_a^i \mathsf{map}_b^j} = E_{i,j}^{\mathsf{map}_{a'}^i \mathsf{map}_{b'}^j} \implies v_{i,j}(a, b) = v_{i,j}(a', b').$$

This fact will become of particular importance in the following section. Recall that, at each iteration of the for-loop, we must find the smallest encoding $E_{i,j}^{<}$ of $E_{i,j}$ which is coherent with previous iterations of the for-loop (and thus with all $E_{i',j'}^{<}$ where $(i', j') <_{lex} (i, j)$). To do so, we will pick a couple $(a, b) \in A_i \times A_j$ compatible[9] with the current labelling coset such that $E_{i,j}^{\mathsf{map}_a^i \mathsf{map}_b^i}$ is minimal and set $E_{i,j}^{<}$ to $E_{i,j}^{\mathsf{map}_a^i \mathsf{map}_b^i}$. To ensure, in further iterations of the for-loop, that all couples considered are compatible with this semi-local encoding $E_{i,j}^{<}$, the current value of the labelling coset $\mathcal{C}$ must be updated, which entails to store the constraint

$$\forall \sigma \in \mathcal{C}, \vartheta_{i,j}(\varphi(\sigma)) = v_{i,j}(a, b)$$

i.e. the couple $(\vartheta_{i,j}, v_{i,j}(a, b))$ must be added to a relation within a fixed-point computation. However, the couple (a, b) was picked arbitrary amongst all couples (a', b') compatible with $\mathcal{C}$ yielding the minimal encoding of $E_{i,j}$. For this storage of the couple $(\vartheta_{i,j}, v_{i,j}(a, b))$ to be definable, it must be the case that the value of $v_{i,j}(a, b)$ does not depend on this choice, which is exactly what Corollary 4.32 proves.

To prove Theorem 4.29, it is only left to show that $\vartheta_{i,j}$ and $v_{i,j}$ are FPC-definable.

Lemma 4.33. *There is a* FPC *formula* $\vartheta(\mu, \nu, R, \vec{s}, \vec{t})$ *with* $\mathrm{type}(\vec{s}) = \mathrm{type}(\vec{t}) = \mathrm{element}^2\mathrm{number}^2$ *such that, for any* $\sigma \in \varphi(\pi\Gamma)$ *and* $i \neq j \leq m$,

$$\vartheta(\mathfrak{A}, i, j, \mathrm{graph}(\iota(\sigma))) = \mathrm{graph}(\iota_3(\vartheta_{i,j}(\sigma))).$$

There is a FPC *formula* $\mathsf{v}(\mu, \nu, x, y, \vec{s}, \vec{t})$ *with* $\mathrm{type}(\vec{s}) = \mathrm{type}(\vec{t}) = \mathrm{element}^2\mathrm{number}^2$ *such that, for any* $i \neq j \leq m$,

$$\mathsf{v}(\mathfrak{A}, i, j, a, b) = \mathrm{graph}(\iota_3(v_{i,j}(a, b))).$$

Proof. Let us first define $v_{i,j}$ as a formula $\mathsf{v}(\mu, \nu, x, y, \vec{s}, \vec{t})$. The variables μ, ν track the pair of colour-classes we are currently handling and x, y track the component of $v_{i,j}$ we are defining, i.e. we aim to obtain $\mathsf{v}(\mathfrak{A}, i, j, a_i, a_j) = \mathrm{graph}(v_{i,j}(a_i, a_j))$. Note that $v_{i,j}(a_i, a_j)$ is represented as acting on $A \times A \times \Omega_{i,j}$. Thus, the tuples of variables $\vec{s}$ and $\vec{t}$ are meant to represent individual elements of this set, and for readability purposes, we name those variables in accordance with our definition of $v_{i,j}$ above: $\vec{s} = (a_s, b_s, \mu_s, \nu_s)$

[9]The precise $\mathsf{FP} + \mathsf{ord}$ definition of this compatibility relation will be defined in the following section

and $\vec{t} = (a_t, b_t, \mu_t, \nu_t)$; with (μ_s, ν_s) and (μ_t, ν_t) representing elements of $\Omega_{i,j}$.

$$\mathsf{v}(\mu,\nu,x,y,\vec{s},\vec{t}) := \begin{cases} (x \notin A_\mu \vee y \notin A_\nu) \wedge \vec{s} = \vec{t} \\ (a_s \notin A_\mu \vee b_s \notin A_\nu) \wedge \vec{s} = \vec{t} \\ \begin{cases} x \in A_\mu \wedge y \in A_\nu \\ a_s \in A_\mu \wedge b_s \in A_\nu \\ a_s = a_t \wedge b_s = b_t \\ \mathsf{slMorph}(\mu,\nu,R_g,\mu_s\nu_s,\mu_t\nu_t)[R_g(\alpha,\beta)/\xi(\alpha,\beta)] \end{cases} \end{cases}$$

where

$$\xi(\alpha,\beta) := \begin{cases} \alpha \in A_\mu \wedge \beta \in A_\mu \wedge \exists\lambda, \mathsf{map}(\mu,x,\alpha,\lambda) \wedge \mathsf{map}(\mu,a_s,\beta,\lambda) \\ \alpha \in A_\nu \wedge \beta \in A_\nu \wedge \exists\lambda, \mathsf{map}(\nu,y,\alpha,\lambda) \wedge (\mathsf{map}(\nu,b_s,\beta,\lambda)) \end{cases}$$

Recall that map defines the local-labellings map^i_a as shown in Lemma 4.16, that $\mathsf{slMorph}$ is the formula defining the morphisms $m_{i,j}$, as shown in Lemma 4.23 and that $F[G/H]$ denotes the formula F where each occurence of the subformula G is substituted by the subformula H. Thus, for any suitable $\vec{a} = (i, j, a_i, a_j, a, b)$, an assignment of the free variables $(\mu, \nu, x, y, a_s, b_s)$ of ξ, $\xi(\mathfrak{A}, \vec{a})$ defines the graph of $(\mathsf{map}^i_a \mathsf{map}^j_b)^{-1} \mathsf{map}^i_{a_i} \mathsf{map}^j_{a_j} \in \Gamma_i\Gamma_j$. As such, the last clause in the definition of v correctly defines the graph (on the variables $\mu_s\nu_s, \mu_t\nu_t$) of $m_{i,j}((\mathsf{map}^i_a \mathsf{map}^j_b)^{-1} \mathsf{map}^i_{a_i} \mathsf{map}^j_{a_j})$.

We now turn to the definition of the morphisms $\vartheta_{i,j}$. We provide a formula $\vartheta(\mu, \nu, R, \vec{s}, \vec{t})$, where once again, (μ, ν) tracks the couple of colour-classes at hand, and now, R should be the graph of an element $g \in \iota(\mathcal{G})$.

As in the definition of v, $\vec{s}$ and $\vec{t}$ represent the domain $A \times A \times \Omega_{i,j}$ of the permutational image of the morphism $\vartheta_{i,j}$, and we name each individual variable a_s, b_s, μ_s, ν_s (resp. a_t, b_t, μ_t, ν_t) to improve readability.

$$\vartheta(\mu,\nu,R,\vec{s},\vec{t}) := \begin{cases} a_s = a_t \\ b_s = b_t \\ \mathsf{slMorph}(\mu,\nu,R_g,\mu_s\nu_s,\mu_t\nu_t)[R_g(x,y)/\theta(x,y)] \end{cases}$$

where

$$\theta(x,y) := \exists w, R(a_s,x,a_s,w) \wedge R(b_s,w,b_s,y)$$

One should be careful not to confuse R_g, which is the second-order variable in $\mathsf{slMorph}$ that should take as value the graph of a permutation in $\Gamma_i\Gamma_j$, and R, which is used in ϑ as a place-holder for the graph of a permutation in $\mathcal{G}$, so that $\vartheta(\mu, \nu, R, \vec{s}, \vec{t})$ defines a morphism as explained in Definition 3.10. □

4.7 Conclusion of the proof

We can now present formally our *representation* of labelling cosets. Any such $\mathcal{C} \subseteq \pi\Gamma$ is represented by a couple $(m_\mathcal{C}, v_\mathcal{C})$, where $m_\mathcal{C} : \mathcal{G} \to \mathrm{Sym}(\Omega)$ and $v_\mathcal{C} \in \mathrm{Sym}(\Omega)$ for

some set Ω, such that

$$\mathcal{C} = \{\varphi^{-1}(g), g \in \mathcal{G}, m_{\mathcal{C}}(g) = v_{\mathcal{C}}\} \tag{4.8}$$

where $\mathcal{G}$ and φ are the group and function presented in the last subsection. We now show how this enables every step of Algorithm 3.

Initialisation of the labelling coset (Line Line 3 of Algorithm 3) Recall that, at this stage $\mathcal{C} = \pi\Gamma = \prod \mathcal{O}(A_i)$. Our objective is thus to present a representation of this labelling coset. According to Eq. (4.8), we aim to find a couple (m, v) such that $\varphi(\pi\Gamma) = \{g \in \mathcal{G}, m(g) = v\}$. But we have already shown in Lemma 4.27 that (m_{init}, v_{init}) verifies exactly that equality, and thus represent $\pi\Gamma$. Note that this is a prerequisite for the application of φ^{-1} to be well-defined in Eq. (4.8) (as $\varphi(\pi\Gamma)$ is a strict subset of $\mathcal{G}$). In this case, $\Omega = A^3$.

We are now entering the for-loop, fixing $i < j$. Assume that $\mathcal{C}$ is represented by (m_c, v_c) as in Eq. (4.8). Two operations still must be implemented :

- Finding the lexicographically minimal encoding of $E_{i,j}$ coherent with $\mathcal{C}$ (Line 4 of Algorithm 3).
- Updating our coset $\mathcal{C}$, and thus its representation $(m_{\mathcal{C}}, v_{\mathcal{C}})$ according to this encoding of $E_{i,j}$ (Line 5 of Algorithm 3).

Finding the smallest coherent encoding of $E_{i,j}$ By definition, for an encoding of $E_{i,j}$ to be coherent with $\mathcal{C}$, it needs to be the encoding of $E_{i,j}$ relative to some labelling $\sigma \in \mathcal{C}$. As $\mathcal{C} \subseteq \pi\Gamma$, any such labelling σ must fulfill the equality

$$\sigma_{\restriction A_i \cup A_j} = \mathsf{map}^i_a \mathsf{map}^i_b$$

for some $(a, b) \in A_i \times A_j$. Moreover, given such a couple $(a, b) \in A_i \times A_j$, we can define the corresponding encoding of $E_{i,j}$ using $\mathrm{enc}^{\mathsf{map}(i,a)\vee\mathsf{map}(i,b)}_{E_{i,j}}$ as shown in Lemma 4.15. It is also quite straight-forward, given $(a, b), (a', b') \in A_i \times A_j$ two such couples, to check whether the encoding generated by $\mathsf{map}^i_a \mathsf{map}^j_b$ is lexicographically smaller than the one generated by $\mathsf{map}^i_{a'} \mathsf{map}^j_{b'}$.

Therefore, it is only left to define in FPC a compatibility relation $\mathrm{Comp}_{\mathcal{C}} \subseteq A_i \times A_j$, such that

$$(a, b) \in \mathrm{Comp}_{\mathcal{C}} \iff \exists \sigma \in \mathcal{C}, \sigma_{\restriction A_i \cup A_j} = \mathsf{map}^i_a \mathsf{map}^j_b$$

Then, we can order $\mathrm{Comp}_{\mathcal{C}}$ according to the lexicographical order on the encodings generated by any two couples $(a, b), (a', b')$. Therefore, setting

$$E^{<}_{i,j}(\vec{\mu}, \vec{\nu}) := \exists (a_0, b_0) \in \mathrm{Comp}_{\mathcal{C}}, \textcircled{\wedge} \begin{cases} \mathrm{enc}^{\mathsf{map}(i,a_0)\vee\mathsf{map}(j,b_0)}_{E_{i,j}} \text{ is minimal} \\ \mathrm{enc}^{\mathsf{map}(i,a_0)\vee\mathsf{map}(j,b_0)}_{E_{i,j}}(\vec{\mu}, \vec{\nu}) \end{cases}$$

is a $\mathsf{FP} + \mathsf{ord}$-definition of the minimal encoding of $E_{i,j}$ coherent with $\mathcal{C}$.

Let us now show how the representation of labelling cosets given in Eq. (4.8) enables the definition of $\mathrm{Comp}_{\mathcal{C}}$. For any $(a, b) \in A_i \times A_j$, Theorem 4.29 ensures that

$$\begin{aligned}(a,b) \in \mathrm{Comp}_{\mathcal{C}} &\iff \vartheta_{i,j}^{-1}(v_{i,j}(a,b)) \cap \varphi(\mathcal{C}) \neq \emptyset \\ &\iff \vartheta_{i,j}^{-1}(v_{i,j}(a,b)) \cap m_{\mathcal{C}}^{-1}(v_{\mathcal{C}}) \neq \emptyset\end{aligned}$$

Using the morphism constructions we have defined in Section 3.3 (in particular, in Definition 3.16)

$$\begin{aligned}(a,b) \in \mathrm{Comp}_{\mathcal{C}} &\iff (\vartheta_{i,j} \otimes m_{\mathcal{C}})^{-1}(v_{i,j}(a,b) \oplus v_{\mathcal{C}}) \neq \emptyset \\ &\iff (v_{i,j}(a,b) \oplus v_{\mathcal{C}}) \in \mathrm{Im}(\vartheta_{i,j} \otimes m_{\mathcal{C}})\end{aligned}$$

By induction hypothesis, $m_{\mathcal{C}}$ and $v_{\mathcal{C}}$ are $\mathsf{FP} + \mathsf{ord}$ definable, and we showed in Theorem 4.29 that $\vartheta_{i,j}$ and $v_{i,j}(a, b)$ are FPC definable. We have showed in Lemma 3.18 that under those assumptions, $\vartheta_{i,j} \otimes m_{\mathcal{C}}$ is definable and therefore, by Lemma 3.14, a generating set for $\mathrm{Im}(\vartheta_{i,j} \otimes m_{\mathcal{C}})$ is definable. According to Lemma 3.19, $(v_{i,j}(a, b) \oplus v_{\mathcal{C}})$ is also definable, and as permutation group membership is decidable in $\mathsf{FP} + \mathsf{ord}$,[10], we can construct in $\mathsf{FP} + \mathsf{ord}$ a formula that holds iff $(a, b) \in \mathrm{Comp}_{\mathcal{C}}$, which concludes our implementation of line 4 of Algorithm 3.

Updating the labelling coset representation At this point, we have successfully used our new representation of labelling cosets to obtain a canonical encoding $E^{<}_{i,j}$ of the edge-relation between two colour-classes A_i, A_j coherent with a given labelling coset $\mathcal{C}$. We must now enforce our induction hypothesis, and show that we can define an encoding of the resulting, further restricted labelling coset

$$\mathcal{C}' := \{\sigma \in \mathcal{C} \mid E^{\sigma}_{i,j} = E^{<}_{i,j}\}$$

Most of the work was already done in the last paragraph: setting $m_{\mathcal{C}'} := m_{\mathcal{C}} \otimes \vartheta_{i,j}$ and $v_{\mathcal{C}'} := v_{\mathcal{C}} \oplus v_{i,j}(a_0, b_0)$ for any (a_0, b_0) minimal in $\mathrm{Comp}_{\mathcal{C}}$ yields the desired result. Note that Corollary 4.32 ensures that the value of $v_{\mathcal{C}'}$ does not depends on which couple (a_0, b_0) we use.

This concludes our informal presentation of our encoding of labelling cosets in $\mathsf{FP} + \mathsf{ord}$. However, before we introduce the formulae implementing this idea, there are a few things left unclear about the recursive construction of a complex morphism in $\mathsf{FP} + \mathsf{ord}$. In particular, notice, in the last step we have depicted, that the use of the $\otimes$ operator induces an increase of the base set Ω on which $v_{\mathcal{C}'}$ and elements in $\mathrm{Im}(m_{\mathcal{C}'})$ act. However, this growth is quite tame, and we will see how we can actually define this sequence of morphisms in $\mathsf{FP} + \mathsf{ord}$.

A shift in the representation of morphisms

Defining a sequence of morphisms with growing codomains raises the problem of the fixed arity of relations in fixed-point logic. However, we have already defined, in the

[10]Note that this is the only use of the ord operator so far

previous section, the building blocks for the morphisms $m_{\mathcal{C}}$ to be considered all along the run of Algorithm 3, and we will now see how this enables us to avoid such a growth of the arity of the formulae defining the relevant morphisms, through a small shift of representation.

Let $W \subseteq [m]^2$ be the set of pairs of colours that have already been processed in the for-loop of Algorithm 3. Then, it is easy to see that (up to reordering of its components) $m_{\mathcal{C}}$ is equal to

$$\Theta_W := \left(\bigotimes_{(i,j)\in W} \vartheta_{i,j} \right) \otimes m_{init}$$

W defines a "window", a set of indices on which the values of the $\vartheta_{i,j}$ are currently relevant. Let $W(i,j) := \{(i',j') \mid (i',j') <_{lex} (i,j)\}$ so that $\Theta_{W(i,j)}$ is exactly the morphism $m_{\mathcal{C}}$ encoding the value of $\mathcal{C}$ when starting to treat the couple (i,j). After one iteration of the for-loop, the morphism $m_{\mathcal{C}'}$ would be $m_{\mathcal{C}'} = \Theta_{W(i,j)\cup\{(i,j)\}}$. We will now show that the family of morphisms $(\Theta_W)_{W\in\mathcal{P}([m]^2)}$ (or rather, a somehow nicer encoding of those morphisms, as we will see below) are directly definable within FPC. In this context, only the permutation $v_{\mathcal{C}} \in \mathrm{Im}(\Theta_{W(i,j)})$ and the encoding of the edge relation associated with the labelling coset $\Theta^{-1}_{W(i,j)}(v_{\mathcal{C}}) \subseteq \pi\Gamma$ will need to be carried along the fixed-point computation.

Note that, at this stage, the codomain growth problem we have mentioned subsists: when $W \subset W'$, the permutations in $\Theta_{W'}(\mathcal{G})$ act on a larger set than those in $\Theta_W(\mathcal{G})$. This can now be easily circumvented by building a morphism equivalent to Θ_W whose codomain is a constant permutation group (which should be codomain of $\Theta_{[m]^2}$), for all values of W. The following morphisms satisfy those requirements:

$$\Xi_W := \left(\bigotimes_{(i,j)\in[m]^2} \underline{\vartheta}^W_{i,j} \right) \otimes m_{init}$$

where

$$\underline{\vartheta}^W_{i,j} := \begin{cases} \vartheta_{i,j} & \text{if } (i,j) \in W \\ \mathrm{Id} & \text{otherwise} \end{cases}$$

Note that, for any values of W, i, j and $g \in \mathcal{G}$, $\underline{\vartheta}^W_{i,j}(g) \in \mathrm{Sym}(A^2 \times (A^<)^2)$. As such, for any W, Lemma 3.17 enables the representation of $\bigotimes_{(i,j)} \underline{\vartheta}^W_{i,j}$ as a morphism $\otimes\underline{\vartheta}^W : \mathcal{G} \to \mathrm{Sym}((A^<)^2 \times A^2 \times (A^<)^2)$, and thus, the $\otimes$-product of $\otimes\underline{\vartheta}^W$ with m_{init} yields:

$$\Xi_W(g) \in \mathrm{Sym}((A^<)^2 \times A^2 \times (A^<)^2) \times \mathrm{Sym}(A \times A \times A)$$

Recall that we can identify $\mathrm{Sym}(X) \times \mathrm{Sym}(Y)$ with a subgroup of $\mathrm{Sym}(X \times Y)$. Let us now show that Ξ_W is definable in FPC. We start by defining the morphisms $\underline{\vartheta}$:

$$\underline{\vartheta}(R_W, \mu, \nu, R, \vec{s}, \vec{t}) := \begin{cases} R_W(\mu,\nu) \wedge \vartheta(\mu,\nu,R,\vec{s},\vec{t}) \\ \neg R_W(\mu,\nu) \wedge \vec{s} = \vec{t} \end{cases}$$

where, for all $i < j$, $\vartheta(\mathfrak{A}, i, j)$ defines the morphism $\vartheta_{i,j}$, as was showed in Lemma 4.33. In the same way, for any W, i, j, $\underline{\vartheta}(\mathfrak{A}, W, i, j)$ defines $\underline{\vartheta}^W_{i,j}$. Building on $\underline{\vartheta}$, we can construct the inner $\otimes$-product of morphisms in the definition of Ξ_W:

$$\mathsf{compound}\vartheta(R_W, R, \mu_s\nu_s\vec{s}, \mu_t\nu_t\vec{t}) := \circledwedge \begin{cases} \mu_s = \mu_t \wedge \nu_s = \nu_t \\ \underline{\vartheta}(R_W, \vec{\nu}s, R, \vec{s}, \vec{t}) \end{cases}$$

The definition of $\mathsf{compound}\vartheta$ is a direct application of Lemma 3.18. It remains to build the $\otimes$-product of this morphism with m_{init}:

$$\Xi(R_W, R, \mu_s\nu_s b_s\vec{s}\vec{s'}, \mu_t\nu_t b_t\vec{t}\vec{t'}) := \circledwedge \begin{cases} \mu_s = \mu_t \wedge \nu_s = \nu_t \wedge b_s = b_t \\ \circledvee \begin{cases} b_s = 0 \wedge \mathsf{compound}\vartheta(R_W, R, \mu_s\nu_s\vec{s}, \mu_t\nu_t\vec{t}) \\ b_s = 1 \wedge \mathsf{initMorph}(R, \vec{s'}, \vec{t'}) \end{cases} \end{cases}$$

where $\mathsf{initMorph}$ is the FPC-formula shown in Lemma 4.28 to define m_{init}.

Defining the corresponding FP + ord formulae

We are now ready to delve into the formal definition of the formula that canonises graphs with abelian colours in FP + ord. We aim to build formulae $\phi_E, \phi_{v_\mathcal{C}}$ such that, given two relations $R_E, R_{v_\mathcal{C}}$ encoding respectively the values of $E^< = \bigcup E^<_{i,j}$ and the permutation $v_\mathcal{C}$ resulting from the iteration of the for-loop pertaining to the couple (i, j) in Algorithm 3, $\phi_E(\mathfrak{A}, R_E, R_{v_\mathcal{C}})$ and $\phi_{v_\mathcal{C}}(\mathfrak{A}, R_E, R_{v_\mathcal{C}})$ encode the values $E'^<$ and $v_{\mathcal{C}'}$ obtained after one additional iteration, that is, after treating the couple $(i, j+1)$ (or $(i+1, i+2)$ if $j = m$). Once such a couple of formulae is defined, the result of the algorithm is easily definable using a simultaneous fixed-point, which can be simulated in FPC.

Theorem 4.34. *There is a* FP + ord *formula* $\varphi_{E^<}$ *that canonises graphs with abelian colours.*

Proof. Let us summarise how the relations R_E and $R_{v_\mathcal{C}}$ are meant to encode the state of the computation. The following invariants will be maintained along the fixed-point computation:

- R_E is a ternary, numerical relation, such that, for any $\mu, \nu \leq |A|$ and $\lambda \in \{0, 1\}$, $R_E(\mu, \nu, \lambda)$ holds iff $E^<$ contains the bit λ at position (μ, ν).[11]
- $R_{v_\mathcal{C}}$ is a relation of type $(\text{number}^2 \cdot \text{element}^2 \cdot \text{number}^2)^2$, which is the graph of a permutation in $(A^<)^2 \times A^2 \times (A^<)^2$.

The initial values of those two relations are easily definable:

$$R_{\mathsf{init}E}(\mu, \nu, \lambda) := \bot$$
$$R_{\mathsf{init}v}(\vec{s}, \vec{t}) := (\vec{s} = \vec{t})$$

[11] We use this encoding rather than a binary numerical relation in order to allow (and detect) the presence of undefined positions in $E^<$.

Under the conditions outlined above, we can check, given a couple i, j, if it has been already treated:

$$\mathsf{treated}(R_E, \mu, \nu) := \bigwedge \begin{cases} \mu < \nu \\ \forall x \in A_\mu^<, y \in A_\nu^<, \exists \lambda, \lambda' \leq 1, R_E(x, y, \lambda) \wedge R_E(y, x, \lambda') \end{cases}$$

And therefore, we can find the next couple to be treated:

$$\mathsf{new}(R_E, \mu, \nu) := \bigwedge \begin{cases} \mu < \nu \\ \forall \mu' \leq m, \nu' \leq m, \mathsf{treated}(R_E, \mu', \nu') \iff (\mu'\nu' <_{lex} \mu\nu) \end{cases}$$

Now, suppose that we have in context a couple of variables (μ, ν) denoting the couple of colours (i, j) to treat. We can define the permutation $v \oplus v_{init}$, where v is the permutation defined by R_v:

$$v_\Xi(R_v, b_s \vec{s} \vec{s'}, b_t \vec{t} \vec{t'}) := b_s = b_t \wedge \bigvee \begin{cases} b_s = 0 \wedge R_v(\vec{s}, \vec{t}) \\ b_s = 1 \wedge \mathsf{initValue}(\vec{s'}, \vec{t'}) \end{cases}$$

where $\mathsf{initValue}$ is the formula presented in Lemma 4.28 that defines the morphism v_{init}.

To ease reading, we introduce a formula $\mathsf{let}v(R_v, \mu, \nu, x, y)$ such that, if $(i, j) \in [m]^2$ and $(a, b) \in A_i \times A_j$, $\mathsf{let}v(\mathfrak{A}, R_v, i, j, x, y)$ defines the family of permutations

$$v_{(i,j) \leftarrow \mathsf{map}_a^i \mathsf{map}_b^i} := (i', j') \mapsto \begin{cases} v_{i,j}(a, b) & \text{if } i' = i, j' = j \\ v(i', j') & \text{otherwise} \end{cases}$$

that is, $\mathsf{let}v$ changes the value of v on the (i, j) component, and set it to $v_{i,j}(a, b)$ (which is the value defined in Theorem 4.29), leaving all other components of v unchanged.

$$\mathsf{let}v(R_v, \mu, \nu, x, y, \mu_s \nu_s \vec{s}, \mu_t \nu_t \vec{t}) := \bigwedge \begin{cases} \mu_s = \mu_t \wedge \nu_s = \nu_t \\ \bigvee \begin{cases} \mu = \mu_s \wedge \nu = \nu_s \wedge \mathsf{v}(\mu, \nu, x, y, \vec{s}, \vec{t}) \\ (\mu \neq \mu_s \vee \nu \neq \nu_s) \wedge R_v(\mu_s \nu_s \vec{s}, \mu_t \nu_t \vec{t}) \end{cases} \end{cases}$$

Recall that v is the formula defining $v_{i,j}$ given in Lemma 4.33. Combining this with v_Ξ, we obtain

$$\mathsf{let}v_\Xi(R_v, \mu, \nu, x, y, b_s \vec{s} \vec{s'}, b_t \vec{t} \vec{t'}) := v_\Xi(R_v, b_s, \vec{s} \vec{s'}, b_t \vec{t} \vec{t'})[R_v(\vec{u}, \vec{v}) / \mathsf{let}v(R_v, \mu, \nu, \vec{u}, \vec{v})]$$

We can now define a formula $\mathsf{comp}(R_v, \mu, \nu, x, y)$ that holds on i, j, a, b iff the encoding of $E_{i,j}$ through $\mathsf{map}_a^i \mathsf{map}_b^j$ is compatible with $\mathcal{C}$. That is, comp defines the relation $\mathrm{Comp}_\mathcal{C}$ that we introduced at the beginning of Section 4.7. Recall that $(a, b) \in \mathrm{Comp}_\mathcal{C}$ is equivalent to

$$v_{i,j}(a, b) \in \vartheta_{i,j}(\varphi(\mathcal{C}))$$

and by induction hypothesis, $\varphi(\mathcal{C})$ is encoded by the morphism $\Xi_{\mathsf{treated}(\mathfrak{A}, R_E)}$. Therefore, $\mathsf{comp}(R_E, R_v, \mu, \nu, x, y)$ should express the fact that

$$v_{i,j}(a, b) \in \Xi_{\mathsf{treated}(\mathfrak{A}, R_E) \cup \{i,j\}}(\mathcal{G})$$

$$\mathsf{comp}(R_E, R_v, \mu, \nu, x, y) := \left(\mathsf{let} v_\Xi(R_v, \mu, \nu, x, y, \vec{s}, \vec{t}) \in \left\langle \mathrm{Im}\Xi(\mu, \nu, x, y, \vec{p}, \vec{s}, \vec{t})\right\rangle\right)_{\vec{p};\vec{s},\vec{t}}$$

where, $(F \in \langle G\rangle)_{\vec{p},\vec{s},\vec{t}}$ is the $\mathsf{FP}+\mathsf{ord}$-definable membership test quantifier introduced in Definition 3.4, and, according to the definition of Ξ and Lemma 3.14,

$$\mathrm{Im}\Xi(\mu, \nu, x, y, \vec{s}, \vec{t}) := \Xi(R_W, R, \vec{s}, \vec{t}) \left[\begin{array}{l} R_W(\alpha, \beta)/\mathsf{Union}(R_E, \mu, \nu, \alpha, \beta) \\ R(\vec{s}', \vec{t}')/\mathsf{gen}\mathcal{G}(\vec{p}, \vec{s}', \vec{t}') \end{array}\right]$$

where Union defines the new window $W \cup \{(i, j)\}$ as follows:

$$\mathsf{Union}(R_E, \mu, \nu, \alpha, \beta) := \mathsf{treated}(R_E, \alpha, \beta) \vee (\mu = \alpha \wedge \nu = \beta)$$

and $\mathsf{gen}\mathcal{G}$ is the formula defining a generating set for $\mathcal{G}$ defined in Lemma 4.24.

This use of the membership test quantifier constitutes our sole use of the ord operator in this proof. Following the path we set at the beginning of this section, we need to order $\mathsf{comp}(\mathfrak{A}, R_E, R_v, i, j)$ according to the lexicographical ordering of the encodings of $E_{i,j}$ the couples $(a, b) \in A_i \times A_j$ induce. We do so in the following formula, although we do not restrict ourselves to couples within $\mathsf{comp}(\mathfrak{A}, R_E, R_v, i, j)$. Let us first define a general formula for the comparison of two numerical relations of the same arity:

$$\mathsf{str}_{\leq}(R_1, R_2) := \vee \begin{cases} \forall \vec{\mu}, R_1(\vec{\mu}) \iff R_2(\vec{\mu}) \\ \exists \vec{\mu}, \wedge \begin{cases} R_1(\vec{\mu}) \wedge \neg R_2(\vec{\mu}) \\ \forall \vec{\nu} <_{lex} \vec{\mu}, R_1(\vec{\nu}) \iff R_2(\vec{\nu}) \end{cases} \end{cases}$$

Using Lemma 4.15, $\mathrm{enc}_E^{\mathsf{map}(\mu,x,z,\kappa)\vee\mathsf{map}(\nu,y,z,\kappa)}$ defines the numerical relation $E_{i,j}^{\mathsf{map}_a^i\mathsf{map}_b^j}$, for any coherent valuation (i, j, a, b) of (μ, ν, x, y), and as such

$$\mathsf{enc}_{\leq}(\mu, \nu, x, y, x', y') := \mathsf{str}_{\leq}(R_1, R_2) \left[\begin{array}{l} R_1(\vec{\lambda})/(\mathrm{enc}_E^{\mathsf{map}(\mu,x,z,\kappa)\vee\mathsf{map}(\nu,y,z,\kappa)})(\vec{\lambda}) \\ R_2(\vec{\lambda})/(\mathrm{enc}_E^{\mathsf{map}(\mu,x',z,\kappa)\vee\mathsf{map}(\nu,y',z,\kappa)})(\vec{\lambda}) \end{array}\right]$$

holds on (i, j, a, b, a', b') iff $E_{i,j}^{\mathsf{map}_a^i\mathsf{map}_b^j} \leq_{lex} E_{i,j}^{\mathsf{map}_{a'}^i\mathsf{map}_{b'}^j}$.

The following formula holds on R_E, R_v, a, b iff (a, b) is an adequate choice of elements to define the encoding of the edge-relation on the first untreated couple of colours (i, j), that is, if $(a, b) \in \mathrm{Comp}_{\mathcal{C}}$ and amongst couples in $\mathrm{Comp}_{\mathcal{C}}$, it yields the minimal encoding of $E_{i,j}$:

$$\mathsf{base}(R_E, R_v, x, y) := \exists \mu, \nu, \textcircled{\wedge} \begin{cases} \mathsf{new}(R_E, \mu, \nu) \\ \mathsf{comp}(R_E, R_v, \mu, \nu, x, y) \\ \forall x', y', \textcircled{\vee} \begin{cases} \neg\mathsf{comp}(R_E, R_v, \mu, \nu, x', y') \\ \mathsf{enc}_{\leq}(\mu, \nu, x, y, x', y') \end{cases} \end{cases}$$

We are now able to define the encoding of the edge relation over this new pair of colour-classes:

$$\phi_E(R_E, R_v, \vec{\lambda}) := \begin{cases} R_E(\mu, \nu, \vec{\lambda}) \\ \quad \exists \mu, \nu, x, y, \begin{cases} \mathsf{new}(R_E, \mu, \nu) \wedge \lambda_1 \in A^<_\mu \wedge \lambda_2 \in A^<_\nu \\ \mathsf{base}(R_E, R_v, x, y) \\ \quad \exists s, t, \begin{cases} \mathsf{map}(\mu, x, s, \lambda_1) \\ \mathsf{map}(\nu, y, t, \lambda_2) \\ (\lambda_3 = 1 \wedge E(s,t)) \vee (\lambda_3 = 0 \wedge \neg E(s,t)) \end{cases} \end{cases} \\ \quad \exists \mu, \nu, x, y, \begin{cases} \mathsf{new}(R_E, \mu, \nu) \wedge \lambda_1 \in A^<_\nu \wedge \lambda_2 \in A^<_\mu \\ \mathsf{base}(R_E, R_v, x, y) \\ \quad \exists s, t, \begin{cases} \mathsf{map}(\mu, x, t, \lambda_1) \\ \mathsf{map}(\nu, y, s, \lambda_2) \\ (\lambda_3 = 1 \wedge E(s,t)) \vee (\lambda_3 = 0 \wedge \neg E(s,t)) \end{cases} \end{cases} \end{cases}$$

Note that the two large similar clauses of the disjunction treat edges going in both directions between the two colour-classes. It is only left to update the permutation $v_{\mathcal{C}}$. Recall that we should provide its value for the new element (i, j) of the window:

$$\phi_v(R_E, R_v, \mu_s\nu_s\vec{s}, \mu_t\nu_t\vec{t}) := \exists \mu, \nu, \begin{cases} \mathsf{new}(R_E, \mu, \nu) \\ \quad \begin{cases} (\mu \neq \mu_s \vee \nu \neq \nu_s) \wedge R_v(\mu_s\nu_s\vec{s}, \mu_t\nu_t\vec{t}) \\ \quad \exists x, y, \begin{cases} \mu = \mu_s \\ \nu = \nu_s \\ \mathsf{base}(R_E, R_v, x, y) \\ \mathsf{let}v(R_v, \mu, \nu, x, y, \mu_s\nu_s\vec{s}, \mu_t\nu_t\vec{t}) \end{cases} \end{cases} \end{cases}$$

This concludes the definition of ϕ_E, ϕ_v. Throughout this proof, we have shown that, if S_E, S_v are two relations on $\mathfrak{A}$ that properly encode the values, when entering the for-loop of Algorithm 3, of $E^< = \bigcup_{(i',j')<_{lex}(i,j)} E^<_{i,j}$ and $v_{\mathcal{C}}$ (according to Eq. (4.8))—respectively, the relations

$$T_E := \phi_E(\mathfrak{A}, S_E, S_v)$$
$$T_v := \phi_v(\mathfrak{A}, S_E, S_v)$$

encode in the same way the values of $E^<$ and $\mathcal{C}$ *after* Line 5 of Algorithm 3. Therefore, the sequence $(R_{E,n}, R_{v,n})_{n\in\mathbb{N}}$ of pairs of relations on $\mathfrak{A}$ defined by

$$R_{E,0} := R_{\mathrm{init}E}(\mathfrak{A}) \qquad R_{v,0} := R_{\mathrm{init}v}(\mathfrak{A})$$
$$R_{E,i+1} := \phi_E(\mathfrak{A}, R_{E,i}, R_{v,i}) \qquad R_{v,i+1} := \phi_v(\mathfrak{A}, R_{E,i}, R_{v,i})$$

precisely depicts the iteration of the for-loop over all pairs of colours, and it reaches a fixed-point exactly when all pairs of colours have been treated, and the resulting value of R_E defines a graph over the ordered domain which is isomorphic to (A, E).

As a last remark on the correctness of those formulae, recall that Corollary 4.32 ensures that our last application of $\mathsf{let}v$ is correct, as all couples $(a, b) \in \mathsf{base}(\mathfrak{A}, T_E, T_v)$ yield the same permutation $\mathsf{v}(\mathfrak{A}, i, j, a, b)$ (where i and j are the colours of a and b resp.).

It is only left to use a simultaneous fix-point operator to define the last value of the sequence $(R_{E,n}, R_{v,n})$ depicted above. While this sequence is neither monotone nor inflationary, the *trace trick*, presented in the proof of Theorem 1.31 allows the simulation of this iteration for m^2 steps. Because each iteration treats a new pair of colour-classes, and there are less than $m^2 < |A|^2$ such pairs, applying this technique for $|A|^2$ iterations yields the desired fix-point of the sequence, and thus the canonical copy of (A, E).

While the combination of simultaneous fixed-point and the trace trick do not present any difficulty, the resulting formulae are quite tedious to present, so we omit them here.

Finally, note that the preorder $\prec^{\mathfrak{A}}$ can trivially be defined on $A^<$, and we have already defined, within FPC, the canonical copies $\Gamma_i^<$ of the ordered groups Γ_i in Lemma 4.19. This concludes the canonisation within $\mathsf{FP} + \mathsf{ord}$ of graphs with abelian colours. □

As we have argued at the beginning of this chapter, this result has important implications on the expressive power of $\mathsf{FP} + \mathsf{ord}$:

Theorem 4.35. $\mathsf{FP} + \mathsf{ord}$ *captures* P *on any class of structures with abelian colours.*

Proof. This is a direct consequence of Lemma 4.11, in conjunction with Theorem 4.34. □

Corollary 4.36. $\mathsf{FP} + \mathsf{ord} > \mathsf{FP} + \mathsf{rk}$.

Proof. As mentioned in Section 4.1, it was shown by Lichter [Lic23] that there is a P-decidable query over a class of structures with abelian colours that is not $\mathsf{FP} + \mathsf{rk}$ definable. Yet, Theorem 4.35 implies that this query is definable in $\mathsf{FP} + \mathsf{ord}$ and therefore $\mathsf{FP}+\mathsf{ord} \neq \mathsf{FP}+\mathsf{rk}$. We have shown in the last chapter that $\mathsf{FP}+\mathsf{rk} \leq \mathsf{FP}+\mathsf{ord}$, which concludes the proof. □

Chapter 5

Subgroup Computation

In the previous chapter, we have shown that $\mathsf{FP} + \mathsf{ord}$ is strictly more expressive than $\mathsf{FP} + \mathsf{rk}$, by showing that it can canonise structures with Abelian colours, that $\mathsf{FP} + \mathsf{rk}$ cannot distinguish. We have done so by using the group-theoretic framework to graph (or structure) canonisation, as defined in Section 4.3. In turn, applying this framework required to define generating sets for accessible subgroups of groups for which a generating set was already defined. While this constitutes one of the three operations enabled by the Schreier-Sims algorithm (Proposition 1.69), we have shown this operation not to be definable within $\mathsf{FP} + \mathsf{ord}$ in general (Theorem 3.9). Yet, in the presence of Abelian colours, the assumption that each colour class's group is *abelian* enabled the use of the morphism framework, as defined in Section 3.3.

At the end of Section 2.3, amidst the definition of our representation of permutation groups as relational structures, we remarked that this setting introduced a middle-ground in the ordering of structures: we can consider *ordered* sets of permutations, over an *unordered* domain. Structures with Abelian colours lie in that middle-ground, as by definition, such structures come equipped with a linear-ordering over the groups acting on each individual colour-class; and this ordering was vital (together with transitivity) to construct a first labelling coset (in Lemma 4.16).

In this chapter, we aim to show that, under the sole assumption that the provided generating set of permutations is ordered — that is, in a more general setting than in the previous chapter — we can carry out the third operation of the Schreier-Sims framework — a weaker operation than the canonisation of structures. Said differently, while we have shown in Section 3.2 that the order-invariance condition on generating sets is too restrictive to define generating sets for accessible subgroups in general, when an ordering on our base group is provided (hence undermining the isomorphism-invariance restriction), the whole of the Schreier-Sims framework is definable.

In the first section, we define the problem formally. In the second section, we solve it, by defining a simulation of the Schreier-Sims algorithm within $\mathsf{FP} + \mathsf{ord}$. Finally, we apply this result to a somehow artificial class, denoted $\mathrm{PBCG}^<$, that we will introduce in the third and final section, and show that $\mathsf{FP} + \mathsf{ord}$ can define a generating set for the automorphism groups of structures within $\mathrm{PBCG}^<$. $\mathrm{PBCG}^<$ is a class of structures that generalises the notion of structure with abelian colours. As such, in this last section,

we show a weaker result (computation of a generating set for the automorphism group vs. canonisation) that applies to a wider range of structures.

5.1 Ordered permutation groups

In the statement of Theorem 3.9, we have already stated the problem we aim to solve, although in the unordered setting. Let us restate this definition, while introducing the ordered setting:

Definition 5.1. Let Σ be a signature, $\mathcal{K} \subseteq \mathrm{STRUC}[\Sigma]$, and suppose that $\mathbf{H}$ is a function, mapping any structure $\mathfrak{A} \in \mathcal{K}$ to a group $\mathbf{H}(\mathfrak{A}) \leq \mathrm{Sym}(A^T)$ for some fixed type T.

A logic $\mathcal{L}$ *witnesses the k-accessibility of* $\mathbf{H}$ *in* $\mathcal{K}$ *(uniformly)* if there are $\mathcal{L}[\Sigma]$ formulae $\varphi_{\mathbf{G}}(\vec{p}, \vec{s}, \vec{t})$ and $\varphi_{\in}(\vec{\mu}, X)$ where $\mathrm{type}(\vec{s}) = \mathrm{type}(\vec{t}) = T$, $\mathrm{type}(\vec{\mu}) = \mathrm{number}^k$, and X is a second-order relation variable of type T^2, such that for any $\mathfrak{A} \in \mathcal{K}$ with $n := |A|$:

- We can inductively define a decreasing chain of groups $(\mathbf{G}_i(\mathfrak{A}))_{i=0}^{n^k}$ as follows:
  - $\mathbf{G}_0(\mathfrak{A}) := \langle \varphi_{\mathbf{G}} \rangle_{\vec{p},\vec{s}\vec{t}}(\mathfrak{A})$
  - $\mathbf{G}_{i+1}(\mathfrak{A}) := \{\sigma \in \mathbf{G}_i(\mathfrak{A}) \mid \mathfrak{A}, i, \mathrm{graph}(\sigma) \models \varphi_{\in}\}$, where i is seen as a k-tuple of numerical values (assigned to $\vec{\mu}$)
- $\mathbf{G}_{n^k}(\mathfrak{A}) = \mathbf{H}(\mathfrak{A})$
- For all $i < n^k$, $|\mathbf{G}_i(\mathfrak{A}) : \mathbf{G}_{i+1}(\mathfrak{A})| \leq n^k$

We say that $\mathcal{L}$ witnesses the k-accessibility of $\mathbf{H}$ in $\mathcal{K}$ *in the ordered setting* if, in the formula $\varphi_{\mathbf{G}}$ above, $\vec{p}$ is a tuple consisting only of *numerical* variables.

Using this formalism, we aim to show that, in the ordered setting, FP + ord can define a generating set for any group $\mathbf{H}$ for which $\varphi_{\mathbf{G}}, \varphi_{\in}$ witness the k-accessibility. Remark that there is a short-coming of this definition: it requires a formula $\varphi_{\in}$ taking as input a second-order variable X (ranging over graphs of permutations). As such, we cannot expect to represent all the instances of the problem at hand as one unified class of structures over some fixed signature, but rather, we must rely on the definability of such a formula $\varphi_{\in}$.

5.2 The Schreier-Sims algorithm within FP + ord

This section is devoted to the proof of the following theorem:

Theorem 5.2. *Let Σ be a signature, $\mathcal{K} \subseteq \mathrm{STRUC}[\Sigma]$ and suppose that $\mathbf{H}$ is a function, mapping any structure $\mathfrak{A} \in \mathcal{K}$ to a group $\mathbf{H}(\mathfrak{A}) \leq \mathrm{Sym}(A^T)$ for some fixed type T.*

If $\mathsf{FP} + \mathsf{ord}$ witnesses the k-accessibility of $\mathbf{H}$ in $\mathcal{K}$ in the ordered setting, there is a formula $\varphi_{\mathbf{H}} \in (\mathsf{FP} + \mathsf{ord})[\Sigma]$ defining a generating set for $\mathbf{H}(\mathfrak{A})$ for any $\mathfrak{A} \in \mathcal{K}$.

This result should be contrasted with the unordered setting, where, as shown in Theorem 3.9, such a formula $\varphi_{\mathbf{H}}$ does not exist.

This result relies on a partial simulation of the Schreier-Sims framework within $\mathsf{FP}+\mathsf{ord}$, which was already given in Algorithms 1 and 2. For convenience, we reproduce those algorithms here.

Algorithme 1 : Sifting Procedure

Input : $(\sigma_{i,j})$, a SGS for G along $(H_i)_{i=0}^k$, a subgroup chain with conditional membership tests $(P_i)_{i=1}^k$; and $\tau \in \mathrm{Sym}(\Omega)$
Result : Does $\tau \in G$?

```
1  Function sift(g, λ) is
       // Invariant :  g ∈ H_λ ⟺ τ ∈ G
2      if λ = k then
3          if g = Id then
4              return true
5          else
6              return false outputting (g, λ)
7      else
8          if ∃!j s.t. P_{λ+1}(σ_{λ,j}^{-1} · g) then
9              return sift((σ_{λ,j}^{-1} · g), λ + 1)
10         else
11             return false outputting (g, λ)
12 return sift(τ, 0)
```

Recall moreover that, when $(H_i)_{i=0}^k$ is adequate, those algorithms run in polynomial time. Proposition 1.69 enables the use of Algorithms 1 and 2 to obtain a generating set for an accessible subgroup $H \leq \mathrm{Sym}(X)$. Suppose that the accessibility of H is witnessed by the decreasing chain of subgroups $(G_i)_{i=0}^m$. Then, we evaluate Algorithms 1 and 2 on the subgroup chain

$$G_0 \geq G_1 \geq \ldots G_m = H \geq \mathrm{Stab}_H(x_1) \geq \cdots \geq \mathrm{Stab}_H(x_1, \ldots, x_{|X|}) = 1$$

where $(x_i)_{i=1}^{|X|}$ is an ordered enumeration of X. Indeed, in such a case, after the termination of Algorithm 2, $\{\sigma_{i,j}, i > m\}$ constitutes a generating set for H.

In our current case however, where $\mathbf{H}(\mathfrak{A}) \leq \mathrm{Sym}(A^T)$, such an ordered enumeration of the domain of the group is not achievable and as such, we cannot expect to

Algorithme 2 : Constructing a SGS

Input : S, a generating set for G, $(H_i)_{i\leq k}$, an adequate subgroup chain for G, with conditional membership tests $(P_i)_{i=1}^k$.

Output : $(\sigma_{i,j})_{i<k}$, a SGS for (H_i)

```
Initialize an array (σ_{i,j}) with empty cells, except σ_{i,0} := Id for all i ≤ k
ℓ := S
while ℓ ≠ ∅ do
    Let τ be the first element of ℓ and remove it from ℓ
    if sift(τ, 0) rejects on (σ_{i,j}), outputting (g, i) then
        j := min{λ, σ_{i,λ} = ∅}
        σ_{i,j} := g
        ℓ := {g · σ_{a,b}, a ≤ i, σ_{a,b} ≠ ∅} ⊕ {σ_{a,b} · g, a ≥ i, σ_{a,b} ≠ ∅} ⊕ ℓ
return (σ_{i,j})
```

simulate those algorithms fully. This constitutes the main obstacle to the definition of a generating set for $\mathbf{H}$ in $\mathsf{FP} + \mathsf{ord}$.

To overcome this issue, we will use the ord operator to bypass the need for this chain of point-wise stabilisers. That is, our method to prove Theorem 5.2 will be to construct iteratively a *partial* strong generating set, collecting a coset transversal only for the naturally ordered part of the adequate chain of subgroups.

$$\mathbf{G}_0(\mathfrak{A}) \geq \cdots \geq \mathbf{G}_{n^k}(\mathfrak{A})$$

while we keep an unstructured set of generators for $\mathbf{G}_{n^k}(\mathfrak{A})$. Therefore, this partially strong generating set structure will be made of two parts: a structured array of m rows and n^k columns, defining coset representatives for $\mathbf{G}_{i+1}(\mathfrak{A})$ in $\mathbf{G}_i(\mathfrak{A})$, and an unstructured list of permutations in $\mathbf{H}(\mathfrak{A})$ Moreover, the sifting procedure should be adapted according to this new notion of partially strong generating sets.

We structure the proof as follows: in the first subsection, we formalise this notion of partially strong generating sets, and show how it can be represented in extensions of FPC. Then, we will provide an implementation in $\mathsf{FP} + \mathsf{ord}$ of the sifting procedure in this new setting, and conclude with the translation of the construction procedure, and the resulting formula defining a generating set for $\mathbf{H}(\mathfrak{A})$.

Partially strong generating sets

We now show how to adapt the structure of SGS to defer the handling of the stabiliser chain to the ord operator.

Definition 5.3. Given an k-adequate tower of subgroups $(G_i)_{i=0}^m$ for $G \leq \mathrm{Sym}(X)$, and $d \leq m$ a *d-partially strong generating set* for G is a couple $(\mathcal{R}, \mathcal{S})$, where:

- $\mathcal{R}$ is a list of d lists of permutations, such that, for each $i < d$, the i-th inner list, $\mathcal{R}(i) := \{\mathcal{R}(i, \lambda), \lambda \leq |X|^d\}$ is a transversal of G_{i+1} in G_i

- $\mathcal{S} \subseteq \mathrm{Sym}(X)$ is a generating set for G_d.

In the setting of Theorem 5.2, we aim to represent a n^k-partially strong generating set for the chain:

$$\mathbf{G}_0(\mathfrak{A}) \geq \cdots \geq \mathbf{G}_{n^k}(\mathfrak{A}) = \mathbf{H}(\mathfrak{A}) \geq \mathrm{Stab}_{\mathbf{H}(\mathfrak{A})}(a_1) \geq \cdots \geq \mathrm{Stab}_{\mathbf{H}(\mathfrak{A})}(a_1, \ldots, a_n)$$

Since both the index of $\mathbf{G}_i(\mathfrak{A})$ in $\mathbf{G}_{i+1}(\mathfrak{A})$ and the length of this sequence are bounded by $|A|^k$, we can index the permutations in $\mathcal{R}$ with two tuples of k variables. When defining the construction procedure, we will see how we can bound the size of $\mathcal{S}$ by $|A|^{2|T|}$. This yields the following representation of SGS in $\mathsf{FP} + \mathsf{ord}$:

Definition 5.4. Fix k an integer. Let $(\mathbf{K}_i)_{i=0}^{n^k}$ be functions mapping a Σ-structure $\mathfrak{A}$ with $|A| = n$ to groups over A^T for some fixed type T, such that $(\mathbf{K}_i(\mathfrak{A}))_{i=0}^{n^k}$ is k-adequate. A representation of a n^k-partially strong generating set $(\mathcal{R}, \mathcal{S})$ for $(\mathbf{K}_i(\mathfrak{A}))$ is a couple of relations R, S over $\mathfrak{A}$ such that:

- $\mathrm{type}(R) = \mathrm{number}^{2k} \cdot T^2$, and for each $\vec{\mu}, \vec{\nu} \in (A^<)^k$, $R^{\mathfrak{A}}(\vec{\mu}, \vec{\nu}) := \{(\vec{s}, \vec{t}) \in A^{T \cdot T}, (\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \in R^{\mathfrak{A}}\}$ is either empty or the graph of a permutation.
- For any fixed $\vec{\mu}$, the set of all permutations $\sigma \in \mathrm{Sym}(A^T)$ such that $\mathrm{graph}(\sigma) = R^{\mathfrak{A}}(\vec{\mu}, \vec{\nu})$ for some $\vec{\nu}$ is equal to $\{\mathcal{R}(i, j), j \leq |A|^k\}$, where i is the integer encoded by $\vec{\mu}$.
- $\mathrm{type}(S^{\mathfrak{A}}) = \mathrm{number}^{2|T|} \cdot T \cdot T$ and for any $\vec{\lambda} \in (A^<)^{2|T|}$, $S^{\mathfrak{A}}(\vec{\lambda})$ is either empty or the graph of a permutation.
- The set of all permutations $\sigma \in \mathrm{Sym}(A^T)$ such that $\mathrm{graph}(\sigma) = S^{\mathfrak{A}}(\vec{\lambda})$ for some $\vec{\lambda}$ is equal to $\mathcal{S}$.
- For each $\vec{\mu}$, the set of $\vec{\nu}$ such that $R^{\mathfrak{A}}(\vec{\mu}, \vec{\nu}) \neq \emptyset$ form an intial segment of $(A^<)^k$ (for the natural encoding). The same holds for $S^{\mathfrak{A}}$ w.r.t. $\vec{\lambda}$.

This last condition mainly plays a role as an invariant in the iterative definitions to come.

For brevity, we refer to those structures of partially strong generating sets as *PSGS*. If $(\mathcal{R}, \mathcal{S})$ is such a PSGS, we refer to $\mathcal{R}$ as the *transversal table* and $\mathcal{S}$ as the *residual list* of the PSGS. In what follows, we often identify a PSGS $(\mathcal{R}, \mathcal{S})$ with its representation $(R^{\mathfrak{A}}, S^{\mathfrak{A}})$.

With those definitions in mind, notice that, if we manage to build a n^k-PSGS for $(\mathbf{G}(\mathfrak{A}))_{i=0}^{n^k}$, its residual list will be a generating set for $\mathbf{H}(\mathfrak{A})$, which is exactly the aim of Theorem 5.2.

Remark that we require the rows and columns of both $R^{\mathfrak{A}}$ and $S^{\mathfrak{A}}$ to be indexed over the *ordered* domain. As a counterpart, we need to show how to order those indexing

sets. While this will effectively be done in the following subsections, we provide an intuition for this right now.

Since $\mathsf{FP} + \mathsf{ord}$ witnesses the k-accessibility of $\mathbf{H}$ in $\mathcal{K}$ *in the ordered setting*, $\varphi_{\mathbf{G}}(\vec{p}, \vec{s}, \vec{t})$ provides an *ordered* enumeration $\mathcal{X} = (g_1, \ldots, g_{n^{|p|}})$ of a generating set for the largest group $\mathbf{G}(\mathfrak{A})$ in the sequence. To construct a n^k-partially strong generating set for $\mathbf{G}(\mathfrak{A})$ through the sequence $(\mathbf{G}_i(\mathfrak{A}))$, we will insert the elements of $\mathcal{X}$ one by one into an initially trivial n^k-PSGS (i.e. where $\mathcal{S}$ and each list in $\mathcal{R}$ contain only the identity permutation), so as to obtain, after λ such insertions, a n^k-PSGS for the chain of subgroups:

$$\langle g_1, \ldots, g_\lambda \rangle \cap \mathbf{G}_0(\mathfrak{A}) \geq \langle g_1, \ldots, g_\lambda \rangle \cap \mathbf{G}_1(\mathfrak{A}) \geq \ldots \langle g_1, \ldots, g_\lambda \rangle \cap \mathbf{G}_{n^k}(\mathfrak{A}) \tag{5.1}$$

The astute reader will remember that, in Algorithm 2, a saturation step (given on line 8) was needed to ensure that the SGS did in fact represent a group (in particular, that the membership test defined by the sifting procedure was closed by composition). In a similar way, to obtain an actual PSGS for the chain of subgroups of Eq. (5.1), we need to apply a saturation procedure to the PSGS, that will be duly defined.

Since elements are added iteratively, in an order entirely dependent on the ordering of $\mathcal{X}$, we can order cosets of $\mathbf{G}_{i+1}(\mathfrak{A})$ in $\mathbf{G}_i(\mathfrak{A})$ depending on the order in which elements belonging to those cosets were found. In the same way, we can order the indexing of S depending on the order in which those generators of $\mathbf{H}(\mathfrak{A})$ were found. Using this fact as an induction hypothesis, it is then clear that the saturation procedure can also be conducted in an iterative, canonically ordered fashion.

Having defined the central data structure of our proof, we are now ready to delve into the definition in $\mathsf{FP} + \mathsf{ord}$ of the partial simulation of the Schreier-Sims framework. Let us fix, for the rest of this chapter, the type T of tuples on which the groups defined by $\mathbf{G}$ and $\mathbf{H}$ act.

The sifting procedure

Since the construction procedure make calls to the sifting procedure, it seems sensible to start by defining the latter. In what follows, suppose that R and S are second-order variables representing the current value of our n^k-PSGS. We now define $\Sigma \sqcup \{R\}$-FPC formulae $\mathrm{sift}_{\mathrm{res}}(\Gamma_\sigma, s, t), \mathrm{sift}_{\mathrm{level}}(\Gamma_\sigma, \vec{\mu})$ which specify the result of the first n^k steps of the sifting procedure on the chain of subgroups represented by $(R^{\mathfrak{A}}, S^{\mathfrak{A}})$ on input σ, for any suitable structure $\mathfrak{A}$.

Lemma 5.5. *There are* FPC*-formulae* $\mathrm{sift}_{\mathrm{res}}(R, \Gamma_\sigma, \vec{s}, \vec{t})$ *and* $\mathrm{sift}_{\mathrm{level}}(R, \Gamma_\sigma, \vec{\mu})$ *such that, given a structure* $\mathfrak{A} \in \mathcal{K}$*, a transversal table* $R^{\mathfrak{A}}$ *over* $\mathfrak{A}$*, and a permutation* $\sigma \in \mathrm{Sym}(A^T)$*:*

- $(\mathfrak{A}, R^{\mathfrak{A}}, \mathrm{graph}(\sigma), \vec{a}) \models \mathrm{sift}_{\mathrm{level}}$ *iff* $\vec{a}$ *encodes the integer* i *which is the row of the SGS defined by* $R^{\mathfrak{A}}$ *reached during the computation of* $\mathrm{sift}(\sigma, 0)$ *as defined in*

Algorithm 1.[1]

- $\mathrm{sift}_{\mathrm{res}}(\mathfrak{A}, R^{\mathfrak{A}}, \mathrm{graph}(\sigma))$ *is the graph of the permutation which is the value of the input to the last recursive call to* sift *before the* n^k*-th, on initial input* $(\sigma, 0)$.[2]

Proof. We first show that we can build a formula $\mathrm{sift}_{\mathrm{eval}}(R, R_\sigma, \vec{\mu}, \vec{s}, \vec{t})$ that defines the trace of the value of the input permutation to the various recursives calls to sift in the computation of $\mathrm{sift}(\sigma, 0)$. Precisely, if i is the integer encoded by $\vec{a}$ and $\mathrm{sift}(\tau, i)$ is called at some point in the computation of $\mathrm{sift}(\sigma, 0)$, $\mathrm{sift}_{\mathrm{eval}}(\mathfrak{A}, R^{\mathfrak{A}}, \sigma, \vec{a}) = \mathrm{graph}(\tau)$.

$$\mathrm{sift}_{\mathrm{eval}}(R, R_\sigma) := \mathrm{ifp}_{\Theta,\vec{\mu},\vec{s},\vec{t}} \begin{cases} \vec{\mu} = 0 \wedge \sigma(\vec{s}) = \vec{t} \\ \exists \vec{\nu}, \begin{cases} \varphi_{\in}(\vec{\mu}, X)[X(\vec{x}, \vec{y})/\chi(\vec{x}, \vec{y})] \\ \chi(\vec{s}, \vec{t}) \end{cases} \end{cases} \tag{5.2}$$

where

$$\chi(\vec{x}, \vec{y}) := \exists \vec{z}, \begin{cases} \Theta(\vec{\mu} - 1, \vec{x}, \vec{z}) \\ R(\vec{\mu} - 1, \vec{\nu}, \vec{y}, \vec{z}) \end{cases}$$

and where by $\vec{\mu} - 1$, we mean the tuple encoding the predecessor of the integer encoded by $\vec{\mu}$ (which is obviously definable in FPC). Recall that R is our given transversal table and $\varphi_{\in}$ the conditional membership formula from Definition 5.1, that takes as input a numerical tuple, and a relation encoding a permutation.

Let us explain this definition. We claim that this inflationary fixed-point definition is such that, after k iterations, the relation to which Θ evaluates to the trace of k recursive calls (starting from $\mathrm{sift}(\sigma, 0)$) to the sift function as defined in Algorithm 1. Recall that, by definition of the ifp operator, Θ initially evaluates to the empty relation, and as such, after one iteration, it is easy to see that Θ evaluates to $\{0\} \times \mathrm{graph}(\sigma)$. This models the fact that, before any recursive call to sift, the only call to sift in the trace is $(\sigma, 0)$.

Let us now show the correctness of the recursive part of this fix-point definition, starting with the definition of χ. Notice that, if $R(\mathfrak{A}, i-1, j) = \mathrm{graph}(\tau)$ and $\Theta(\mathfrak{A}, i-1) = \mathrm{graph}(\rho)$ — that is, if the transversal table contains the permutation τ at position $(i-1, j)$ and, by induction hypothesis, the permutation resulting of $i-1$ steps of sifting is ρ — then, $\chi(\mathfrak{A}, R, \Theta, i, j) = \mathrm{graph}(\rho^{-1}\tau)$. That is, χ is precisely the composition of the permutation defined by $\Theta(\vec{\mu} - 1)$ with the inverse of the permutation defined by $R(\vec{\mu} - 1, \vec{\nu})$, as constructed in Lemma 2.11. Equation (5.2) effectively assigns the permutation defined by χ to $\Theta(i)$, for the value of $\vec{\nu}$ which encodes the unique j (if

[1] Said differently, i is equal to the minimum between n^k and the number of recursive calls during the evaluation of $\mathrm{sift}(\sigma, 0)$

[2] That is, either the permutation output of Algorithm 1 if it rejects before the chain of pointwise stabilisers, or the unique τ such that $\mathrm{sift}(\tau, n^k)$ is called during the evaluation of $\mathrm{sift}(\sigma, 0)$

such a j exists) such that $R(i-1,j)$ and $\Theta(i-1)$ belong to the same coset of $\mathbf{G}_i(\mathfrak{A})$ in $\mathbf{G}_{i-1}(\mathfrak{A})$ (as checked using $\varphi_\in$) which is precisely the operation conducted within the else-clause on line 7 of Algorithm 1.

Now, if at the i-th recursive call $\Theta(i-1)$ does not belong to any coset defined by $R(\mathfrak{A}, i-1)$, none of the two disjuncts in Eq. (5.2) hold, and $\Theta(\vec{\nu}) = \emptyset$ holds for any number of iterations of the formula within the ifp operator, and for any $\vec{\nu}$ encoding an integer $j \geq i$.

From this stems the definition of $\mathrm{sift}_{\mathrm{level}}$:

$$\mathrm{sift}_{\mathrm{level}}(R, R_\sigma, \vec{\mu}) := \circledwedge \begin{cases} \exists \vec{s}, \vec{t}, \mathrm{sift}_{\mathrm{eval}}(R, R_\sigma, \vec{\mu}, \vec{s}, \vec{t}) \\ \forall \vec{s}, \vec{t}, \neg \mathrm{sift}_{\mathrm{eval}}(R, R_\sigma, \vec{\mu}+1, \vec{s}, \vec{t}) \end{cases}$$

From there, $\mathrm{sift}_{\mathrm{res}}$ is also easily definable, as $\mathrm{sift}_{\mathrm{res}}(\mathfrak{A}, R, \mathrm{graph}(\sigma))$ should evaluate to $\mathrm{sift}_{\mathrm{eval}}(\mathfrak{A}, R, \mathrm{graph}(\sigma), i)$, where i is the minimum between n^k and the number of iterations after which the sift procedure halts, i.e., the unique integer in $\mathrm{sift}_{\mathrm{level}}(\mathfrak{A}, R, \mathrm{graph}(\sigma))$:

$$\mathrm{sift}_{\mathrm{res}}(R, \sigma, \vec{s}, \vec{t}) := \exists \vec{\mu}, \mathrm{sift}_{\mathrm{level}}(R, \sigma, \vec{\mu}) \wedge \mathrm{sift}_{\mathrm{eval}}(R, \sigma, \vec{\mu}, \vec{s}, \vec{t}) \qquad \square$$

Now, suppose that (R, S) represent a d-PSGS $(\mathcal{R}, \mathcal{S})$, and $\sigma \in \mathrm{Sym}(X)$. Going through the first d recursive calls of $\mathrm{sift}(\sigma, 0)$ on $\mathcal{R}$ (notice that during the first d recursive calls, Algorithm 1 only involves the first d rows of $\mathcal{R}$), two outcomes can be reached:

- either at some recursive call (τ_i, i) with $i < d$, there is no coset of G_{i+1} in G_i to which τ_i belongs, which implies that $\tau_i \notin G$, and thus $\sigma \notin G$
- or we reach the d-th recursive call. Let us denote τ the unique permutation such that $\mathrm{sift}(\tau, d)$ is called. Then, $\sigma \in G \iff \tau \in G_d \iff \tau \in \langle S \rangle$.

Using $\mathrm{sift}_{\mathrm{res}}$ and $\mathrm{sift}_{\mathrm{level}}$ together with the ord operator, it is therefore easy to build a formula $\mathrm{sift}_{\mathrm{bool}}(R, S, \sigma)$ which holds on a n^k-PSGS representation (R, S) of G iff $\sigma \in G$:

$$\mathrm{sift}_{\mathrm{bool}}(R, S, \sigma) := \mathrm{sift}_{\mathrm{level}}(\sigma, n^k - 1) \wedge (\mathrm{sift}_{\mathrm{res}}(\sigma, \vec{s}, \vec{t}) \in \langle S(\vec{s}, \vec{t}) \rangle)_{\vec{\lambda}, \vec{s}\vec{t}}$$

where we have used the membership quantifier introduced in Definition 3.4. We are now ready to delve into the construction of the PSGS representation.

Construction of a PSGS

To ease the presentation, we divide the construction procedure into smaller parts. We first show how to add a single permutation to a PSGS structure. In a second time, we deal with saturation, and finally, we provide a formula for the whole construction, which iterates the insertion formulae over the generating set of $\mathbf{G}(\mathfrak{A})$ provided by $\varphi_{\mathbf{G}}$, and then apply the saturation formulae.

From sifting to insertion Before we define the insertion formulae, we direct the reader's attention to the fact that what motivates the existence of a saturation mechanism is that, after a single insertion of a permutation σ in a (partial) strong generating set, the resulting structure is not necessarily a SGS anymore. As such, a definition is necessary to state precisely the post-conditions of our insertion procedure.

Definition 5.6. *(Assumptions of Definition 5.3)* A *d-PSGS prototype* for $G \leq \mathrm{Sym}(X)$ through $(G_i)_{i=0}^m$ is a couple $(\mathcal{R}, \mathcal{S})$ such that:

- $\mathcal{R}$ is a list of length d of lists of permutations. Each of those inner list contain at most $|X|^k$ permutations.
- $\mathcal{S}$ is a list of permutations.
- The set of permutations contained in $(\mathcal{R}, \mathcal{S})$ generates G.
- For each $i < d$, the set of permutations in $\mathcal{R}(i)$ is a subset of G_i, and each of those permutation belong to a different coset of G_{i+1} in G_i. $\mathcal{S}$ is a subset of G_d.

That is, PSGS prototypes differ from PSGS in that we only expect $\mathcal{R}(i)$ to be a *subset* of a transversal of the cosets of G_{i+1} in G_i. In this context, the additional condition that $\bigcup \mathcal{R} \cup \mathcal{S}$ generates G is necessary to maintain a correspondence between PSGS prototypes and the groups they define.

We are now ready to introduce the insertion formulae ins, which assume $(R^{\mathfrak{A}}, S^{\mathfrak{A}})$ to be a PSGS prototype, and evaluate to a larger PSGS prototype:

Lemma 5.7. *There is a* FPC*-formula* $\mathrm{ins}_{\mathrm{table}}(R, R_\sigma, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t})$ *and a* $\mathsf{FP} + \mathsf{ord}$*-formula* $\mathrm{ins}_{\mathrm{list}}(R, S, R_\sigma, \vec{\lambda}, \vec{s}, \vec{t})$ *such that, given a* Σ*-structure* $\mathfrak{A}$*, a subgroup* $\mathbf{K}(\mathfrak{A}) \leq \mathbf{G}(\mathfrak{A})$*, a permutation* $\sigma \in \mathbf{G}(\mathfrak{A})$ *and a PSGS prototype* $(R^{\mathfrak{A}}, S^{\mathfrak{A}})$ *for* $\mathbf{K}(\mathfrak{A})$ *through*

$$\mathbf{K}(\mathfrak{A}) \geq \mathbf{K}(\mathfrak{A}) \cap \mathbf{G}_1(\mathfrak{A}) \geq \cdots \geq \mathbf{K}(\mathfrak{A}) \cap \mathbf{G}_{n^k}(\mathfrak{A})$$

The relations $R' = \mathrm{ins}_{\mathrm{table}}(\mathfrak{A}, R^{\mathfrak{A}}, \mathrm{graph}(\sigma))$ *and* $S' = \mathrm{ins}_{\mathrm{list}}(\mathfrak{A}, R^{\mathfrak{A}}, S^{\mathfrak{A}}, \mathrm{graph}(\sigma))$ *form a PSGS prototype for* $\langle \mathbf{K}(\mathfrak{A}), \sigma \rangle$ *through the chain of subgroup*

$$\langle \mathbf{K}(\mathfrak{A}), \sigma \rangle \geq \langle \mathbf{K}(\mathfrak{A}), \sigma \rangle \cap \mathbf{G}_1(\mathfrak{A}) \geq \cdots \geq \langle \mathbf{K}(\mathfrak{A}), \sigma \rangle \cap \mathbf{G}_{n^k}(\mathfrak{A})$$

Proof. We first depict the expected behaviour of ins. Consider τ the result of the partial sifting of σ, that is $\tau := \mathrm{perm}(\mathrm{sift}_{\mathrm{res}}(\mathfrak{A}, R^{\mathfrak{A}}, \mathrm{graph}(\sigma)))$ and i the unique index of $R^{\mathfrak{A}}$ such that $(\mathfrak{A}, R^{\mathfrak{A}}, \mathrm{graph}(\sigma), i) \models \mathrm{sift}_{\mathrm{level}}$.

- If $i = n^k - 1$, then τ should be added to S. To make sure that S always contains less than $n^{2|T|}$ elements, we only add σ to S if $\tau \notin \langle S(\vec{\lambda}, \vec{s}, \vec{t}) \rangle_{\vec{\lambda}}(\mathfrak{A})$ (this can be tested using the ord operator).

- Otherwise, τ is not represented in the partial transversal $R(i+1)$. Therefore, τ must be added to $R(\vec{i}{+}1, j)$, where j is the minimal value for which $R(\mathfrak{A}, i{+}1, j) = \emptyset$.

We now give the definitions of $\mathrm{ins}_{\mathrm{table}}$ and $\mathrm{ins}_{\mathrm{list}}$, where the index i is encoded by the tuple $\vec{\mu}$, and j by the tuple $\vec{\nu}$:

$$\mathrm{ins}_{\mathrm{table}}(R, \sigma, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := \begin{cases} R(\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \\ \bigvee \begin{cases} \mathrm{sift}_{\mathrm{level}}(R, \sigma, \vec{\mu}) \wedge \vec{\mu} \neq n^k - 1 \\ \forall \vec{s'}, \vec{t'}, \neg R(\vec{\mu}, \vec{\nu}, \vec{s'}, \vec{t'}) \\ \bigwedge \forall \vec{\nu'} < \vec{\nu}, \exists \vec{s'}, \vec{t'}, R(\vec{\mu}, \vec{\nu'}, \vec{s'}, \vec{t'}) \\ \forall \vec{\nu'} < \vec{\nu}, \neg \varphi_{\in}(\vec{\mu}, X)[X(\vec{x}, \vec{y}) / R_{\rho^{-1}\tau}(\vec{\nu'}, \vec{x}, \vec{y})] \\ \tau(\vec{s}) = \vec{t} \end{cases} \end{cases}$$

$$\mathrm{ins}_{\mathrm{list}}(R, S, \sigma, \vec{\lambda}, \vec{s}, \vec{t}) := \begin{cases} \forall \vec{\lambda'} < \vec{\lambda}, \exists \vec{s'}, \vec{t'}, S(\lambda', \vec{s'}, \vec{t'}) \\ \bigwedge \begin{cases} S(\vec{\lambda}, s, t) \\ \bigvee \begin{cases} \forall \vec{s'}, \vec{t'}, \neg S(\lambda, \vec{s'}, \vec{t'}) \\ \bigwedge \begin{cases} \mathrm{sift}_{\mathrm{level}}(R, \sigma, n^k - 1) \\ \tau(s) = t \end{cases} \\ \neg(\tau \in \langle S \rangle)_{\vec{\lambda}, \vec{s}, \vec{t}} \end{cases} \end{cases} \end{cases}$$

where τ and $\rho^{-1}\tau$ are the permutations defined by

$$R_\tau(\vec{s}, \vec{t}) := \mathrm{sift}_{\mathrm{res}}(T, \sigma, \vec{s}, \vec{t})$$

$$R_{\rho^{-1}\tau}(\vec{\nu'}, \vec{s}, \vec{t}) := \exists \vec{z}, \bigwedge \begin{cases} R_\tau(\vec{x}, \vec{z}) \\ R(\vec{\mu}, \vec{\nu'}, \vec{y}, \vec{z}) \end{cases}$$

The last clause of $\mathrm{ins}_{\mathrm{list}}$ ensures that the group $\langle S' \rangle$ is strictly larger than $\langle S \rangle$, therefore by a multiplicative factor. As such, since $|\mathbf{H}(\mathfrak{A})| \leq |A^T|!$, only $|A^T| \log(|A^T|)$ insertions into S can effectively take place throughout the construction procedure (which ensures that tuples of arity $2|T|$ are large enough to represent indices of any S). □

Remark that the ordering of the cosets has been used here to choose an indexing for the potential new element to insert within $\mathcal{R}$.

Iterating insertions over the whole generating set defined by $\varphi_{\mathbf{G}}$, we obtain:

Corollary 5.8. *There is a couple of* $\mathsf{FP} + \mathsf{ord}$ *formulae* $\mathsf{proto}_{\mathrm{table}}(R, S, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t})$ *and* $\mathsf{proto}_{\mathrm{list}}(R, S, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t})$ *which define a PSGS prototype for* $\mathbf{G}(\mathfrak{A})$ *through* $(\mathbf{G}_i(\mathfrak{A}))_{i=0}^{n^k}$.

Proof. Suppose that we have already added a subset X of the permutations defined by $\varphi_{\mathbf{G}}$, and suppose additionally that $I^{\mathfrak{A}}$ is a numerical second-order relation containing

exactly those indices i for which we have already inserted $\mathrm{perm}(\varphi_{\mathbf{G}}(\mathfrak{A}, i))$. In this context, the index of the next permutation to be inserted can be defined in FPC, and the updated value of I too:

$$\mathrm{step}_{ind}(R', S', I, \vec{p}) := \forall \vec{q} < \vec{p}, I(\vec{q})$$

With step_{ind} defined, we can easily define formulae to update (R, S) accordingly:

$$\mathrm{step}_{table}(R', S', I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := \bigvee \begin{cases} R'(\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \\ \bigwedge \begin{cases} \forall \vec{s'}, \vec{t'}, \neg R'(\vec{\mu}, \vec{\nu}, \vec{s'}, \vec{t'}) \\ \exists \vec{p}, \bigwedge \begin{cases} \neg I(\vec{p}) \wedge \mathrm{step}_{ind}(R', S', I, \vec{p}) \\ \mathrm{ins}^*_{\mathrm{table}}(R', S', R_\sigma, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \end{cases} \end{cases} \end{cases}$$

$$\mathrm{step}_{list}(R', S', I, \vec{\lambda}, \vec{s}, \vec{t}) := \bigvee \begin{cases} S'(\vec{\lambda}, \vec{s}, \vec{t}) \\ \bigwedge \begin{cases} \forall \vec{s'}, \vec{t'}, \neg S'(\vec{\lambda}, \vec{s'}, \vec{t'}) \\ \exists \vec{p}, \bigwedge \begin{cases} \neg I(\vec{p}) \wedge \mathrm{step}_{ind}(R', S', I, \vec{p}) \\ \mathrm{ins}^*_{\mathrm{list}}(R', S', R_\sigma, \vec{\lambda}, \vec{s}, \vec{t}) \end{cases} \end{cases} \end{cases}$$

where the formula $\mathrm{ins}^*_{\mathrm{table}}$ (resp. $\mathrm{ins}^*_{\mathrm{list}}$) is the formula $\mathrm{ins}_{\mathrm{table}}$ (resp. $\mathrm{ins}_{\mathrm{list}}$) where each occurrence of $R_\sigma(\vec{x}, \vec{y})$ has been substitued by $\varphi_{\mathbf{G}}(\vec{p}, \vec{x}, \vec{y})$. Those step formulae merely iterate the insertion formulae defined in the last lemma over all permutations defined by $\varphi_{\mathbf{G}}$. It is only left to consider the simultaneous fixed-point of those three formulae, however, we must ensure that the initial values of the fixed-point second-order variables are correctly set:

$$\mathsf{proto}_{\mathrm{table}}(R, S, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := (\text{s-ifp}_{\vec{\mu}\vec{\nu}\vec{s}\vec{t}R';\vec{\lambda}\vec{s}\vec{t}S';\vec{p}I} \mathrm{step}'_{table}; \mathrm{step}'_{list}; \mathrm{step}_{ind})$$
$$\mathsf{proto}_{\mathrm{list}}(R, S, \vec{\lambda}, \vec{s}, \vec{t}) := (\text{s-ifp}_{\vec{\lambda}\vec{s}\vec{t}S';\vec{\mu}\vec{\nu}\vec{s}\vec{t}R';\vec{p}I} \mathrm{step}'_{list}; \mathrm{step}'_{table}; \mathrm{step}_{ind})$$

Indeed, by definition of the s-ifp operator, all second-order variables are intiated to the empty relations, while we expect the iteration of the step-formulae to apply initially on the relations R, S given as input to the formulae $\mathsf{proto}_{\mathrm{table}}, \mathsf{proto}_{\mathrm{list}}$. This motivates the following definition of the formulae step′:

$$\mathrm{step}'_{table}(R, S, R', S', I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := \bigvee \begin{cases} \forall \vec{p}, \neg I(\vec{p}) \wedge R(\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \\ \mathrm{step}'_{table}(R', S', I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \end{cases}$$

$$\mathrm{step}'_{list}(R, S, R', S', I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := \bigvee \begin{cases} \forall \vec{p}, \neg I(\vec{p}) \wedge S(\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \\ \mathrm{step}'_{list}(R', S', I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \end{cases}$$

□

Saturating a PSGS We have now defined almost all the parts of the construction procedure. We now show how to turn a PSGS prototype for some group $\mathbf{K}(\mathfrak{A})$ into a PSGS for $\mathfrak{A}$. Remark that, the only difference between the two notions is that if (R, S) is a PSGS for some chain of subgroups $(\mathbf{K}_i(\mathfrak{A}))$, the i-th row of R constitutes a transversal of $\mathbf{K}_{i+1}(\mathfrak{A})$ in $\mathbf{K}_i(\mathfrak{A})$ (see Definitions 5.3 and 5.6).

Lemma 5.9. *There is a couple of* $(\mathsf{FP} + \mathsf{ord})[\Sigma]$ *formulae* $\mathsf{sat}_{\mathrm{table}}(R, S, \vec{\mu}, \vec{s}, \vec{t})$ *and* $\mathsf{sat}_{\mathrm{list}}(R, S, \vec{\lambda}, \vec{s}, \vec{t})$ *such that, given a structure* $\mathfrak{A}$ *and a PSGS prototype* $(R^{\mathfrak{A}}, S^{\mathfrak{A}})$ *for some group* $\mathbf{K}(\mathfrak{A})$*,* $R' := \mathsf{sat}_{\mathrm{table}}(\mathfrak{A})$ *and* $S' := \mathsf{sat}_{\mathrm{list}}(\mathfrak{A})$ *form a PSGS for the same group.*

Proof. Following Algorithm 2, we consider all products of the form $\sigma\tau$ with $\sigma, \tau \in \bigcup \mathcal{R} \cup \mathcal{S}$, and insert them (as defined by the ins formulae) into the PSGS prototype defined by (R, S), until saturation (and thus a fixed-point) is reached.

It is vital not to insert simultaneously two such products, as inserting two different permutations at the same index would break the very structure of PSGS prototypes[3]. This is where the ordering on the indexings of R and S show most important, as they enable the definition of an ordered enumeration of products of couples of elements appearing in (R, S), and we first exhibit this enumeration. In the remainder of this proof, we use tuples of the form $\vec{n} = (b_n, \vec{\mu}_n, \vec{\nu}_n, \vec{\lambda}_n)$ to represent the variables indexing the following enumeration:

$$\mathrm{enum}(R, S, \vec{m}, \vec{n}, \vec{s}, \vec{t}) := \exists \vec{u}, \bigwedge \begin{cases} \bigvee \begin{cases} b_m = 0 \wedge R(\vec{\mu}_m, \vec{\nu}_m, \vec{s}, \vec{u}) \\ b_m = 1 \wedge S(\vec{\lambda}_m, \vec{s}, \vec{u}) \end{cases} \\ \bigvee \begin{cases} b_n = 0 \wedge R(\vec{\mu}_n, \vec{\nu}_n, \vec{u}, \vec{t}) \\ b_n = 1 \wedge S(\vec{\lambda}_n, \vec{u}, \vec{t}) \end{cases} \end{cases}$$

Using the same method as in Corollary 5.8, we can use this enumeration to keep track of all pairs of elements whose product has already been inserted in (R, S) in a second-order relation I. After one step, here is how I should be updated:

$$\mathsf{step}^{\mathsf{sat}}_{ind}(R, S, I, \vec{m}, \vec{n}) := \bigwedge \begin{cases} \mathrm{Bij}(X)[X(\vec{x}, \vec{y})/\mathrm{enum}(R, S, \vec{m}, \vec{n}, \vec{x}, \vec{y})] \\ \forall \vec{m}'\vec{n}' < \vec{m}\vec{n}, \bigvee \begin{cases} I(\vec{m}', \vec{n}') \\ \neg \mathrm{Bij}(\mathrm{enum}(R, S, \vec{m}', \vec{n}')) \end{cases} \end{cases}$$

where $\mathrm{Bij}(X)$ is a formula which holds iff X is a $2k$-ary relation which is the graph of a bijection. With this upkeeping of the set of indices defined, an iteration of the saturation process can be defined as follows:

$$\mathsf{step}^{\mathsf{sat}}_{table}(R, S, I, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) := \bigvee \begin{cases} R(\vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \\ \bigwedge \begin{cases} \forall \vec{s}', \vec{t}', \neg R(\vec{\mu}, \vec{\nu}, \vec{s}', \vec{t}') \\ \exists \vec{m}, \vec{n}, \bigwedge \begin{cases} \neg I(\vec{m}, \vec{n}) \wedge \mathsf{step}^{\mathsf{sat}}_{ind}(R, S, I, \vec{m}, \vec{n}) \\ \mathrm{ins}^{*}_{\mathrm{table}}(R, R_\sigma, \vec{\mu}, \vec{\nu}, \vec{s}, \vec{t}) \end{cases} \end{cases} \end{cases}$$

[3]This would lead inserting the *union* of the graphs of the two permutations, which is not the graph of a permutation

$$\mathsf{step}^{\mathsf{sat}}_{list}(R,S,I,\vec{\lambda},\vec{s},\vec{t}) := \begin{cases} S(\vec{\lambda},\vec{s},\vec{t}) \\ \circledvee \begin{cases} \forall \vec{s}',\vec{t}', \neg S(\vec{\lambda},\vec{s}',\vec{t}') \\ \circledwedge \begin{cases} \exists \vec{m},\vec{n}, \circledwedge \begin{cases} \neg I(\vec{m},\vec{n}) \wedge \mathsf{step}^{\mathsf{sat}}_{ind}(R,S,I,\vec{m},\vec{n}) \\ \mathrm{ins}^{*}_{\mathrm{list}}(R,S,R_\sigma,\vec{\mu},\vec{\nu},\vec{s},\vec{t}) \end{cases} \end{cases} \end{cases}$$

where, here, $\mathrm{ins}^{*}_{\mathrm{table}}$ (resp. $\mathrm{ins}^{*}_{\mathrm{list}}$) is the formula $\mathrm{ins}_{\mathrm{table}}$ (resp. $\mathrm{ins}_{\mathrm{list}}$), where each occurence of $R_\sigma(\vec{x},\vec{y})$ is substituted by $\mathrm{enum}(R,S,\vec{m},\vec{n})$. Notice that only one permutation is added per iteration of

$$(R,S,I) \mapsto (\mathsf{step}_{table}(\mathfrak{A},R,S,I), \mathsf{step}_{list}(R,S,I), \mathsf{step}_{ind}(R,S,I))$$

To obtain a completely saturated couple (R,S), it is only left to consider the least fix-point of this iteration. This is once again an application of the simultaneous fixed-point operator, where special care is taken with regards to initial values:

$$\mathsf{sat}_{\mathrm{table}}(R,S) := (\text{s-ifp}_{\vec{\mu}\vec{\nu}\vec{s}\vec{t}R;\vec{\lambda}\vec{s}\vec{t}S;\vec{m},\vec{n},I}\mathsf{step}^{\mathsf{sat}}_{table}; \mathsf{step}^{\mathsf{sat}}_{list}; \mathsf{step}^{\mathsf{sat}}_{ind})$$
$$\mathsf{sat}_{\mathrm{list}}(R,S) := (\text{s-ifp}_{\vec{\lambda}\vec{s}\vec{t}S;\vec{\mu}\vec{\nu}\vec{s}\vec{t}R;\vec{m},\vec{n},I}\mathsf{step}^{\mathsf{sat}}_{list}; \mathsf{step}^{\mathsf{sat}}_{table}; \mathsf{step}^{\mathsf{sat}}_{ind})$$

Once again, in those expressions, the formulae $\mathsf{step}^{\mathsf{sat}}_{table}$ and $\mathsf{step}^{\mathsf{sat}}_{list}$ should be substituted by formulae taking into account the case where $I = \emptyset$ (to detect the initial situtation), where they should evaluate respectively to the values of R and S given as input to $\mathsf{sat}_{\mathrm{table}}$ and $\mathsf{sat}_{\mathrm{list}}$. This is done exactly as in the proof of Corollary 5.8. Note that, to avoid the variable capture of R and S, those variables should be substituted in the definition of the $\mathsf{step}^{\mathsf{sat}}$ formulae, so that the s-ifp operator does not bind them.

It is only left to show that, if (R,S) is a PSGS prototype,

$$(R',S') := (\mathsf{sat}_{\mathrm{table}}(\mathfrak{A},R,S), \mathsf{sat}_{\mathrm{list}}(A,R,S))$$

constitutes a PSGS for the same group. That is, for any $i < n^k$, we must show that $R'(i)$ defines a transversal of the cosets of $\mathbf{G}_{i+1}(\mathfrak{A})$ in $\mathbf{G}_i(\mathfrak{A})$.

The following argument is a direct adaptation of [FHL80, Theorem 1]. Notice that (R',S') is a PSGS if any element $g \in \mathbf{K}(\mathfrak{A})$ can be written as a product $r_1 r_2 \dots r_{n^k-1} s$ with $r_i \in R'(i)$ and $s \in S'$. Indeed, in such a case, if $g \in G_i$ and $g \notin G_{i+1}$, it must hold that $r_1 = r_2 = \dots = r_{i-1} = \mathrm{Id}$ (as those are the initially set unique representatives in R' of the coset $\mathrm{Id}G_{\lambda+1}$ in G_λ), and thus, r_i is a representative of the coset gG_{i+1}.

Now, consider $g \in \mathbf{K}(\mathfrak{A})$, and let us show that g can be written as such a product. Because $\mathbf{K}(\mathfrak{A}) = \langle \bigcup R' \cup S' \rangle$, let $s_1 \dots s_m$ be a decomposition of g in elements of $\bigcup R' \cup S'$. Define the *index* of such an element $x \in \bigcup R' \cup S'$, denoted $i(x)$, as the unique i such that $x \in R'(i)$ if such an i exists, or $i(x) = n^k$ if $x \in S'$. Let y be an element of minimal index whose predecessor x in the sequence $s_1, \dots, s_m$ is such that $i(x) \geq i(y)$. Because x and y both belong to $\bigcup R' \cup S'$, their product has

been inserted in the table, and therefore their product admit a decomposition of the form $r_1 r_2 \dots r_{n^k-1} s$ with $r_i \in R'(i)$ and $s \in S'$. Moreover, since $xy \in G_{i(y)}$, for all $j < i(y)$, $r_j = \mathrm{Id}$. Therefore, rewriting xy as $r_{i(y)+1} \dots r_{n^k-1}$ in $s_1 \dots, s_m$ yields another decomposition of g. This process can be iterated until no such y can be found, at which point the sequence is such that, for any $\mu < \nu$, $i(s_\mu) < i(s_\nu)$, i.e. we have found an adequate decomposition of g. □

Conclusion

We now conclude our proof that $\mathbf{H}(\mathfrak{A})$ admits an $\mathsf{FP} + \mathsf{ord}$ definable generating set:

Proof of Theorem 5.2. The proto formulae already define a PSGS prototype for $\mathbf{G}(\mathfrak{A})$ through $(\mathbf{G}_i(\mathfrak{A}))_{i=0}^{n^k}$. Applying the sat formulae then yields an actual PSGS for this chain of subgroup, and the residual list constitutes a generating set for $\mathbf{H}(\mathfrak{A})$:

$$\varphi_{\mathbf{H}}(\vec{\lambda}, \vec{s}, \vec{t}) := \mathsf{sat}_{\mathrm{list}}(R, S)[R/\mathsf{proto}_{\mathrm{table}}, S/\mathsf{proto}_{\mathrm{list}}]$$ □

While our proof is now complete, there is a final remark to be made: we have used the ord operator very scarcely. Indeed, the ord operator has only been used in $\mathrm{ins}_{\mathrm{list}}$ to ensure that we are not adding "superfluous" elements to S. This ensured that the size of S could not reach $\log_2(\mathrm{Sym}(A^T)) \leq |A|^{|T|} \log(|A|^{|T|}) \leq |A|^{2|T|}$.

We will now show that, when $\mathbf{G}_0$ is abelian, we can enforce a polynomial bound on the size of S in another way, without using the ord operator at all.

Corollary 5.10. *Let Σ be a signature, $\mathcal{K} \subseteq \mathrm{STRUC}[\Sigma]$ and suppose that $\mathbf{H}$ is a function, mapping any structure $\mathfrak{A} \in \mathcal{K}$ to a group $\mathbf{H}(\mathfrak{A}) \leq \mathrm{Sym}(A^T)$ for some fixed type T.*

If FPC witnesses the k-accessibility of $\mathbf{H}$ in $\mathcal{K}$ in the ordered setting, and for any $\mathfrak{A} \in \mathcal{K}$, the largest group $\mathbf{G}(\mathfrak{A})$ in the chain of subgroups witnessing the accessibility of $\mathbf{H}(\mathfrak{A})$ is abelian, there is a formula $\varphi_{\mathbf{H}} \in \mathsf{FPC}[\Sigma]$ defining a generating set for $\mathbf{H}(\mathfrak{A})$ for any $\mathfrak{A} \in \mathcal{K}$.

Sketch of proof. The fact that all the groups in the chain are abelian has a radical implication on the saturation procedure: we do not need to consider products of permutations of the form $\sigma\tau$ where either σ or τ belong to S.

To illustrate this, consider a PSGS prototype for a chain of abelian subgroups $G_1 \geq \dots \geq G_m$, two integers $i < j < m$, and let us review the argument used in Lemma 5.9 to show that (R', S') was a PSGS. Here, given $g \in G_1$, we can once again write g as $s_1 \dots s_m$ where $s_i \in R' \cup S'$ but here, the commutativity of the groups enable us to directly sort this sequence into an index-increasing one, i.e. a sequence $s'_1 \dots s'_m$ such that for all $\lambda \leq \mu$, $i(s'_\lambda) \leq i(s'_\mu)$ (where $i(x)$ is the index of x, as defined in the

proof of Lemma 5.9). As such, it is only needed, in order to turn $s'_1 \dots s'_m$ into a *strictly* index-increasing decomposition of g, to sift all products of pairs of elements x, y which belong to the *same* row $R(i)$, for each i.

This implies that, in the sat procedure, we can actually reduce the indexing tuples to be of the form $\vec{m} = (\vec{\mu}_m, \vec{\nu}_m)$ (that is, to track positions within R) and change the enumeration accordingly. This in turn implies that the total number of insertions into $(\mathcal{R}, \mathcal{S})$ is bounded by the size of our generating set for $\mathbf{G}(\mathfrak{A})$ — which is bounded by $|A|^{|\vec{p}|}$ — and the number of products of elements from the transversal table — which is bounded by $|A|^{2(|\vec{\mu}|+|\vec{\nu}|)} = |A|^{4k}$.

In this context, no further filtering of the permutations added to S is necessary to enforce a polynomial bound on S. As such, in the restricted setting where $\mathbf{G}(\mathfrak{A})$ is abelian, we can change the size of $\vec{\lambda}$ used in the relation variable S from $2|T|$ to $4k$, and simply remove the clause of ins_{list} used to ensure that we only insert elements which actually increase the group generated by S. All other formulae in the proof of Theorem 5.2 must be changed accordingly to reflect the changed length of λ, but otherwise remain the same, and yield the desired result. □

5.3 Coloured Graph Automorphisms

In Chapter 4, we have shown that FP + ord is able to canonise structures with Abelian colours. Such structures have recently played an important role in the quest of a logic for P, as the class of structures shown to separate FP + rk from CPT (and P) in [Lic23] are known to have Abelian colours.

Recall that a structure $\mathfrak{A}$ has abelian colours if it is equipped with a total pre-order $\preceq^{\mathfrak{A}}$ and relation $\Phi^{\mathfrak{A}}$ such that, for each i, $\Phi(\mathfrak{A}, i)$ defines some transitive, abelian (and ordered) group Γ_i acting on the i-th colour class of $\mathfrak{A}$ (where the colour classes are defined by $\preceq^{\mathfrak{A}}$). Recall also that we have shown in Lemma 4.20 that in such a case, FPC can refine $\prec$ and redefine Φ accordingly so that, for each i, $\Gamma_i = \text{Aut}(\mathfrak{A}_i)$.

We will now see that, under this condition that for each i, $\Gamma_i = \text{Aut}(\mathfrak{A}_i)$, we can relax most of the group-theoretic assumptions on the groups defined by $\Phi(\mathfrak{A}, i)$, and still define the automorphism group of $\mathfrak{A}$ within FP + ord. Precisely, we will show that, if for all i, $|\text{Aut}(\mathfrak{A}_i)| < |A|^k$ for some fixed k, and $\Phi(\mathfrak{A}, i)$ *generates* $\text{Aut}(\mathfrak{A}_i)$, FP + ord defines the automorphism group of $\mathfrak{A}$. Note that we no longer require the groups to be abelian, transitive, or even fully enumerated by Φ, but merely that Φ provides an ordered generating set for those groups (and that their sizes remain polynomially bounded by $|A|$). Even more so, when the groups $\text{Aut}(\mathfrak{A}_i)$ *are* abelian, Corollary 5.10 applies, and we can define a generating set for $\text{Aut}(\mathfrak{A})$ in FPC. This should be contrasted with the fact that the class of structures considered contains the CFI-structures from [Lic23], which separate FP + rk from P: in this specific context, a strong distinction seems to exist between the Automorphism problem (which we will thus show to be FPC definable), and the Canonisation problem (which cannot be defined in FP + rk, by [Lic23]).

This is due, at least in part, to the fact that the ordering of the groups $\mathrm{Aut}(\mathfrak{A}_i)$ provided by Φ imply that the classes of structures under consideration are not union-closed, which implies that we cannot easily obtain an isomorphism test from the definability of the automorphism groups of the structure through Proposition 1.73. Whether the Isomorphism problem for such structures is definable in $\mathsf{FP} + \mathsf{ord}$ in the general case remains, to our knowledge, open.

A generalisation of bounded colour-classes

In order to introduce our generalisation, we review the group-theoretic framework to the automorphism of coloured graphs, introduced in [Bab79]. We have already presented this idea in Section 1.2.

Let $\mathfrak{A}$ be a graph over a set A with n vertices equipped with a total pre-order $\preceq^{\mathfrak{A}}$ defining a colouring $c : A \to [m]$, we denote $\mathfrak{A}_i$ the subgraph of $\mathfrak{A}$ induced by $c^{-1}(i)$ for any $i \in c(A)$. $\mathrm{Aut}_c(\mathfrak{A})$ is the group of colour preserving automorphisms of $\mathfrak{A}$. Let $\mathcal{I} := \binom{[m]}{2}$, and fix a linear-ordering of $\mathcal{I}$. We denote s_λ the λ-th element of I with regards to this ordering. For any $s = \{i, j\} \in \mathcal{I}$, we denote $\mathfrak{A}_s$ the subgraph of $\mathfrak{A}$ induced by $c^{-1}(i) \cup c^{-1}(j)$. We can define the following tower of subgroups:

$$G_\lambda := \left\{ \sigma \in \prod_{i \in [m]} \mathrm{Aut}(\mathfrak{A}_i) \middle| \forall \kappa < \lambda, \sigma_{\restriction A_{s_\kappa}} \in \mathrm{Aut}_c(\mathfrak{A}_{s_\kappa}) \right\} \tag{5.3}$$

By definition, for $\lambda, \kappa \leq \binom{m}{2}$, if $\lambda \leq \kappa$ then $G_\kappa \leq G_\lambda$. We aim to delimit under which hypothesis this chain of subgroups witnesses the accessibility of $\mathrm{Aut}_c(\mathfrak{A})$. Notice that $G_{\binom{m}{2}} = \mathrm{Aut}_c(\mathfrak{A})$, and that conditional membership for $G_{\lambda+1}$ in G_λ is clearly P-decidable. Thus, we are left with two conditions to fulfil:

(1) A generating set for $G_0 = \prod_{i \in [m]} \mathrm{Aut}(\mathfrak{A}_i)$ can be computed in polynomial time.

(2) There is a constant ℓ such that $\forall \lambda < \binom{m}{2}, |G_\lambda : G_{\lambda+1}| \leq n^\ell$

Fulfilling those conditions depends on the restrictions we set on the colour-classes.

We now introduce a quite general class of coloured graphs that fulfils (2): Polynomially bounded colour-class graphs. We say that $\mathfrak{A}$ is a *k-Polynomially Bounded Colour-class* graph (denoted $\mathfrak{A} \in \mathrm{PBCG}_k$) if $\mathfrak{A}$ is a graph equipped with a colouring such that, for every colour i of $\mathfrak{A}$, $|\mathrm{Aut}(\mathfrak{A}_i)| < n^k$.

Lemma 5.11. *If $\mathfrak{A} \in \mathrm{PBCG}_k$, then for all $\lambda < \binom{m}{2}$, $|G_\lambda : G_{\lambda+1}| < n^{2k}$.*

Proof. For all λ, let i, j such that $i < j$ and $\{i, j\} = s_\lambda$. $\pi_\lambda(\sigma) := (\sigma_{\restriction A_i}, \sigma_{\restriction A_j})$ defines a morphism $\pi_\lambda : G_0 \to \mathrm{Aut}(\mathfrak{A}_i) \times \mathrm{Aut}(\mathfrak{A}_j)$. For $\lambda < \binom{m}{2}$ and $\sigma, \tau \in G_\lambda$, if $\pi_\lambda(\sigma) = \pi_\lambda(\tau)$, then $\sigma G_{\lambda+1} = \tau G_{\lambda+1}$. Indeed, we then have :

$$(\sigma^{-1}\tau)_{\restriction A_{s_\lambda}} = 1 \in \mathrm{Aut}(\mathfrak{A}_{s_\lambda})$$

Therefore,

$$\begin{aligned}|G_\lambda : G_{\lambda+1}| &\leq |\mathrm{Im}(\pi_\lambda)| \\ &\leq |\mathrm{Aut}(\mathfrak{A}_i)| \cdot |\mathrm{Aut}(\mathfrak{A}_j)| \\ &\leq n^k \cdot n^k = n^{2k}\end{aligned}$$ □

In order to fulfil (1), we additionally require a generating set for $\mathrm{Aut}(\mathfrak{A}_i)$ to be explicitly given, for each i. However, as was the case for structures with Abelian colours, we require this enumeration to be indexed on the numerical domain. We obtain the following definition:

Definition 5.12. An instance of $\mathrm{PBCG}_k^<$ is a coloured graph $\mathfrak{A} = (A, E^{\mathfrak{A}}, \preceq^{\mathfrak{A}}, \Phi^{\mathfrak{A}})$ such that $(A, E^{\mathfrak{A}}, \preceq^{\mathfrak{A}}) \in \mathrm{PBCG}_k$, and Φ is a relation of type $\mathrm{number}^{k+1} \cdot \mathrm{element}^2$ such that for every i, $\mathrm{Aut}(\mathfrak{A}_i) = \langle\Phi\rangle_{\vec{p}}(\mathfrak{A}, i)$.

Remark that we have assumed the generating set for $\mathrm{Aut}(\mathfrak{A}_i)$ to be given by a $k+3$-ary relation. In the context where $|\mathrm{Aut}(\mathfrak{A}_i)| < n^k$, and Φ is indexed over the numerical domain, this can be enforced by a simple trick:

Remark 5.13. For $k_1 < k_2$, consider $\mathrm{PBCG}_{k_1,k_2}^<$ be the class of coloured graphs $\mathfrak{A} = (A, E^{\mathfrak{A}}, \preceq^{\mathfrak{A}}, \Phi^{\mathfrak{A}})$ such that $(A, E^{\mathfrak{A}}, \preceq^{\mathfrak{A}}) \in \mathrm{PBCG}_{k_1}$, and Φ is a relation of type $\mathrm{number}^{k_2+1}\mathrm{element}^2$ such that for every i, $\mathrm{Aut}(\mathfrak{A}_i) = \langle\Phi\rangle_{\vec{p}}(\mathfrak{A}, i)$. There is an FPC-interpretation from $\mathrm{PBCG}_{k_1,k_2}^<$ to $\mathrm{PBCG}_{k_1}^<$.

Proof. We first define a formula that holds on $i, \vec{x}$ if $\Phi(i, \vec{x})$ is the graph of a permutation distinct from all permutations defined by $\Phi(i, \vec{y})$, for $\vec{y} < \vec{x}$:

$$\mathsf{new}(\mu, \vec{p}) := \forall \vec{q} < \vec{p}, \exists s, t, \Phi(\mu, \vec{p}, s, t) \wedge \neg\Phi(\mu, \vec{q}, \vec{s}, \vec{t})$$

We are now able to define a reindexing Φ' of Φ, which is indexed over tuples of $k_1 + 1$ numerical variables instead of $k_2 + 1$, that contains each permutation at most once:

$$\Phi'(\mu, \vec{x}, s, t) := \exists \vec{p}, \circledwedge \begin{cases} \mathsf{new}(\mu, \vec{p}) \\ (\#\vec{q}, (\vec{q} < \vec{p} \wedge \mathsf{new}(\mu, \vec{q}))) = \vec{x} \\ \Phi(\mu, \vec{p}, s, t) \end{cases}$$

Because $|\langle\Phi\rangle_{\vec{p}}(\mathfrak{A}, i)| < n^{k_1}$ for all i, and the permutations enumerated by Φ' are pairwise distinct, all permutations defined by Φ must occur in Φ'. (Note that if Φ contained $v < n^{k_1}$ permutations, $\Phi'(\mathfrak{A}, v + c)$ is empty for all c.) □

Theorem 5.14. $\mathsf{FP} + \mathsf{ord}$ *defines the automorphism group of* $\mathrm{PBCG}_k^<$ *structures.*

Proof. This is a direct application of Theorem 5.2, and it is only left to provide definitions for $\varphi_{\mathbf{G}}$, where $\mathbf{G}(\mathfrak{A}) := \prod_{i\in[m]} \mathrm{Aut}(\mathfrak{A}_i)$; and $\varphi_\in$ according to the chain of subgroups defined in Eq. (5.3).

Suppose that $(S_i)_{i\in[m]}$ is a family of sets of permutation sets such that S_i generates $\mathrm{Aut}(\mathfrak{A}_i)$ for all $i \in [m]$. Then, $\prod_{i\in[m]} \mathrm{Aut}(\mathfrak{A}_i) = \langle \bigcup_{i\in[m]} S_i \rangle$. As such, under the hypothesis of $\mathrm{PBCG}_k^<$, we can define $\varphi_{\mathbf{G}}(\mu, \vec{p}, s, t) := \Phi(\mu, \vec{p}, s, t)$, and it is easy to see that $\langle \varphi_{\mathbf{G}} \rangle_{\mu\vec{p}}(\mathfrak{A}) = \mathbf{G}(\mathfrak{A})$.

We now provide a definition for $\varphi_\in$ in accordance with Eq. (5.3). First, we set a fixed linear-ordering of $\mathcal{I}$ which is definable in FP, with $\mathcal{I}$ encoded over $(A^<)^2$:

$$((\mu_1, \nu_1) \leq (\mu_2, \nu_2)) := \begin{cases} \mu_1 < \nu_1 \wedge \mu_2 < \nu_2 \\ \begin{cases} \mu_1 < \mu_2 \\ \mu_1 = \mu_2 \wedge \nu_1 \leq \nu_2 \end{cases} \end{cases}$$

We are now ready to give the definition of $\varphi_\in$:

$$\varphi_\in(\mu, \nu, R) := \forall x, y, (x \in A_\mu \wedge y \in A_\nu) \implies \exists x', y', \begin{cases} R(x, x') \wedge R(y, y') \\ E(x, y) \iff E(x', y') \end{cases}$$

If $R^{\mathfrak{A}}$ is the graph of a permutation $\sigma \in G_\lambda$, and $s_\lambda = \{i, j\}$ where $i < j$, $(\mathfrak{A}, i, j, R^{\mathfrak{A}}) \models \varphi_\in$ iff σ respects the edge relation on $\mathfrak{A}_{i,j}$. As such, $\sigma \in \mathrm{Aut}_c(\mathfrak{A}_{i,j})$, which is the expected condition. This concludes the proof as we can now apply Theorem 5.2 on the chain of subgroups defined by $(\varphi_{\mathbf{G}}, \varphi_\in)$, which yields a generating set for $\mathbf{H}(\mathfrak{A}) = \mathrm{Aut}_c(\mathfrak{A})$. □

Interestingly, if we were to reinstate the commutativity condition, and assume all the $\mathrm{Aut}(\mathfrak{A}_i)$ to be abelian groups, we can use Corollary 5.10 instead of Theorem 5.2:

Corollary 5.15. FPC *can define a generating set for the automorphism group of all structures in* $\mathrm{PBCG}_k^<$ *such that each colour class has an abelian group of automorphisms.*

This last result should be contrasted with Lichter's breakthrough [Lic23] $\mathsf{FP} + \mathsf{rk} < \mathsf{P}$. Indeed, the structures shown by Lichter to be indistinguishable in $\mathsf{FP} + \mathsf{rk}$ (which is strictly more expressive than FPC) are coloured in such a way that each colour class has a polynomially bounded, abelian, ordered automorphism group. As such, while $\mathsf{FP} + \mathsf{rk}$ cannot distinguish those structures, Corollary 5.15 applies, and FPC can define their automorphism groups.

While there is generally a polynomial-time reduction from the Graph Isomorphism problem over $\mathcal{C}$ to the Graph Automorphism problem over $\mathcal{C}$, this requires the class of graphs $\mathcal{C}$ at hand to be *union-closed*. In the current context, this is not the case. Indeed, proving that the class of "Abelian" $\mathrm{PBCG}_k{}^<$ is union-closed would entail, given two such structures $\mathfrak{A}, \mathfrak{B}$, to build, within the logic at hand, an ordered enumeration of the automorphisms of the subgraph induced by $A_i \cup B_i$, for each i. It is a corollary of Lichter's result and Corollary 5.15 that this cannot be done in neither FPC nor $\mathsf{FP} + \mathsf{rk}$. As a final digression, note that this type of arguments — where fixed-point logics are unable to handle all potential swaps between two colour-classes — is already present in Section 3.2, where we showed that $\mathsf{FP} + \mathsf{ord}$ is unable to define some accessible subgroups in the unordered context.

Conclusion

In this PhD dissertation, we have introduced ord, an operator which enables the definition of the order of a group generated by a definable set of permutations. We showed that this operator is a generalisation of rk, the rank operator (Section 3.4) and that FP + ord is strictly more expressive than FP + rk (Chapter 4). More precisely, we have shown FP + ord to capture polynomial-time computation over the class of structures which was shown to separate FP + rk from P. As such, FP + ord constitutes a new candidate logic for P.

This capture result relies on the definition within FP + ord of the canonisation algorithm defined by Babai and Luks in [BL83]. The definition of the ord operator was motivated precisely by algorithms of this kind, which pervasively represent permutation groups by generating sets of permutations, harnessing the polynomial-time operations enabled by the Schreier-Sims algorithm. However, actually implementing such an algorithm in an isomorphism-invariant context has led us to consider different representations, as isomorphism-invariance hinders the definability of certain subgroups, as showcased in Section 3.2. More precisely, we have shown in this section that the definition of a generating set for an accessible subgroup H of a group G provided by a generating set — another fundamental application of the Schreier-Sims algorithm — is in general incompatible with isomorphism-invariance.

This led us to study restricted cases in which this issue can be circumvented. First, we have considered the situation where H is a morphism-definable subgroup of G and a morphism can be defined on G which cancels itself exactly on elements on H. In such a case, we have shown in Section 3.3 that, together with the generating set of G, this morphism *itself* constitutes a satisfying representation of H, in the sense that the primitive operations on groups enabled by the ord operator (as studied in Section 3.1) can be transferred to this new representation in FP + ord. This result is central to the canonisation of structures with abelian colours presented in Chapter 4. In Chapter 5, we considered another restricted case: when the generating set for G is ordered. We showed that in such circumstances, a generating set for any accessible subgroup of H is definable. We did so by defining a partial simulation of the Schreier-Sims algorithm.

Interestingly, the results of both Chapters 4 and 5 shared a common feature: the additional assumptions at hand enabled the representation of *cosets*. Handling cosets is quite important in the group-theoretic algorithms for Graph Isomorphism, and even more so in the context of Graph Canonisation. Yet, the representation of cosets used in such an algorithmic context is not isomorphism-invariant.

Ultimately, our results spotlight the deep connection between the quest of a logic for P and the computational study of permutation groups. First, Corollary 4.36 implies that $\mathsf{FP} + \mathsf{rk}$ is unable to define the order of a group provided by a generating set of permutations. Moreover, the inability to represent certain groups by generating sets of permutations showcased in Section 3.2 could be a first step towards actual inexpressibility results in CPT or $\mathsf{FP} + \mathsf{ord}$. Finally, our results show that the expressivity of isomorphism-invariant models of computation can benefit from additional group-theoretic primitives.

This last assertion is the original motivation for our work. In order to confirm this work hypothesis, we set out to consider a minimal set of group-theoretic operations which would effectively increase the expressive power of fixed-point logic. As such, many other group primitives have yet to be studied in a logical context. In particular, one way to circumvent the limitations exhibited in Section 3.2 would be to introduce a *group sort*, not unlike the *numerical sort* which was needed to define FPC. In such a case, strong restrictions on the quantification of group variables would have to be enforced to maintain polynomial-time model-checking for the resulting logic. Finding such syntactic restrictions which also provide sufficient expressive power to capture a robust set of group-theoretic operations could constitute a challenge in and of itself.

Another further direction of research would be to extend the study of $\mathsf{FP}+\mathsf{ord}$. First, one could wonder whether $\mathsf{FP} + \mathsf{ord}$ captures P. While we do not believe this to be the case, obtaining a separation result will necessarily rely on the introduction of new techniques. Indeed, we have shown in Chapter 4 that $\mathsf{FP}+\mathsf{ord}$ captures all structures on which abelian colours can be defined, which encompasses all known generalisations of the CFI-construction, over ordered graphs. Although we have not shown this precisely, it is to be expected that, over unordered graphs, $\mathsf{FP} + \mathsf{ord}$ expresses the generalised CFI-query. Indeed, this query has been shown to encode the satisfiability of a system of equations over rings, and as noted in Remark 3.23, this problem is definable in $\mathsf{FP} + \mathsf{ord}$. Hence, it seems that CFI-structures might not be the right framework to obtain inexpressibility results for $\mathsf{FP} + \mathsf{ord}$. Instead, it might be interesting to try to obtain generalisations of *multipedes* [GS96]. Like CFI-structures, this class of structures have been shown to separate FPC from P. Additionally, multipedes are *rigids*, i.e., they have no non-trivial automorphisms. Having shown that $\mathsf{FP} + \mathsf{ord}$ is able to leverage the automorphisms of the structure at hand, the rigidity of multipedes might reduce its utility.

Finally, in the face of the unfitness of the representation of groups by generating sets of permutations (in the isomorphism-invariance context), we have already argued that finding new isomorphism-invariant representations of permutations groups is an interesting problem to tackle, as it may provide capture results for $\mathsf{FP} + \mathsf{ord}$ on additional classes of structures. To make this assertion precise, recall that our canonisation result in Chapter 4 relies on the simulation of the canonisation procedure from [BL83] (see Algorithm 3). In the same article, the authors also introduced a notion of *composition width of a group*. This is a metric of the structural complexity of the group which parameterizes the time-complexity of the canonisation algorithm mentionned. That

is, if one can produce a labeling coset of a group of bounded-composition width on which to initiate the canonisation procedure, this procedure runs in polynomial-time. Still in [BL83], it is shown how this fact enables the polynomial-time canonisation of graphs of bounded colour-class size, and also of graphs of bounded degree. Finding satisfactory representations of groups of bounded composition-width in $\mathsf{FP} + \mathsf{ord}$ thus constitutes a first step towards capture results over those important classes of graphs.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press, 2009. ISBN: 978-0-521-42426-4. DOI: 10.1017/CBO9780511804090. (Visited on 08/07/2024).

[ABD07] Albert Atserias, Andrei Bulatov, and Anuj Dawar. "Affine Systems of Equations and Counting Infinitary Logic". In: *Automata, Languages and Programming*. Ed. by Lars Arge et al. Berlin, Heidelberg: Springer, 2007, pp. 558–570. ISBN: 978-3-540-73420-8. DOI: 10.1007/978-3-540-73420-8_49.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN: 978-0-201-53771-0.

[AV91] Serge Abiteboul and Victor Vianu. "Generic Computation and Its Complexity". In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. STOC '91. New York, NY, USA: Association for Computing Machinery, Jan. 1991, pp. 209–219. ISBN: 978-0-89791-397-3. DOI: 10.1145/103418.103444. (Visited on 01/11/2025).

[AV95] S. Abiteboul and V. Vianu. "Computing with First-Order Logic". In: *Journal of Computer and System Sciences* 50.2 (Apr. 1995), pp. 309–335. ISSN: 0022-0000. DOI: 10.1006/jcss.1995.1025. (Visited on 02/24/2025).

[Bab16] László Babai. *Graph Isomorphism in Quasipolynomial Time*. Jan. 2016. DOI: 10.48550/arXiv.1512.03547. arXiv: 1512.03547 [cs, math]. (Visited on 06/18/2022).

[Bab79] László Babai. "Monte-Carlo Algorithms in Graph Isomorphism Testing". In: *Université de Montréal Technical Report, DMS* 79-10 (1979). URL: https://people.cs.uchicago.edu/~laci/lasvegas79.pdf.

[Bar+01] David A. Mix Barrington et al. "On the Complexity of Some Problems on Groups Input as Multiplication Tables". In: *Journal of Computer and System Sciences* 63.2 (Sept. 2001), pp. 186–200. ISSN: 00220000. DOI: 10.1006/jcss.2001.1764. (Visited on 05/03/2024).

[Bar72] Roland Barthes. *Le degré zéro de l'écriture suivi de Nouveaux essais critiques*. Points 35. Paris: Éd. du Seuil, 1972. ISBN: 978-2-02-000610-1.

[BGS97] Andreas Blass, Yuri Gurevich, and Saharon Shelah. *Choiceless Polynomial Time*. May 1997. DOI: 10.48550/arXiv.math/9705225. arXiv: math/9705225. (Visited on 07/13/2022).

[BL83] László Babai and Eugene M. Luks. "Canonical Labeling of Graphs". In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing - STOC '83*. ACM Press, 1983, pp. 171–183. ISBN: 978-0-89791-099-6. DOI: 10.1145/800061.808746. (Visited on 09/25/2022).

[Bla24] Alan F. Blackwell. *Moral Codes: Designing Alternatives to AI*. The MIT Press, Aug. 2024. ISBN: 978-0-262-37920-5. DOI: 10.7551/mitpress/14872.001.0001. (Visited on 02/04/2025).

[BM91] David A. Mix Barrington and Pierre McKenzie. "Oracle Branching Programs and Logspace versus P". In: *Information and Computation* 95.1 (Nov. 1991), pp. 96–115. ISSN: 08905401. DOI: 10.1016/0890-5401(91)90017-V. (Visited on 05/06/2024).

[CFI92] Jin-Yi Cai, Martin Fürer, and Neil Immerman. "An Optimal Lower Bound on the Number of Variables for Graph Identification". In: *Combinatorica* 12.4 (Dec. 1992), pp. 389–410. ISSN: 1439-6912. DOI: 10.1007/BF01305232. (Visited on 06/18/2022).

[CH82] Ashok Chandra and David Harel. "Structure and Complexity of Relational Queries". In: *Journal of Computer and System Sciences* 25.1 (Aug. 1982), pp. 99–128. ISSN: 0022-0000. DOI: 10.1016/0022-0000(82)90012-5. (Visited on 06/18/2022).

[Cho59] Noam Chomsky. "On Certain Formal Properties of Grammars". In: *Information and Control* 2.2 (June 1959), pp. 137–167. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(59)90362-6. (Visited on 01/26/2025).

[Daw+09] Anuj Dawar et al. "Logics with Rank Operators". In: *2009 24th Annual IEEE Symposium on Logic In Computer Science*. Aug. 2009, pp. 113–122. DOI: 10.1109/LICS.2009.24.

[Daw+13] Anuj Dawar et al. "Definability of Linear Equation Systems over Groups and Rings". In: *Logical Methods in Computer Science* Volume 9, Issue 4 (Nov. 2013), p. 725. ISSN: 1860-5974. DOI: 10.2168/LMCS-9(4:12)2013. (Visited on 01/24/2025).

[Daw15] Anuj Dawar. "The Nature and Power of Fixed-Point Logic with Counting". In: *ACM SIGLOG News* 2.1 (Jan. 2015), pp. 8–21. DOI: 10.1145/2728816.2728820. (Visited on 10/25/2024).

[DFS22] Anuj Dawar and Felipe Ferreira Santos. "Separating LREC from LFP". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. Haifa Israel: ACM, Aug. 2022, pp. 1–13. ISBN: 978-1-4503-9351-5. DOI: 10.1145/3531130.3533368. (Visited on 05/10/2024).

[DM96] John D. Dixon and Brian Mortimer. *Permutation Groups*. Vol. 163. Graduate Texts in Mathematics. New York, NY: Springer, 1996. ISBN: 978-1-4612-6885-7 978-1-4612-0731-3. DOI: 10.1007/978-1-4612-0731-3. (Visited on 02/22/2025).

[DRR08] Anuj Dawar, David Richerby, and Benjamin Rossman. "Choiceless Polynomial Time, Counting and the Cai–Fürer–Immerman Graphs". In: *Annals of Pure and Applied Logic* 152.1 (Mar. 2008), pp. 31–50. ISSN: 0168-0072. DOI: 10.1016/j.apal.2007.11.011. (Visited on 06/18/2022).

[EF95] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Ed. by S. Feferman et al. Perspectives in Mathematical Logic. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. ISBN: 978-3-662-03184-1 978-3-662-03182-7. DOI: 10.1007/978-3-662-03182-7. (Visited on 07/09/2022).

[Fag74] Ronald Fagin. "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets". In: *Complexity of Computation (Proc. SIAM-AMS Sympos., New York, 1973)*. Vol. Vol. VII. SIAM-AMS Proc. Amer. Math. Soc., Providence, RI, 1974, pp. 43–73.

[FHL80] Merrick Furst, John Hopcroft, and Eugene Luks. "Polynomial-Time Algorithms for Permutation Groups". In: *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*. SFCS '80. USA: IEEE Computer Society, Oct. 1980, pp. 36–41. DOI: 10.1109/SFCS.1980.34. (Visited on 06/18/2022).

[GH98] F. Gire and H.K. Hoang. "An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct". In: *Information and Computation* 144.1 (July 1998), pp. 40–65. ISSN: 08905401. DOI: 10.1006/inco.1998.2712. (Visited on 06/06/2024).

[GM95] E. Gradel and G.L. Mccolm. "On the Power of Deterministic Transitive Closures". In: *Information and Computation* 119.1 (May 1995), pp. 129–135. ISSN: 08905401. DOI: 10.1006/inco.1995.1081. (Visited on 05/06/2024).

[GP19] Erich Grädel and Wied Pakusa. "Rank Logic Is Dead, Long Live Rank Logic!" In: *The Journal of Symbolic Logic* 84.1 (Mar. 2019), pp. 54–87. ISSN: 0022-4812, 1943-5886. DOI: 10.1017/jsl.2018.33. (Visited on 07/30/2023).

[GR02] Mikael Goldmann and Alexander Russell. "The Complexity of Solving Equations over Finite Groups". In: *Information and Computation* 178.1 (Oct. 2002), pp. 253–262. ISSN: 0890-5401. DOI: 10.1006/inco.2002.3173. (Visited on 12/01/2024).

[Gra+15] Erich Gradel et al. "Characterising Choiceless Polynomial Time with First-Order Interpretations". In: *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. Kyoto, Japan: IEEE, July 2015, pp. 677–688. ISBN: 978-1-4799-8875-4. DOI: 10.1109/LICS.2015.68. (Visited on 05/10/2024).

[Gro17] Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. 1st ed. Cambridge University Press, Aug. 2017. ISBN: 978-1-107-01452-7 978-1-139-02886-8. DOI: 10.1017/9781139028868. (Visited on 06/18/2022).

[GS86] Yuri Gurevich and Saharon Shelah. "Fixed-Point Extensions of First-Order Logic". In: *Annals of Pure and Applied Logic* 32 (Jan. 1986), pp. 265–280. ISSN: 0168-0072. DOI: 10.1016/0168-0072(86)90055-2. (Visited on 08/07/2024).

[GS96] Yuri Gurevich and Saharon Shelah. "On Finite Rigid Structures". In: *The Journal of Symbolic Logic* 61.2 (1996), pp. 549–562. ISSN: 0022-4812. DOI: 10.2307/2275675. JSTOR: 2275675. (Visited on 02/28/2025).

[Gur88] Yuri Gurevich. "Logic and the Challenge of Computer Science". In: *Current Trends in Theoretical Computer Science ed. Egon Boerger* (July 1988). URL: https://www.microsoft.com/en-us/research/publication/logic-challenge-computer-science/ (visited on 06/18/2022).

[Hod93] Wilfrid Hodges. *Model Theory*. 1st ed. Cambridge University Press, Mar. 1993. ISBN: 978-0-521-30442-9 978-0-521-06636-5 978-0-511-55157-4. DOI: 10.1017/CBO9780511551574. (Visited on 06/29/2024).

[Hol10] Bjarki Holm. "Descriptive Complexity of Linear Algebra". PhD thesis. University of Cambridge, 2010. URL: http://bjarkiholm.com/publications/Holm_2010_phd-thesis.pdf.

[Imm83] Neil Immerman. "Languages Which Capture Complexity Classes". In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing - STOC '83*. Not Known: ACM Press, 1983, pp. 347–354. ISBN: 978-0-89791-099-6. DOI: 10.1145/800061.808765. (Visited on 08/04/2024).

[Imm86] Neil Immerman. "Relational Queries Computable in Polynomial Time". In: *Information and Control* 68.1 (Jan. 1986), pp. 86–104. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(86)80029-8. (Visited on 06/18/2022).

[Imm99] Neil Immerman. *Descriptive Complexity*. New York, NY: Springer New York, 1999. ISBN: 978-1-4612-6809-3 978-1-4612-0539-5. DOI: 10.1007/978-1-4612-0539-5. (Visited on 07/13/2022).

[Isa08] I. Martin Isaacs. *Finite Group Theory*. Graduate Studies in Mathematics v. 92. Providence, R.I: American Mathematical Society, 2008. ISBN: 978-0-8218-4344-4.

[Kle56] S. C. Kleene. "Representation of Events in Nerve Nets and Finite Automata". In: *Automata Studies. (AM-34)*. Ed. by C. E. Shannon and J. McCarthy. Princeton University Press, 1956, pp. 3–42. ISBN: 978-0-691-07916-5. JSTOR: j.ctt1bgzb3s.4. URL: http://www.jstor.org/stable/j.ctt1bgzb3s.4 (visited on 02/05/2025).

[Koz77] Dexter Kozen. "Lower Bounds for Natural Proof Systems". In: *18th Annual Symposium on Foundations of Computer Science (Sfcs 1977)*. Oct. 1977, pp. 254–266. DOI: 10.1109/SFCS.1977.16. (Visited on 02/03/2025).

[Lag22] U. Dal Lago. "Implicit Computation Complexity in Higher-Order Programming Languages: A Survey in Memory of Martin Hofmann". In: *Mathematical Structures in Computer Science* 32.6 (June 2022), pp. 760–776. ISSN: 0960-1295, 1469-8072. DOI: 10.1017/S0960129521000505. (Visited on 02/05/2025).

[Lib04] Leonid Libkin. *Elements of Finite Model Theory*. Berlin, Heidelberg: Springer, 2004. ISBN: 978-3-642-05948-3 978-3-662-07003-1. DOI: 10.1007/978-3-662-07003-1. (Visited on 08/08/2024).

[Lic23] Moritz Lichter. "Separating Rank Logic from Polynomial Time". In: *J. ACM* 70.2 (Mar. 2023), 14:1–14:53. ISSN: 0004-5411. DOI: 10.1145/3572918. (Visited on 11/30/2024).

[Luk82] Eugene M. Luks. "Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time". In: *Journal of Computer and System Sciences* 25.1 (Aug. 1982), pp. 42–65. ISSN: 0022-0000. DOI: 10.1016/0022-0000(82)90009-5. (Visited on 06/18/2022).

[Neu87] P.M. Neumann. "Some Algorithms for Computing with Finite Permutation Groups". In: *Proceedings of Groups - St. Andrews 1985*. Ed. by Colin Matthew Campbell and E. F. Robertson. London Mathematical Society Lecture Note Series. Cambridge: Cambridge University Press, 1987, pp. 59–92. ISBN: 978-0-521-33854-7. DOI: 10.1017/CBO9780511600647.006. (Visited on 10/31/2024).

[Ott17] Martin Otto. *Bounded Variable Logics and Counting: A Study in Finite Models*. Lecture Notes in Logic. Cambridge: Cambridge University Press, 2017. ISBN: 978-1-107-16794-0. DOI: 10.1017/9781316716878. (Visited on 08/14/2024).

[Pak10] Wied Pakusa. "Finite Model Theory with Operators from Linear Algebra". In: 2010. URL: https://www.semanticscholar.org/paper/Finite-Model-Theory-with-Operators-from-Linear-Pakusa/32bd5ab2fde31e3876621140f4 (visited on 01/22/2025).

[Pak15] Wied Pakusa. “Linear Equation Systems and the Search for a Logical Characterisation of Polynomial Time”. PhD thesis. RWTH Aachen University, 2015. URL: http://publications.rwth-aachen.de/record/567588/files/567588.pdf?subformat=pdfa (visited on 07/19/2022).

[Rei08] Omer Reingold. “Undirected Connectivity in Log-Space”. In: *Journal of the ACM* 55.4 (Sept. 2008), pp. 1–24. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/1391289.1391291. (Visited on 05/06/2024).

[Ros10] Benjamin Rossman. “Choiceless Computation and Symmetry”. In: *Fields of Logic and Computation*. Ed. by Andreas Blass, Nachum Dershowitz, and Wolfgang Reisig. Vol. 6300. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 565–580. ISBN: 978-3-642-15024-1 978-3-642-15025-8. DOI: 10.1007/978-3-642-15025-8_28. (Visited on 07/11/2023).

[Sim67] Charles C. Sims. “Graphs and Finite Permutation Groups”. In: *Mathematische Zeitschrift* 95.1 (1967), pp. 76–86. ISSN: 0025-5874, 1432-1823. DOI: 10.1007/BF01117534. (Visited on 06/17/2024).

[Sim70] Charles C. Sims. “Computational Methods in the Study of Permutation Groups”. In: *Computational Problems in Abstract Algebra*. Ed. by John Leech. Pergamon, Jan. 1970, pp. 169–183. ISBN: 978-0-08-012975-4. DOI: 10.1016/B978-0-08-012975-4.50020-5. (Visited on 01/12/2023).

[Sim71] Charles C. Sims. “Computation with Permutation Groups”. In: *Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation*. SYMSAC ’71. New York, NY, USA: Association for Computing Machinery, Mar. 1971, pp. 23–28. ISBN: 978-1-4503-7786-7. DOI: 10.1145/800204.806264. (Visited on 01/12/2023).

[Sto76] Larry J. Stockmeyer. “The Polynomial-Time Hierarchy”. In: *Theoretical Computer Science* 3.1 (Oct. 1976), pp. 1–22. ISSN: 0304-3975. DOI: 10.1016/0304-3975(76)90061-X. (Visited on 08/07/2024).

[SW19] Pascal Schweitzer and Daniel Wiebking. “A Unifying Method for the Design of Algorithms Canonizing Combinatorial Objects”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. Phoenix AZ USA: ACM, June 2019, pp. 1247–1258. ISBN: 978-1-4503-6705-9. DOI: 10.1145/3313276.3316338. (Visited on 07/02/2024).

[Tor04] Jacobo Torán. “On the Hardness of Graph Isomorphism”. In: *SIAM Journal on Computing* 33.5 (Jan. 2004), pp. 1093–1108. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S009753970241096X. (Visited on 06/06/2024).

[Var82] Moshe Y. Vardi. “The Complexity of Relational Query Languages (Extended Abstract)”. In: *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. STOC ’82. New York, NY, USA: Association for Computing Machinery, May 1982, pp. 137–146. ISBN: 978-0-89791-070-5. DOI: 10.1145/800070.802186. (Visited on 06/18/2022).

[Vin49] Leonardo da Vinci. *Paragone, a Comparison of the Arts by Leonardo Da Vinci, with an Introduction and English Translation by Irma A. Richter.* Oxford University Press, 1949.

Index

Notations