universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

"Comparing the Relative Complexity of Equivalence Relations via Strong Reduction Functions"

verfasst von / submitted by

Michael Mortinger, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 821
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Mathematik
Betreut von / Supervisor:	o. Univ.-Prof. i.R. Dr. Sy-David Friedman

Acknowledgement

Writing a master's thesis in times of a global pandemic is a solitary affair. All the more important is the support you get from all the people in your life.

First of all, I would like to express my gratitude to my supervisor Sy-David Friedman for his patience and his help. He gave me the opportunity to write my master's thesis in one of my favourite ares of mathematical logic; complexity theory.

I would also like to thank Vera Fischer for recommending Sy-David Friedman as my supervisor and putting me in touch with him.

Furthermore, I want to thank my parents, especially my mother, for supporting me, not only during the writing process of this thesis, but also during the many years I studied Mathematics at the University of Vienna.

Last but not least, I also want to thank my friends, who always contributed a big deal to keeping my life in balance.

Abstract

The goal of this thesis is to compare the relative complexity of different equivalence relations. This is done by so-called strong reduction functions in which, instead of reducing a pair of elements from one equivalence relation to another, the reduction is performed on each element individually. This notion of reduction is more appropriate than polynomial time reduction when, for example, comparing the computational complexity of the isomorphism problem for different classes of structures.

The first chapter provides a rough introduction to some basics of model theory and complexity theory. Furthermore it offers some examples of classes of structures which will be used to establish our framework throughout the thesis.

In the second chapter we introduce the notions of strong isomorphism reducibility and potential reducibility and use the concepts of invariantization and canonization to gain some insight into the structure of strong isomorphism degrees. We will see that there is a rich structure, in fact we will show that the structure of strong isomorphism degrees below the relation LOP contains a copy of the countable atomless boolean algebra. Further, we investigate whether the notion of strong isomorphism redction $\leq_{\text{iso}}$ and the notion of potential reducibility $\leq_{\text{pot}}$ are distinct and can only answer this by making some complexity theoretical assumption. At the end of this section we look at strong reduction functions that allows us to compare arbitrary equivalence relations. We use this concept to show that one can separate the complexity classes P and #P assuming that $\leq_{\text{iso}}$ and $\leq_{\text{pot}}$ are distinct.

Chapter 3 focuses on four elementary problems: The recognition problem, the invariant problem, the canonization problem and the first member problem. We show that those problems are of strictly increasing complexity and that it is not possible in general to reduce one of these problems to an earlier one even with the help of an oracle.

Finally, in chapter 4 the benchmark relations id, E_λ and E_σ are introduced. We show that the reducibility hierarchy, even when considering only equivalence relations apart from isomorphism relations, has a rich structure and compare it to our established hierarchy of strong isomorphism degrees. In the last section of this chapter we look at finitary equivalence relations, which are equivalence relations with only finitely many non trivial equivalence classes, i.e. only finitely many equivalence classes consist of more than one element.

Kurzfassung

Das Ziel dieser Arbeit ist es, die relative Komplexität verschiedener Äquivalenzrelationen zu vergleichen. Dies geschieht durch so genannte starke Reduktionsfunktionen, bei denen nicht ein Tupel von einer Äquivalenzrelation auf eine andere reduziert wird, sondern die Reduktion für jedes Element einzeln angewandt wird. Diese Art von Reduktion ist besser geeignet um die Komplexität verschiedener Isomorphieprobleme zu vergleichen als die gewöhnliche Polynomzeit-Reduktion.

Das erste Kapitel bietet eine kurze Einführung in einige Grundlagen der Modelltheorie und der Komplexitätstheorie. Darüber hinaus bietet es einige Beispiele für Klassen von Strukturen, die im Laufe der Arbeit zur Entwicklung unserer Theorie verwendet werden.

Im zweiten Kapitel führen wir die Begriffe der starken Isomorphie-Reduzierbarkeit und der potenziellen Reduzierbarkeit ein und verwenden die Konzepte der Invariantisierung und Kanonisierung, um einen Einblick in die Struktur der starken Isomorphie-Grade zu gewinnen. Wir werden sehen, dass es eine reichhaltige Struktur gibt. Tatsächlich werden wir zeigen, dass die Struktur der starken Isomorphiegrade unterhalb der Relation LOP eine Kopie der abzählbaren atomfreien booleschen Algebra enthält. Ferner untersuchen wir, ob der Begriff der starken Isomorphie-Reduzierbarkeit $\leq_{\text{iso}}$ und der Begriff der potentiellen Reduzierbarkeit $\leq_{\text{pot}}$ unterschiedlich sind, und können dies nur beantworten, indem wir einige komplexitätstheoretische Annahmen treffen. Am Ende dieses Abschnitts betrachten wir starke Reduktionsfunktionen, die es uns ermöglichen, beliebige Äquivalenzrelationen zu vergleichen. Wir verwenden dieses Konzept, um zu zeigen, dass man die Komplexitätsklassen P und #P seperieren kann, wenn man annimmt dass $\leq_{\text{iso}}$ und $\leq_{\text{pot}}$ verschieden sind.

Kapitel 3 konzentriert sich auf vier elementare Probleme: Das Erkennungsproblem, das Invariantenproblem, das Kanonisierungsproblem und das Problem des ersten Mitglieds. Wir zeigen, dass diese Probleme von streng zunehmender Komplexität sind und dass es im Allgemeinen nicht möglich ist eines dieser Probleme auf ein früheres zu reduzieren, selbst mit Hilfe eines Orakels.

Schließlich, in Kapitel 4 werden die Benchmark-Relationen id, E_λ und E_σ eingeführt. Wir zeigen, dass die Reduzierbarkeitshierarchie auch ohne Isomorphierelationen eine reichhaltige Struktur hat, und vergleichen sie mit der Hierarchie der starken Isomorphiegrade. Im letzten Abschnitt dieses Kapitels betrachten wir pseudo-endliche Äquivalenzrelationen, d.h. Äquivalenzrelationen mit nur endlich vielen nicht trivialen Äquivalenzklassen.

Contents

1 Introduction

In order for the thesis to be self-contained, several basic concepts and results will be introduced to facilitate a better understanding of the thesis. At first, we give some basic definitions.

1.1 Notation and convention

A *natural number* is a number in the set $\mathbb{N} = \{0, 1, 2, \ldots\}$. We use $\mathbb{N}^*$ to denote the set $\mathbb{N} \setminus \{0\}$. If $n \geq 1$, then we denote by $[n]$ the set $\{1, \ldots, n\}$ and understand that $[0] = \emptyset$. For a positive real number x, we denote by $\lfloor x \rfloor$ the largest $n \in \mathbb{N}$ such that $n \leq x$. We denote by $\log x$ the logarithm of x to the base 2. By $\mathbb{N}[x]$ we denote the set $\{p(x) \mid p(x) = \Sigma_{k=0}^{n} a_k x^k$ for some $n \geq 0$ and $a_k \in \mathbb{N}$ for $0 \leq k \leq n\}$ and call it's elements *polynomials*. The *degree* of a polynomial, denoted by $deg(p)$, is the largest n such that $a_n \neq 0$. Finally, we say that a condition holds for a *large enough* n if it holds for every $n \geq N$ for some $N \in \mathbb{N}$.

Alphabet and strings
Throughout the thesis Σ denotes the set $\{0, 1\}$ and is called the (binary) *alphabet* and its elements are called *bits*. The set of strings of length exactly n over Σ is denoted by Σ^n. $\Sigma^{\leq n} := \bigcup_{k=0}^{n} \Sigma^k$, and Σ^* is used to denote the set of all finite strings, i.e. $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. Note that there is a unique string ε of length 0 called "empty string". The length of a string is denoted by absolute value, that is $|x| = n$ if and only if $x \in \Sigma^n$. For $x, y \in \Sigma^*$, we denote by $x^\frown y$ the string of length $|x| + |y|$, where the last bit of x is followed by the first bit of y. If it is clear from the context we write xy instead of $x^\frown y$. By 1^n (0^n) we denote the string $11 \ldots 1$ $(00 \ldots 0)$ of length n. For $x \in \Sigma^*$ and $k \leq |x|$, we denote by $x \upharpoonright k$ the string that consists of the first k bits of x and call a string y an *initial segment* of x if $y = x \upharpoonright k$ for some $k < |x|$. A subset $L \subseteq \Sigma^*$ is called a *language*. The complement of L is denoted $\overline{L} = \Sigma^* \setminus L$.

Tuples
Let $x, y \in \Sigma^*$, where $x = x_1 \ldots x_k$, $y = y_1 \ldots y_l$ with $x_1, \ldots, x_k, y_1, \ldots, y_l \in \Sigma$. We code an ordered pair of string by an easily computable, invertible bijective pairing function $\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ that maps $(x, y) \mapsto x_1 x_1 \ldots x_k x_k 01 y_1 y_1 \ldots y_l y_l$. For $n-$tuples $(x^{(1)}, \ldots, x^{(n)})$ the iteration is performed as follows: $\langle (x^{(1)}, \ldots, x^{(n)}) \rangle = \langle x^{(1)}, \langle x^{(2)}, \ldots, x^{(n)} \rangle \rangle$.

Equivalence relation
A set $E \subseteq \Sigma^* \times \Sigma^*$ is an *equivalence relation* on Σ^* if the following three properties hold:

- (reflexivity) For all $x \in \Sigma^*$, $(x, x) \in E$.
- (symmetry) For all $x, y \in \Sigma^*$, $(x, y) \in E$ implies $(y, x) \in E$.
- (transitivity) For all $x, y, z \in \Sigma^*$, if $(x, y) \in E$ and $(y, z) \in E$, then $(x, z) \in E$.

An equivalence relation E can be encoded as a language by taking tuples of each pair in E. In this way we can study the computational complexity of classes of languages which represent equivalence relations. We will write $(x, y) \in E$ or simply xEy instead of $\langle x, y \rangle \in L_E$, where L_E is the language on Σ induced by E.
The *equivalence classes* of x with respect to E is the set $\{y \in \Sigma^* \mid (xEy\}$. It is denoted $[x]_E$, of it the context is clear, simply $[x]$.

Lexicographic order $<_{\text{lex}}$
For two strings $x, y \in \Sigma^*$ we denote by $<_{\text{lex}}$ the length-lexicographical order on Σ^*, i.e. $x <_{\text{lex}} y$ if either $|x| < |y|$ or $|x| = |y|$ and if the i-th bit is the leftmost bit where x is different from y then $x_i < y_i$.

1.2 Background from Complexity Theory

We assume some basic knowledge of complexity theory but give the following definitions as a reminder. For more details we refer the reader to [8], [5], as well as [10].

Definition 1.2.1 (Big O Notation)
For two functions $f, g : \mathbb{N} \to \mathbb{N}$, we say that

(i) $f = O(g)$ if there is a constant c such that $f(n) \leq c \cdot g(n)$ for large enough n,

(ii) $f = \Omega(g)$ if $g = O(f)$, and

(iii) $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

Definition 1.2.2 (Decision problem)
A *decision problem* Q for a language L is: given $x \in \Sigma^*$, decide whether or not $x \in L$

We will use the terms 'language' and 'problem' interchangeably, since it is common practice in complexity theory.

Complexity Classes

Our standard model of computation is that of deterministic and nondeterministic Turing machines, usually denoted $\mathbb{M}$. We make no further mentions of the model except where it is relevant.

Definition 1.2.3 (Time bounded Turing machines)
We call a Turing machine $\mathbb{M}$, with $t : \mathbb{N} \to \mathbb{N}$ *t-time bounded* if for every $x \in \Sigma^*$ there exists a complete run of $\mathbb{M}$ on input x that has a length at most $t(|x|)$. By DTIME(t) (NTIME(t)) we denote the class of problems Q such that there is some $c \in \mathbb{N}$ and some $(c \cdot t + c)$-time bounded $\mathbb{M}$ that decides Q.

Definition 1.2.4 (P, NP and co-NP)
The classes

$$\mathrm{P} := \bigcup_{k \in N} \mathrm{DTIME}(n^k) \qquad \text{and} \qquad \mathrm{NP} := \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(n^k)$$

are called deterministic (and nondeterministic) polynomial time. The class co-NP is defined as the class of languages whose complements are in NP, i.e.

$$\text{co-NP} \; = \; \{\overline{L} \mid L \in \mathrm{NP}\}.$$

Remark 1.2.5 (P versus NP)
The biggest open problem in complexity theory is whether P = NP or not. It is strongly believed that this is not the case. For the rest of this thesis we will always assume that P $\neq$ NP unless stated otherwise.

Definition 1.2.6 (Polynomial time reduction)
Let Q, Q' be problems. A function $f : \Sigma^* \to \Sigma^*$ is a *polynomial time reduction* from Q to Q' if f is computable in polynomial time and for all $x \in \Sigma^*$

$$x \in Q \iff f(x) \in Q'.$$

We say Q is *polynomial time reducible* to Q' and write $Q \leq_p Q'$ if such a f exists. If both $Q \leq_p Q'$ and $Q' \leq_p Q$ we write $Q \equiv_p Q'$ and say Q and Q' are *polynomial time equivalent*.

If is easy to see that $\leq_p$ is transitive and that $\equiv_p$ is an equivalence relation.

Definition 1.2.7 (Hard and complete problems)
Let CC be a complexity class and let Q, Q' be problems.

(i) We say a problem Q' is CC-*hard* if $Q \leq_p Q'$ for every $Q \in$ CC.

(ii) If additionally $Q' \in$ CC, then Q' is called CC-*complete*.

We give a quick reminder of the polynomial hierarchy, for more details see e.g. [13].

Definition 1.2.8 (Polynomial hierarchy)
For every $k \geq 1$, a language L is in Σ_k^{P} if there is a (deterministic) Turing machine $\mathbb{M}$ running in time polynomial in the first input, such that

$$x \in L \iff \exists y_1 \, \forall y_2 \, \ldots \, Q_k y_k \, \mathbb{M}(x, y_1, \ldots, y_k) = 1$$

where $Q_k = \forall$ of k is even, and $Q_k = \exists$ if k is odd.

We say L is in Π_k^{P} if there is a (deterministic) Turing machine $\mathbb{M}$ running in time polynomial in the first input, such that

$$x \in L \iff \forall y_1 \exists y_2 \ldots Q_k y_k \, \mathbb{M}(x, y_1, \ldots, y_k) = 1$$

where $Q_k = \exists$ of k is even, and $Q_k = \forall$ if k is odd.

Finally for the polynomial hierarchy we write $\mathrm{PH} := \bigcup_{k \in \mathbb{N}} \Sigma_k^{\mathrm{P}}$

It is well known that:

Theorem 1.2.9 If P = NP, then the polynomial hierarchy collapses, i.e. P = PH.

1.3 Background from Model Theory

A *vocabulary* τ is a finite set of constant symbols, function symbols and relation symbols. A $\tau-$*structure* is a pair $\mathcal{A} = (A, (s^{\mathcal{A}})_{s \in \tau})$ where A is a non-empty set called the *universe* of $\mathcal{A}$, $s^{\mathcal{A}} \in A$ if s is a constant, $s^{\mathcal{A}} : A^n \to A$ is s is an $n-$ary function symbol, and $s^{\mathcal{A}} \subseteq A^m$ is s is an $m-$ary relation symbol. For $s \in \tau$ we call $s^{\mathcal{A}}$ the $\mathcal{A}-$*interpretation* of s.
All structures in this thesis are assumed to be finite and have $[n]$ as their universe for some $n \in \mathbb{N}$.
Therefore, we can identify a structure $\mathcal{A}$ with its encoding, that is a nonempty string $\ulcorner \mathcal{A} \urcorner \in \Sigma^*$. We assume that the mappings $\mathcal{A} \mapsto \ulcorner \mathcal{A} \urcorner$ and $\ulcorner \mathcal{A} \urcorner \mapsto \mathcal{A}$ are computable in polynomial time and that for every vocabulary τ there is a polynomial $q_\tau \in \mathbb{N}[x]$ such that $|A| \leq |\ulcorner \mathcal{A} \urcorner| \leq q_\tau(|A|)$ for every $\tau-$structure $\mathcal{A}$.

A class C of $\tau-$structures is *closed under isomorphism* is for all structures $\mathcal{A}$ and $\mathcal{B}$

$$\mathcal{A} \in \mathrm{C} \text{ and } \mathcal{A} \cong \mathcal{B} \text{ imply } \mathcal{B} \in \mathrm{C}.$$

From now on we will only consider classes C (and D) which are in P, are closed under isomorphism and contain arbitrary large finite structures. Moreover, in a fixed class all structures will have the same vocabulary.

The following examples will be helpful in the next chapter.

Example 1.3.1

(i) The classes SET (sets), BOOLE (boolean algebras), FIELD (fields), GROUP (groups), ABELIAN (abelian groups) and CYCLIC (cyclic groups) with vocabularies $\tau_{\mathrm{SET}} = \emptyset$, $\tau_{\mathrm{BOOLE}} = \{\cap, \cup\}$, $\tau_{\mathrm{FIELD}} = \{+, \times\}$, $\tau_{\mathrm{GROUP}} = \tau_{\mathrm{ABELIAN}} = \tau_{\mathrm{CYCLIC}} = \{\circ\}$ respectively. Here $\cap, \cup, +, \times, \circ$ are the usual binary function symbols of the corresponding structures.

(ii) The class GRAPH of (simple and undirected) graphs, $\tau_{\mathrm{GRAPH}} = \{E\}$ where E is a binary relation symbol.

(iii) The class ORD of linear orders ($\tau_{\mathrm{ORD}} = \{<\}$) with a binary relation symbol $<$.

(iv) The class LOP of *linear orders with a distinguished point* and the class LOU of *linear orders with a unary relation*. Let $\tau_{\mathrm{LOP}} := \tau_{\mathrm{ORD}} \cup \{c\}$ with a constant symbol c and $\tau_{\mathrm{LOU}} := \tau_{\mathrm{ORD}} \cup \{U\}$ with a unary relation symbol U.

Remark 1.3.2
There is a one-to-one correspondence between strings in Σ^* and structures in LOU. A string $x_1 \dots x_n = x \in \Sigma^n$ corresponds to a structure $\mathcal{A} \in \mathrm{LOU}$ with universe $A = [n]$ and where $<^{\mathcal{A}}$ is interpreted as the natural ordering on $[n]$ and $U^{\mathcal{A}} := \{i \in [n] \mid x_i = 1\}$.

For a vocabulary τ let $\mathrm{STR}[\tau]$ denote the class of all τ-structures and $\mathcal{O}[\tau]$ denote the class of all *ordered* τ-structures. A structure $\mathcal{A}$ is said to be ordered if the reduct $\mathcal{A} \restriction \tau_{\mathrm{ORD}} \in \mathrm{ORD}$.

2 The isomorphism relation

In mathematics it is a natural question to ask whether two structures (e.g. graphs or groups) are isomorphic. The isomorphism problem (for two structures of the same class) is defined as follows:

Iso(C)
Instance: $\mathcal{A}, \mathcal{B} \in$ C
Problem: Is $\mathcal{A} \cong \mathcal{B}$?

One main technique for determining the complexity of a problem is showing how the difficulty of the problem relates to the difficulty of other known problems. The relative difficulty of computational problems are often compared using polynomial time (many-one) reductions, a function by which an instance of one problem gets transformed (encoded) as an instance of the other problem.
The set

$$\text{Iso(C)} := \{(\mathcal{A}, \mathcal{B}) \mid \mathcal{A}, \mathcal{B} \in \text{C} \,\text{and}\, \mathcal{A} \cong \mathcal{B}\}$$

is the set of positive instances of the isomorphism problem of a class C, i.e. the set of pairs of isomorphic structures in C.
A polynomial time reduction from Iso(C) to Iso(D) is a function $f : C \times C \to D \times D$ computable in polynomial time such that for all $\mathcal{A}, \mathcal{B} \in$ C

$$(\mathcal{A}, \mathcal{B}) \in \text{Iso(C)} \iff f(\mathcal{A}, \mathcal{B}) \in \text{Iso(D)}.$$

The function computing this reduction has access to both structures (in an instance) of the problem. However it would be more natural to reduce each structure individually when comparing the complexity of the isomorphism problem of different classes. This motivates the notion of strong isomorphism reduction. Here, the contents are mainly taken from [3].

2.1 Strong Isomorphism Reduction $\leq_{\text{iso}}$

Definition 2.1.1 (Strong isomorppphism reduction)
Let C and D be classes. We say that C is *strongly isomorphism reducible* to D if there is a function $f :$ C $\to$ D computible in polynomial time such that for all $\mathcal{A}, \mathcal{B} \in$ C

$$\mathcal{A} \cong \mathcal{B} \iff f(\mathcal{A}) \cong f(\mathcal{B}).$$

We then write $f :$ C $\leq_{\text{iso}}$ D. If C and D have the same *strong isomoprhism degree*, i.e. C $\leq_{\text{iso}}$ D and D $\leq_{\text{iso}}$ C, we write C $\equiv_{\text{iso}}$ D.

It is easy to see that $\leq_{\text{iso}}$ is a partial order (i.e., reflexive, transitive and antisymmetric) and $\equiv_{\text{iso}}$ is an equivalence relation.

Example 2.1.2

(i) Since the multiplicative group of a finite field is cyclic, we get FIELD $\leq_{\text{iso}}$ CYCLIC by sending a field to its multiplicative group.

(ii) If C $\subseteq$ D then $\mathtt{id}_{\text{C}}$: C $\leq_{\text{iso}}$ D, e.g. CYCLIC $\leq_{\text{iso}}$ ABELIAN $\leq_{\text{iso}}$ GROUP.

(iii) ORD $\equiv_{\text{iso}}$ SET $\equiv_{\text{iso}}$ CYCLIC

We can reduce the notation of strong isomorphism reduction to the notion of general polynomial time reduction.

Remark 2.1.3 A function $f : C \to D$ induces a function $\hat{f} : C \times C \to D \times D$ with

$$\hat{f}(\mathcal{A}, \mathcal{B}) := (f(\mathcal{A}), f(\mathcal{B})) . \tag{2.1}$$

Then

$$f : \text{C} \leq_{\text{iso}} \text{D} \iff \hat{f} : \textsc{Iso}(\text{C}) \leq_{\text{p}} \textsc{Iso}(\text{D}).$$

It is possible to construct polynomial time reductions from Iso(C) to Iso(D) that are not of the form 2.1. We will later present classes C, D such that

$$\textsc{Iso}(\text{C}) \leq_{\text{p}} \textsc{Iso}(\text{D}) \quad \text{but} \quad \text{C} \not\leq_{\text{iso}} \text{D}.$$

Our goal is to get a better understanding of the relation $\leq_{\text{iso}}$. First we see, that there is a maximum element.

Proposition 2.1.4 (Maximum element of $\leq_{\text{iso}}$)
GRAPH is a maximum element of $\leq_{\text{iso}}$, i.e. C $\leq_{\text{iso}}$ GRAPH for all classes C.

Proof. The encoding of an arbitrary τ-structure $\mathcal{A} \in \text{STR}[\tau]$ as an undirected simple graph yields a strong isomorphism reduction from STR[τ] to GRAPH. In particular the restriction to a subclass C $\subseteq$ STR[τ] shows that C $\leq_{\text{iso}}$ GRAPH. □

The class GRAPH is not the only maximal element as we will see in the next lemma.

Lemma 2.1.5 Let BIP be the class of bipartite graphs, then GRAPH $\equiv_{\text{iso}}$ BIP.

Proof. We only show that there exists a strong isomorphism reduction f : GRAPH $\leq_{\text{iso}}$ BIP, $f : G \mapsto G^*$. The other direction follows directly from Proposition 2.1.4. Given a finite graph $G = (V, E)$ we construct $G^* = (V^*, E^*)$ as follows: Let $V^* = V \cup E$ and $E^* = \{(v, e) \mid v \in V, e \in E, \text{and } v \text{ is incident to } e \text{ in } G\}$. The constructed G^* is clearly bipartite because $\{V, E\}$ is a partition of its vertices such that every edge in G^* is between a vertex in V and a vertex in E. Now suppose that $G_1, G_2 \in$ GRAPH. It is easy to see that, if $G_1 \cong G_2$, then $G_1^* \cong G_2^*$. In fact any isomorphism $\varphi : G_1 \mapsto G_2$ induces an isomorphism $\psi : G_1^* \mapsto G_2^*$ by the definition of G^*. For the converse, first note that G is

connected if and only if G^* is connected. Without loss of generality we assume that G_1 and G_2 are connected. Otherwise, we look at each connected component individually. Suppose that ψ is the isomorphism that maps G_1^* to G_2^* Then ψ maps V_1 either to V_2 or E_2, since both graphs are bipartite. In the case where $\psi(V_1) = V_2$ and $\psi(E_1) = E_2$, ψ induces an isomorphism between G_1 and G_2. In the other case, i.e. where $\psi(V_1) = E_2$ and $\psi(E_1) = V_2$ every vertex in E_1 in G_1^* has degree 2, thus it must happen that each vertex in V_2 in G_2^* has also degree 2, and therefore each vertex in V_1 in G_1^* has degree 2. This means that all vertices of the two graphs have degree 2 and G_1^* must be a simple cycle (with an even number of vertices). The same holds for G_2^*. This simply implies that both G_1 and G_2 are simply cycles, and both have the same length, thus $G_1 \cong G_2$. This concludes the proof. □

To further investigate the structure of strong isomorphism degrees, we will describe the isomorphism types of a structure with the help of an invariant and the related notion of canonization.

Definition 2.1.6 (Invariantization)
A function $\text{Inv} : \text{C} \to \Sigma^*$ is an *invariantization* for a class C if it is computable in polynomial time and for all $\mathcal{A}, \mathcal{B} \in \text{C}$

$$\mathcal{A} \cong \mathcal{B} \iff \text{Inv}(\mathcal{A}) = \text{Inv}(\mathcal{B}).$$

Lemma 2.1.7 Let C, D be a classes, $\text{C} \leq_{\text{iso}} \text{D}$ and D has an invariantization, then also C has an invariantization.

Proof. Let Inv_D be an invariantization for D and $f : \text{C} \leq_{\text{iso}} \text{D}$. We get an invariantization of C by setting $\text{Inv}_\text{C} := \text{Inv}_\text{D} \circ f$. Let $\mathcal{A}, \mathcal{B} \in \text{C}$ then

$$\mathcal{A} \cong \mathcal{B} \iff f(\mathcal{A}) \cong f(\mathcal{B}) \iff \text{Inv}\left(f(\mathcal{A})\right) = \text{Inv}\left(f(\mathcal{B})\right).$$

□

Of all classes with an invariantization LOU is a maximum element. More precise, we have:

Proposition 2.1.8 For a class C the following are equivalent:

(i) C has an invariantization.

(ii) $\text{C} \leq_{\text{iso}} \text{LOU}$.

(iii) There is a class D of ordered structures such that $\text{C} \leq_{\text{iso}} \text{D}$

Proof.
(i) ⇒ (ii) Follows immediately from Remark 1.3.2 .
(ii) ⇒ (iii) is trivial.
(iii) ⇒ (i) The only automorphism of an ordered structures is the identity, thus for every structure $\mathcal{A} \in \text{D}$ there is a unique structure $\mathcal{A}'$ such that $\mathcal{A} \cong \mathcal{A}'$ and $<^{\mathcal{A}'}$ is interpreted as the natural linear ordering of the universe of A'. Thus the mapping $\mathcal{A} \mapsto \ulcorner \mathcal{A}' \urcorner$ is an invariantization of D. Applying Lemma 2.1.7 yields an invariantization of C. □

The concept of an invariantization is deeply linked with one of the biggest open question in descriptive complexity. Even though it is well known that on ordered structures least-fix-point logic captures polynomial time (Immerman-Vardi theorem), it is still an open question whether there is a logic capturing P on all finite structures. If the class GRAPH has an invariantization, then there is a logic capturing P on all finite structures.
Another useful tool is the so called canonization:

Definition 2.1.9 (Canonization)
A *canonization* for a class C is a polynomial time computable function $\mathrm{Can} : \mathrm{C} \to \mathrm{C}$ such that the following conditions hold:

(i) For all $\mathcal{A}, \mathcal{B} \in \mathrm{C}$
$$\mathcal{A} \cong \mathcal{B} \iff \mathrm{Can}(\mathcal{A}) = \mathrm{Can}(\mathcal{B}).$$

(ii) For all $\mathcal{A} \in \mathrm{C}$
$$\mathcal{A} \cong \mathrm{Can}(\mathcal{A}).$$

It is easy to see that every canonization yields an invariantization.

Lemma 2.1.10 Let C be a class and $\mathrm{Can} : \mathrm{C} \to \mathrm{C}$ a canonization for C. Then the mapping $\mathcal{A} \mapsto \ulcorner \mathrm{Can}(A) \urcorner$ is an invariantization for C.

Proof. Plugging in the definitions, yield

$$\mathcal{A} \cong \mathcal{B} \iff \mathrm{Can}(\mathcal{A}) = \mathrm{Can}(\mathcal{B}) \iff \underbrace{\ulcorner \mathrm{Can}(\mathcal{A}) \urcorner}_{\mathrm{Inv}(\mathcal{A})} = \underbrace{\ulcorner \mathrm{Can}(\mathcal{B}) \urcorner}_{\mathrm{Inv}(\mathcal{B})}.$$

□

The natural question arises wether the converse is also true, i.e. given an invariantization, can we construct a canonization. In many ares of mathematics we can indeed do this. For example if we consider the class FIELD. For every $\mathcal{K} \in$ FIELD the pair $(p_\mathcal{K}, n_\mathcal{K})$, where $p_\mathcal{K}$ is the characteristic of $\mathcal{K}$ and $n_\mathcal{K}$ is the dimension over the prime field, is an invariantization for $\mathcal{K}$. Since for a pair (p, n) we can construct the canonical galois field[1] $\mathcal{F}_{p^n}$, the mapping $\mathcal{K} \mapsto \mathcal{F}_{p_\mathcal{K}^{n_\mathcal{K}}}$ is an canonization for the class FIELD.
This canonization has a further property, namely it is a canonization with polynomial time enumeration.

Definition 2.1.11 (Canonization with polytime enumeration)
Let Can be the canonization of a class C. The *enumeration induced by* Can is a sequence of structures
$$\mathcal{A}_1, \mathcal{A}_2, \ldots$$
where each $\mathcal{A}_k \in \mathrm{Im}\,(\mathrm{Can})$ and $i < j$ iff $\ulcorner \mathcal{A}_i \urcorner <_{\mathrm{lex}} \ulcorner \mathcal{A}_j \urcorner$. We say that C has a *canonization with polytime enumeration* if the mappings $\mathcal{A}_n \mapsto 1^n$ and $1^n \mapsto \mathcal{A}_n$ are computable in polynomial time.

[1] This is possible since all fields of order p^k (where p is a prime and k is a positive integer) are isomorphic.

Remark 2.1.12 It is worth pointing out that the following are equivalent:

(i) The mapping $\mathcal{A}_n \mapsto 1^n$ is computable in polynomial time.

(ii) We get an invariantization Inv of C be setting

$$\mathrm{Inv}(\mathcal{A}) := 1^n \quad \Longleftrightarrow \quad \mathrm{Can}(\mathcal{A}) = \mathcal{A}_n.$$

Recall that 1^n is the unary encoding of the natural number n. So by $\mathcal{A}_n \mapsto 1^n$ and $1^n \mapsto \mathcal{A}_n$ we mean that given a structure (i.e. an element in the sequence) we can compute it's index and vice versa.
Many classes of structures that we presented so far have canonizations with polynomial time enumerations, e.g. SET, ORD, LOP, ABELIAN, CYCLIC and FIELD. We also saw classes of structures that do not have canonizations with polynomial time enumeration, for example

(i) the class BOOLE since the mapping $1^n \mapsto \mathcal{A}_n$ will not be computable in polynomial time because the number of non-isomorphic structures is "too low". (see Example 2.2.2)

(ii) the class LOU has the opposite problem since there are up to isomorphism "too many" structures.

Theorem 2.1.13 Let C, D be classes that have canonizations with polynomial time enumerations. Then $\mathrm{C} \equiv_{\mathrm{iso}} \mathrm{D}$.

Proof. Let $\mathrm{Can}_{\mathrm{C}}, \mathrm{Can}_{\mathrm{D}}$ be canonizations with polynomial time enumerations of the classes C, D respectively. We get a strong isomorphism reduction $f : \mathrm{C} \to \mathrm{D}$ by setting

$$f(\mathcal{A}) := \mathcal{B} \iff \mathrm{Can}_{\mathrm{C}}(\mathcal{A}) = \mathcal{A}_n,$$

where $\mathcal{A}_n$ is the n-th element in the enumeration. By symmetry we also get $\mathrm{D} \leq_{\mathrm{iso}} \mathrm{C}$ and therefore $\mathrm{C} \equiv_{\mathrm{iso}} \mathrm{D}$. □

Remark 2.1.14 A close look at the previous proof shows that we already obtain $\mathrm{C} \leq_{\mathrm{iso}} \mathrm{D}$ if the mappings $\mathcal{A}_n \mapsto 1^n$ and $1^n \mapsto \mathcal{B}_n$ are computable in polynomial time. By this observation we get, for example, $\mathrm{BOOLE} \leq_{\mathrm{iso}} \mathrm{CYCLIC}$.

Corollary 2.1.15 The classes SET, FIELD, ABELIAN, CYCLIC, ORD and LOP all have the same strong isomorphism degree.

2.2 Potential Reduction $\leq_{\mathrm{pot}}$

In this section we introduce a notion of reduction that purely relies on the number of isomorphism types.

Definition 2.2.1 (Number of equivalence classes)
For a class C, let $\mathrm{C}(n)$ be the subclass which contains all structures of cardinality at most n and $\#\mathrm{C}(n)$ denotes the number of isomoprhism types of $\mathrm{C}(n)$, i.e.

$$\mathrm{C}(n) := \{\mathcal{A} \in \mathrm{C} \mid |A| \leq n\} \quad \text{and} \quad \#\mathrm{C}(n) := |\mathrm{C}(n)/_{\cong}|.$$

Example 2.2.2

(i) $\#\mathrm{BOOLE}(n) = \lfloor \log n \rfloor$

(ii) $\#\mathrm{CYCLIC}(n) = n$.

(iii) $\#\mathrm{SET}(n) = \#\mathrm{ORD}(n) = n + 1$.

(iv) $\#\mathrm{LOP}(n) = \sum_{i=1}^{n} i = \frac{(n+1)\cdot n}{2}$ and $\#\mathrm{LOU}(n) = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

(v) For every vocabulary τ there is a polynomial q_τ such that for every $C \subseteq STR[\tau]$ we have $\#\mathrm{C}(n) \leq 2^{q_\tau(n)}$.

(vi) $\#\mathrm{GROUP}(n) \leq n^{O(log^2 n)}$ (This has been proven in [12])

Definition 2.2.3 (Potential reduction)
Let C, D be classes. We say that C is *potential reducible* to D (denoted by $\mathrm{C} \leq_{\mathrm{pot}} \mathrm{D}$), if there is a polynomial $p \in \mathbb{N}[x]$ such that $\#\mathrm{C}(n) \leq \#\mathrm{D}(p(n))$, for all $n \in \mathbb{N}$. If $\mathrm{C} \leq_{\mathrm{pot}} \mathrm{D}$ and $\mathrm{D} \leq_{\mathrm{pot}} \mathrm{C}$, denoted by $\mathrm{C} \equiv_{\mathrm{pot}} \mathrm{D}$, then C and D have the same *potential reducibility degree*.

The importance of the reduction defined above is due to the next lemma.

Lemma 2.2.4 Let C, D be classes. If $\mathrm{C} \leq_{\mathrm{iso}} \mathrm{D}$, then $\mathrm{C} \leq_{\mathrm{pot}} \mathrm{D}$.

Proof. Let $f : \mathrm{C} \leq_{\mathrm{iso}} \mathrm{D}$. There there exists a polynomial $p \in \mathbb{N}[x]$ such that for all $\mathcal{A} \in \mathrm{C}$ the cardinality of the image is bounded by this polynomial, i.e. $|f(A)| \leq p(|A|)$, where $f(A)$ denotes the universe of $f(\mathcal{A})$. As f strongly preserves isomorphisms, it induces a one-to-one map from $\{\mathcal{A} \in \mathrm{C} \mid |A| \leq n\}/_{\cong}$ to $\{\mathcal{B} \in \mathrm{D} \mid |B| \leq p(n)\}/_{\cong}$. □

We will mainly use the contrapositive of the statement above to show that two classes have not the same strong-isomorphism-degree.

Corollary 2.2.5 Let C, D be classes. If $\mathrm{C} \not\leq_{\mathrm{pot}} \mathrm{D}$, then $\mathrm{C} \not\leq_{\mathrm{iso}} \mathrm{D}$.

We immediately can state some simple consequences.

Proposition 2.2.6

(i) $\mathrm{CYCLIC} \not\leq_{\mathrm{iso}} \mathrm{BOOLE}$ and $\mathrm{LOU} \not\leq_{\mathrm{iso}} \mathrm{LOP}$.

(ii) For all classes C we have $\mathrm{C} \leq_{\mathrm{pot}} \mathrm{LOU}$ and $\mathrm{LOU} \equiv_{\mathrm{pot}} \mathrm{GRAPH}$.

(iii) $\mathrm{LOP} \lneq_{\mathrm{pot}} \mathrm{GROUP} \lneq_{\mathrm{pot}} \mathrm{LOU}$.

(iv) LOP $\lneq_{\mathrm{iso}}$ GROUP $\lneq_{\mathrm{iso}}$ GRAPH.

Proof. Using the Corollary 2.2.5, we get

(i) follows from Example 2.2.2.

(ii) holds by Example 2.2.2 and Proposition 2.1.4.

(iii) is a direct consequence of Example 2.2.2.

(iv) GROUP $\leq_{\mathrm{iso}}$ GRAPH follows from Proposition 2.1.4 and

$$\mathrm{LOP} \leq_{\mathrm{iso}} \mathrm{CYCLIC} \leq_{\mathrm{iso}} \mathrm{GROUP}$$

holds by Corolarry 2.1.15 and Example 2.1.2. The fact that

$$\mathrm{GRAPH}, \not\leq_{\mathrm{iso}} \mathrm{GROUP} \quad \text{and} \quad \mathrm{GROUP} \not\leq_{\mathrm{iso}} \mathrm{LOP}$$

follows from (3) and the fact that LOP $\equiv_{\mathrm{pot}}$ GRAPH.

□

In Figure 2.1 we use the convention that
C $\dashrightarrow$ D if C $\leq_{\mathrm{iso}}$ D but it is not known whether D $\leq_{\mathrm{iso}}$ C or not,
C $\mapsto$ D if C $\leq_{\mathrm{iso}}$ D but D $\not\leq_{\mathrm{iso}}$ C, and
C $\leftrightarrow$ D if C $\equiv_{\mathrm{iso}}$ D.

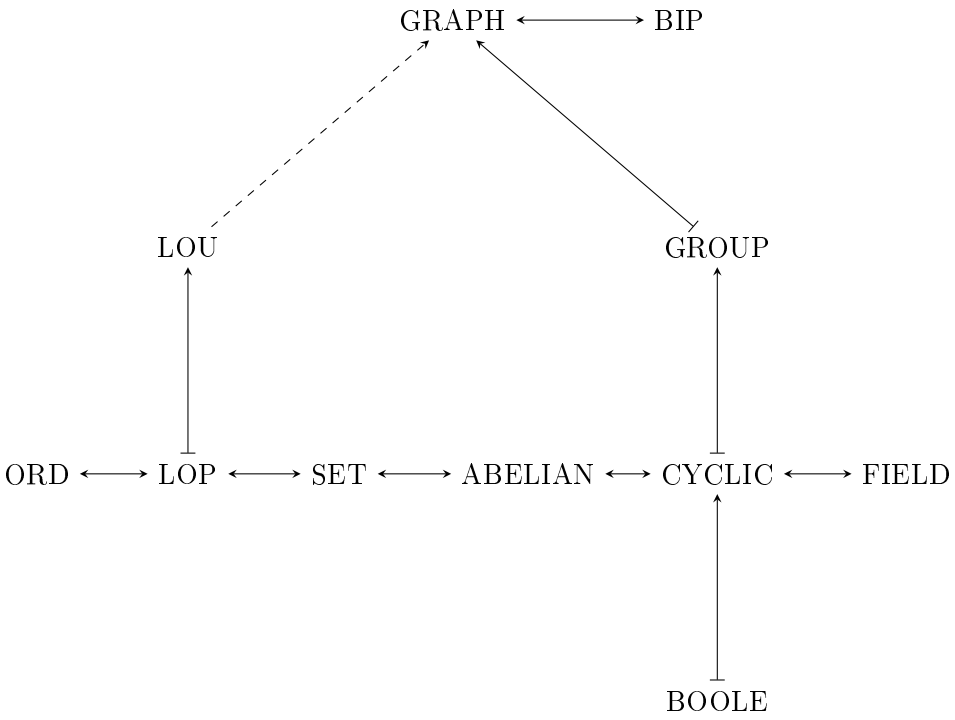

Figure 2.1: Structure of Strong Isomorphism Degrees

We will later see that LOU $\not\leq_{\mathrm{iso}}$ GROUP.

2.3 Structure of $\leq_{\mathsf{iso}}$ below LOP

Before we can show the main result of this section, Theorem 2.3.18, we need to do some preliminary work.

2.3.1 Boolean Algebras

In this section we introduce the concept of Boolean algebras and show that, up to isomorphism, there is only one countable atomless Boolean algebra. Here, the contents will be taken from [7].

Definition 2.3.1 (Lattice)
A *lattice* $(L, \wedge, \vee)$ is a nonempty set L with two binary operations $\wedge$ and $\vee$ that are commutative, associative and satisfy the absorption laws, i.e.

$$x \vee (x \wedge y) = x$$
$$x \wedge (x \vee y) = x$$

for all $x, y \in L$.

Alternatively a lattice can be viewed as a partially ordered set.

Definition 2.3.2 (Alternative definition of a lattice)
A partially ordered set $(L, \leq)$ is a lattice if for all $x, y \in L$

$$x \leq y \quad \Longleftrightarrow \quad x \wedge y = x, \tag{2.2}$$

where $\wedge$ is the binary operation from the definition above.

Note that (2.2), together with the absorption laws is equivalent to

$$x \leq y \quad \Longleftrightarrow \quad x \wedge y = x \quad \Longleftrightarrow \quad x \vee y = y.$$

Definition 2.3.3 (Distributive lattice)
A lattice $(L, \wedge, \vee)$ is *distributive* if the following holds:

$$(x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z)$$
$$(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z),$$

for all $x, y, z \in L$.

Definition 2.3.4 (Boolean algebras)
A *Boolean algebra* $(B, \wedge, \vee, 0, 1, \neg)$ is a distributive lattice $(B.\wedge, \vee)$ with a least element 0 and greatest element 1 and with an additional unary operator $\neg$ that satisfies

$$\neg x \wedge x = 0$$
$$\neg x \vee x = 1$$

for all $x \in B$.

Definition 2.3.5 (Atom)
Let $(P, \leq)$ be a poset with a least element 0. An *atom* is a nonzero element $a \in P$ such that there is no $x \in P$ with $0 < x < a$.

Observe that an atomless Boolean algebra cannot be finite. Since we are interested in atomless Boolean algebras the next theorem will be useful.

Theorem 2.3.6 (Countable atomless Boolean algebras are isomorphic)
The first order theory of countable atomless Boolean algebras is $\omega-$categorical, i.e. any two countable atomless Boolean algebras are isomorphic.

We will show this using the back-and-forth method, common in model theory showing isomorphisms between countably infinite structures.

Proof. Let $(A, \wedge, \vee, 0, 1, \neg)$ and $(B, \wedge, \vee, 0, 1, \neg)$ be countable atomless Boolean algebras. We will successively construct bigger finite subalgebras

$$\{0,1\} = A_0 \subset A_1 \subset A_2 \subset \ldots \qquad \text{and} \qquad \{0,1\} = B_0 \subset B_1, \subset B_2 \subset \ldots$$

such that

$$\bigcup_{i \in \mathbb{N}} A_i = A \qquad \text{and} \qquad \bigcup_{i \in \mathbb{N}} B_i = B,$$

and isomorphisms $f_0 : A_0 \to B_0, f_1 : A_1 \to B_1, f_2 : A_2 \to B_2, \ldots$ with

$$f_0 \subset f_1 \subset f_2 \subset \ldots .$$

Then $f := \bigcup_{i \in \mathbb{N}} f_i$ will be the isomoprhism that witnesses $A \cong B$. We start by enumerating the Elements of A and B as

$$A = \{a_0, a_1, a_2, \ldots\} \qquad \text{and} \qquad B = \{b_0, b_1, b_2, \ldots\}.$$

Trivially f_0 is given by $f_0(0) = 0, f_0(1) = 1$.
Now suppose that n is even and we have already constructed A_n, B_n and f_n. Since A_n is generated by its atoms $p_0, p_1, \ldots p_k$ $(k \leq n)$ we know that B_n has atoms $f_n(p_0), f_n(p_1), \ldots, f_n(p_k)$. Let x be the first element in our enumeration $a_0, a_1, a_2, \ldots$ that is not in A_n, then A_{n+1} is generated by A_n and x. Using the discjunction normal form, every element of A_{n+1} can be written as a disjunction of elements of

$$X = \{p_0 \wedge x, p_0 \wedge \neg x, \ldots, p_k \wedge x, p_k \wedge \neg x\}.$$

Hence A_{n+1} is finite and its atoms are the nonzero elements of X.
Now define $x_i := p_i \wedge x$ for $i = 0, \ldots, k$ and pick $y_i \in B$ according to the following rules:

$$\begin{aligned} y_i &= 0 && \text{if } x_i = 0 \\ y_i &= f_n(p_i) && \text{if } x_i = p_i \\ 0 < y_i &< f_n(p_i) && \text{if } 0 < x_i < p_i \end{aligned}$$

which is always possible because B is atomless.
Define $y = \bigvee_{i=0}^{k} y_i$ and let $B_{n+1} \subset B$ generated by B_n and y. Like before every element of B_{n+1} can be written as a conjunction of elements in

$$Y = \{f_n(p_0) \wedge y, f_n(p_0) \wedge \neg y, \ldots, f_n(p_k) \wedge y, f_n(p_k) \wedge \neg y\}.$$

Hence B_{n+1} is also finite and its atoms are the nonzero elements of Y. Therefore we can define f_{n+1} such that $f_{n+1}(x) = y$ and $f_{n+1} \restriction A_n = f_n$. This induces a bijection from the atoms of A_{n+1} to the atoms of B_{n+1}, and therefore f_{n+1} is well-defined.
If n is odd we switch the roles of A and B, i.e. x will then be the first element of $b_0, b_1, b_2, \ldots$ that is not in B_n and so on. By this construction we make sure that

$$\bigcup_{i \in N} A_i = A \qquad \text{and} \qquad \bigcup_{i \in \mathbb{N}} B_i = B$$

and since in every step $f_i : A_i \to B_i$ is an isormorphism it follows that $\bigcup_{i \in \mathbb{N}} f_i = f : A \to B$ is an isomorphism. □

2.3.2 Embedding the partial order of the countable atomless Boolean algebra into the structure of $\leq_{\text{iso}}$

As mentioned earlier the structure of $\leq_{\text{iso}}$ between LOU and GRAPH is linked with open questions in descriptive complexity theory. We are now in a position to show that the structure of $\leq_{\text{iso}}$, even below LOP is quite rich. In fact the partial ordering of the countable atomless Boolean algebra is embeddable into the partial ordering induced by $\leq_{\text{iso}}$ on the degrees of strong isomorphism reducibility below LOP. For this section we again will mainly follow [3].

Definition 2.3.7 (Value-polynomial)
A function $f : \mathbb{N} \to \mathbb{N}$ is called *value-polynomial* if it is strictly increasing and $f(n)$ can be computed in $\bigcup_{k \in \mathbb{N}} \text{DTIME}\big(f(n)^k\big)$. We will denote that class of all value-polynomial functions by VP .

Remark 2.3.8 Note that $f(n)$ does not need to be computable in time polynomial in n.

For $f \in$ VP, we define a subclass of LOU as follows:

$$\text{C}_f := \{\mathcal{A} \in \text{LOP} \mid |A| \in \text{Im}(f)\}.$$

Note that

$$\#\text{C}_f(n) = \sum_{\substack{k \in \mathbb{N} \\ f(k) \leq n}} f(k),$$

since there are exactly $f(k)$ pairwise non-isomorphic structures of cardinality $f(k)$ in LOP.

Proposition 2.3.9 Let $f \in \mathrm{VP}$ and assume that for every polynomial $p \in \mathbb{N}[x]$ there is an $n \in \mathbb{N}$ such that

$$\sum_{\substack{k\in\mathbb{N}\\ f(2k)\leq n}} f(2k) \;>\; \sum_{\substack{k\in\mathbb{N}\\ f(2k+1)\leq p(n)}} f(2k+1). \tag{2.3}$$

Then $\mathrm{C}_{g_0} \not\leq_{\mathrm{pot}} \mathrm{C}_{g_1}$, where $g_0, g_1 : \mathbb{N} \to \mathbb{N}$ are defined by $g_0(n) := f(2n)$, $g_1(n) := f(2n+1)$.

Proof. We will prove this by contradiction. Assume that $\mathrm{C}_{g_0} \leq_{\mathrm{pot}} \mathrm{C}_{g_1}$. Then there is a polynomial $p \in \mathbb{N}[x]$ such that $\#C_{g_0}(n) \leq \#C_{g_1}(p(n))$, $\forall n \in \mathbb{N}$. Choose n such that (2.3) holds. Then

$$\#C_{g_0}(n) \;=\; \sum_{\substack{k\in\mathbb{N}\\ f(2k)\leq n}} f(2k) \;>\; \sum_{\substack{k\in\mathbb{N}\\ f(2k+1)\leq p(n)}} f(2k+1) \;=\; \#C_{g_1}(p(n))$$

is a contradiction and therefore $\mathrm{C}_{g_0} \not\leq_{\mathrm{pot}} \mathrm{C}_{g_1}$. □

We will now construct a countable atomless boolean algebra which we will use later in the proof of Theorem 2.3.18.

Lemma 2.3.10 (Construction of a countable Boolean algebra)
The set

$$V \;=\; \{\mathrm{Im}(f) \mid f \in \mathrm{VP}\} \cup \{A \subseteq \mathbb{N} \mid A \text{ is finite}\} \tag{2.4}$$

are the elements of a countable Boolean algebra $\mathcal{V} = (V, \cap, \cup)$.
Let $\equiv$ be the equivalence relation defined on V, where for $b, b' \in \mathrm{V}$

$$b \equiv b' \iff (b \setminus b') \cup (b' \setminus b) \text{ is finite}, \tag{2.5}$$

then the factor algebra $\mathcal{V}/\equiv$ is a countable atomless Boolean algebra.

Remark 2.3.11 Note that cofinite sets are contained in V since they are of the form $\mathrm{Im}(f)$ for some $f \in \mathrm{VP}$, i.e. for a finite set A, let $f : \mathbb{N} \to \mathbb{N} \setminus A$ be the strictly increasing enumeration of the set $\mathbb{N} \setminus A$. It is easy to find a polynomial $p \in \mathbb{N}[x]$ such that $f(k)$ can be computed in $\mathrm{DTIME}(p(f(k))$ for all $k \in \mathbb{N}$.

Proof. We need to verify that for $f, g \in \mathrm{VP}$, the sets

$$\mathbb{N} \setminus \mathrm{Im}(f), \quad \mathrm{Im}(f) \cap \mathrm{Im}(g), \quad \text{and} \quad \mathrm{Im}(f) \cup \mathrm{Im}(g)$$

are images of value-polynomial functions if they are infinite. We don't need to worry about finite sets since they are elements of V by definition.
Assume that $\mathbb{N} \setminus \mathrm{im}(f)$ is infinite. Since f is a value-polynomial, we can choose an algorithm $\mathbb{A}$ and a polynomial $p \in \mathbb{N}[x]$ such that for every $n \in \mathbb{N}$ the algorithm computes $f(n)$ in $\mathrm{DTIME}(p(f(n)))$. Let $h : \mathbb{N} \to (\mathbb{N} \setminus \mathrm{im}(f))$ be the function enumerating $\mathbb{N} \setminus \mathrm{im}(f)$. Clearly h is increasing and surjective. We will show that h is a value-polynomial too.
We define an algorithm $\mathbb{B}$ that inductively computes pairs $(h(0), m_0), (h(1), m_1), \ldots$ with

$$f(m_n) \;<\; h(n) \;<\; f(m_n + 1) \qquad \forall n \in \mathbb{N}.$$

Note that if $f(0) > 0$ and hence $h(0) = 0$, we set $(h(0), m_0) = (0, -1)$. For $n \geq 1$ the algorithm $\mathbb{B}$ gets $(h(n), m_n)$ from $(h(n-1), m_{n-1})$ by performing the following list of instructions:

(1) Set $k := h(n-1) + 1$ and $l := m_{n-1}$.

(2) Simulate $\mathbb{A}$ with input $l + 1$ for at most $p(k)$ steps.

(3a.) If $\mathbb{A}$ does not halt or if it outputs $f(l+1)$ with $f(l+1) > k$, then $(h(n), m_n) = (k, l)$.

(3b.) Otherwise, i.e. if $f(l+1) = k$, set $k := k + 1$ and $l := l + 1$ and goto (2).

It is easy to see that $\mathbb{B}$ computes all pairs $(h(k), m_k)$, $0 \leq k \leq n$ in DTIME$(p(h(n)))$.
Now assume that $\mathrm{im}(f) \cap \mathrm{im}(g)$ is infinite and let $p, q \in \mathbb{N}[x]$ be polynomials such that p, q compute $f(n), g(n)$ in DTIME$(p(f(n)))$, DTIME$(q(g(n)))$ respectively. We construct an algorithm $\mathbb{A}$ that outputs pairs $(n, h(n))$ such that $\mathrm{Im}(h) = \mathrm{Im}(f) \cap \mathrm{Im}(g)$ in the following way:

(1) Set $k, c_f, c_g := 0$.

(2) Set $l := f(c_f)$.

(3) Compute $g(c_g)$.

(4a) If $l = g(c_g)$ output $(k, f(k))$ and let $k := k + 1$, $c_f := c_f + 1$, $c_g := c_g + 1$ and goto (2).

(4b) If $l < g(c_g)$ set $c_f := c_f + 1$ and goto (2).

(4c) Otherwise, i.e. if $l > g(c_g)$ let $c_g := c_g + 1$ and goto (3).

It is easy to see that $h(n)$ can be computed in DTIME$((p+q)(h(n)))$ and therefore h is a value-polynomial-function.
To see that $\mathrm{im}(f) \cup \mathrm{im}(g)$ is the image of a value-polynomial-function you just need to slightly modify the algorithm described above. □

Definition 2.3.12 (Stretching function)
We recursively define a "stretching functions" $\varphi : \mathbb{N} \to \mathbb{N}$ as follows:

$$\varphi(n) := \begin{cases} 0, & \text{for } n = 0 \\ 1, & \text{for } n = 1 \\ (\varphi(0) + \ldots \varphi(n-1))^{n-1}, & \text{for } n \geq 2 \end{cases}$$

Remark 2.3.13 This stretching function φ has the following properties:

(i) $\varphi \in$ VP, and

(ii) if $\varphi^{-1}(m) < \varphi^{-1}(n+1)$, then $\varphi^{-1}(m) \leq \varphi^{-1}(n)$.

For the rest of the section φ will always be this stretching function.

We introduce the notation $\subseteq^*$ as it will be usefull later. For $f, g \in \mathrm{VP}$, we define

$$f \subseteq^* g \iff \mathrm{im}(f) \setminus \mathrm{im}(g) \text{ is finite.}$$

Definition 2.3.14 (Inverse of an increasing function)
For an increasing[2] function $f : \mathbb{N} \to \mathbb{N}$ we define the inverse $f^{-1} : \mathbb{N} \to \mathbb{N}$ as follows:

$$f^{-1}(n) := \max\{k \mid f(k) \leq n\},$$

and $\max \emptyset := 0$.

Lemma 2.3.15 Let $f, g : \mathbb{N} \to \mathbb{N}$ be increasing.

(i) f^{-1} is non-decreasing, $f^{-1} \leq \mathrm{id}_{\mathbb{N}}$, $f^{-1} \circ f = \mathrm{id}_{\mathbb{N}}$, and $f(f^{-1}(n)) \leq n$ for all $n \geq f(0)$.

(ii) $(f \circ g)^{-1} = g^{-1} \circ f^{-1}$.

(iii) If $f \in VP$, then f^{-1} is computable in polynomial time.

Proof.

(i) Let $n < m$, then $f(n) < f(m)$ and by definition $f^{-1}(n) \leq f^{-1}(m)$. Since $\mathrm{id}_{\mathbb{N}} : n \mapsto n$ is the *slowest increasing* function from $\mathbb{N} \to \mathbb{N}$ and $\mathrm{id}_{\mathbb{N}}^{-1} = \mathrm{id}_{\mathbb{N}}$, it follows that for all other increasing functions $f : \mathbb{N} \to \mathbb{N}$; $f^{-1}(n) \leq \mathrm{id}_{\mathbb{N}}$. The remaining two claims follow by our definition of inverse functions.

(ii) Using the fact that f, g are increasing and the definition of inverse functions we get

$$(f \circ g)^{-1}(n) = \max\{k \mid f(g(k)) \leq n\} = g^{-1}(\max\{k \mid f(k) \leq n\}) = g^{-1} \circ f^{-1}(n).$$

(iii) Follows from the definition of functions in VP, i.e. the fact that $f(n)$ is computable in $\bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(f(n)^k)$.

□

We introduce another notation. Let $f : \mathbb{N} \to \mathbb{N}$, then $f^{\Sigma} : \mathbb{N} \to \mathbb{N}$ is defined by

$$f^{\Sigma}(n) := \sum_{k \leq n} f(k).$$

Lemma 2.3.16 Let $f, g : \mathbb{N} \to \mathbb{N}$ be functions and g be increasing. Then $(f \circ g)^{\Sigma} \leq f^{\Sigma} \circ g$.

Proof. Using the definition we get

$$(f \circ g)^{\Sigma}(n) = \sum_{k \leq n} f(g(k)) = \sum_{\substack{k \leq g(n) \\ k \in \mathrm{im}(g)}} f(k) \leq \sum_{k \leq g(n)} f(k) = f^{\Sigma} \circ g(n).$$

The second equality uses the fact that g is increasing. □

[2] If we speak of increasing, we mean strictly increasing, i.e. for $n < m$ we have that $f(n) < f(m)$.

Lemma 2.3.17 If $f \in \mathrm{VP}$, then $\#\mathrm{C}_f(n) = (f^{\Sigma} \circ f^{-1})(n)$ for all $n \in \mathbb{N}$.

Proof.

$$\#\mathrm{C}_f(n) \;=\; \sum_{\substack{k\in\mathbb{N}\\ f(k)\le n}} f(k) \;=\; \sum_{\substack{k\in\mathbb{N}\\ k\le f^{-1}(n)}} f(k) \;=\; (f^{\Sigma}\circ f^{-1})(n).$$

□

All the preparation we did so far was to prove the following theorem.

Theorem 2.3.18 (Embedding a countable p.o. below LOU)
The mapping $f \mapsto \mathrm{C}_{\varphi\circ f}$; $(\mathrm{VP}, \subseteq^*) \to (\mathrm{C} \subseteq \mathrm{LOU}, \le_{\mathrm{iso}})$ is one-to-one. For all $f, g \in \mathrm{VP}$:

(i) if $f \subseteq^* g$ and $g \not\subseteq^* f$, then $\mathrm{C}_{\varphi\circ f} \le_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}$,

(ii) if $\mathrm{C}_{\varphi\circ f} \le_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}$, then $f \subseteq^* g$.

Proof. First we show that the mapping $f \mapsto \mathrm{C}_{\varphi\circ f}$ is injective. Therefore we assume that $\mathrm{Im}(\varphi \circ f) = \mathrm{Im}(\varphi \circ g)$. Since φ is injective and f and g are increasing it follows that $\mathrm{Im}(f) = \mathrm{Im}(g)$.

(i) The set $\mathrm{Im}(\varphi \circ f) \setminus \mathrm{Im}(\varphi \circ g)$ is finite because by assumption the set $\mathrm{Im}(f) \setminus \mathrm{Im}(g)$ is finite. Since φ is injective and the set $\mathrm{Im}(\varphi \circ g) \setminus \mathrm{Im}(\varphi \circ f)$ is infinite since $\mathrm{Im}(g) \setminus \mathrm{Im}(f)$ is infinite. Therefore we get $\mathrm{C}_{\varphi\circ f} \le_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}$ by defining a function that sends the finitely many structures of $\mathrm{C}_{\varphi\circ f} \setminus \mathrm{C}_{\varphi\circ g}$ to $\mathrm{C}_{\varphi\circ g} \setminus \mathrm{C}_{\varphi\circ f}$ and which is the identity on all other structures of $\mathrm{C}_{\varphi\circ f}$.

(ii) We are going to prove an equivalent version of the statement, i.e.

$$\text{for } f, g \in \mathrm{VP}. \text{ If } f \not\subseteq^* g, \text{ then } \mathrm{C}_{\varphi\circ f} \not\le_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}. \tag{2.6}$$

We are going to prove this by contradiction. Assume that $\mathrm{C}_{\varphi\circ f} \le_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}$ and therefore $\mathrm{C}_{\varphi\circ f} \le_{\mathrm{pot}} \mathrm{C}_{\varphi\circ g}$. Hence there is a polynomial $p \in \mathbb{N}[x]$ such that $\#\mathrm{C}_{\varphi\circ f}(n) \le \#\mathrm{C}_{\varphi\circ g}(p(n))$ for all $n \in \mathbb{N}$. It suffices to show that there is a $k \in \mathbb{N}$ such that $\#\mathrm{C}_{\varphi\circ f}(k) \not\le \#\mathrm{C}_{\varphi\circ g}(p(k))$. For this purpose we choose k such that

$$g(0) < f(k), \quad f(k) \in \mathrm{Im}(f) \setminus \mathrm{Im}(g), \quad \text{and} \quad p(\varphi(f(k))) < \varphi(f(k)+1). \tag{2.7}$$

This is possible because of the assumption in (2.6) and the fact that φ grows faster then every polynomial.
We now compute $\#\mathrm{C}_{\varphi\circ g}(p(\varphi(f(k))))$ using Lemma 2.3.17 and of Lemma 2.3.15 (ii);

$$\#\mathrm{C}_{\varphi\circ g}\big(p(\varphi(f(k)))\big) \;=\; (\varphi\circ g)^{\Sigma}\circ(\varphi\circ g)^{-1}\big(p(\varphi(f(k)))\big) \;=\; (\varphi\circ g)^{\Sigma}\circ(g^{-1}\circ\varphi^{-1})\big(p(\varphi(f(k)))\big).$$

Using $p(\varphi(f(k))) < \varphi(f(k)+1)$ (see 2.7) and the property (ii) of remark 2.3.13 we get

$$(\varphi \circ g)^{\Sigma} \circ (g^{-1} \circ \varphi^{-1})\big(p(\varphi(f(k)))\big) \;\le\; (\varphi \circ g)^{\Sigma} \circ g^{-1}\big(f(k)\big).$$

We further simplify using $f(k) \notin \mathrm{im}(g)$ (2.7), Lemma 2.3.16 as well as Lemma 2.3.15 and get

$$(\varphi\circ g)^{\Sigma}\circ g^{-1}\big(f(k)\big) \;=\; (\varphi\circ g)^{\Sigma}\circ g^{-1}\big(f(k)-1\big) \;\leq\; \varphi^{\Sigma}\circ g\circ g^{-1}\big(f(k)-1\big) \;\leq\; \varphi^{\Sigma}\big(f(k)-1\big).$$

Now we perform a strict estimate which is also the key part of this proof. By definition of φ we get

$$\varphi^{\Sigma}\big(f(k)-1\big) \;<\; \varphi\big(f(k)\big)$$

and finally

$$\varphi\big(f(k)\big) \;\leq\; (\varphi\circ f)^{\Sigma}\circ(\varphi\circ f)^{-1}\big(\varphi(f(k))\big) \;=\; \#\mathrm{C}_{\varphi\circ f}(\varphi(f(k))).$$

This shows us that $\mathrm{C}_{\varphi\circ f} \not\leq_{\mathrm{pot}} \mathrm{C}_{\varphi\circ g}$, hence $\mathrm{C}_{\varphi\circ f} \not\leq_{\mathrm{iso}} \mathrm{C}_{\varphi\circ g}$ which concludes the proof. □

2.4 Are the notion of $\leq_{\mathrm{iso}}$ and $\leq_{\mathrm{pot}}$ distinct?

Definition 2.4.1 (Double-exponential time)
The complexity classes *double-exponential time* and *nondeterministic double-exponential time* are defined as follows:

$$2\mathrm{EXP} := \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}\left(2^{2^{n^k}}\right) \qquad \text{and} \qquad \mathrm{N2EXP} := \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}\left(2^{2^{n^k}}\right).$$

The complexity class U2EXP contains precisely those problems Q that are accepted by double-exponential time bounded unambiguous Turing machines; these are nondeterministic Turing machines which on every input $x \in Q$ have at most one accepting run. As usual we define co-U2EXP $:= \{\Sigma^* \setminus Q \mid Q \in \mathrm{U2EXP}\}$.

Theorem 2.4.2 (Separation of $\leq_{\mathrm{iso}}$ and $\leq_{\mathrm{pot}}$)
If U2EXP $\cap$ co-U2EXP $\neq$ 2EXP, then the relations of $\leq_{\mathrm{iso}}$ and $\leq_{\mathrm{pot}}$ are distinct.

Proof. Let $Q \in \mathrm{U2EXP} \cap \text{co-U2EXP}$ then there exists a $d \geq 2$ and a nondeterministic turing machine $\mathbb{M}$ with the following four properties:

(M1) The set of terminal states of $\mathbb{M}$ is given by $\{'\mathrm{yes}', '\mathrm{no}', '\mathrm{maybe}'\}$.

(M2) For an input $x \in \Sigma^*$, every run of $\mathbb{M}$ stops after exactly $2^{2^{|x|^d}}$ steps.

(M3) For $x \in Q$ exactly one run stops in $'\mathrm{yes}'$ and none in $'\mathrm{no}'$ and vice versa, i.e. for $x \notin Q$ exactly one run stops in $'\mathrm{no}'$ and none in $'\mathrm{yes}'$.

(M4) In every nonterminal state the machine $\mathbb{M}$ has exactly two different choices for the next step.

For $n \in \mathbb{N}$ let $l(n) := 2^{2^{n^d}}$. For $x \in \Sigma^n$, every run of $\mathbb{M}$ on input x can uniquely be identified by a string $r \in \Sigma^{l(n)}$ and the other way around. Since there are 2^n different bit-strings we set $m(n) := 2^n$ and enumerate all strings of Σ^n by $x_1, x_2, \ldots, x_{m(n)}$ in lexicographic ordering. Let $s = s_1 s_2 \ldots s_{m(n)}$ be a binary string of length $m(n) \cdot l(n)$, with $|s_i| = l(n)$ for $1 \leq i \leq m(n)$. We call such a string s a *decision string* if every s_i corresponds to run of $\mathbb{M}$ on x_i that ends in either 'yes' or 'no'. By our assumption (M3) we get that:

$$\text{For every } n \in \mathbb{N}, \text{ there exists exactly one decision string } s \text{ of length } m(n) \cdot l(n). \quad (2.8)$$

With every $s \in \Sigma^{m(n) \cdot l(n)}$ we associate a structure $\mathcal{A}(s)$ over the vocabulary $\tau = Zero, One, R$, where $Zero$ and One are unary relation symbols and R is a binary relation symbol. For $\mathcal{A}(s)$ let

$$\begin{aligned} A(s) &:= [m(n) \cdot l(n)], \\ R^{\mathcal{A}(s)} &:= \{(j, j+1) \mid 1 \leq j \leq m(n) \cdot l(n) - 1\}. \\ Zero^{\mathcal{A}(s)} &:= \begin{cases} \{j \mid \text{the } j\text{th bit of } s \text{ is } 0\}, & \text{if } s \text{ is a decision string} \\ \emptyset, & \text{otherwise.} \end{cases} \\ One^{\mathcal{A}(s)} &:= \begin{cases} \{j \mid \text{the } j\text{th bit of } s \text{ is } 1\}, & \text{if } s \text{ is a decision string} \\ \emptyset, & \text{otherwise.} \end{cases} \end{aligned}$$

By this construction of $\mathcal{A}(s)$ and (2.8) we get for every $s, s' \in \Sigma^{m(n) \cdot l(n)}$

$$\mathcal{A}(s) \ncong \mathcal{A}(s') \iff \text{either } s \text{ or } s' \text{ is a decision string.} \quad (2.9)$$

Now let D_n be the class containing, up to isomorphism, the structures $\mathcal{A}(s)$ for $s \in \Sigma^{m(n) \cdot l(n)}$. It follows from the construction that

(D1) $|A| = m(n) \cdot l(n)$, for all $\mathcal{A} \in \mathrm{D}_n$, and

(D2) $|\mathrm{D}_n/\cong| = 2$.

We also construct for every $n \in \mathbb{N}$ the class C_n where every structure in C_n is either isomorphic to the complete graph $K_{m(n) \cdot l(n)}$ or its complement, i.e. the empty graph $\overline{K_{m(n) \cdot l(n)}} = ([m(n) \cdot l(n)], \emptyset)$ of order $m(n) \cdot l(n)$. Again by construction it is easy to see that

(C1) $|B| = m(n) \cdot l(n)$, for all $\mathcal{B} \in \mathrm{C}_n$

(C2) $|\mathrm{C}_n/\cong| = 2$.

Finally, we set

$$\mathrm{C} := \bigcup_{n \in \mathbb{N}} \mathrm{C}_n \qquad \text{and} \qquad \mathrm{D} := \bigcup_{n \in \mathbb{N}} \mathrm{D}_n.$$

By construction, it is easy to see that $\mathrm{C} \leq_{\text{pot}} \mathrm{D}$.
Now we claim that if there is a strong isomorphism reduction $f : \mathrm{C} \leq_{\text{iso}} \mathrm{D}$, then there is $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$

$$f(\mathrm{C}_n/\cong) \;=\; \mathrm{D}_n/\cong$$

which is supposed to be understood as a map from the isomorphism classes in C_n to the isomorphism classes of D_n.
To show this, we observe that by (C2) and (D2) it suffices to show that $f(\mathrm{C}_n) \subseteq \mathrm{D}_n$ for large enough n. Since f is computable in polynomial time there is a $c \in \mathbb{N}$ such that for all $\mathcal{A} \in \mathrm{C}_n$

$$|B| \;\leq\; \left(2^n \cdot 2^{2^{n^d}}\right)^c,$$

where B is the universe of $\mathcal{B} = f(\mathcal{A})$.
We now choose $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$

$$\left(2^n \cdot 2^{2^{n^d}}\right)^c \;<\; 2^{n+1} \cdot 2^{2^{(n+1)^d}}. \tag{2.10}$$

To see that such an n_0 exists we perform basic algebraic transformations.

$$\begin{aligned}
(2^n \cdot 2^{2^{n^d}})^c \;&<\; 2^{n+1} \cdot 2^{2^{(n+1)^d}} && \\
(2^{n+2^{n^d}})^c \;&<\; 2^{n+1+2^{(n+1)^d}} && \mid log_2 \\
c \cdot n + c \cdot 2^{n^d} \;&<\; n + 1 + 2^{(n+1)^d} && \text{, since } d \geq 2 \\
c \cdot n + c \cdot 2^{n^d} \;&<\; n + 1 + 2^{n^d+2n+1} \;\leq\; n + 1 + 2^{(n+1)^d} && \\
c \cdot n + c \cdot 2^{n^d} \;&<\; n + 1 + 2^{2n+1} \cdot 2^{n^d} \;\leq\; n + 1 + 2^{(n+1)^d}. &&
\end{aligned}$$

Since c is constant the linear parts become negligible for large n, it is now easy to see that such an n_0 exists.
Hence, for $n \geq n_0$

$$f\left(\bigcup_{k \leq n} \mathrm{C}_k\right) \;\subseteq\; \bigcup_{k \leq n} \mathrm{D}_k.$$

Since $\bigcup_{k \leq n} \mathrm{C}_k$ and $\bigcup_{k \leq n} \mathrm{D}_k$ contain, up to isomorphism, the same number of structures, the claim follows.
Now we have everything to prove the statement of the theorem, so assume that $f : \mathrm{C} \leq_{\text{iso}} \mathrm{D}$. We construct an algorithm $\mathbb{A}$ that shows that $Q \in 2\text{EXP}$. Let n_0 be as in (2.10). Hence for $x \in \Sigma^n$, with $n \geq n_0$ the algorithm $\mathbb{A}$ computes

$$\mathcal{A}(s) = f\left(K_{m(n)\cdot l(n)}\right) \quad \text{and} \quad \overline{\mathcal{A}(s')} = f\left(\overline{K_{m(n)\cdot l(n)}}\right).$$

They are in D_n and nonisomorphic as f preserves isomorphisms as well as nonisomorphisms. By (2.9) exactly one of s and s' is a decision string, i.e. we get a run of M on input x that ends in 'yes' or in 'no'. The algorithm $\mathbb{A}$ answers accordingly. □

2.5 If $\leq_{\text{iso}} \neq \leq_{\text{pot}}$, then P $\neq$ #P

In this section we will continue with our analysis of the two reductions $\leq_{\text{iso}}$ and $\leq_{\text{pot}}$ defined in the previous chapter. The main goal of this section, first shown in [3], is to show that we would get P $\neq$ #P if we could separate the two notions of reduction without any complexity-theoretic assumption.

Definition 2.5.1 (Complexity class #P)
The complexity class #P is the set of functions $f_{\mathbb{M}} : \Sigma^* \to \mathbb{N}$ such that there exists a polynomial time nondeterministic turing machine $\mathbb{M}$ such that for all $x \in \Sigma^*$, $f_{\mathbb{M}}(x)$ is the number of accepting paths of $\mathbb{M}$ on input x.

In this section we consider equivalence relations on all of Σ^*. We start by a extending the isomorphism relation to a more general equivalence relation.

Definition 2.5.2 (Extended isomorphism relation)
For a class C, let $E(\mathrm{C})$ be the equivalence relation on Σ^* defined by

$$E(\mathrm{C}) := \{(\mathcal{A}, \mathcal{B}) \mid \mathcal{A}, \mathcal{B} \in \mathrm{C} \text{ and } \mathcal{A} \cong \mathcal{B}\} \cup \{(x, y) \mid x, y \in \Sigma^*, x \notin \mathrm{C} \text{ and } y \notin \mathrm{C}\}.$$

Since C is in P, $E(\mathrm{C})$ is clearly in NP.

We define the set

$$\mathrm{CC(eq)} := \{E \mid E \text{ equivalence relation on } \Sigma^*, E \in \mathrm{CC}\}$$

Clearly $E(\mathrm{C}) \in \mathrm{NP(eq)}$ for all classes C and $E(\mathrm{LOU}) \in \mathrm{P(eq)}$.
We generalize the notion of strong isomoprhism reduction to arbitrary equivalence relations.

Definition 2.5.3 (Strong equivalence reduction)
Let E and E' be equivalence relations on Σ^*. We say that E is *strongly equivalence reducible* to E' if there is a function $f : \Sigma^* \to \Sigma^*$ computable in polynomial time such that for all $x, y \in \Sigma^*$

$$xEy \iff f(x)E'f(y).$$

We then write $f : E \leq_{\text{eq}} E'$. If $E \leq_{\text{eq}} E'$ and $E' \leq_{\text{eq}} E$, we write $E \equiv_{\text{eq}} E'$.

Let PROP be the set of all *formulas of propositional logic* and TAUT be the set of *tautologies* and note that we view all formulas as strings. We define the equivalence relation E_{equiv} on Σ^* induced by logical equivalent formulas

$$E_{\text{equiv}} := \{(\alpha, \beta) \mid \alpha, \beta \in \textsc{Prop} \text{ and } (\alpha \leftrightarrow \beta) \in \textsc{Taut}\} \cup \{(x, y) \mid x, y \notin \textsc{Prop}\}.$$

Note that $E_{\mathrm{equiv}} \in$ co-NP(eq) and that there is a correspondence between strong isomorphism reduction and strong equivalence reduction, i.e.

$$\mathrm{C} \leq_{\mathrm{iso}} \mathrm{D} \iff E(\mathrm{C}) \leq_{\mathrm{eq}} E(\mathrm{D}).$$

We will as well generalize the notion of potential reducibility to arbitrary equivalence relations.

Definition 2.5.4 (Potential reduction)
Let $E, E' \subseteq \Sigma^* \times \Sigma^*$ be equivalence relations. We say that E is *potential reducible* to E' (denoted by $E \leq_{\mathrm{pot}} E'$), if there is a polynomial $p \in \mathbb{N}[x]$ such that $|\Sigma^{\leq n}/E| \leq |\Sigma^{\leq p(n)}/E'|$, for all $n \in \mathbb{N}$.

We will now show that for classes $E(\mathrm{C})$ as defined above this new notion of potential reduction coincides with the old one.

Proposition 2.5.5 Let C, D be classes. Then

$$\mathrm{C} \leq_{\mathrm{pot}} \mathrm{D} \iff E(\mathrm{C}) \leq_{\mathrm{pot}} E(\mathrm{D}). \tag{2.11}$$

Proof. Let C be a class of τ-structures and D be a class of μ-structures and let $p_\tau, p_\mu \in \mathbb{N}[x]$ be such that for every τ-structure $\mathcal{A}$ and μ-structure $\mathcal{B}$ the following holds:

$$|A| \leq |\ulcorner \mathcal{A} \urcorner| \leq p_\tau(|A|) \quad \text{and} \quad |B| \leq |\ulcorner \mathcal{B} \urcorner| \leq p_\mu(|B|). \tag{2.12}$$

First assume that C $\leq_{\mathrm{pot}}$ D, i.e. there is a polynomial $p \in \mathbb{N}[x]$ such that $\#\mathrm{C}(n) \leq \#\mathrm{D}(p(n))$. Then

$$|\Sigma^{\leq n}/E(\mathrm{C})| \leq \#\mathrm{C}(n) + 1 \leq \#\mathrm{D}(p(n)) + 1 \leq |\Sigma^{\leq p_\mu(p(n))}/E(\mathrm{D})|.$$

Therefore $E(\mathrm{C}) \leq_{\mathrm{pot}} E(\mathrm{D})$.

Now assume that $E(\mathrm{C}) \leq_{\mathrm{pot}} E(\mathrm{D})$, i.e. there is a polynomial $p \in \mathbb{N}[x]$ such that $|\Sigma^{\leq n}/E(\mathrm{C})| \leq |\Sigma^{\leq p(n)}/E(\mathrm{D})|$. Then

$$\#\mathrm{C}(n) + 1 \leq |\Sigma^{\leq p_\tau(n)}/E(\mathrm{C})| \leq \left|\Sigma^{p(p_\tau(n))}/E(\mathrm{D})\right| \leq \#\mathrm{D}(p(p_\tau)) + 1.$$

This concludes the proof. □

Lemma 2.5.6 Let E, E' be equivalence relations on Σ^*. If $E \leq_{\mathrm{eq}} E'$, then $E \leq_{\mathrm{pot}} E'$.

The proof is analogous to the proof of Lemma 2.2.4.

Corollary 2.5.7 Let E, E' be equivalence relations on Σ^*. If $E \not\leq_{\mathrm{pot}} E'$, then $E \not\leq_{\mathrm{eq}} E'$.

We can show that the notions of $\leq_{\mathrm{eq}}$ and $\leq_{\mathrm{pot}}$ are distinct under weaker assumptions than in Theorem 2.4.2, in fact the notion of strong equivalence reduction is finer than that of potential reducibility.

Proposition 2.5.8 If P≠NP, then the relations of $\leq_{\text{eq}}$ and $\leq_{\text{pot}}$ do not coincide on NP(eq).

Proof. Let $Q \in \text{NP}\backslash\text{P}$, we define E_Q by

$$xE_Qy \iff \Big(x = y \vee \Big(x = b^\frown z \wedge y = (1-b)^\frown z,\, z \in Q, b \in \Sigma\Big)\Big).$$

Since $Q \in \text{NP}\backslash\text{P}$, we know that $E_Q \in \text{NP(eq)}$. Let id be the identity on Σ^*. It is easy to see that $E_Q \leq_{\text{pot}} \text{id}$. Since $Q \notin \text{P}$, we get $E_Q \not\leq_{\text{eq}} \text{id}$. Otherwise any strong equivalence reduction $f : E_Q \leq_{\text{eq}} \text{id}$ would yield a polynomial time decision procedure to decide whether xE_Qy and therefore $Q \in \text{P}$ which contradicts our assumption.
Thus the notions of $\leq_{\text{eq}}$ and $\leq_{\text{pot}}$ are distinct on NP(eq). □

We will now generalize the notion of canonization (and that of the enumeration induced by a canonization) that we introduced earlier.

Definition 2.5.9 (Canonization)
A *canonization* for an equivcalence relation $E \in \text{CC(eq)}$ is a polynomial time computable function $\text{Can} : \Sigma^* \to \Sigma^*$ such that the following conditions hold:

(i) For all $x, y \in \Sigma^*$
$$xEy \iff \text{Can}(x) = \text{Can}(y).$$

(ii) For all $x \in \Sigma^*$
$$xE\,\text{Can}(x)$$

Definition 2.5.10 (Enumeration induced by Can)
Let Can be a canonization of E. The *enumeration induced by* Can is a sequence of strings

$$x_1, x_2, \ldots$$

where each $x_k \in \text{Im(Can)}$ and $i < j$ iff $x_i <_{\text{lex}} x_j$.

Clearly if E has a canonization, then $E \in \text{P}$.
A common approach to check whether xEy is to compute $\text{Can}(x)$ and $\text{Can}(y)$ and check whether $\text{Can}(x) = \text{Can}(y)$.

Lemma 2.5.11 If P = NP, then every $E \in \text{P(eq)}$ has a canonization, more precise, the mapping sending each $x \in \Sigma^*$ to the $<_{\text{lex}}$-first member of $[x]$ is a canonization.

Proof. Let $E \in \text{P(eq)}$ and assume P = NP. Then the polynomial hierarchy collapses, i.e. P = PH. Therefore it suffices to show that the mapping described above is somewhere in the polynomial hierarchy. This is easy to show, since there is an alternating polynomial time algorithm $\mathbb{A}$ with a finite number of alternations, that on input $x \in \Sigma^*$ guesses existentially $y \in \Sigma^*$ such that $|y| \leq |x|$ and xEy and then $\mathbb{A}$ guesses universally $z \in \Sigma^*$ with $|z| \leq |x|$ and xEz. Finally $\mathbb{A}$ outputs y if $y \leq z$ and rejects otherwise. □

Lemma 2.5.12 Let $E \in \text{P}(eq)$ be an equivalence relation with a canonization Can. Then the problem defined below is in #P:

INDEX OF Can	
Instance:	$x \in \Sigma^*$
Problem:	Compute i (in binary) such that $\text{Can}(x)$ is the ith element in the enumeration induced by Can.

Proof. Consider a nondeterministic polynomial time algorithm $\mathbb{A}$ that runs as follows: On input $x \in \Sigma^*$, it first computes $y := \text{Can}(x)$. Then $\mathbb{A}$ guesses existentially $z \in \Sigma^*$ with $|z| \leq |y|$. Finally it accepts if $\text{Can}(z) = z$ and $z \leq y$. The number of accepting runs of $\mathbb{A}$ on input x is

$$|\{z \mid z \leq \text{Can}(x) \text{ and } \text{Can}(z) = z\}|\,.$$

□

Theorem 2.5.13 If the relations of strong equivalence reduction and potential reduction do not coincide on NP(eq), then P $\neq$ #P.

Proof. Our goal is to show that if P = #P, then the relation of strong equivalence reduction and potential reduction coincide on NP(eq). To see this start by assuming that P = #P. Let $E, E' \in \text{NP}(eq)$ be equivalence relations and assume that $E \leq_{\text{pot}} E'$, i.e. there is a polynomial $p \in \mathbb{N}[x]$ such that $|\Sigma^{\leq n}/E| \leq |\Sigma^{\leq p(n)}/E'|$ for all $n \in \mathbb{N}$. We show that $E \leq_{\text{eq}} E'$.
Since we assumed P = #P, we have P = NP and therefore $E, E' \in \text{P}(eq)$. Thus, by Lemma 2.5.11, there are canonizations Can_E and $\text{Can}_{E'}$, and there are polynomial time algorithms $\mathbb{A}$ and $\mathbb{A}'$ that solve the problem of Lemma 2.5.12 for E and E' respectively. The following nondeterministic polynomial time algorithm computes $f : E \leq_{\text{eq}} E'$:
On input $x \in \Sigma^*$, it computes $\text{Can}_E(x)$ and $n := |\text{Can}_E(x)|$ and guesses a string $x' \in \Sigma^*$ with $\text{Can}_{E'}(x') = x'$. It then simulates $\mathbb{A}$ and $\mathbb{A}'$ and checks if $\text{Can}_E(x)$ and x' have the same position in the enumeration induced by Can_E and $\text{Can}_{E'}$ respectively. If so, it outputs x', otherwise it rejects. Since $|\Sigma^{\leq n}/E| \leq |\Sigma^{\leq p(n)}/E'|$ such a $x' \in \Sigma^{\leq p(n)}$ with $\text{Can}_{E'}(x) = x'$ at the same position as $\text{Can}_E(x)$ exists. Since by assumption P = NP, it follows that f is computable in polynomial time. □

The existence of a maximum element of P(eq) and NP(eq) with respect to strong eqivalence reduction was studied in [3]. More specifically, they proved the following theorem:

Theorem 2.5.14 (Maximum element of P(eq) and NP(eq)) The following hold:

(i) If E=NE, then P(eq) has a maximum element.

(ii) If NP=co-NP, then NP(eq) has a maximum element.

3 Recognition, Invariant, Canonization and First Member problems

This chapter follows the contents of [1] and [2]. So far we mostly looked at equivalence relations 'being in P' or 'having an invariantization/canonization' as a property. In this chapter we will now turn our attention to the underlying problems of those definitions. We consider the following four problems:
The *recognition problem*:

P(eq)
Instance: $x, y \in \Sigma^*$
Problem: Determine if xEy?

The *invariant problem*:

Inv(eq)
Instance: $x \in \Sigma^*$
Problem: Calculate $\mathrm{Inv}_E(x)$.

The *canonization problem*:

Can(eq)
Instance: $x, \in \Sigma^*$
Problem: Calculate $\mathrm{Can}_E(x)$.

And the *first member problem*:

Lexfirst(eq)
Instance: $x \in \Sigma^*$
Problem: Calculate y such that xEy and $y <_{\mathrm{lex}} x$ and for all z with xEz we have $z \not<_{\mathrm{lex}} y$.

Definition 3.0.1 (Invariantizitaion)
For $x \in \Sigma^*$ and $E \subseteq \Sigma^* \times \Sigma^*$ we define Inv_E analogously to Definition 2.1.6, i.e. $\mathrm{Inv}_E : \Sigma^* \to \Sigma^*$ is a polynomial time computable function such that

$$xEy \iff \mathrm{Inv}_E(x) = \mathrm{Inv}_E(y).$$

Note that we are only interested in polynomial time solutions for those problems, therefore a solution for one of the problem automatically yields a solution for all of the previous problems, i.e.

$$\textsc{Rec}(\mathrm{eq}) \leq_p \textsc{Inv}(\mathrm{eq}) \leq_p \textsc{Can}(\mathrm{eq}) \leq_p \textsc{Lexfirst}(\mathrm{eq}). \tag{3.1}$$

Naturally the question arises whether they are also polynomial time equivalent or if those problems are of increasing complexity. We showed in Lemma 2.5.11 that P = #P would yield

$$\textsc{Rec}(\text{eq}) \equiv_p \textsc{Inv}(\text{eq}) \equiv_p \textsc{Can}(\text{eq}) \equiv_p \textsc{Lexfirst}(\text{eq}).$$

The main goal of the next section is to show that this is not the case in general.

3.1 Nonreducibility

To show that the reductions in (3.1) are strict we will look at a special kind of Turing machines, namely the so called oracle machines.

Definition 3.1.1 (Oracle machine)
Let $O \subseteq \Sigma^*$ be a problem. An *orcale machine with oracle* O is a Turing machine $\mathbb{M}$ (later called $\mathbb{M}^O$) with an additional work tape, called *oracle tape* and three additional states $s_{query}, s_{yes}, s_{no}$. If $\mathbb{M}$ enters the s_{query} state the machine does two things in a single computational step; it reads the content of the oracle tape and it changes to either the state s_{yes} or s_{no} accordingly.

Before we can prove that main theorem we need the following lemma:

Lemma 3.1.2 Let X be a non empty set of cardinality $2k$ and let $R \subseteq X \times X$ be a binary relation on X. Suppose that for every $x \in X$ there are less than k elements $y \in X$ such that $(x, y) \in R$. Then there exists $\tilde{x}, \tilde{y} \in X$, with $\tilde{x} \neq \tilde{y}$ and $(\tilde{x}, \tilde{y}), (\tilde{y}, \tilde{x}) \notin R$.

Proof. This can be seen by a simple counting argument. On the one hand the number of all two-element subsets of X is given by $\left|[X]^2\right| = \binom{2k}{2} = k(2k-1)$ but on the other hand

$$\left| \bigcup_{x \in X} \Big\{ \{x, y\} \mid (x, y) \in R \Big\} \right| \leq 2k \cdot (k-1) < k \cdot (2k-1).$$

Therefore there must be distinct $\tilde{x}, \tilde{y} \in X$, such that neither $(\tilde{x}, \tilde{y})$ nor $(\tilde{y}, \tilde{x})$ is in R. □

Theorem 3.1.3 (Nonreducibility)

(i) There is an equivalence relation E on Σ^* such that the invariant problem is not polynomial time reducible to the recognition problem, even with an oracle for E.

(ii) There are an equivalence relation E on Σ^* and a polynomial time computable solution Inv_E for its invariant problem, such that there is no polynomial time algorithm that solves the canonization problem for E even with Inv_E as an oracle.

(iii) There is an equivalence relation E on Σ^* and a polynomial time solution Can_E for its canonization problem such that there is no polynomial time reduction from the first member problem to the canonization problem even with Can_E as an oracle.

Proof. Let $\mathbb{M}_0, \mathbb{M}_1, \ldots$ be the enumeration of all polynomial time oracle machines with alphabet Σ and for every k let $p_k \in \mathbb{N}[x]$ be the time bound of the machine $\mathbb{M}_k$. We construct E in steps and at the end we let $E := \bigcup_k E_k$. We start by letting E_0 be the equality relation on Σ^* At each step we either set $E_k := E_{k-1}$ or $E_k := E_{k-1} \cup \{(x,y),(y,x)\}$ for some $x, y \in \Sigma^{d_k}$, where $d_0 < d_1 < \ldots$ is a strictly increasing sequence. Thus E has the following properties:

(E1) Each equivalence class has at most two members.

(E2) If an equivalence class has two members, the both have the same length.

(E3) For each length there is at most one equivalence class with two members.

The sequence $d_0 < d_1 < \ldots$ is choosen such that $p_k(d_k) < d_{k+1}$ and $p_k(d_k) < 2^{d_k-1}$. Thus, the machine $\mathbb{M}_k$ on an input of length d_k asks fewer than 2^{d_k-1} queries and each query is shorter than d_{k+1}. At the end of each step we show that $\mathbb{M}_k$ (even with an oracle for the easier problem) cannot correctly solve the harder problem in polynomial time. Since this is the case for every polynomial time bounded oracle machine $\mathbb{M}$, we get the desired result.
For the rest of the proof we only need to look at an arbitrary step k and use $d = d_k, \mathbb{M} = \mathbb{M}_k, p = p_k$ as well as $E = E_k$ and $E_{-1} = E_{k-1}$ for better readability.

(i) Let x' be the result of $\mathbb{M}$ on input x with an oracle for E. There are two cases.
Case 1 If there are x, y, with $x \neq y$ but $x' = y'$, we just set $E := E_{-1}$. $\mathbb{M}^E$ fails to solve the invariant problem for E because $\mathbb{M}^E(x) = x' = y' = \mathbb{M}^E(y)$ but $(x,y) \notin E$.
Case 2 Suppose that $x' \neq y'$ for all $x \neq y$. We say that x *affects* y if $\mathbb{M}$ queries E about (x,y) or (y,x) in the computation of y'. Since by assumption each run of $\mathbb{M}$ has at most 2^{d-1} queries we know that each y can be affected by at most 2^{d-1} elements x. By Lemma 3.1.2 (with $X = \Sigma^d$ and R being the "affected by" relation) there are distinct x, y such that neither xRy nor yRx. Now let $E := E_{-1} \cup \{(x,y),(y,x)\}$. Since x and y do not affect each other, the computation of x' and y' do not change if we use E instead of E_{-1} as an oracle. But now $\mathbb{M}^E$ fails to solve the invariant problem for E because $(x,y) \in E$ but $\mathbb{M}^E(x) = x' \neq y' = \mathbb{M}^E(y)$.

(ii) We will construct the desired invariantization Inv_E simultaneously with E. At the begining $\mathrm{Inv}_{E_0}(x) = x^\frown 1$. It is easy to see tat Inv_0 indeed is an invariantization for E_0. Again for better readability we will use $\mathrm{Inv} := \mathrm{Inv}_E$ and $\mathrm{Inv}_{-1} := \mathrm{Inv}_{E_{-1}}$.
For each x we define

$$\mathrm{Inv}^x(z) := \begin{cases} 0^{d+1}, & \text{if } z = x \\ \mathrm{Inv}_{-1}(z), & \text{otherwise.} \end{cases}$$

Now let x' be the output of $\mathbb{M}$ on input x using Inv^x as an oracle. As before we have two cases.
Case 1 If there is an x with $x' \neq x$, just set $\mathrm{Inv} := \mathrm{Inv}^x$. Note that this does

not affect the computation of x'. Since $\mathrm{Inv} : \Sigma^d \to \Sigma^{d+1}$ is injective, we just set $E := E_{-1}$. Now $(x, x') \notin E$ but $\mathbb{M}^{\mathrm{Inv}}$ fails to solve the canonization problem for E.
Case 2 Suppose that $x' = x$ for all x. Then by Lemma 3.1.2 we can choose distinct x, y such that $\mathbb{M}$ does not query Inv^x about y in the computation of x' and $\mathbb{M}$ does not query Inv^y about x in the computation of y'. We now set

$$\mathrm{Inv}(z) := \begin{cases} 0^{d+1}, & \text{if } z = x \text{ or } z = y \\ \mathrm{Inv}_{-1}(z), & \text{otherwise.} \end{cases}$$

and $E := E_{-1} \cup \{(x, y), (y, x)\}$. Now $\mathbb{M}^{\mathrm{Inv}}$ fails to solve the canonization problem for E because $(x, y) \in E$ but $\mathbb{M}^{\mathrm{Inv}}(x) = x \neq y = \mathbb{M}^{\mathrm{Inv}}(y)$.

(iii) We use the same idea as in (ii) and construct $\mathrm{Can} := \mathrm{Can}_E$ alongside E. Initially Can_{E_0} is the identity on Σ^*. For each x we define

$$\mathrm{Can}^x(z) := \begin{cases} 1^d, & \text{if } z = x \\ \mathrm{Can}_{-1}(z), & \text{otherwise.} \end{cases}$$

Similar as before we let x' be the result of $\mathbb{M}$ with input x using the oracle for Can^x. We will again look at the two cases.
Case 1 If there is an x with $x' \neq x$, then we set $\mathrm{Can} := \mathrm{Can}^x$ and $E := E_{-1} \cup \{(x, 1^d), (1^d, x)\}$. The first member problem is not solvable by $\mathbb{M}^{\mathrm{Can}}$ because $\mathbb{M}^{\mathrm{Can}}(x) = 1^d \neq x$ but clearly x is the first member in the equivalence class $[x]$.
Case 2 On the other hand, suppose that $x' = x$ for all x. Therefore in particular $(1^d)' = 1^d$. Choose $x \in \Sigma^d \setminus \{1^d\}$ such that $\mathbb{M}$ does not query the oracle about x in the computation of $(1^d)'$. By our assumption this is possible since $\mathbb{M}$ uses fewer than $p(d) < 2^{d-1}$ queries. We now set $\mathrm{Can} := \mathrm{Can}^x$. This does not change the computations of x' and $(1^d)'$. Finally let $E := E_{-1} \cup \{(x, 1^d), (1^d, x)\}$. Now $\mathbb{M}^{\mathrm{Can}}$ fails to solve the first member problem of E because $\mathbb{M}^{\mathrm{Can}}(1^d) = 1^d \neq x$ but x is the first member of the equivalence class $[1^d]$. □

Remark 3.1.4 Note that the previous proof relies on the fact that the changes we make to E, Inv and Can at a certain step do not change the computation of any previous step.

4 Benchmark relations and structural results

This last chapter is largely motivated by [6]. We have already seen in Chapter 2 that the structure of $\leq_{\text{eq}}$ is quite rich even when restricting it to isomorphism relations. In this chapter we first identify some benchmark equivalence relations: id, E_λ and E_σ (see Definition 4.1.1) and show that there is even a rich structure of $\leq_{\text{eq}}$ between E_λ and id, more precise we will construct an inifinite chain and and infinite antichain between E_λ and id. We then show that there is an initial segment of the $\leq_{\text{eq}}$-hierarchy consisting of certain equivalence relations in P. Finally we consider equivalence relations with only finitely many non trivial equivalence classes and those whose equivalence classes are all finite.

4.1 The Benchmark Equivalence Relations id, E_λ, E_σ

We will start by introducing some canonical examples of equivalence relations, which will serve us as benchmarks for the hierarchy of strong equivalence reductions. Later we will even compare how these benchmark equivalence relations relate to those in the previous chapter.

Definition 4.1.1 (Benchmark equivalence relations)
Let id, E_λ and E_σ be equivalence relations on Σ^* defined as follows:

(i) Let *id* be the *identity relation*, i.e.

$$(x, y) \in id \iff x = y.$$

(ii) We denote by E_λ the *equality of length relation*, i.e.

$$(x, y) \in E_\lambda \iff |x| = |y|.$$

(iii) Let E_σ be the equivalence relation such that for all $x, y \in \Sigma^*$ we have

$$(x, y) \in E_\sigma \iff \text{if } l_m \leq |x| < l_{m+1} \text{ and } l_n \leq |y| < l_{n+1}, \text{ then } m = n,$$

where $(l_k)_{k\in\mathbb{N}}$ is the *super-exponential sequence*, i.e. $l_0 = 0$ and for all $k \geq 1$, $l_{k+1} = 2^{l_k}$.

The relative complexity of those benchmark equivalence relations is strictly increasing.

Lemma 4.1.2 $E_\sigma \lneq_{\text{eq}} E_\lambda \lneq_{\text{eq}} id$.

Proof. To see that $E_\lambda \leq_{\text{eq}} id$ it suffices to define the strong equivalence reduction function $f : \Sigma^* \to \Sigma^*$ by $f(x) = 0^{|x|}$. For $E_\sigma \leq_{\text{eq}} E_\lambda$ we define a strong equivalence reduction function $g : \Sigma^* \to \Sigma^*$ by $g(x) = 0^k$ where $k = max\{k \in \mathbb{N} \mid l_k \leq |x|\}$.
To see the non-reducibility apply Corollary 2.5.7 to $\#id(n) \geq 2^n, \#E_\lambda(n) = n$, and $\#E_\sigma(n) \geq \log n$. □

We can create new equivalence relations E_n by replacing the sequence l_k in the definition of E_σ by "sparser" sequences to obtain an infinite descending chain of equivalence relations below E_σ.
We will state the following result without a proof.

Corollary 4.1.3
There is an infinite sequence $(E_n)_{n\in\mathbb{N}}$ of equivalence relations such that $E_{n+1} \lneq_{\text{eq}} E_n \lneq_{\text{eq}} E_\sigma$ for all $n \in \mathbb{N}$.

4.1.1 The structure between E_λ and id

Using binary encoding, we will view $\mathbb{N}$ as a subset of Σ^*. In particular we will speak of polynomial time computable functions from $\mathbb{N}$ to $\mathbb{N}$ and subset of $\mathbb{N}$ in various different complexity classes.

Definition 4.1.4 (Good set)

(i) We call a set $A \subseteq \mathbb{N}$ *good* if A is infinite, $A \in \mathrm{P}$ and $a_n \mapsto n$ is computable in polynomial time for all $n \geq 1$, where $(a_n)_{n\geq 1}$ is the enumeration of A in strictly increasing order.

(ii) Let $A \subseteq \mathbb{N}$ be a good set and $(a_n)_{n\geq 1}$ be the enumeration of A in strictly increasing order. We define $L_A : \Sigma^* \to \mathbb{N}$ by

$$L_A(x) = \begin{cases} 0, & \text{for } |x| < a_1 \\ n, & \text{for } a_n \leq |x| < a_{n+1}. \end{cases}$$

(iii) Let $A \subseteq \mathbb{N}$ be a good set, $(a_n)_{n\geq 1}$ the enumeration of A in strictly increasing order and let $\varphi : \mathbb{N} \to \mathbb{N}$ be a polynomial time computable function such that $\varphi(n) \leq a_n$ for all $n \geq 1$. We define an equivalence relation $E_{\varphi,A}$ as follows:

$$E_{\varphi,A} = \{(x, y) \mid L_A(x) = L_A(y) \text{ and} \\ \text{if } L_A(x) \neq 0, \text{ then } x \text{ and } y \text{ agree on the first } \varphi(L_A(x)) \text{ bits}\}.$$

Lemma 4.1.5 If $A \subseteq \mathbb{N}$ is a good set, then L_A is polynomial time computable.

Proof.
Let A be a good set and $(a_n)_{n\geq 1}$ the enumeration of all elements in A in strictly increasing

order. Then, by definition there are a polynomial time computable function $\psi : a_n \mapsto n$ and a polynomial time computable function $\chi_A : \mathbb{N} \to \Sigma$ recognising A, i.e.

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A. \end{cases}$$

We compute L_A in the following way: Given $x \in \Sigma^*$, we compute $|x|$. Then run $\chi_A(m)$ for $m = |x|, |x|-1, \ldots 0$ until we find the first m such that $\chi_A(m) = 1$. This can clearly be done in polynomial time in $|x|$, since we run at most $|x|$ computations of χ_A on inputs of size at most $|x|$. If $\chi_A(m) = 1$, then $m \in A$ and therefore $m = a_n$ for some $n \geq 1$. For the largest m such that $\chi_A(m) = 1$ and $m = a_n$, we have that $|x| < a_{n+1}$. Now

$$L_A(x) = \begin{cases} \chi_A(m), & \text{if there is a } m \text{ as described above} \\ 0, & \text{if the computation does not return any } m \text{ with } \chi_A(m) = 1. \end{cases}$$

Not that $L_A(x) = 0$ if and only if $|x| < a_1$. This is exactly the L_A from the definition above. □

From the definition of $E_{\varphi,A}$ it is easy to see that $E_{\varphi,A} \leq_{\text{eq}} id$. The following theorem describes the close relation between the complexity of $E_{\varphi,A}$ and the growth rate of φ and $n \mapsto a_n$, which is the function that enumerates the elements of A.

Theorem 4.1.6
Let $A \subseteq \mathbb{N}$ be a good set and $(a_n)_{n\geq 1}$ the enumeration of A in strictly increasing order. Let $\varphi, \psi : \mathbb{N} \to \mathbb{N}$ be polynomial time computable functions with $\varphi(n), \psi(n) \leq a_n$ for all $n \geq 1$. Then the following hold:

(i) If $\varphi(n) \leq \psi(n)$ for all n, then $E_{\varphi,A} \leq_{\text{eq}} E_{\psi,A}$.

(ii) If φ is increasing, $\varphi(n) = \Omega(log\, a_n)$ and for any polynomial $p \in \mathbb{N}[x]$, it holds that

$$\psi(n) \neq O(\varphi(p(a_n))),$$

then $E_{\psi,A} \not\leq_{\text{eq}} E_{\varphi,A}$.

(iii) Let B be a good set and let $(b_n)_{n\geq 1}$ be the enumeration of B in strictly increasing order. If for all $n \geq 1$, $a_n \leq b_n$, then $E_{\varphi,B} \leq_{\text{eq}} E_{\varphi,A}$.

Proof.
Let $A, (a_n)_{n\geq 1}$, and $\varphi, \psi : \mathbb{N} \to \mathbb{N}$ be as in the theorem.

(i) Let $\varphi(n) \leq \psi(n)$ for all $n \in \mathbb{N}$. We define a function $f : \Sigma^* \to \Sigma^*$ and show that it is a strong equivalence reduction function from $E_{\varphi,A}$ to $E_{\psi,A}$. Let $x \in \Sigma^*$. If $|x| < a_1$ (then $L_A(x) = 0$) and we define $f(x) := \varepsilon$. If on the other hand $L_A(x) > 0$ then we have $\varphi(L_A(x)) \leq a_{L_A(x)} \leq |x|$. In this case we define $f(x) := x'$ where $|x'| = |x|$ and x' agree on the first $\varphi(L_A(x))$ bits with x and is 0 on the remaining bits.

We now show that f is the desired function. Let $x, y \in \Sigma^*$. If $(x,y) \in E_{\varphi,A}$ then, by definition, $L_A(x) = L_A(y)$ and x and y agree on the first $\varphi(L_A(x))$ bits hence by construction $f(x) = f(y)$ and therefore $(f(x), f(y)) \in E_{\psi,A}$. If on the other hand $(x,y) \notin E_{\psi,A}$, then either $L_A(x) \neq L_A(y)$ or $L_A(x) = L_A(y) > 0$ but x and y do not agree on the first $\varphi(L_A(x))$ bits. If $L_A(x) \neq L_A(y)$, then $L_A(f(x)) \neq L_A(f(y))$ and therefore $(f(x), f(y)) \notin E_{\psi,A}$. If $L_A(x) = L_A(y) > 0$ but they don't agree on the first $\varphi(L_A(x))$ bits, then $f(x)$ and $f(y)$ don't agree on the first $\varphi(L_A(f(x))) = \varphi(L_A(x))$ bits, which implies that $(f(x), f(y)) \notin E_{\psi,A}$.

(ii) Assume $E_{\psi,A} \leq_{\text{eq}} E_{\varphi,A}$. Let f be the strong reduction function from $E_{\psi,A}$ to $E_{\varphi,A}$. Fix $n \geq 1$. Let $N_\varphi(n)$ be the number of $E_{\varphi,A}$-equivalence classes which consist of strings x with $L_A(x) = n$ and define $N_\psi(n)$ analogously . It is easy to see that $N_\varphi(n) = 2^{\varphi(n)}$ and $N_\psi(n) = 2^{\psi(n)}$. Since f is a strong reduction function, f is a one to one function on the equivalence classes of $E_{\psi,A}$. Thus the $N_\psi(n)$ many distinct $E_{\psi,A}$-equivalence classes are mapped via f to $N_\psi(n)$ many distinct $E_{\varphi,A}$-equivalence classes. Let

$$S(n) := \{x \in \Sigma^* \mid |x| = a_n\}.$$

It is easy to see that for each $x \in S(n)$ we have $L_A(x) = n$, and for any $y \in \Sigma^*$ with $L_A(y) = n$, there is $x \in S(n)$ with $(x,y) \in E_{\psi,A}$ (since $\psi(n) \leq a_n$), i.e. every $E_{\psi,A}$-equivalence class contains an element of $S(n)$. Let $p \in \mathbb{N}[x]$ be the polynomial time bound of f, i.e. $|f(x)| \leq p(|x|)$ for all $x \in \Sigma^*$ Without loss of generality, we can assume that p and is monotone increasing and $p(n) \geq n$ for all n. Considering the $E_{\varphi,A}$-equivalence classes $\{[f(x)] \mid x \in S(n)\}$, we obtain

$$N_\psi(n) \leq \sum_{k \leq p(a_n)} N_\varphi(k) \leq p(a_n) N_\varphi(p(a_n)).$$

The second inequality follows from the hypothesis that φ and p are increasing. Taking logarithm on both ends yield

$$\psi(n) \leq \varphi(p(a_n)) + O(\log a_n).$$

Since $\varphi(n) = \Omega(\log a_n)$, we have $\psi(n) = O(\varphi(p(a_n)))$, which contradicts our assumption.

(iii) Suppose that $a_n \leq b_n$ for all $n \geq 1$. Therefore $L_B(x) \leq L_A(x)$ for all $x \in \Sigma^*$ and, if $L_B(x) > 0$, then $a_{L_B(x)} \leq a_{L_A(x)} \leq |x|$. We define

$$f(x) = \begin{cases} \varepsilon & \text{if } L_B(x) = 0 \\ x \upharpoonright a_{L_B(x)} & \text{otherwise.} \end{cases}$$

Clearly f is polynomial time computable, since $L_A(x)$ and $L_B(x)$ are computable in time polynomial in $|x|$ and we obtain $x \upharpoonright a_{L_B(x)}$ by checking on all successive truncations of x. We only need to verify that f is a strong equivalence reduction

function. Suppose that $(x,y) \in E_{\varphi,B}$ and, therefore $L_B(x) = L_B(y) > 0$. Hence x and y agree on the first $\varphi(L_B(x))$ bits. By definition of f, $f(x)$ and $f(y)$ both have length $a_{L_B(x)}$. We have $L_A(f(x)) = L_B(x)$ and $L_A(f(y)) = L_B(y)$. Therefore, $L_A(f(x)) = L_A(f(y))$ and $f(x)$ and $f(y)$ agree on the first $\varphi(L_A(f(x))) = \varphi(L_B(x))$ bits, so $(f(x), f(y)) \in E_{\varphi,A}$.
Suppose on the other hand that $(x,y) \notin E_{\varphi,B}$. Then either $L_B(x) \neq L_B(y)$ or $L_B(x) = L_B(y) > 0$ but x and y do not agree on the first $\varphi(L_B(x))$ bits. Assume that $L_B(x) \neq L_B(y)$, then we have $L_A(f(x)) \neq L_A(f(y))$ since $L_A(f(x)) = L_B(x)$ and $L_A(f(y)) = L_B(y)$. Now assume that $L_B(x) = L_B(y)$ but x and y do not agree on the first $\varphi(L_B(x))$ bits, then $L_A(f(x)) = L_A(f(y)) > 0$ but $f(x)$ and $f(y)$ do not agree on the first $\varphi(L_A(f(x)))$ bits, since $f(x) = x \restriction a_{L_B(x)}$ and $f(y) = y \restriction a_{L_B(x)}$, and $\varphi(L_A(f(x))) \leq \varphi(L_B(x)) \leq a_{L_B(x)}$. In both cases $(f(x), f(y)) \notin E_{\varphi,A}$. Therefore f is a strong equivalence reduction from $E_{\varphi,B}$ to $E_{\varphi,A}$. □

For the rest of the section we look at applications of Theorem 4.1.6 with a fixed good set $A = \mathbb{N} \setminus \{0\}$. Thus $a_n = n$ in the enumeration of A and $L_A(x) = |x|$ for all $x \in \Sigma^*$. For simplicity, let $E_\varphi := E_{\varphi,A}$ for this particular A.
To sum it up, let $\varphi : \mathbb{N} \to \mathbb{N}$ be increasing and polynomial time computable such that $\varphi(n) \leq n$ for all $n \in \mathbb{N}$, and E_φ the equivalence relation defined as follows:

$$E_\varphi = \{(x,y) \mid |x| = |y| \text{ and } x \restriction \varphi(|x|) = y \restriction \varphi(|x|)\}.$$

It is easy to see from the definition that $E_\lambda \leq_{\text{eq}} E_\varphi \leq_{\text{eq}} \text{id}$. We are now ready to construct an infinite ascending chain of equivalence relations between E_λ and id.

Proposition 4.1.7
There is an infinite sequence $(\varphi_m)_{m\geq 1} : \Sigma^* \to \Sigma^*$ of polynomial time computable functions such that $E_{\varphi_m} \lneq_{\text{eq}} E_{\varphi_{m+1}}$ for all $m \geq 1$.

Proof.
Define $\varphi_m(n) := \min\{n, (\log n)^m\}$ for each $m \geq 1$. Then each φ_m is computable in polynomial time and

$$\log n \leq \varphi_m(n) \leq n \qquad \forall n \in \mathbb{N}.$$

Since $\varphi_m(n) \leq \varphi_{m+1}(n)$ for all n we can apply Theorem 4.1.6 (i) and get $E_{\varphi_m} \leq_{\text{eq}} E_{\varphi_{m+1}}$. On the other hand $\varphi_m(p(n)) = \Theta((\log n)^m)$ for every $p \in \mathbb{N}[x]$. Since $(\log n)^{m+1} \neq O((\log n)^m)$, we have that $\varphi_{m+1}(n) \neq O(\varphi(p(n))$. Therefore we can apply Theorem 4.1.6 (2) and get $E_{\varphi_{m+1}} \not\leq_{\text{eq}} E_{\varphi_m}$. □

One can easily verify that the construction of φ_m in the above proof can be modified to obtain longer chains, e.g. we could use functions such as $\log n(\log\log n)^k$, $\log n(\log\log n)(\log\log\log n)^k$, etc. .With this construction one can squeeze more infinite ascending sequences between those in the proof above. As a consequence, one can even embed the linear order of the ordinal ω^k, for any finite k, into the degree of strong equivalence reduction between E_λ and id.

To prove the next statement we need a function with certain properties.

Definition 4.1.8 (Cantor pairing function)
The function $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \langle m_1, m_2 \rangle \mapsto \frac{1}{2}(m_1 + m_2)(m_1 + m_2 + 1) + m_2$ is called the *Cantor pairing function*.

Remark 4.1.9 The Cantor pairing function π has the following properties:

(i) π is computable in polynomial time.

(ii) π is strictly increasing in both arguments, i.e. $\pi(m_1 + 1, m_2) > \pi(m_1, m_2)$ and $\pi(m_1, m_2 + 1) > \pi(m_1, m_2)$ for all $m_1, m_2 \in \mathbb{N}$

(iii) π is bijective.

(iv) Let $k := \pi(m_1, m_2)$. The decoding functions

$$(\cdot)_1 = \begin{cases} \mathbb{N} \to \mathbb{N} \\ k \mapsto m_1 \end{cases} \qquad \text{and} \qquad (\cdot)_2 = \begin{cases} \mathbb{N} \to \mathbb{N} \\ k \mapsto m_2 \end{cases}$$

are computable in polynomial time.

We will now construct an infinite antichain of equivalence relations between E_λ and id, i.e. we will construct infinitely many pairwise incomparable equivalence relations.

Proposition 4.1.10
There is an infinite sequence $(\varphi_m)_{m \in \mathbb{N}} : \Sigma^* \to \Sigma^*$ of polynomial time computable functions such that $E_{\varphi_m} \not\leq_{\text{eq}} E_{\varphi_{m'}}$ for any $m \neq m'$.

Proof.
To prove the statement we construct a sequence $(\varphi_m)_{m \in \mathbb{N}}$ of functions in a way such that we can apply Theorem 4.1.6 (ii). Let $(N_k)_{k \in \mathbb{N}}$ be an increasing sequence of natural numbers defined by induction as follows:

$$\begin{aligned} N_0 &= 1, \\ N_{k+1} &= 2^{1+(\log N_k)^2} \end{aligned}$$

Define φ_m for every $m \in \mathbb{N}$ as follows:

$$\varphi_m(n) = \begin{cases} \min\{n, (\log n)^2\} & \text{if } N_k \leq n < N_{k+1} \text{ and } (k)_1 = m, \\ \min\{n, (\log N_k)^2\} & \text{if } N_k \leq n < N_{k+1} \text{ and } (k)_1 \neq m, \end{cases} \tag{4.1}$$

where $(\cdot)_1$ is the first decoding function of the Cantor pairing function. Clearly, each φ_m is computable in polynomial time. It is easy to see that each φ_m is increasing since each φ_m is composed of taking the minimum of (strictly) increasing functions.
To see that $\varphi_m(n) = \Omega(\log a_n)$, first note that $a_n = n$ since $A = \mathbb{N} \setminus \{0\}$. Therefore it suffices to show that $\varphi_m(n) = \Omega(\log n)$. To see this we fix m. In the case that $(k)_1 = m$, we have $\varphi_m(n) = (\log n)^2 = \Omega(\log n)$ for large enough n. If on the other hand $(k)_1 \neq m$, we have $\log N_{k+1} = O(\varphi_m(N_k))$ since asymptotically, $\varphi_m(N_k) = (\log N_k)^2$

and $\log N_{k+1} = 1 + (\log N_k)^2$ (by definition of N_{k+1}).
To complete the proof we need to show that, if $m \neq m'$, then $\varphi_m(n) \neq O(\varphi_{m'}(p(n)))$ for any $p \in \mathbb{N}[x]$. Let $d = \deg(p)$, then

$$O\left(\varphi_{m'}(p(n))\right) = O\left((\log p(n))^2\right) = O\left((\log n^d)^2\right) = O\left(d^2 \cdot (\log n)^2\right) = O\left(\varphi_{m'}(n)\right).$$

Therefore it suffices to show that $\varphi_m(n) \neq O(\varphi_{m'}(n))$ for $m \neq m'$. Suppose the contraposition, i.e. assume $\varphi_m(n) = O(\varphi_{m'}(n))$. Then there is some constant $C > 0$ and $N \in \mathbb{N}$, such that

$$\varphi_m(n) \leq C\varphi_{m'}(n) \qquad \forall n \geq N.$$

Suppose N is large enough that $(\log n)^2 \leq n$ for all $n \geq N$. It follows that for all k such that $N_k \geq N$,

$$\varphi_m(N_{k+1} - 1) \leq C\varphi_{m'}(N_{k+1} - 1).$$

If additionally $(k)_1 = m$, then by (4.1) we have

$$\varphi_m(N_{k+1} - 1) = (\log(N_{k+1} - 1))^2 \leq C(\log N_k)^2.$$

This would imply that, as a function in k, we have $\log(N_{k+1} - 1) = O(\log N_k)$. This is impossible since there are infinitely many k with $(k)_1 = m$ and

$$\log(N_{k+1} - 1) = \Theta(\log N_{k+1}) = \Theta\left(1 + (\log N_k)^2\right) = \Theta\left((\log N_k)^2\right) \neq O(\log N_k).$$

This is a contradiction to $\varphi_m(n) = O(\varphi_{m'}(n))$ and therefore concludes our proof. □

The functions we constructed in the proof above have an interesting property that we will now explore.

Corollary 4.1.11
There is an assignment $X \mapsto E_X$, from all polynomial time computable subsets of $\mathbb{N}$ into the equivalence relations between E_λ and id such that

$$X \subseteq Y \iff E_X \leq_{\text{eq}} E_Y.$$

Proof.
Let the functions φ_m be as in the previous proof. For each polynomial-time computable $X \subseteq \mathbb{N}$, we define

$$\varphi_X(n) := \max\{\varphi_m(n) \mid m \in X\} \quad \text{and} \quad E_X := E_{\varphi_X}.$$

It is easy to verify that each φ_X is polynomial-time computable, since

$$\varphi_X(n) = \begin{cases} \min\{n, (\log n)^2\} & \text{if } N_k \leq n < N_{k+1} \text{ and } (k)_1 \in X, \\ \min\{n, (\log N_k)^2\} & \text{if } N_k \leq n < N_{k+1} \text{ and } (k)_1 \notin X. \end{cases}$$

For $X \subseteq Y \subseteq \mathbb{N}$ and $X, Y \in \mathrm{P}$, we have $\varphi_X(n) \leq \varphi_Y(n)$ for all $n \in \mathbb{N}$ and therefore, by Theorem 4.1.6 (i), we have $E_X \leq_{\text{eq}} E_Y$. On the other hand if $X \not\subseteq Y$, then there exists $m \in X \setminus Y$ and therefore $E_{\varphi_m} \not\leq_{\text{eq}} E_Y$ similar as in Proposition 4.1.10.

Corollary 4.1.12
Any finite partial order can be embedded into the the relation $\leq_{\text{eq}}$ between E_λ and *id*.

Proof.
We have seen in Corollary 4.1.11 that every finite partial order can be embedded into a finite Boolean algebra. We also showed in Corolarry 4.1.11 that any finite Boolean algebra is embeddable into the degrees of strong equivalence reductions between E_λ and *id*. Therefore our claim follows. □

4.1.2 The hierarchy of $\leq_{\text{eq}}$

In this section we will look at isomorphism relations for finite structures as discussed in Chapter 2 as well as the new established benchmark relations and show how they are related.
We also discuss another interesting equivalence relation:

Definition 4.1.13 (Clique relation)
Let $x, y \in \Sigma^*$ be the encodings of finite graphs. The *clique relation*, denoted CLIQ is defined as follows:

$$(x, y) \in \text{CLIQ} \iff \text{the maximal cliques in } x \text{ and } y \text{ have the same size.}$$

It was shown in [11] that CLIQ is DP-complete.

Definition 4.1.14 (Complexity class DP)
The complexity class DP is defined as follows:

$$\text{DP} := \{L \mid L = L_1 \cap L_2 \text{ such that } L_1 \in \text{NP and } L_2 \in \text{co-NP}\}.$$

Note that

$$\text{P} \subseteq \text{NP} \cap \text{co-NP} \subseteq \text{NP} \cup \text{co-NP} \subseteq \text{DP}.$$

We will collect results from earlier as well as some new ones and then give an overview of the structure of strong equivalence reduction.

Proposition 4.1.15

(i) Let E be the isomorphism relation of a class of finite structures, then $E \leq_{\text{eq}}$ GRAPH.

(ii) LOU $\equiv_{\text{eq}}$ id.

(iii) Let E be the isomorphism relation of one of the following classes

$$\text{SET}, \text{ORD}, \text{LOP}, \text{ABELIAN}, \text{CYCLIC or FIELD},$$

then $E \equiv_{\text{eq}} E_\lambda$.

Proof.

(i) See Proposition 2.1.4.

(ii) See Remark 1.3.2.

(iii) Follows from Example 2.2.2(ii), (iii) and Corollary 2.1.15. □

Proposition 4.1.16

(i) BOOLE $\leq_{\text{eq}} E_\lambda \leq_{\text{eq}}$ GROUP $\leq_{\text{eq}}$ GRAPH.

(ii) id $\not\leq_{\text{eq}}$ GROUP

(iii) $E_\sigma \leq_{\text{eq}}$ BOOLE

(iv) $E_\lambda \leq_{\text{eq}}$ CLIQ

(v) id $\not\leq_{\text{eq}}$ CLIQ

(vi) GRAPH $\not\leq_{\text{eq}}$ CLIQ

Proof.

(i) Follows from Remark 2.1.14 and Proposition 2.2.6.

(ii) Note that $\#\text{id}(n) = \sum_{k=0}^{n} 2^k = 2^{n+1} - 1$. Therefore by Corollary 2.5.7 the claim follows.

(iii) Also follows from Corollary 2.5.7 since $\#\text{BOOLE}(n) = \lfloor \log n \rfloor = \Theta(\log n)$ and $\#E_\sigma(n) < \log \log n$.

(iv) This is witnessed by the strong reduction function $x \mapsto K_{|x|}$, where $K_{|x|}$ is the complete graph of order $|x|$.

(v) Note that $\#\text{CLIQ}(n)$. Thus by Corollary 2.5.7 the claim follows.

(vi) Since id $\leq_{\text{eq}}$ GRAPH, it follows that GRAPH $\not\leq_{\text{eq}}$ CLIQ. □

We summarize the results of reducibility and non-reducibility in Figure 4.1 and use the convention that
C $\dashrightarrow$ D if C $\leq_{\text{iso}}$ D but it is no known whether D $\leq_{\text{iso}}$ C or not,
C $\mapsto$ D if C $\leq_{\text{iso}}$ D but D $\not\leq_{\text{iso}}$ C, and
C $\leftrightarrow$ D if C $\equiv_{\text{iso}}$ D.

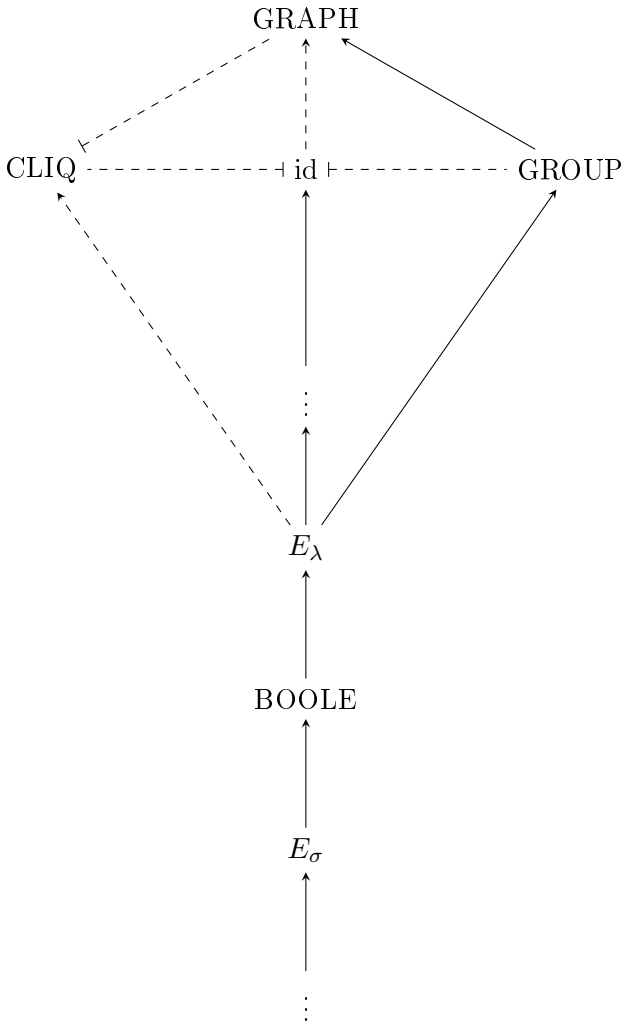

Figure 4.1: Structure of Strong Equivalence Degrees

So far, none of the statements in this chapter required any knowledge on the relations of P, NP and co-NP.
We will collect some of those statements here:

Proposition 4.1.17 The following are equivalent

(i) P=NP

(ii) CLIQ $\leq_{\text{eq}} E_\lambda$

(iii) CLIQ $\equiv_{\text{eq}} E_\lambda$

Proof. Clearly (ii) and (iii) are equivalent since we already showed that $E_\lambda \leq_{\text{eq}}$ CLIQ. To see that (i) $\Rightarrow$ (ii), we assume that P=NP. Then, given a finite graph G, there is a

polynomial time algorithm to determine, whether G contains a clique of size k. Let n be the number of vertices of G. By setting $k = 2, \ldots, n$, we can determine the maximal k such that G has a clique of size k in polynomial time. By running this algorithm and outputting a string with length of this maximal k we get a strong reduction function from CLIQ to E_λ. To show that $(ii) \Rightarrow (i)$, note that NP$\subseteq$DP and that CLIQ is DP-complete as a set. If there is a polynomial time computable strong reduction function f : CLIQ $\leq_{\text{eq}} E_\lambda$, then CLIQ $\in$P, as well as every DP set. □

This means if P = NP, then CLIQ is of the same complexity as E_λ and thus not a separate benchmark as shown above.

Proposition 4.1.18 If P $\neq$ NP, then CLIQ $\not\leq_{\text{eq}}$ id.

Proof. CLIQ $\leq_{\text{eq}}$ id would imply that CLIQ $\in$P. □

Proposition 4.1.19 If NP $\neq$ co-NP, then CLIQ $\not\leq_{\text{eq}}$ GRAPH, and in particular CLIQ $\not\leq_{\text{eq}}$ id and CLIQ $\not\leq_{\text{eq}}$ GROUP.

Proof. CLIQ $\leq_{\text{eq}}$ GRAPH would imply that CLIQ $\in$NP. Since co-NP $\subseteq$ DP, it would follow that NP = co-NP = DP. □

Corollary 4.1.20 If P=NP, then $E \leq_{\text{eq}}$ id for every $E \in$NP.

4.2 Finitary equivalence relations

In this section we show that there is an initial segment of the $\leq_{\text{eq}}$ hierarchy consisting of certain P equivalence relations. We also show that if P$\neq$NP, then there are NP equivalence relations strictly above id. We obtain these results by considering a special class of equivalence relations:

Definition 4.2.1 (Finitary equivalence relation)
An equivalence relation E is said to be *finitary* if E has only finitely many non-trivial equivalence classes.

We will state the following proposition without proof:

Proposition 4.2.2 Let E be a finitary equivalence relation on Σ^*, then the following are equivalent:

(i) $E \in$ P,

(ii) Each equivalence class of E is in P, and

(iii) $E \leq_{\text{eq}}$ id.

We will consider two subclasses of finitary equivalence relations. The first class consists of equivalence relations with only finitely many equivalence classes. The canonical example

is the following:
For $n \in \mathbb{N} \setminus \{0\}$ let $\equiv_n$ denote the congruence relation mod n, i.e.

$$x \equiv_n y \iff x \equiv y \bmod n,$$

for $x, y \in \mathbb{N}$.
Clearly $\equiv_n \leq_{\text{eq}} \equiv_{n+1} \leq_{\text{eq}}$ id for all n. Moreover, if $E \leq_{\text{eq}} \equiv_n$ for any equivalence relation E, then $E \in$P and E has at most n equivalence classes.
We state the following without proof.

Proposition 4.2.3 Let E be an equivalence relation on Σ^* with exactly n equivalence classes. Then the following are equivalent:

(i) $E \in$ P.

(ii) There is a polynomial time computable equivalence relation F with infinitely many equivalence classes such that $E \leq_{\text{eq}} F$.

(iii) For any equivalence relation F with at least n equivalence classes, we have $E \leq_{\text{eq}} F$.

(iv) $E \equiv_{\text{eq}} (\equiv_n)$.

Therefore the P equivalence relation with finitely many equivalence classes are an initial segment of the $\leq_{\text{eq}}$ hierarchy.
We now consider the next subclass of finitary equivalence relations, namely equivalence relations induced by a single set.

Definition 4.2.4 For any subset $S \subseteq \Sigma^*$, we define an equivalence relation R_S on Σ^* as follows:

$$R_S := \{(x, y) \mid x = y \text{ or } x, y \in S\}.$$

Equivalence relations that are constructed in this way are clearly finitary since their only non-trivial equivalence class is generated by S, more precisely it is the set equivalence class $S \times S$.
The following lemma is an observation about their mutual reducibility.

Lemma 4.2.5 Let $S, T \subset \Sigma^*$. If $R_S \leq_{\text{eq}} R_T$ then either $S \leq_p T$ as sets or $R_S \leq_{\text{eq}}$ id.

Proof. Suppose that $f : R_S \leq_{\text{eq}} R_T$ is a srong equivalence reduction. Then for any $x, y \in S$ we have $(f(x), f(y)) \in R_T$. There are two cases.
Case 1 For any $x \in S$ we have $f(x) \in T$ (and of course for $x \notin S, f(x) \notin T$). Thus f is polynomial time reduction from S to T.
Case 2 For any $x \in S$ we have $f(x) \notin T$. In this case $f(x) = f(y)$ for all $x, y \in S$. But then clearly R_S is strong equivalence reducible to id, in fact $f : R_S \leq_{\text{eq}}$ id. □

Remark 4.2.6

(i) The relation id itself is of the form $R_\emptyset$.

(ii) There exists a co-infinite set $S \subseteq \Sigma^*$ such that $R_S \lneq_{\text{eq}} \text{id}$, e.g.

$$S = \{x \in \Sigma^* \mid x(i) \neq 0 \text{ for some } i < |x|\}.$$

Proposition 4.2.7 Let $\emptyset \neq S \subseteq \Sigma^*$. Then the following hold:

(i) $S \in$NP $\Leftrightarrow R_S \in$NP.

(ii) If S is NP-hard, then R_S is NP-hard as a set.

(iii) If $R_T \leq_{\text{eq}} R_S$ for all $T \subseteq \Sigma^*$ with $T \in$NP, then S is NP-hard.

(iv) If P$\neq$NP and S is NP-hard, then $R_S \not\leq_{\text{eq}} \text{id}$.

Proof. The ($\Rightarrow$) direction of (i) follows from the definition. For ($\Leftarrow$) of (i) and (ii) we fix $a \in S$ and note that for any $x \in \Sigma^*$, we have that $x \in S \iff (x, a) \in R_S$. Therefore the mapping $x \mapsto (x, a)$ is a polynomial time reduction from S to R_S as sets. This reduction and $R_S \in$NP together imply that $S \in$ NP while this reduction and the fact that S is NP-hard imply that R_S is NP-hard. Now (iv) follows from (ii), since otherwise id would be NP-hard which would imply P = NP. Finally, to prove (iii), we assume $R_T \leq_{\text{eq}} R_S$ for all NP subsets $T \subseteq \Sigma^*$. By Remark 4.2.6 we know that $\text{id} \leq_{\text{eq}} R_S$ and it follows that S is co-infinite?. We now look at two cases. If P = NP, then any nonempty, proper subset of Σ^* is NP-hard. Suppose now that P $\neq$ NP. Since the statement must hold for all $T \in$ NP we can assume that T is NP-hard. Then $R_T \not\leq_{\text{eq}} \text{id}$ by (iv). Therefore by Lemma 4.2.5 there must by a polynomial time reduction from T to S, and hence S is NP-hard. □

We will define a notation that is dual to the notion of finitary equivalence relation.

Definition 4.2.8 (Finite equivalence relation)
We call an equivalence relation E *finite* if every equivalence class of E is finite.

Remark 4.2.9 The isomorphism relation for finite structures, e.g. GRAPH, and the relations $E_{\varphi,A}$ that we defined earlier are finite.

We will show that the notion of finite equivalence relations and the notion of finitary equivalence relations are in a sense orthogonal in terms of reducibility.

Proposition 4.2.10 Let E, F be equivalence relations on Σ^*. Furthermore, assume that E is finitary and F is finite. Then the following hold:

(i) If $E \leq_{\text{eq}} F$, then $E \leq_{\text{eq}} \text{id}$ and $E \in$P.

(ii) If $F \leq_{\text{eq}} E$, then $F \leq_{\text{eq}} \text{id}$ and $F \in$P.

Proof. To show (i) we use Proposition 4.2.2. Therefore we only need to show that every E-equivalence class is in P. Since $E \leq_{\text{eq}} F$, there must be a polynomial time strong equivalence reduction function which maps the equivalence classes of E to the equivalence classes of F. Since the latter one are finite and thus in P, the equivalence classes of E

also lie in P.
For (ii),let f be the strong equivalence reduction function from F to E. Let

$$X = \{x \in \Sigma^* \mid [x]_E = \{x\}\}.$$

We now define a strong reduction from F to id in the following way: Let $g(x) := f(x)$ if $x \in f^{-1}(X)$ and let $g(x)$ the lexicographical first element of the E-equivalence class of $f(x)$ if $x \notin f^{-1}(X)$. It is easy to see that g is a strong equivalence reduction function from F to id. □

We now define a finite equivalence relation induced by a single set.

Definition 4.2.11 For any $S \subseteq \Sigma^*$, we define an equivalence relation

$$D_S := \{(x, y) \mid x = y \text{ or } x \restriction (|x| - 1) = y \restriction (|y| - 1) \in S\}.$$

Clearly D_S is a finite equivalence relation since Σ is finite. Note that D_S does not have finitely many equivalence classes.

Proposition 4.2.12 Let $S \subseteq \Sigma^*$. The following are equivalent:

(i) $D_S \in \mathrm{P}$,

(ii) $S \in \mathrm{P}$,

(iii) $D_S \leq_{\mathrm{eq}} \mathrm{id}$,

(iv) $D_S \equiv_{\mathrm{eq}} \mathrm{id}$.

Proof.
(i) ⇔ (iii) follows from Proposition 4.2.2.
(ii) ⇒ (i) and (iv) ⇒ (iii) is trivial.
(iii) ⇒ (iv) follows from the fact that for every $S \subseteq \Sigma^*$ the mapping $x \mapsto x^\frown 0$ is a strong reduction from id to D_S.
(i) ⇒ (ii) For any $x \in \Sigma^*$,

$$x \in S \iff (x^\frown 0, x^\frown 1) \in D_S,$$

and therefore if $D_S \in \mathrm{P}$ then $S \in \mathrm{P}$.

Corollary 4.2.13 Let $S \subseteq \Sigma^*$. Then the following hold:

(i) $S \in \mathrm{NP} \Leftrightarrow D_S \in \mathrm{NP}$.

(ii) If S is NP-hard then D_S is NP-hard as a set.

(iii) If $\mathrm{P} \neq \mathrm{NP}$ and S is NP-hard, then $\mathrm{id} \lneq_{\mathrm{eq}} D_S$.

Open Questions

At this time there are still many open questions concerning the relative complexity of different equivalence relations. We will present a few selected ones related to the content of this thesis.

It is well known that there is a logic capturing P on finite ordered structures (Immerman-Vardi Theorem) but it is a major open problem in descriptive complexity if there is an analogous logic for all finite structures.

Open Question 1 Is there a logic capturing P on all finite structures?

We have seen that the separation of $\leq_{\text{iso}}$ and $\leq_{\text{pot}}$ would have big consequences in complexity theory but we were not able to separate them without strong assumptions.

Open Question 2 Is it possible to separate the relations of $\leq_{\text{iso}}$ and $\leq_{\text{pot}}$ without any complexity theoretical assumptions.

As we have seen numerous times it is possible to prove reducibility results about $\leq_{\text{eq}}$ without assumptions on the relationship between complexity classes. We call these results *absolute*. The following open problems might have absolute answers.

Open Question 3 Does $E \leq_{\text{eq}} \text{id}$ for all $E \in \text{P}$?

Open Question 4 Does $E \leq_{\text{eq}} \text{GRAPH}$ for all finite $E \in \text{NP}$?

Bibliography

[1] Andreas Blass and Yuri Gurevich. *Equivalence relations, invariants, and normal forms, II*, pages 24–42. 01 1970.

[2] Andreas Blass and Yuri Gurevich. Equivalence relations, invariants, and normal forms, ii. In E. Börger, G. Hasenjaeger, and D. Rödding, editors, *Logic and Machines: Decision Problems and Complexity*, pages 24–42, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.

[3] Sam Buss, Yijia Chen, Jörg Flum, Sy-David Friedman, and Moritz Müller. Strong isomorphism reductions in complexity theory. *J. Symbolic Logic*, 76(4):1381–1402, 2011.

[4] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory.* Perspectives in Mathematical Logic. Springer, 1995.

[5] Lance Fortnow and Joshua A. Grochow. Complexity classes of equivalence problems revisited. *CoRR*, abs/0907.4775, 2009.

[6] Su Gao, Caleb Ziegler, et al. On polynomial-time relation reducibility. *Notre Dame Journal of Formal Logic*, 58(2):271–285, 2017.

[7] Steven Givant. *Atomless Boolean Algebras*, pages 134–141. Springer New York, New York, NY, 2009.

[8] Erich Grädel. *Finite model theory and its applications.* Springer, Berlin; New York, 2007.

[9] Leonid Libkin. *Elements of Finite Model Theory.* Springer, August 2004.

[10] Moritz Müller. Introduction to theoretical computer science, 2016.

[11] C.H. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984.

[12] L. Pyber. Asymptotic results for permutation groups. In *Groups And Computation*, 1991.

[13] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, October 1976.

Index