Effective Axiomatizations of Simple Ordered Structures

Jackson Meyer-Lee

April 30, 2021
Version: Final Draft
Advisor: Steven Lindell

Abstract

Axiomatization in finite model theory is the process of finding a set of axiom formulas from which all other formulas in a theory can be derived. Applied to the theory of a structure like “binary strings” or “natural number arithmetic,” this allows us to find the precise minimal mathematical definition of that structure. This has consequences on the computability of a structure; if a theory has a recursively enumerable axiomatization, then it is itself recursively enumerable. For example, the theory of binary strings can be axiomatized using just the axioms of a linear order, definable induction, and a maximal element. To do this, a technique called an “Ehrenfeucht-Fraisse game" is required, which shows that different models are equivalent for all formulas with at most n unique variables.

However, the pigeonhole principle is logically equivalent to definable induction, so anything which can be derived from definable induction can be derived from the pigeonhole principle. Thus, it should be possible to axiomatize the theory of binary strings with the pigeonhole principle instead of definable induction, with interesting consequences and extensions as the pigeonhole principle (unlike induction) applies to sets without a linear order. For example, the same axiomatization technique can be applied to other ordered structures, like trees, which have a very different form of order.

Contents

1

Introduction

1.1 Preliminaries

Model Theory lies within the wider area of mathematical logic, an overlapping subfield of mathematics and computer science. Mathematical logic is concerned with reasoning about logic itself, such as figuring out what can or can't be proven in which logic systems. However, the results of these abstract branches bear meaningful results on our ability to understand the problems and structures that we work with in computer science. By effectively axiomatizing these structures, we gain insight into their fundamental properties as well as whether problems involving them are necessarily solvable.

Model theory is concerned with two main components of logical languages: the set of valid symbols in the logic, known as the grammar, and meaning or interpretation of the symbols, called the semantics. A particular grammar in model theory is known as a *signature*, whereas a particular interpretation of a signature is a *structure*. A signature generally consists of constant symbols which stand for specific values, functional symbols which stand for functions that take a number of values to a single value, and relational symbols which stand for a possible relationship between two values. For example, we could consider the signature $\sigma = \{\{0, 1\}, \{+\}, \emptyset\}$: a signature which contains constant symbols 0 and 1 and a binary function symbol $+$, with no relational symbols. These symbols are arranged into sequences known as *formulas*, which are themselves built from discrete building blocks called terms. Terms are any constant symbol, variable, or sequence $f(t_1, \ldots, t_n)$ where f is an n-ary function symbol and t_i are terms. For example, on the signature σ above, 0, $+(0, 1)$, and $+(0, +(0, 0))$ are all terms, although $+(x, y)$ would most commonly be written with the notation $(x + y)$ for familiarity. The most simple of formulas, known

2

as atomic formulas, either are an n-ary relation on n terms or the formula $t_1 = t_2$ for any two terms t_1 and t_2. For σ, $0 = (0 + 1)$ is an atomic formula. More complex formulas are constructed out of propositional symbols like $\neg, \wedge, \vee$ and quantifiers like $\forall x$ and $\exists y$ in combination with atomic formulas. A formula with no variables except those specified by a quantifier is said to have no free variables, and called a *sentence*. In the case of σ, a sentence might look something like $\varphi = (\forall x)(\neg(0 = x + 1))$. In a standard structure, this sentence φ would be interpreted as saying "there is no number x such that $x + 1 = 0$." A structure grants a truth value to a sentence by granting a meaning to the symbols involved; if a sentence φ is true in a structure S, then S *satisfies* φ and S is said to be a *model* of φ denoted $S \vDash s$. We can consider the naturals $\mathcal{N}$ and integers $\mathcal{Z}$ as structures on the signature σ mentioned above. Then, the sentence φ would be true in the natural numbers but not in the integers, so we would write $\mathcal{N} \vDash \varphi$ and $\mathcal{Z} \nvDash \varphi$. A set of sentences is known as a *theory*, and a structure is a model of a theory if it satisfies every sentence in the theory. Similarly, a sentence or theory is "satisfiable" if there exists a model that makes the sentence (or every sentence of a theory) true.

Structures themselves consist of a domain, or set of possible values for terms, as well as definitions for functional and relational symbols. Finite model theory is concerned with structures that possess finite domain. This is notably different from classical model theory, as many major theorems of classical model theory fail for finite structures, such as the Compactness Theorem and Godel's Completeness Theorem. (Libkin 2013) Within this paper, however, we will study classes of finite structures as a whole instead of individual finite structures; for example, we will study the whole class of binary strings rather than any particular binary string, meaning a structure that encapsulates the idea of "a binary string" rather than any particular binary string structure. Thus, at times we will use psuedofinite structures to characterize a class, structures which have infinite domain but also retain the property that any sentence which they model is also modeled by a finite structure. Thus, they cannot possess first order definable properties which inherently involve their infiniteness.

As a practical example of a structure and a signature, one can consider the natural numbers $\mathcal{N}$ as a structure on the signature $\sigma = \{\{0, 1\}, \{+\}, \emptyset\}$ by interpreting 0 and 1 as the respective natural numbers and $+$ as the standard addition. Thus, $(\forall x)(x + 1 = 1 + x)$ is satisfied by the natural numbers. In fact, every true first order sentence about the natural numbers with addition is somewhat intuitively satisfied by $\mathcal{N}$ with the standard definitions. This set of every true first order sentence about natural number addition is known as Presburger arithmetic. It

can be *effectively axiomatized*, meaning that there is a recursive enumerable set of sentences from which every sentence in the theory of natural numbers can be derived. This recursively enumerable set of axioms is known as the axioms of Presburger arithmetic.(Stansifer 1984).

Effective Axiomatization is an important task in Finite Model Theory. When working with finite structures that so often appear in Computer science, an effective axiomatization of a structure implies that the theory of the structure is decidable, meaning that it's possible for a computer to programmatically solve any problem about that structure which can be stated in first order logic.

1.2 Background

1.2.1 Binary Strings

Binary strings are a particular example structure that is useful for the axiomatizations studied in this paper. A binary string can be seen as a unary predicate on a finite linear order; essentially, this is a function that outputs 1 or 0 operating on a sequence of digits that has a start and an end: the binary string 1001 can be seen as a structure with domain $\{v_1, v_2, v_3, v_4\}$ operating on the signature with a binary relation $<$ and a unary predicate (relation) P, defining the linear ordering $v_i < v_j$ if $i < j$, and defining the unary predicate P as $P = \{v_1, v_4\}$ (meaning that $P(v_1)$ is true and $P(v_3)$ is false). However, as mentioned above, it's much more interesting to discuss a class of finite structures rather than a particular finite structure, so we will examine the class of binary strings as a whole rather than a particular binary string. We will refer to the first order sentences defining this class as "the theory of binary strings," which happens to be axiomatizable by the sentences for a linear ordering and the induction principle.

1.2.2 Linear Ordering

A linear ordering is a relation on a set typically denoted by $<$ and defined as a transitive, trichotomous binary relation, meaning that it is transitive and for all elements a, b in the set, exactly one of $a < b$, $b < a$, or $a = b$ is true. Axiomatically, it

can be defined by exactly the sentences for trichotomy and transitivity.
Trichotomy:

$$\begin{aligned}(\forall x)((\forall y)((x = y \wedge \neg(x < y) \wedge \neg(y < x)) \\ \vee (\neg(x = y) \wedge x < y \wedge \neg(y < x)) \\ \vee (\neg(x = y) \wedge \neg(x < y) \wedge y < x))\end{aligned}$$

Transitivity:

$$(\forall x)((\forall y)((\forall z)(x < y) \wedge (y < z) \rightarrow (x < z)))$$

As these sentences are the axioms of a linear ordering, any model of these sentences contains a definition of $<$ that obeys the properties any linear order.

1.2.3 Well-foundedness

Well-foundedness is a property of binary relations on a set, typically examined in relation to orderings like linear orderings. It is the property that there is some minimal element of the set; for a well-founded relation $<$ on a nonempty set S, there is some x in S such that there is no $y \in S$ where $x < y$. In other words, there is some x in S with no smaller element. When applied to a linear ordering, this property is called *well-ordering*, and it has stronger consequences: it means that every nonempty subset $A \subseteq S$ has some least element, an element $x \in A$ such that $x < y$ for all $y \in A$ where $x \neq y$. In terms of first order logic, it is presented as

$$(\exists x)S(x) \rightarrow (\exists y)[S(y) \wedge (\forall z < y)\neg S(z)]$$

Meaning that if there exists an x with the property S (such as membership in a set), then there exists a y with the property S such that every z less than y does not have the property S. y is clearly a least element, as for every x with property S, we know that x is not $< y$ as otherwise $\neg S(x)$. By the trichotomy of a linear ordering, either $x = y$ or $y < x$. This "least element" property is often cited as the "well-ordering principle."

1.2.4 Induction Principle

Mathematical Induction is generally thought of as a proof technique, but can also be interpreted as a logical statement. As a proof technique, it means that if you can prove some property is true for zero, and if it is true for some natural n it is true about $n+1$, then the property is true about all natural numbers. However, if we interpret our property as a subset of the naturals (the subset which contains every natural with that property) then induction is the idea that any subset of the natural numbers which contains zero and is closed under successor (the $n+1$ idea) contains all of them.

However, the principle of induction can easily be generalized beyond the naturals to any finite set, as it truly requires little more than a well-ordering to operate. Thus, the only symbol required to define it in a logical language is the well ordering $<$. From there, strong induction can be defined in first order logic as

$$[(\forall x)[(\forall y < x)P(y)] \rightarrow P(x)] \rightarrow (\forall z)P(z).$$

Pieced apart, this first order formula simply states that if it is the case that if for all x some property P holds for all y less than x then it holds at x, then it is also the case that the property holds for universally. The base case of the induction, equivalent to the $P(0)$ case, is actually embedded within this statement. If we consider some least element x of any finite structure with a linear ordering (which exists per the well-ordering principle), the sentence $(\forall y < x)P(y)$ is trivially true. Thus, if $[(\forall x)[(\forall y < x)P(y)] \rightarrow P(x)]$, $P(x)$ must also be true, as otherwise there would exist an x for which the sentence $[(\forall y < x)P(y)] \rightarrow P(x)]$ is false. This reliance upon the existence of some least element x should be carefully noted; the principle of induction on a linear order is closely related to the well-ordering principle. In fact, they are logically equivalent!

To go from the well-foundedness principle to the principle of induction, consider the contrapositive of the well-foundedness principle: If there does not exist a y such that $\neg P(y)$ and for all $z < y$ $P(z)$, then there does not exist an x such that $\neg P(x)$. Passing these through DeMorgan's laws, we find that if for all y either $P(y)$ or there is some $z < y$ such that $\neg P(z)$ (or both), then for all x we have $P(x)$. That premise can be restated through the equivalence of $\varphi \rightarrow \psi$ and $\neg(\varphi) \vee \psi$ as "for all y, if $P(z)$ for all $z < y$ then $P(y)$, which is exactly the premise of the inductive principle. The conclusion already is in the form of the inductive principle, so we have shown that the well foundedness principle is equivalent to the inductive principle. Similarly,

reasoning from induction to the well foundedness principle can be done very quickly through the contrapositive.

1.2.5 Pigeonhole Principle (PHP)

The pigeonhole principle is essentially the core idea of musical chairs; it is impossible to fit n objects into less than n slots. More formally, it is the idea if a set A is finite, and B is a strict subset of A, then any function $f : A \to B$ cannot be injective, meaning that there must be some distinct values $x, y \in A$ such that $f(x) = f(y)$. This seems somewhat intuitive, but interestingly enough is also equivalent to the principle of mathematical induction on finite sets!

Consider the definition of mathematical induction given above, that any subset of the natural numbers which contains 0 and is closed under successor (has the property that if $n \in P$, then $(n + 1) \in P$) contains all natural numbers. Then, From the PHP to mathematical induction::

Let S be a subset of the natural numbers which is closed under successor and includes 0. Suppose, for the sake of contradiction, that there is some $m \in \mathcal{N}$ such that $m \notin S$. Define a function $f : \{0 \dots m\} \to \{0 \dots m - 1\}$ by:

$$f(x) = \begin{cases} x - 1 & x \notin S \\ x & x \in S \end{cases}$$

Suppose that f is not injective; then, there is some $y \in \mathcal{N}$ such that $f(x_1) = y$ and $f(x_2) = y$ but $x_1 \neq x_2$. Note that the only two possibilities are $f(y) = y$ and $f(y + 1) = y$, as f is always either the identity or the predecessor. But if $f(y) = y$, then $y \in S$. By the fact that S is closed under successor, therefore $y+1 \in S$. But then $f(y + 1) = y + 1$, so we know that $f(y + 1) = y$ is not the case. As there can only be one x such that $f(x) = y$, f must be injective. But by the pigeonhole principle, there cannot be an injective function from a finite set to a strict subset. Thus, this contradicts the fact that m exists at all, and thus all $n \in \mathcal{N}$ are in S where S is a subset of the naturals closed under successor and including 0. This is the principle of mathematical induction.

In the other direction, we will show that mathematical induction leads to the pigeonhole principle. Let A and B be finite sets such that $|A| > |B|$, and let

$f : A \to B$. Proceed by induction on $|B|$ to show that f is not injective: when $|B|$=1, B only has 1 element, b. As $|A| > |B|$, there is some $a \in A$ such that $a \notin B$. Thus, $f(a) = b$ and $f(b) = b$ as f takes everything in A to b, but $a \neq b$ as $a \notin B$ but $b \in B$. Thus, f is not injective. In the inductive case, assume that $|B| > 1$ and consider some $b \in B$. If the size of the preimage $|f^{-1}(b)|$ is great than 1, then we immediately find a counterexample to injectivity by looking at two different elements of the preimage. Otherwise, examine $A' = A \setminus f^{-1}(b)$ and $B' = B \setminus \{b\}$ and define $f' : A' \to B'$ as $f'(x) = f(x)$. As $|B'| < |B|$, the inductive hypothesis tells us that f' is not injective. Thus, there is a pair $x, y \in A'$ such that $f'(x) = f'(y)$ but $x \neq y$. By definition of f', we have $f(x) = f(y)$ but $x \neq y$, so f is not injective. Thus, we have shown by induction that any function from a finite set to a strictly smaller (nonempty) set cannot be injective. As proper subsets of finite sets are always strictly smaller, this is equivalent to the pigeonhole principle.

First order logic cannot make quantifier statements over sets with a single formula, so instead the expression of the pigeonhole Principle takes the form of an "axiom schema," an infinite list of axioms defined by a specific pattern of construction. To begin this construction, we must first define injectivity in first order logic. A function can be thought of as a binary predicate $P(x, y)$ that returns true if $f(x) = y$. Then, injectivity means that if $P(x, z)$ and $P(y, z)$, then $x = y$. Formalized in first order logic, that is the sentence

$$\forall x, y, z[[P(x, z) \wedge P(y, z)] \to (x = y)]$$

As producing formulas about sets is difficult in first order logic, it is much simpler to transform the pigeonhole principle into a statement about functions that take a set to itself, rather than a strict subset. These functions are called *endomorphisms*. To do this, consider an endomorphism $f : A \to A$; note that it defines a function $f' : A \to f(A)$. If f is injective then so is f', and by the contrapositive of the pigeonhole principle, we know that $f(A)$ is not a proper subset of A. But as f is an endomorphism, it must be a subset of A, so therefore $f(A) = A$. As the image of f is equal to its codomain, we know that f is surjective. This gives us the pigeonhole principle applied to endomorphisms: every injective endomorphism is surjective. The other direction of equivalence can be quickly verified by taking the contrapositive: every non surjective endomorphism is not injective. A non surjective endomorphism defines a function that takes its domain to a proper subset, giving us the pigeonhole principle once again.

An endomorphism in first order logic can be defined by

$$\forall x[\exists y P(y,x) \rightarrow \exists z P(x,z)]$$

i.e. The first order statement that every element in the range is also in the domain. Defining surjectivity in first order logic in the general case is actually somewhat tricky, because our predicate P is much more useful for talking about the image of the function f that it represents rather than the codomain. However, defining a surjective endomorphism is actually significantly easier, as we can take advantage of the fact that the codomain is the domain:

$$\forall x[\exists y P(x,y) \rightarrow \exists z P(z,x)]$$

In words, this states that if there is some y such that $f(x) = y$ (which is true if x in the domain and thus codomain of f), then there is some z such that such that $f(z) = x$, so x is in the image of f. Thus, If x is in the codomain, it is in the image, which is the definition of surjectivity.

Finally, we can restate the pigeonhole principle in first order logic for a single function by translating "if a function is a endomorphism and injective, then it is surjective:

$$\forall x[\exists y P(y,x) \rightarrow \exists z P(x,z)] \wedge \forall x,y,z[[P(x,z) \wedge P(y,z)] \rightarrow (y = z)] \rightarrow \forall x[\exists y P(x,y) \rightarrow \exists z P(z,x)]$$

However, as mentioned earlier, this is the first order sentence for the pigeonhole principle for a single function f, not a general first order definition of the pigeonhole principle. To generalize it to all functions, we must define it as an axiom schema, a pattern which we extend to an infinite list of sentences. This means that we must include a sentence for every single possible function f. Instead of interpreting $P(x,y)$ as a function f, we can also interpret it as a metavariable standing in for a formula $\varphi(x,y)$ that is true whenever $f(x) = y$. Thus, for all such formulas $\varphi(x,y)$, we have an instance of the axiom schema of the pigeonhole principle, a single sentence that applies to a given function.

1.2.6 Finite Model Theory

Finite Model Theory is a powerful tool to reason about decidability and complexity, as the data structures involved in computability problems are usually realistically finite. In the case of binary strings, we can use n-equivalence to show that every model of the theory of binary strings is also a model of a finite structure, and then use that to show the theory of finite or infinite strings is decidable. n equivalence is one of the most powerful tools in finite model theory, coming directly from the idea of an "Ehrenfeucht–Fraïssé game."

10

2

Literature Review

2.1 Related Work

2.1.1 Ehrenfeucht–Fraïssé games

Väänänen describes how Ehrenfeucht–Fraïssé games games can be used as powerful tools to distinguish (or show somewhat equivalent) logical structures. Ehrenfeucht–Fraïssé games are powerful tools for finite model theory as many of the other theorems in model theory fail for finite models, yet the EF game technique remains extremely effective. The central idea of an EF game is to show that two structures are equivalent with respect to a given logical system, up to a certain quantifier depth n, which is then frequently inducted upon to show that they are equivalent up to all quantifier depths.

An EF game consists of two players: the Spoiler and the Duplicator, as well as two structures on the same language, $\mathcal{A}$ and $\mathcal{B}$. The spoiler is attempting to pick elements from the two structures in order to create two structures $\mathcal{A}_n$ and $\mathcal{B}_n$ such that they are not elementarily equivalent (meaning that they satisfy the same logical sentences). The duplicator, on the other hand, is attempting to pick elements such that they are equivalent at the end of the game. A game is played for a predetermined n-rounds, and in each round, the Spoiler picks an element from one structure while the Duplicator picks an element from the other. If and only if the resulting substructures are the "same" at the end of n rounds, the Duplicator wins. If the duplicator wins every n-move EF game, the two structures are said to be n-equivalent.

The trick is in the resulting notion that the substructures are the "same." More formally, this process of picking an element from each structure is inducing a partial map $\pi : \mathcal{A} \to \mathcal{B}$, and the duplicator wins if this map is a partial isomorphism, essentially meaning that if it can be extended to an isomorphism $\pi*$ with a domain as the minimal substructure of A which contains $\mathrm{dom}(\pi)$ and range as the minimal substructure of B which contains $\mathrm{rng}(\pi)$. Here, an isomorphism on a pair of structures is a bijection on the domains which preserves the evaluation of the constants, and the evaluations of the n-ary functions and relations. The minimal substructure of $\mathcal{A}$ generated by a subset of the domain X can be found through the union of X with the representation of every constant symbol, as well as every evaluation of every functional symbol of every term already in the set. That last part can be thought about recursively, so it applies even to the evaluations of other functional symbols, although this gets complicated quickly so customarily EF games are played between structures with no functional symbols.(Väänänen 2011)

For example, consider an EF game played between two structures A and B on a signature with no symbols. Thus, any one to one partial map between the structures is a partial isomorphism, as it's not hard to preserve the evaluation of symbols when there are no symbols. Therefore, the duplicator will win any EF game that ends before one of the structures runs out of unique elements to pick and maintain the one to one map. Thus, a 2 turn EF game with a 4 element structure and a 3 element structure on the empty signature shows that they are 2-equivalent, but a 4 turn game shows that they are not 4-equivalent (or any $n > 4$).

This notion of n-equivalence is quite powerful; in effect, if two structures $\mathcal{A}$ and $\mathcal{B}$ are n-equivalent, then they cannot be distinguished by any formula of quantifier rank n (where rank here means the depth of nested quantifiers). This means that if and only if a formula of rank n is satisfied by one structure, it is satisfied by the other. Intuitively, if we recall that the negation of a universal ($\forall$) quantifier is an existential quantifier, every first order quantified sentence can be thought of as an existential sentence, and so the quantifier rank can be imagined as the number of variables that the existential statements are specifiying information about. Then, intuitively, if the duplicator wins the n move Ehrenfeucht-Fraisse game, for any element that satisfies an existential statement in one structure, the duplicator can pick an element that satisfies the same statement in the other structure, up to n times, due to the fact that there is always a partial isomorphism with at least n elements in the domain. This was actually proved by Ehrenfeucht's original paper: he uses an induction method, reasoning that if every quantifier layer of a rank n formula but the very outermost (a formula of quantifier rank $n - 1$) cannot distinguish the two structures, and the

duplicator can pick some nth element that maintains a partial isomorphism between the two structures, then picking that element cannot possibly cause the resulting formula to distinguish the two structures, as the extra layer of quantification can be interpreted as an existence statement and the nth chosen element can satisfy that existence statement only if it's isomorphic equivalent element does.(Ehrenfeucht 1961)

Continuing down this line of thought, showing that the duplicator wins every Ehrenfeucht-Fraisse game for every $n \in \mathcal{N}$ leads to the conclusion that there is no formula in the logic used for the game which can distinguish the two structures. When the logic used is first order logic, this is termed "elementary-equivalence." However, this doesn't necessarily indicate that the two structures are truly equivalent, only that the logic being used is not capable of distinguishing them. For example, Libkin proves that the property of "having an even number of elements" is not expressible over linear orders because for any finite linear order of size 2^n, it is n-equivalent to a finite linear order of size $2^n + 1$. Thus, any formula that is satisfied by a linear order of even size is also satisfied by a linear order of odd size.(Libkin 2013).

2.1.2 Kees-Doets

Kees Doets uses this EF game technique to great effect to establish n-equivalencies to finite structures. The core idea of this technique is to take a theory with a semi-infinite model, like the theory of all binary strings, show how segments of a semi-infinite structure are n-equivalent to a finite structure, and from there establish that any model of the structure is a model of a finite structure.

Recall the definition of binary strings as a structure consisting of a unary predicate (a relation on a single variable that returns true or false) on a linearly ordered finite chain (a chain just being a pairwise comparable list). Thus, the signature we are concerned with is one with no constant symbols, no functional symbols, a unary predicate P, and a binary predicate $<$.

Then, consider the idea of the "theory of binary strings," the set of all first order sentences which are true about all binary strings. For any sentence in this theory, any structure with a linearly ordered finite chain domain is a model of the theory, as those structures are the binary strings. To figure out what statements are actually in this

theory, we must axiomatize it, finding a set of sentences from which every sentence in the theory can be derived. As we want all models with linearly ordered finite domains, it seems intuitive that the axioms would resemble the sentences describing a linear order and a finite domain. We already saw the first order definition of a linear order earlier, but finiteness is not a first order property and cannot be defined in first order logic without a specific upper bound. This can be proven with a standard compactness argument.

Instead, Doets's candidate axioms are the axioms of a linear order, the axiom schema for the definable well ordering principle (which we already proved is equivalent to induction), and an axiom stating that there is a maximal element greater than all others. He makes use of the fact that the definable well ordering principle is equivalent to definable induction, allowing us to apply the property to the whole structure. The maximal element is still necessary, however. Refer to the theory containing these axioms as the theory T. It should be noted that there are infinite models of these axioms; the natural numbers $\mathcal{N}$ along with $\aleph_0$ and any interpretation of the unary relation P satisfy these axioms. However, Doets makes clever use of EF games to show that any statement true about binary strings (linear order with finite domain) can also be derived from these axioms.

Doets starts by proving a lemma which will grant us the psuedo-finiteness of all models of T:

Lemma 2.1.1. *Every model of T is n-equivalent to a finite structure.*

Proof. Let $\mathfrak{A}$ be a model of T with domain A, a linear order interpretation of the binary relation $<$, and an interpretation of the unary predicate denoted by U. This is written as $\mathfrak{A} = \langle A; <, U \rangle$.

First, Doets establishes that the property of being n-equivalent to a finite structure is definable in first order logic. This is due to the fact there are a finite number of quantifier rank n or less sentences in a given finite language. Doets uses this to define the n-characteristic σ of a model $\mathfrak{M}$, the conjunction of all sentence of quantifier rank $\leq n$ satisfied by $\mathfrak{M}$. Then, some portion of these n-characteristics have finite models; if a model $\mathfrak{N}$ can satisfy one of these n-characteristics then it satisfies all the same sentences as a finite model up to quantifier rank n, and thus is n equivalent to a finite structure. Thus, the first order definition of being n-equivalent to a finite structure is just the disjunction of all n-characteristics of finite models.

Symbolically, if we let Γ be the set of all n-characteristics of finite models, then the property of being n-equivalent to a finite structure is stated by the formula

$$\bigvee_{\tau \in \Gamma} \tau$$

Next, Doets considers the substructure $[1, a]$ of $\mathfrak{A}$ with domain of $\{x \in A : x \leq a\}$, where 1 is the least element of our structure (guaranteed to exist by the well ordering principle, which is itself equivalent to induction). By establishing that the formula which states that the substructure $[1, x]$ is n-equivalent to a finite structure is inductive, we can use definable induction (because we included definable well ordering in our axioms) to show that every substructure $[1, a]$ is n-equivalent to a finite structure, including the substructure where a is the maximal element, which is just the original structure $\mathfrak{A}$.

Establishing this induction requires an EF game. The base case, $[1, 1]$, is trivial, as $[1, 1]$ is a single element structure and trivially finite. In the inductive step, the inductive hypothesis tells us that $[1, a]$ is n-equivalent to a finite structure $\mathfrak{B}$; let this finite structure be written $\mathfrak{B} = \langle \{1, \ldots, m\}; <, V \rangle$ where $<$ is the standard interpretation of the binary relation $<$ and V is a unary predicate on $\{1, \ldots, m\}$. Doets shows with an EF game that $[1, a + 1]$ (where $a + 1$ denotes the immediate successor of a, which exists thanks to the well ordering principle) is n-equivalent to $\mathfrak{B}' = \langle \{1, \ldots, m + 1\}; <, V' = V \cup \{m + 1 : U(a + 1)\} \rangle$ using an EF game. Before diving into this game, we must understand V', the interpretation of the unary predicate symbol on $\{1, \ldots, m + 1\}$ in our new structure. V' acts identically to V except in the case of $m + 1$, where it is true for $m + 1$ if and only if $a + 1$ is true for U in $\mathfrak{A}$. This makes the EF game relatively simple, given that we know that there is a winning EF game between $[1, a]$ and $\mathfrak{B}$ by the inductive hypothesis. If the Spoiler picks any element in $[1, a]$ or $\mathfrak{B}$, the Duplicator picks the corresponding element in the other set which it picks in the winning EF game between $[1, a]$ and $\mathfrak{B}$. If the Spoiler picks $a + 1$ or $m + 1$, the Duplicator simply chooses the other; this preserves $<$, as $x < a + 1$ for all $x \in [1, a + 1]$ and likewise $y < m + 1$ for all y in $\mathfrak{B}'$, and the unary predicate V' was specifically defined such that $U(a + 1)$ if and only if $V(m + 1)$. Thus, by an EF-game, we know that the property that an initial segment $[1, x]$ of $\mathfrak{A}$ is n-equivalent to a finite structure is inductive. By induction, every initial segment $[1, x]$ of $\mathfrak{A}$ is n-equivalent to a finite structure. As $\mathfrak{A}$ has a greatest element, it is itself an initial segment, so it is n-equivalent to a finite structure. □

With the bulk of the work done, Doets moves on to establish that any model of T is pseudofinite, meaning every first order sentence has same truth value in both the pseudofinite model and a finite model of T. For any given first order sentence of quantifier rank n, this is guaranteed by the n-equivalence of T to a finite model.

Finally, because any sentence that is true in T is true in some finite model, and as discussed all finite models of T represent binary strings, it is a quick corollary that T axiomatizes the theory of binary strings. To verify this, simply suppose that it didn't, that there is some sentence θ in the theory of binary strings (meaning that it is true for all binary strings) which cannot be derived from T. The contrapositive of completeness tells us that there must be some model of T where θ is false; however, the lemma proven above would indicate that θ is false in a finite model of T, which is a binary string, contradicting the fact that θ is in the the theory of binary strings.

(Doets et al. 1989).

2.1.3 Axiomatization

One of the primary challenges in finite model theory is the axiomatization of finite structures. Axioms are fundamental principles, the ideas that are taken as true and from which all else are derived. In terms of model theory, the axioms of a structure are the set of sentences which uniquely characterize the structure up to isomorphism. Because there are a finite number of possible values in a finite structure and thus finitely many ways they can relate, any finite structure can be axiomatized by the conjunction of every possible way the values can relate. However, extending this to classes of finite structures becomes more complicated. It's not guaranteed that every class of finite structures has a finite axiomatization, for example. This has important implications on the decidability of the class of finite structures; for example, the Rosen paper states that a class of finite structures is axiomatizable by a single sentence of a certain form if and only if it has complexity class NP.(Rosen 2002)

Axiomatization of the theory of a class of structures is a fundamental step to understanding that class of structures, as it is essentially reducing the class down to a precise list of mathematical definitions that describe what it means to be a member of that class. Then, any problem that can be stated about the structure in the language of the axioms can be reasoned about using the axioms. In fact, effectively axiomatizing a class of structures can sometimes directly indicate that first order

problems concerning that structure are decidable and computable, meaning that a computer algorithm could solve any problem that first order logic can describe about the structure. More formally, the statement is:

Theorem 2.1.1. *Any effectively axiomatizable complete theory is decidable.*

Proof. Effectively axiomatizable simply means that there are a recursively enumerable number of axioms; however, due to Craig's Theorem, a recursively enumerable list of axioms yields a recursive (decidable) list of axioms. Craig's Theorem obtains this recursive list from the recursively enumerable list simply by mapping between each sentence A_i in the recursively enumerable list and the conjunction $B_i = \bigvee_i A_i$ of i many copies of A_i. While equivalent in truth value to the set of all A_i, the set of all B_i is easily decided by an algorithm which checks whether a sentence is of the form of i many conjunctions of the formula A_i. Thus, any effective axiomatization may also be considered a recursive axiomatization. (Craig 1953)

A complete theory is one where for every sentence φ that can be formulated in the language of the theory, either φ or $\neg\varphi$ is in the theory. In other words, every statement in the language has a discrete truth value in regard to the theory. When we talk about "the theory of a structure," we are typically discussing a complete theory, because these theories consist of every statement that is true for that structure. Then, if a sentence isn't in the theory, this means that it is false for that structure, so the negation of it is true for that structure and in the theory.

The above theorem hinges upon the fact that given a recursively enumerable set of axioms, we can derive every possible consequent formula in an enumerable way. Essentially, this can be done in a brute force way by making every possible derivation in some specific order on the axioms of the theory, like brute force generating every possible proof tree. Then, once we have enumerated the consequences of the axioms, we have recursively enumerated the whole theory.

At this point, the proof is very quick: because either φ or $\neg\varphi$ is in the theory, we can decide whether φ is in the theory by simply enumerating the sentences in the theory until we arrive at either φ or $\neg\varphi$. Thus, the theory is decidable and computable. □

For incomplete theories, this same approach yields a weaker result:

Theorem 2.1.2. *Any recursively axiomatizable consistent theory is recursively enumerable.*

Proof. Follow the same approach as above; however, if neither φ nor $\neg\varphi$ is in the theory then then the decision algorithm will never terminate. □

Applied to the axiomatization of binary strings found by Kees Doets earlier, this means that the theory of binary strings is recursively enumerable; it's possible to list out every true first order statement about all binary strings. However, while theories of classes of structures like binary strings are not complete, if the class itself is recursively enumerable, the theories actually decidable, not just recursively enumerable:

Theorem 2.1.3. *A recursively axiomatizable theory of a recursively enumerable class of structures is decidable.*

Proof. The proof is relatively straightforward: Let the theory T be a recursively axiomatizable theory of a recursively enumerable class C of structures. If a sentence θ is in T, this can be found by enumerated T until we arrive at θ. If a sentence θ is not in T, then there is some structure $\mathfrak{A} \in C$ such that θ is false in A. Thus, an algorithm can decide that θ is not in T based upon enumerating the structures of C until it arrives at the structure $\mathfrak{A}$ where θ is false. Thus, because it's possible to compute whether $\theta \in T$ in either case, T is decidable. □

(Smith 2013)

3

Axiomatizating Trees

3.1 Introduction

In the previous section, we saw how we could use EF-games along with definable induction to axiomatize the theory of binary strings, one of the simplest ordered data structures. Rather than strings, however, many common computer science problems are focused around the manipulation of trees, an ordered data structure which allows for more complex representations than the linear structure of strings. Trees have some important characteristics: they have a root element, from which all other elements descend, and the nodes are connected in such a way that there is a single unique path from any node to the root. Defining these properties carefully is crucial to properly axiomatizing trees, as well as understanding the logic which which they can be manipulated.

3.1.1 Definable Trees

The core idea behind representing a tree as a structure is to use the order relation $<$ to represent ancestry on the tree, such that the root of a tree is the minimal element. This means that $<$ is a partial order rather than a linear order, as there can easily be nodes in a tree which are not ancestors of each other. A partial order is similar to a linear order, but it doesn't require all elements in the set to be related or equal; instead of the trichotomy property of linear orders, which dictates that exactly one of $a < b$, $b < a$, and $a = b$ is true, partial orders dictate that at most one of those relations are true. However, to properly represent a tree, not just any partial order will work; for example, a relation which only relates 2 elements of the domain is trivially a partial order, but clearly can't represent a tree with a large

19

number of nodes. Instead, for any element m in the domain M, consider the set $\{n \in M : n < m\}$ of all ancestors of m. For a tree, these ancestors should be a single unique path to the root, so they should be linearly ordered for all nodes m; thus, all descending chains $\{n \in M : n < m\}$ are linearly ordered in a tree. This property isn't too hard to define in first order logic; the idea is that if two nodes x, y share a common descendant, then their relationship is trichotomous; exactly one of $x < y$, $y < x$, or $x = y$ is true:

$$(\forall x, y)((\exists z)(x < z \wedge y < z) \rightarrow (\text{trichotomy between x and y}))$$

This, along with the property of the partial order, is almost enough to define a tree; in fact, these properties define "forests", meaning collections of trees, as well as trees. To define an individual tree, we'd want to consider a maximally connected subset our overall forest, where "connected" is thought of as related; these can be defined as the descendants of a minimal element. It's worth noting that this way of defining a tree specifically identifies it by its root element; this is in common with other common operations on trees, which often associate their value at the root with the value of the operation on the tree. Regardless, the distinction between tree and forest is not incredibly important to the first order logic definition as a structure, as any definable property on a forest structure is also definable on a tree structure, so we will often refer to just the two properties of partial order and linearly ordered ancestors as the properties of trees.

3.1.2 Induction on a Tree

Because trees have a partial ordering instead of a total linear one, the meaning of well-foundedness and induction changes slightly. Well-foundedness is still the idea that any non-empty subset has a minimal element, but while that implies a well-ordering on the linear ordering of binary strings, it just means that each of the descending chains $\{n \in M : n < m\}$ are well-orderings in a tree. Similarly, induction takes on a slightly different meaning; it still always means that if a property is inductive (and true of some base case), then it is universal and true of all nodes in tree, but what it means to be inductive can take on a different interpretation. The previous definition of strong definable induction still makes sense; it essentially states that if a property is strongly inductive (if it is true for all $x < m$, then it is true for m) then it is universal. Applied to trees, the inductive property would

mean "if it is true for all of my ancestors, then it is true for me." It's easy to prove that strong induction on trees is a direct consequence of strong induction on linear orders, simply by examining the ancestor chain of any element and noting that it is a linear order. Recall that "weak" (or standard) inductivity means that if a property is true for m, then it is true for $m + 1$; the parallel in terms of trees would be that if a property is true for a node, it is true for all of its children. This notion of induction can be referred to as "root to leaf" induction (hereby referred to as RTL induction), for the way that the property spreads down through the tree from the root. Formally, RTL induction is the law that a property is RTL inductive and it is true of the root, then it is true of all nodes in the tree. It's not hard to see that this is equivalent to standard linear induction; the naturals can be viewed as a linear tree where each node has one child in one direction, and the RTL inductivity property follows directly from linear induction on the ancestor chains. However, this isn't actually the way induction on trees is commonly done in math and computer science. Often a computer science proof will involve inducting on the height of the tree, which is more related to "leaf to root" induction. When doing induction on the height of a tree, the inductivity statment is that a tree of height n has some property if that same property is true of all trees of lesser height, usually specifically concerning subtrees rooted at the immediate children of the original root. Likewise, "leaf to root" (hereby LTR) inductivity is the statement that if a property is true of all children of a node, then it is true of the node itself. LTR induction says that if a property obeys LTR inductivity, and is true of all the leaves, then it is true of the whole tree. Note that this necessitates the fact that all "branches" (maximal chains) have a leaf, which is not true about RTL induction! More formally, LTR induction requires that all subtrees $\{n \in M : m \leq n\}$ for all nodes m have at least one maximal element. It's important that all subtrees have maximal elements, not just the original tree, because it's easy to model a tree where some elements have a leaf descendant but others don't (imagine a V with one leg going to infinity). This assumption isn't true of all infinite trees, but clearly holds for every finite tree, so isn't an issue if we're trying to study the psuedo-finite models of trees.

3.1.3 Structural Induction

In computer science, we often want to induct over all sorts of data structures, rather than just the natural numbers or trees. The generalized technique for this is called structural induction. The core idea of structural induction is that if a property holds for some "minimal" structures in a set, and it can be shown to hold for some

"recursive" structures which can be defined in terms of other smaller recursive or minimal structures if it holds for those component structures, then that property holds universally over all structures. Applied to trees, we will consider the subtree $(x \uparrow)$ rooted at a given node x as the structure corresponding to that node. Then, it's clear that the smaller structures which $(x \uparrow)$ is composed of are just subtrees rooted at the children of x. The minimal structures are correspondingly the leaf nodes. In this manner, structural induction over the tree becomes very similar to strong induction on the height of the tree, which usually operates on the principle that any node has a larger height than its children, so the strong induction principle allows one to assume the property is true of the children in order to show that it is true of the parent. In this respect, structural induction travels in the Leaf to Root induction, however, it's entirely possible to imagine performing structural induction along an ancestor chain of nodes, which would involve travelling in a Root to Leaf direction.

Structural Induction is a nice form of induction to apply to this case, because its properties are relatively easy to define. The "minimal" structures are just the maximal elements of our tree, and the required principle of "predecessor" or "parent" are relative easy to define. It is important to define this principle, otherwise the validity of induction is less obvious, as it's possible to imagine trees where each node does not have clear parents or children relationship; think of the positive rationals numbers as a unary tree rooted at 0. In this case, it would be difficult to prove that a property is true about a node even if its true for all subtrees of its children, because no node properly has children in the first place! However, having children itself isn't enough; imagine augmenting that rational unary tree such that each node also has an infinite descending chain that is "discrete" like the natural numbers in addition to the previous ordered elements. In this example, every element has a child, but induction still runs into problems trying to relate nodes which descend in the rationals-like chain to their ancestor nodes like previously. As structural induction relies on the idea that any element of a recursively defined structure can be found inside a component structure, it's this exact property we need: any descendant of a node descends from an immediate child of the node; this both guarantees the existence of immediate children, as well as ensures that they can be used for inductive purposes. It should also be noted that this is related to the statement that any descending chain is a well-ordering, which states the node if x descends from the node y, then the subset of all nodes between y and x must have a minimal element. This minimal element is exactly the the immediate child x descends from.

22

3.1.4 The Theory of Finite Trees

The core idea behind axiomatizing the theory of trees is the same as axiomatizing strings: Construct a theory T from the properties of a tree and as few other axioms as possible, so that a finite model of T is just a standard finite tree. Then, we will show that that T is psuedo-finite, which consequently means that any sentence true about all finite trees is true about T, so T effectively axiomatizes the theory of finite trees.

For any model of the previously axiomatized theory of binary strings, Kees Doets proved n-equivalence to a finite structure by inducting the n-equivalence to a finite structure of an initial segment $[1, a]$ over the whole structure, including the maximal element. If we have leaf to root induction, then we can apply a similar technique here, where we induct the n-equivalence to a finite structure of the subtree rooted at a node m up from the leaves all the way to the root.

First, we must carefully construct our candidate theory T. The signature in question contains no constant or function symbols, and the binary predicate $<$, as we are considering all trees, not just trees that store binary information (which would also include a unary predicate P). T must contain the axioms that state that $<$ is a partial order, and that it obeys the "ancestors are linearly ordered" property as mentioned above. Next, we include an axiom that states there is a least element, $(\exists x)(\forall y, x \leq y)$. This is parallel to the maximal element axiom for strings, but by strengthening it to least rather than minimal, we actually ensure that our tree is a single tree rather than a forest. If we wanted to axiomatize the theory of all forests, we could instead replace it with an axiom which states that every ancestor chain has a minimal element, and so consequently every component tree has a root. Finally, we include the axiom schema indicating that structural induction holds for every definable property on the tree. In English, this is the idea that any property P which holds for any node if it holds for all of its children is universal. Note that this embeds the base case of structural induction within it, as the minimal leaf nodes don't have children and thus must be in P in order to satisfy the inductive hypothesis. In first order logic, this is the statement that

$$\forall x \left((\forall y \left(y \text{ is a child of } x \right) \rightarrow P(y)) \rightarrow P(x) \right) \rightarrow \forall z P(z)$$

Note that the "child" relationship is first order definable as a descendant (meaning lesser) element which is not a descendant of any other descendant. This axiom is important because it's the exact idea that induction works on the theory that

consequently leads to the tree being psuedofinite; this form of induction is mutually exclusive with the idea that a tree can have infinite branches which extend endlessly without terminating. Thus, any model of T will be a tree because of the axioms for partial order and linear order of ancestors, and so a finite model of T is a finite tree.

Theorem 3.1.1. *The theory T, containing axioms for a partial order, the linear order of all ancestors of any node, a least element, and an axiom schema for definable structural induction on a tree effectively axiomatizes the theory of finite trees.*

In this case, most of the work again takes place in a lemma showing the n-equivalence to a finite structure:

Lemma 3.1.1. *Every model of T is k-equivalent to a finite structure.*

Proof. Let $\mathfrak{M}$ be a model of T with domain M and partial order interpretation of the binary relation $<$. This is written as $\mathfrak{M} = \langle A; U; > \rangle$.

As proved by Doets earlier, the property of being k-equivalent to a finite structure is definable in first order logic. Then, we will consider the substructures $(m\uparrow)$ with subtree domains $\{n \in M : n \leq m\}$. To proceed, we can prove that the formula which states that $(m\uparrow)$ is n-equivalent to a finite structure is leaf to root inductive, and then use structural induction, as we have the axiom schema for structural induction in our theory, to show that it holds for all $m \in \mathfrak{M}$, including the least element r. This induction will require some EF games.

In the base case, $\{n \in M : n < m\}$ is empty, so m is a leaf; the maximal well ordering principle guarantees us the existence of some leaves. Then, $(m\uparrow)$ is just a single element structure and trivially finite and k-equivalent to a finite structure. In the inductive step, we want to show that $(m\uparrow)$ is k-equivalent to a finite structure $\mathfrak{N}$. By the inductive hypothesis, for all $n \in M$ such that $n \in children(m)$, $(n\uparrow)$ is k-equivalent to a finite structure $\mathfrak{N}_n$. Let

$$\mathfrak{N} = \left(\bigcup_{n \text{ is a child of } m} \mathfrak{N}_n \right) \cup a$$

where a is an additional element added to $\mathfrak{N}$ such that $a < x$ for all $x \in \bigcup \mathfrak{N}_n$. This is a finite structure, as it is the union of finitely of many finite structures. In the case

of an infinite number of children, we can note that there are only finitely many k-equivalence classes of finite structures, and include a copy of a single representative structure $\mathfrak{N}_n$ for any $(n \uparrow)$ in a given equivalence class. Note, however, we must avoid adding infinitely many copies of the same finite structure up to isomorphism to our union in this infinite children case (using the same copy doesn't work, because each copy needs to represent the possible results of a separate EF-game, and thus the structures involved in different EF games must be kept separate in disjoint copies). To do this, just notice that a k-move game can only interact with up to k different substructures of $\mathfrak{N}$, so we only need to include k many disjoint copies of any finite structure $\mathfrak{N}_n$ up to isomorphism. Thus, for any structure $\mathfrak{N}_\mathfrak{n}$ such that there are more than k many nodes which correspond to the finite structure $\mathfrak{N}_n$, only include k many disjoint copies of $\mathfrak{N}_\mathfrak{n}$ in the union. Then, the k move EF game between $(m \uparrow)$ and $\mathfrak{N}$ is relatively simple for the Duplicator to win, because we know there is a winning game between $(n \uparrow)$ and $\mathfrak{N}_n$. If the Spoiler picks any element in $\mathfrak{N}_n$ for some n or picks some $n < m$ in $(m \uparrow)$, the Duplicator picks the corresponding element in the other set which it picks in the winning game between $\mathfrak{N}_n$ and $(n \uparrow)$. If the Spoiler picks $a \in \mathfrak{N}$ or $m \in (m \uparrow)$, the Duplicator chooses the other, which preserves the properties of $<$ as both of them are least elements of their corresponding structures. Thus, the partial map induced by the EF game is a partial isomorphism, so the Duplicator wins the EF game and the property that $(m \uparrow)$ is k-equivalent to a finite structure is structurally inductive. Then, by structural induction, $(m \uparrow)$ is k-equivalent to a finite structure for all $m \in \mathfrak{M}$. This includes $(r \uparrow)$, where r is the least element of $\mathfrak{M}$. But $(r \uparrow)$ is just $\mathfrak{M}$ by the definition of least element, so finally $\mathfrak{M}$ is n-equivalent to a finite string. □

After this, the remaining proof of the Theorem is essentially the same as Kees Doets. The lemma directly leads to the consequence that any model of T is pseudofinite, as any first order sentence of quantifier rank n has the same truth value in the model of T and the finite model that the lemma states that it must be n-equivalent to. Then, a quick contradiction shows that T axiomatizes the theory of finite trees. Suppose there is some sentence θ in the theory of finite trees which cannot be derived from T. The contrapositive of completeness tells us that there must be some model of T for which θ is not true; but by the psuedo-finiteness of this model, there must be a finite model of T where θ is not true. But finite models of T are finite trees, so this contradicts the fact that θ is in the theory of finite trees. Thus, every sentence in the theory of finite trees is derivable from T, so T axiomatizes the theory of finite strings.

4

Conclusion and Future Work

Previous work has been done to axiomatize many finite structures, such as that seen in [Doets]. However, these axiomatizations are often dense and difficult to understand due to relying on advanced results in finite model theory. Axiomatizing based on n-equivalence to a finite structure, or in other words in terms of Ehrenfeucht-Fraisse games, has the remarkable property of essentially only relying on the principle of structural induction. The core idea seems simple: show that a structure is recursively defined, show that the atomic components are n-equivalent to a finite structure (typically because they are in fact finite), and then construct the EF game for the recursive structures based on the respective games of their component structures. In theory, this same angle of attack could be extended to almost any recursively defined structure. The catch, however is exactly that: to use structural induction, we must have a recursively defined structure. The technicalities of what exactly a recursively defined structure is, and even ultimately what structural induction is, are not often well defined in existing literature, as structural induction is often introduced as a proof technique with the brief analogy to mathematical induction. Further work could pursue this element of the proof and attempt to extend this technique to obtain axiomatizations of other, possibly less nicely ordered structures, so long as they are "recursively defined." In another direction, one could try to eschew the induction axiom altogether and replace it with other axioms that accomplish a similar effect, like some sort of well-foundedness, successor, predecessor, or even the pigeonhole principle. This could have implications about what kinds of order are strictly necessary for the proof technique to work, as the pigeonhole principle itself is unordered. However, before that work proceeds, a stronger understanding of the role of structural induction on n-equivalent structures would be helpful, and it is this area that could have interest results such as axiomatizations of more complex structures.

References

Craig, William (Mar. 1953). "On Axiomatizability Within a System". In: *J. Symbolic Logic* 18.1, pp. 30–32. URL: https://projecteuclid.org:443/euclid.jsl/1183731538 (cit. on p. 17).

Doets, Kees et al. (1989). "Monadic Π^1_1-theories of Π^1_1-properties." In: *Notre Dame Journal of Formal Logic* 30.2, pp. 224–240 (cit. on p. 16).

Ehrenfeucht, Andrzej (1961). "An application of games to the completeness problem for formalized theories". In: *Fund. Math* 49.129-141, p. 13 (cit. on p. 13).

Libkin, Leonid (2013). *Elements of finite model theory*. Springer Science & Business Media (cit. on pp. 3, 13).

Rosen, Eric (2002). "Some Aspects of Model Theory and Finite Structures". In: *The Bulletin of Symbolic Logic* 8.3, pp. 380–403. ISSN: 10798986. URL: http://www.jstor.org/stable/3062205 (cit. on p. 16).

Smith, Peter (2013). *An introduction to Gödel's theorems*. Cambridge University Press (cit. on p. 18).

Stansifer, Ryan (Sept. 1984). *Presburger's Article on Integer Airthmetic: Remarks and Translation*. Tech. rep. TR84-639. Cornell University, Computer Science Department. URL: http://techreports.library.cornell.edu:8081/Dienst/UI/1.0/Display/cul.cs/TR84-639 (cit. on p. 4).

Väänänen, Jouko (2011). *Models and Games*. Cambridge Studies in Advanced Mathematics. Cambridge University Press. DOI: 10.1017/CBO9780511974885 (cit. on p. 12).