Chapter 1

STRUCTURES

Algorithms, Turing machines, and modern computer programs all work with finitary objects, objects that usually can be encoded by finite binary strings or just by natural numbers. For this reason, computability theory concentrates on the study of the complexity of sets of natural numbers. To study the computational properties of a countable mathematical structure, the first approach is to set the domain of the structure to be a subset of the natural numbers and then borrow the tools we already have from computability theory. One issue comes up: There might be many bijections between the domain of a structure and the natural numbers, inducing many different *presentations* of the structure with different computability-theoretic properties. The interplay between properties of presentations (computational properties) and properties of isomorphism types (structural properties) is one of the main themes of computable structure theory.

We start this chapter by introducing various ways of representing structures so that we can analyze their computational complexity. These different types of presentations are essentially equivalent, and the distinctions are purely technical and not deep. However, they will allow us to be precise later. At the end of the chapter we prove Knight's theorem that all non-trivial structures have presentations that code any given set.

1.1. Presentations

All the structures we consider are countable. So, unless otherwise stated, "structure" means "countable structure." Furthermore, we usually assume that the domains of our structures are subsets of $\mathbb{N}$. This will allow us to use everything we already know about computable functions on $\mathbb{N}$.

DEFINITION 1.1.1. An *ω-presentation* is nothing more than a structure whose domain is $\mathbb{N}$.[9] Given a structure $\mathcal{A}$, when we refer to *an ω-presentation of $\mathcal{A}$* or to a *copy of $\mathcal{A}$*, we mean an ω-presentation $\mathcal{M}$ which is

[9]The use of the word *presentation* here has nothing to do with its use in group theory. There, a presentation of a group consists a list of generators and a list of relations among them. You might have a group with a computable presentation, meaning that this list of relations is computable, but which has no computable ω-presentation in our sense.

isomorphic to $\mathcal{A}$. An ω-presentation $\mathcal{M}$ is *computable* if all its relations, functions, and constants are uniformly computable; that is, if the set $\tau^{\mathcal{M}}$, defined as

$$\tau^{\mathcal{M}} = \bigoplus_{i \in I_R} R_i^{\mathcal{M}} \oplus \bigoplus_{i \in I_F} F_i^{\mathcal{M}} \oplus \bigoplus_{i \in I_C} \{c_i^{\mathcal{M}}\}, \tag{1}$$

is computable. Note that via standard coding, we can think of $\tau^{\mathcal{M}}$ as a subset of $\mathbb{N}$.

1.1.1. Atomic diagrams. Another standard way of defining when an ω-presentation is computable is via its atomic diagram. Let $\{\varphi_i^{\text{at}} : i \in \mathbb{N}\}$ be an effective enumeration of all atomic τ-formulas with free variables from the set $\{x_0, x_1, \dots\}$. (An *atomic τ-formula* is one of the form $\mathsf{R}(\mathsf{t}_1, \dots, \mathsf{t}_a)$, where R is either "$=$" or R_j for $j \in I_R$, and each t_i is a term built out of the function, constant, and variable symbols.)

DEFINITION 1.1.2. The *atomic diagram* of an ω-presentation $\mathcal{M}$ is the infinite binary string $D(\mathcal{M}) \in 2^{\mathbb{N}}$ defined by

$$D(\mathcal{M})(i) = \begin{cases} 1 & \text{if } \mathcal{M} \models \varphi_i^{\text{at}}[x_j \mapsto j : j \in \mathbb{N}], \\ 0 & \text{otherwise.} \end{cases}$$

It is not hard to see that $D(\mathcal{M})$ and $\tau^{\mathcal{M}}$ are Turing equivalent. We will often treat the ω-presentation $\mathcal{M}$, the real $\tau^{\mathcal{M}}$, and the real $D(\mathcal{M})$ as the same thing. For instance, we define the *Turing degree of the ω-presentation* $\mathcal{M}$ to be the Turing degree of $D(\mathcal{M})$. When we say that $\mathcal{M}$ *is computable from a set X*, that *a set X is computable from* $\mathcal{M}$, that $\mathcal{M}$ is Δ_2^0, that $\mathcal{M}$ is *arithmetic*, that $\mathcal{M}$ is *low*, etc., we mean $D(\mathcal{M})$ instead of $\mathcal{M}$.

Let us also point out that the quantifier-free diagram, which is defined like the atomic diagram but using a listing of the quantifier-free formulas instead, is Turing equivalent to $D(\mathcal{M})$ too.

1.1.2. An example. Unless it is trivial, a structure will have many different ω-presentations—continuum many actually (see Theorem 1.2.1)—and these different ω-presentations will have different computability theoretic properties. For starters, some of them may be computable while others may not. But even among the computable copies of a single structure one may find different computability theoretic properties.

Consider the linear ordering $\mathcal{A} = (\mathbb{N}; \leq)$, where $\leq$ is the standard ordering on the natural numbers. We can build another ω-presentation $\mathcal{M} = (\mathbb{N}; \leq_M)$ of $\mathcal{A}$ as follows. Let $\{k_i : i \in \mathbb{N}\}$ be a one-to-one computable enumeration of the halting problem $0'$. First, order the even natural numbers in the natural way: $2n \leq_M 2m$ if $n \leq m$. Second, place the odd number $2s + 1$ right in between $2k_s$ and $2k_s + 2$, that is, let $2k_s \leq_M 2s + 1 \leq_M 2k_s + 2$. Using transitivity we can then define $\leq_M$ on all pairs of numbers. Thus $2n <_M 2s + 1$ if and only if $n < k_s$, and

$2s+1 <_M 2t+1$ if and only if $k_s < k_t$. (Early codings of sets into ω-presentations of linear orderings appear in [Mar82].)

One can show that $\mathcal{A}$ and $\mathcal{M}$ are two computable ω-presentations of the same structure.[10] However, computationally, they behave quite differently. For instance, the successor function is computable in $\mathcal{A}$ but not in $\mathcal{M}$: In $\mathcal{A}$, $\mathsf{Succ}^{\mathcal{A}}(n) = n+1$ is clearly computable. On the other hand, in $\mathcal{M}$, $\mathsf{Succ}^{\mathcal{M}}(2n) = 2n+2$ if and only if there is no odd number placed $\leq_M$-in-between $2n$ and $2n+2$, which occurs if and only if $n \notin 0'$. Therefore, $\mathsf{Succ}^{\mathcal{M}}$ computes $0'$ and $\mathsf{Succ}^{\mathcal{A}}$ does not.

The reason $\mathcal{A}$ and $\mathcal{M}$ can behave differently despite being isomorphic is that they are not *computably isomorphic*: There is no computable isomorphism between them. To see this, note that if there was one, we could use $\mathsf{Succ}^{\mathcal{A}}$ and the isomorphism to compute $\mathsf{Succ}^{\mathcal{M}}$, contradicting that $\mathsf{Succ}^{\mathcal{M}}$ computes $0'$.

1.1.3. Relaxing the domain. In many cases, it will be useful to consider structures whose domain is a subset of $\mathbb{N}$. We call those $(\subseteq\omega)$-*presentations*. If M, the domain of $\mathcal{M}$, is a proper subset of $\mathbb{N}$, we can still define $D(\mathcal{M})$ by letting $D(\mathcal{M})(i) = 0$ if φ_i^{at} mentions a variable x_j with $j \notin M$. In this case, we have

$$D(\mathcal{M}) \equiv_T M \oplus \tau^{\mathcal{M}}.$$

To see that $D(\mathcal{M})$ computes M, notice that, for $j \in \mathbb{N}$, $j \in M \leftrightarrow D(\mathcal{M})(\ulcorner x_j = x_j \urcorner) = 1$, where $\ulcorner \varphi \urcorner$ is the index of the atomic formula φ in the enumeration $\{\varphi_i^{\mathrm{at}} : i \in \mathbb{N}\}$.

The following observation will simplify many of our constructions later on.

Observation 1.1.3. We can always associate to an infinite $(\subseteq\omega)$-presentation $\mathcal{M}$, an isomorphic ω-presentation $\mathcal{A}$: If $M = \{m_0 < m_1 < m_2 < \cdots\} \subseteq \mathbb{N}$, we can use the bijection $i \mapsto m_i : \mathbb{N} \to M$ to get a copy $\mathcal{A}$ of $\mathcal{M}$, now with domain $\mathbb{N}$. Since this bijection is computable in M, it is not hard to see that $D(\mathcal{A}) \leq_T D(\mathcal{M})$, and furthermore that $D(\mathcal{A}) \oplus M \equiv_T D(\mathcal{M})$.

One of the advantages of $(\subseteq\omega)$-presentations is that they allow us to present finite structures.

1.1.4. Relational vocabularies. A vocabulary is *relational* if it has no function or constant symbols, and has only relational symbols. Every vocabulary τ can be made into a relational one, $\widetilde{\tau}$, by replacing each n-ary function symbol by an $(n+1)$-ary relation symbol coding the graph of the function, and each constant symbol by a 1-ary relation symbol coding it as a singleton. Depending on the situation, this change in

[10]To show that $\mathcal{M}$ is isomorphic to the standard ordering on $\mathbb{N}$, one has to observe that every element of $M = \mathbb{N}$ has finitely many elements $<_M$-below it: $2n$ has at most $2n$, and $2s+1$ has at most $2k_s$.

vocabulary might be more or less significant. For instance, the class of quantifier-free definable sets changes, but the class of $\exists$-definable sets does not (see Exercise 1.1.4). For most computational properties, this change is nonessential; for instance, if $\mathcal{M}$ is an ω-presentation of a τ-structure, and $\widetilde{\mathcal{M}}$ is the associated ω-presentation of $\mathcal{M}$ as a $\widetilde{\tau}$-structure, then $D(\mathcal{M}) \equiv_T D(\widetilde{\mathcal{M}})$ (as it follows from Exercise 1.1.4). Because of this, and for the sake of simplicity, we will often restrict ourselves to relational vocabularies.

EXERCISE 1.1.4. Show that the $\exists$-*diagram of* $\mathcal{M}$ as a τ-structure is m-equivalent to its $\exists$-diagram as a $\widetilde{\tau}$-structure. More concretely, let $\{\varphi_i^\exists : i \in \mathbb{N}\}$ and $\{\widetilde{\varphi}_i^\exists : i \in \mathbb{N}\}$ be the standard effective enumerations of the existential τ-formulas and the existential $\widetilde{\tau}$-formulas on the variables $x_0, x_1, \ldots$. Show that

$$\{i \in \mathbb{N} : \mathcal{M} \models \varphi_i^\exists[x_j \mapsto j : j \in \mathbb{N}]\} \equiv_m \{i \in \mathbb{N} : \widetilde{\mathcal{M}} \models \widetilde{\varphi}_i^\exists[x_j \mapsto j : j \in \mathbb{N}]\}.$$

One could also show these sets are $\equiv_1$-equivalent.

1.1.5. Finite structures and approximations. We can represent finite structures using $(\subseteq\omega)$-presentations. However, when working with infinitely many finite structures at once, we often want to be able to compute things about them uniformly, for instance the sizes of the structures, which we could not do from $(\subseteq\omega)$-presentations (see Exercise 1.1.5). For that reason, we sometimes consider $(\sqsubseteq\omega)$-presentations, which are $(\subseteq\omega)$-presentations whose domains are initial segments of $\mathbb{N}$. Given a finite $(\sqsubseteq\omega)$-presentation, we can easily find the first k that is not in the domain of the structure.

EXERCISE 1.1.5. Show that there exists a computable list $\{\mathcal{M}_n : n \in \mathbb{N}\}$ of $(\subseteq\omega)$-presentations of finite structures whose sizes cannot be computed uniformly, that is, a list such that the domains and relations of the $\mathcal{M}_n$'s are uniformly computable, but there is no computable function f such that $f(n)$ is the size of M_n.

When τ is a finite vocabulary, finite τ-structures can be coded by a finite amount of information. Suppose $\mathcal{M}$ is a finite τ-structure with domain $\{0, \ldots, k-1\}$, and τ is a finite relational vocabulary. Then there are only finitely many atomic τ-formulas on the variables $x_0, \ldots, x_{k-1}$, let us say ℓ_k of them. Assume the enumeration $\{\varphi_i^{\text{at}} : i \in \mathbb{N}\}$ of the atomic τ-formulas is such that those ℓ_k formulas come first, and the formulas mentioning variables beyond x_k come later. Then $D(\mathcal{M})$ is determined by the finite binary string of length ℓ_k that codes the values of those formulas. We will often assume $D(\mathcal{M})$ *is* that string.

When dealing with infinite structures, very often we will want to approximate them using finite substructures. We need to take care of two technical details. First, if τ is an infinite vocabulary, we need to approximate it using finite sub-vocabularies. We assume that all computable vocabularies τ come with an associated effective approximation $\tau_0 \subseteq \tau_1 \subseteq \cdots \subseteq \tau$, where each τ_s is finite and $\tau = \bigcup_s \tau_s$. In general and unless otherwise stated, we let τ_s consist of the first s relation, constant and function symbols in τ, but in some particular cases, we might prefer other approximations. For instance, if τ is already finite, we usually prefer to let $\tau_s = \tau$ for all s. Second, to be able to approximate a τ-structure $\mathcal{M}$ using τ_s-substructures, we need the τ_s-reduct of $\mathcal{M}$ to be *locally finite*, i.e., every finite subset generates a finite substructure. To avoid unnecessary complications, we will just assume τ is relational and, in particular, locally finite. Even if τ is not originally relational, we can make it relational as in Section 1.1.4.

Definition 1.1.6. Given an ω-presentation $\mathcal{M}$, we let $\mathcal{M}_s$ be the finite τ_s-substructure of $\mathcal{M}$ with domain $\{0, \dots, s-1\}$. We call the sequence $\{\mathcal{M}_s : s \in \mathbb{N}\}$ a *finite approximation* of $\mathcal{M}$. We identify this sequence with the sequence of codes $\{D(\mathcal{M}_s) : s \in \mathbb{N}\} \subseteq 2^{<\mathbb{N}}$, which allows us to consider its computational complexity.

In general, when we refer to a $\tau_{|\cdot|}$-*structure*, we mean a τ_s-structure where s is the size of the structure itself. For instance, the structures $\mathcal{M}_s$ above are all $\tau_{|\cdot|}$-structures.

Observation 1.1.7. Here is a simple, but very important observation we will use throughout the book. For each s, $D(\mathcal{M}_s) = D(\mathcal{M}) \restriction \ell_s$, and hence

$$D(\mathcal{M}_0) \subseteq D(\mathcal{M}_1) \subseteq D(\mathcal{M}_2) \subseteq \cdots \quad \text{and} \quad D(\mathcal{M}) = \bigcup_{s \in \mathbb{N}} D(\mathcal{M}_s).$$

The convention here is that for each s, the τ_s-atomic formulas on the variables $\{x_0, \dots, x_{s-1}\}$ are listed before the rest; that is, they are $\varphi_0^{\mathrm{at}}, \dots, \varphi_{\ell_s - 1}^{\mathrm{at}}$ for some $\ell_s \in \mathbb{N}$.

Also, let us remark that the inclusion is an inclusion of stings, not of sets, and so is the union, as defined on page xv.

Thus, from a computational viewpoint, having an ω-presentation is equivalent to having a finite approximation of a structure $\mathcal{M}$. This is why, when we are working with an ω-presentation, we often visualize the structure as being given to us little by little.

Observation 1.1.8. Another simple but important observation is that an $\exists$-formula is true of a tuple $\bar{m}$ in $\mathcal{M}$ if and only if it is true in some finite substructure $\mathcal{M}_s$ that contains $\bar{m}$. Thus, if $\exists$-$Th(\mathcal{M})$ denotes the set of $\exists$-τ-sentences true of $\mathcal{M}$, and $\exists$-$Th(\mathcal{M}_s)$ the set of $\exists$-τ_s-sentences true of $\mathcal{M}_s$, then

$$\exists\text{-}Th(\mathcal{M}) = \bigcup_{s \in \mathbb{N}} \exists\text{-}Th(\mathcal{M}_s),$$

where the union here refers to the union of sets, not sequences.

As a useful technical device, we define the atomic diagram of a finite tuple as the finite binary sequence coding the set of atomic formulas true of the tuple restricted to the smaller vocabulary. Again, we assume that τ is relational.

Definition 1.1.9. Let $\mathcal{M}$ be a τ-structure and let $\bar{a} = \langle a_0, \dots, a_{s-1}\rangle \in M^s$. We define the *atomic diagram of $\bar{a}$ in $\mathcal{M}$, denoted $D_{\mathcal{M}}(\bar{a})$*, as the string in 2^{ℓ_s} such that

$$D_{\mathcal{M}}(\bar{a})(i) = \begin{cases} 1 & \text{if } \mathcal{M} \models \varphi_i^{\mathrm{at}}[x_j \mapsto a_j, j < s], \\ 0 & \text{otherwise.} \end{cases}$$

So, if $\mathcal{M}$ were an ω-presentation and $a_0, \dots a_s, \dots$ were the elements $0, \dots, s, \dots \in M = \mathbb{N}$, then $D_{\mathcal{M}}(\langle a_0, \dots, a_{s-1}\rangle) = D(\mathcal{M}_s)$ as in Definition 1.1.6.

Observation 1.1.10. For every $\sigma \in 2^{<\mathbb{N}}$ and every s with $\ell_s \geq |\sigma|$, there is a quantifier-free τ-formula $\varphi_\sigma^{\mathrm{at}}(x_0, \dots, x_{s-1})$ such that

$$\mathcal{A} \models \varphi_\sigma^{\mathrm{at}}(\bar{a}) \iff \sigma \subseteq D_{\mathcal{A}}(\bar{a})$$

for every τ-structure $\mathcal{A}$ and tuple $\bar{a} \in A^s$, namely

$$\varphi_\sigma^{at}(\bar{x}) \equiv \left(\bigwedge_{i<|\sigma|, \sigma(i)=1} \varphi_i^{at}(\bar{x}) \right) \wedge \left(\bigwedge_{i<|\sigma|, \sigma(i)=0} \neg\varphi_i^{at}(\bar{x}) \right).$$

1.1.6. Congruence structures. It will often be useful to consider structures where equality is interpreted by an equivalence relation. A *congruence τ-structure* is a structure $\mathcal{M} = (M; =^{\mathcal{M}}, \{R_i^{\mathcal{M}} : i \in I_R\}, \{f_i^{\mathcal{M}} : i \in I_F\}, \{c_i^{\mathcal{M}} : i \in I_C\})$, where $=^{\mathcal{M}}$ is an equivalence relation on M, and the interpretations of all the τ-symbols are invariant under $=^{\mathcal{M}}$ (that is, if $\bar{a} =^{\mathcal{M}} \bar{b}$, then $\bar{a} \in R_i^{\mathcal{M}} \iff \bar{b} \in R_i^{\mathcal{M}}$ and $f_j^{M}(\bar{a}) =^{\mathcal{M}} f_j(\bar{b})$ for all relations symbols R_i and function symbols f_j). If $M = \mathbb{N}$, we say that $\mathcal{M}$ is a *congruence ω-presentation*. We can then define $D(\mathcal{M})$ exactly as in Definition 1.1.2, using $=^{\mathcal{M}}$ to interpret equality.

Given a congruence τ-structure, one can always take the quotient $\mathcal{M}/{=^{\mathcal{M}}}$ and get a τ-structure where equality is the standard $\mathbb{N}$-equality. To highlight the difference, we will sometimes use the term *injective ω-presentations* when equality is $\mathbb{N}$-equality.

Lemma 1.1.11. *Given a congruence ω-presentation $\mathcal{M}$ with infinitely many equivalence classes, the quotient $\mathcal{M}/{=^{\mathcal{M}}}$ has an injective ω-presentation $\mathcal{A}$ computable from $D(\mathcal{M})$. Furthermore, the natural projection $\mathcal{M} \to \mathcal{A}$ is also computable from $D(\mathcal{M})$.*

Proof. All we need to do is pick a representative for each $=^{\mathcal{M}}$-equivalence class in a $D(\mathcal{M})$-computable way. Just take the $\mathbb{N}$-least element of each class: Let

$$A = \{a \in M : \forall b \in M\ (b <_{\mathbb{N}} a \Longrightarrow b \neq^{\mathcal{M}} a)\}$$

be the domain of $\mathcal{A}$. Define the functions and relations in the obvious way to get a ($\subseteq\omega$)-presentation of $\mathcal{M}$. To get an ω-presentation, use Observation 1.1.3. □

Therefore, from a computational viewpoint, there is no real difference in considering congruence structures or injective structures.

EXAMPLE 1.1.12. Suppose that $\mathcal{R}$ is a computable ring, and $I \subseteq R$ is a computable ideal. The quotient ring $\mathcal{R}/I$ has a natural congruence ω-presentation where the domain and the operations stay as in $\mathcal{R}$, but the equality relation $=^{\mathcal{R}/I}$ is the equivalence relation induced by I, namely $r =^{\mathcal{R}/I} q \iff r - q \in I$. We can then use the lemma above to get a computable injective ω-presentation of $\mathcal{R}/I$.

EXERCISE 1.1.13. Given a sequence of structures $\{\mathcal{A}_i : i \in \mathbb{N}\}$ and sequence of embeddings $f_{i,i+1}\colon \mathcal{A}_i \hookrightarrow \mathcal{A}_{i+1}$, the *direct limit* of such a sequence is a structure $\mathcal{A}_\infty$ for which there are embeddings $f_{i,\infty}\colon \mathcal{A}_i \to \mathcal{A}_\infty$ that commute with the previous embeddings (i.e, $f_{i,\infty} = f_{i+1,\infty} \circ f_{i,i+1}$ for all $i \in \mathbb{N}$), with the property that there is a increasing sequence of structures $\mathcal{B}_0 \subseteq \mathcal{B}_1 \subseteq \cdots \subseteq \mathcal{B}_\infty$, with $\mathcal{B}_\infty = \bigcup_s \mathcal{B}_s$, that is isomorphic to the original sequence, in the sense that there are isomorphisms $g_i\colon \mathcal{B}_i \to \mathcal{A}_i$ for $i \in \mathbb{N} \cup \{\infty\}$ such that $f_{i,j} \circ g_i = g_j \restriction B_i$ for all $i < j \in \mathbb{N} \cup \{\infty\}$. Prove that if the sequences $\{\mathcal{A}_i : i \in \mathbb{N}\}$ and $\{f_{i,i+1} : i \in \mathbb{N}\}$ of structures and embeddings are computable, then $\mathcal{A}_\infty$ has a computable copy.

1.1.7. Enumerations. Assume τ is a relational vocabulary. An *enumeration of a structure* $\mathcal{M}$ is just an onto map $g\colon \mathbb{N} \to M$. To each such enumeration we can associate a congruence ω-presentation $g^{-1}(\mathcal{M})$ by taking the *pull-back* of $\mathcal{M}$ through g:

$$g^{-1}(\mathcal{M}) = (\mathbb{N}; \sim, \{R_i^{g^{-1}(\mathcal{M})} : i \in I_R\}),$$

where $a \sim b \iff g(a) = g(b)$ and $R_i^{g^{-1}(\mathcal{M})} = g^{-1}(R_i^{\mathcal{M}}) \subseteq \mathbb{N}^{a(i)}$. The assumption that τ is relational was used here so that the pull-backs of functions and constants are not multi-valued. Let us remark that if g is injective, then $\sim$ becomes $=_{\mathbb{N}}$, and hence $g^{-1}(\mathcal{M})$ is an injective ω-presentation. In this case, the assumption that τ is relational is not important, as we can always pull-back functions and constants through bijections.

It is not hard to see that

$$D(g^{-1}(\mathcal{M})) \leq_T g \oplus D(\mathcal{M}).$$

Furthermore, $D(g^{-1}(\mathcal{M})) \leq_T g \oplus \tau^{\mathcal{M}}$, where $\tau^{\mathcal{M}}$ is as in Definition 1.1.1. As a corollary we get the following lemma.

LEMMA 1.1.14. *Let $\mathcal{A}$ be a computable structure in a relational vocabulary and M be an infinite c.e. subset of A. Then, the substructure $\mathcal{M}$ of $\mathcal{A}$ with domain M has a computable ω-presentation.*

PROOF. Just let g be an injective computable enumeration of $\mathcal{M}$. Then $g^{-1}(\mathcal{M})$ is a computable copy of $\mathcal{M}$. □

Throughout the book, there will be many constructions where we need to build a copy of a given structure with certain properties. In most cases, we will do it by building an enumeration of the structure and then taking the pull-back. The following observation will allow us to approximate the atomic diagram of the pull-back, and we will use it countless times.

Observation 1.1.15. Let g be an enumeration of $\mathcal{M}$. Notice that for every tuple $\bar{a} \in M^{<\mathbb{N}}$,

$$D_{g^{-1}(\mathcal{M})}(\bar{a}) = D_{\mathcal{M}}(g(\bar{a})).$$

For each k, use $g \restriction k$ to denote the tuple $\langle g(0), \dots, g(k-1)\rangle \in M^k$. Then $D_{g^{-1}(\mathcal{M})}(\langle 0, \dots, k-1\rangle) = D_{\mathcal{M}}(g \restriction k)$ and the diagram of the pull-back can be calculated in terms of the diagrams of tuples in $\mathcal{M}$ as follows:

$$D(g^{-1}(\mathcal{M})) = \bigcup_{k \in \mathbb{N}} D_{\mathcal{M}}(g \restriction k).$$

1.2. Presentations that code sets

In this section, we show that the Turing degrees of ω-presentations of a non-trivial structure can be arbitrarily high. Furthermore, we prove a well-known theorem of Julia Knight that states that the set of Turing degrees of the ω-presentations of a structure is upwards closed. This set of Turing degrees is called the *degree spectrum* of the structure, and we will study it in detail in Chapter 5. Knight's theorem applies only to non-trivial structures: A structure $\mathcal{A}$ is *trivial* if there is a finite tuple such that every permutation of the domain fixing that tuple is an automorphism. Notice that these structures are essentially finite in the sense that anything relevant about them happens within that finite tuple.

THEOREM 1.2.1 (Knight [Kni98]). *Suppose that X can compute an ω-presentation of a non-trivial τ-structure $\mathcal{M}$. Then there is an ω-presentation $\mathcal{A}$ of $\mathcal{M}$ of Turing degree X.*

Before proving the theorem, let us remark that if instead of an ω-presentation we wanted a $(\subseteq\omega)$-presentation or a congruence ω-presentation, it would be very easy to code X into either the domain or the equality relation of $\mathcal{A}$: Recall that $\mathcal{D}(\mathcal{A}) = A \oplus (=^{\mathcal{A}}) \oplus \tau^{\mathcal{A}}$. Requiring $\mathcal{A}$ to be an injective ω-presentation forces us to code X into the structural part of $\mathcal{A}$, namely $\tau^{\mathcal{A}}$.

PROOF. We will build an X-computable injective enumeration g of $\mathcal{M}$ and let $\mathcal{A} = g^{-1}(\mathcal{M})$. Since g and $\mathcal{M}$ are X-computable, that already gives us $D(\mathcal{A}) \leq_T X$; the actual work comes from ensuring that $D(\mathcal{A}) \geq_T X$. We build g as a limit

$$g = \bigcup_s \bar{p}_s \in M^{\mathbb{N}},$$

where the $\bar{p}_s$ are a nested sequence of injective tuples $\bar{p}_0 \subseteq \bar{p}_1 \subseteq \cdots$ in $M^{<\mathbb{N}}$. Recall from Observation 1.1.15 that we can approximate the atomic diagram of $\mathcal{A}$ by the atomic diagrams of the tuples $\bar{p}_s$:

$$D(\mathcal{A}) = \bigcup_{s \in \mathbb{N}} D_{\mathcal{M}}(\bar{p}_s).$$

Let $\bar{p}_0 = \emptyset$. Suppose now we have already defined $\bar{p}_s$. At stage $s+1$, we build $\bar{p}_{s+1} \supseteq \bar{p}_s$ with the objective of coding the bit $X(s) \in \{0,1\}$ into $D(\mathcal{A})$. The idea for coding $X(s)$ is as follows: We would like to find $a, b \in M \smallsetminus \bar{p}_s$ such that $D_{\mathcal{M}}(\bar{p}_s a) \neq D_{\mathcal{M}}(\bar{p}_s b)$. Suppose we find them and $D_{\mathcal{M}}(\bar{p}_s a) <_{lex} D_{\mathcal{M}}(\bar{p}_s b)$, where $\leq_{lex}$ is the lexicographical ordering on strings in $2^{<\mathbb{N}}$. Then, depending on whether $X(s) = 0$ or 1, we can define $\bar{p}_{s+1}$ to be either $\bar{p}_s ab$ or $\bar{p}_s ba$. To decode $X(s)$, all we have to do is compare the binary strings $D_{\mathcal{A}}\langle 0, \ldots, k_s - 1,\ k_s\rangle$ and $D_{\mathcal{A}}\langle 0, \ldots, k_s - 1,\ k_s + 1\rangle$ lexicographically, where $k_s = |\bar{p}_s|$.

The problem with this idea is that such a and b may not exist, and $D_{\mathcal{M}}(\bar{p}_s a)$ might be the same for all $a \in M$. Since $\mathcal{M}$ is non-trivial, we know there is some bijection of M preserving $\bar{p}_s$ which is not an isomorphism, and hence there exist tuples $\bar{a}$ and $\bar{b} \in (M \smallsetminus \bar{p}_s)^{<\mathbb{N}}$ of the same length with $D_{\mathcal{M}}(\bar{p}_s \bar{a}) \neq D_{\mathcal{M}}(\bar{p}_s \bar{b})$. Furthermore, there exists disjoint such $\bar{a}$ and $\bar{b}$: To see this, take a third tuple disjoint from $\bar{a}$ and $\bar{b}$. Its diagram must be different from that of either $\bar{a}$ or $\bar{b}$ (as those diagrams are different) and we can replace it for $\bar{b}$ or $\bar{a}$ accordingly, two get two disjoint tuples with different diagrams. So we search for such a pair of tuples $\bar{a}, \bar{b}$, say of length h. We also require the pair $\bar{a}, \bar{b}$ to be minimal, in the sense that $D_{\mathcal{M}}(\bar{p}_s a_0, \ldots, a_{i-1}) = D_{\mathcal{M}}(\bar{p}_s b_0, \ldots, b_{i-1})$ for $i < h$; if they are not, truncate them. Suppose $D_{\mathcal{M}}(\bar{p}_s \bar{a}) <_{lex} D_{\mathcal{M}}(\bar{p}_s \bar{b})$ (otherwise replace $\bar{a}$ for $\bar{b}$ in what follows). If $X(s) = 0$, let $\tilde{p}_{s+1} = \bar{p}_s a_0 b_0 a_1 b_1, \ldots, a_{h-1} b_{h-1}$. If $X(s) = 1$, let $\tilde{p}_{s+1} = \bar{p}_s b_0 a_0 b_1 a_1, \ldots, b_{h-1} a_{h-1}$. Finally, to make sure g is onto, we let $\bar{p}_{s+1} = \tilde{p}_{s+1} c$, where c is the $\mathbb{N}$-least element of $M \smallsetminus \tilde{p}_{s+1}$.

To recover X from $D(\mathcal{A})$, we need to also simultaneously recover the sequence of lengths $\{k_s : s \in \mathbb{N}\}$, where $k_s = |\bar{p}_s|$, for which we use the minimality of $\bar{a}$ and $\bar{b}$. Given k_s, we can compute k_{s+1} uniformly in $D(\mathcal{A})$ as follows: k_{s+1} is the least $k > k_s$ such that

$$D_{\mathcal{A}}(0, \ldots, k_s - 1,\ k_s, k_s + 2, k_s + 4, \ldots, k - 3) \neq D_{\mathcal{A}}(0, \ldots, k_s - 1,\ k_s + 1, k_s + 3, k_s + 5, \ldots, k - 2).$$

Once we know which of these two binary strings is lexicographically smaller, we can tell if $X(s)$ is 0 or 1: It is 0 if the former one is $<_{lex}$-smaller than the latter one. $\square$

Notice that for trivial structures, all presentations are isomorphic via computable bijections, and hence all presentations have the same Turing degree. When the vocabulary is finite, all trivial structures are computable.