1. Definitions and Preliminaries

A major part of this chapter serves to review and fix notation and terminology. The material is standard. Readers familiar with the notions addressed might therefore only want to refer back to particular definitions at later points. The main issues of the individual sections are the following:

- Section 1.1 sums up the basics about structures, global relations, logics and types that are relevant for our purposes.

- In Section 1.2 we consider algorithms that deal with structures as inputs and fix some corresponding conventions. Recognizability of classes of finite structures and computability of global relations are discussed.

- The bounded variable fragments of infinitary logic, and the fixed-point logics, are presented in Section 1.3. We also provide some typical examples for the expressive power of these logics.

- Section 1.4 contains some preliminary material about types and definability in the relevant fragments of infinitary logic.

- Section 1.5 deals with interpretations, a concept that plays an important rôle in many definability considerations.

- In Section 1.6 we review the notions of generalized quantifiers and Lindström extensions. In particular we define the class of cardinality Lindström quantifiers.

- Section 1.7 fixes some terminology with respect to the notion of canonization and of complete invariants for arbitrary equivalence relations. We also sketch some technicalities and conventions concerning orderings and preorderings.

1.1 Structures and Types

1.1.1 Structures

We deal with *finite structures* exclusively. $\mathrm{fin}[\tau]$ is the class of all finite τ-structures. Unless explicitly stated otherwise, τ stands for some finite and

purely relational vocabulary. A structure in $\mathrm{fin}[\tau]$ consists of its universe together with interpretations for the symbols in τ. If $\tau = \{R_1, \ldots, R_s\}$, where R_i is a relation or predicate symbol of arity r_i, we write $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \ldots, R_s^{\mathfrak{A}})$ for a τ-structure. Thus $R_i^{\mathfrak{A}} \subseteq A^{r_i}$. The superscripts $^{\mathfrak{A}}$ are mostly omitted when there is no danger of confusion.

It is sometimes convenient in special circumstances to admit some variations and extensions of the basic concept of structures:

(i) In order to deal with fixed tuples of *parameters* over some structure $\mathfrak{A}$, one may think of those parameters as interpretations for a corresponding tuple of extra constant symbols. We here prefer to stick with an entirely relational vocabulary and treat parameters as interpretations for variable symbols. The distinction between parameters and variables becomes purely intentional. The class of all τ-structures with fixed tuples of r parameters is denoted

$$\mathrm{fin}[\tau; r] = \Big\{ (\mathfrak{A}, \bar{a}) \;\Big|\; \mathfrak{A} \in \mathrm{fin}[\tau], \bar{a} \in A^r \Big\}.$$

(ii) In our formalization of fixed-point logic with counting in Chapter 4 we deal with *two-sorted structures*. These are structures over two disjoint universes, one for each sort. Each relation symbol comes with a specification telling which components range over the first sort and which over the second. Similarly terms and in particular variables have a designated status with respect to the sorts. There is a standard way to represent two-sorted structures by ordinary one-sorted structures that have two additional unary predicates to distinguish the sorts. A structure of the form

$$(A, U_1, U_2, \ldots) \quad \text{with } A = U_1 \dot{\cup} U_2$$

can thus naturally encode a two-sorted structure with universes U_i for the two sorts. A binary relation R for instance whose i-th component ranges over the i-th sort for $i = 1, 2$ then gets interpreted as a binary relation over A that satisfies $\forall x \forall y \big(Rxy \to U_1 x \wedge U_2 y\big)$.

(iii) At some places we consider *weighted structures*. These are structures together with some functions from their domains to some external standard domain, mostly and without loss of generality to the set ω of the natural numbers. A standard example is that of graphs (V, E) with weights put on the edges, formalized by a weight function $\nu\colon V^2 \to \omega$.

Linearly ordered structures play a special rôle. Assume that τ contains a designated binary relation symbol $<$ for a linear ordering. Then the class of all finite τ-structures which are linearly ordered is denoted

$$\mathrm{ord}[\tau] = \Big\{ \mathfrak{A} \in \mathrm{fin}[\tau] \;\Big|\; <^{\mathfrak{A}} \text{ a linear ordering of } A \Big\}.$$

When talking of *classes of finite structures* it is generally understood that these are closed with respect to isomorphism. The only exception in our treatment being that in places we restrict attention to structures over *standard*

domains, meaning structures with an initial segment of the natural numbers for their universe. We denote by $\mathrm{stan}[\tau]$ the set of all finite τ-structures over standard domains $n = \{0, \ldots, n-1\}$ [1]:

$$\mathrm{stan}[\tau] = \left\{ \mathfrak{A} \in \mathrm{fin}[\tau] \;\middle|\; A = n, n \geqslant 1 \right\}.$$

There is a direct correspondence between linearly ordered structures and structures over standard domains. Each structure $(\mathfrak{A}, <^{\mathfrak{A}})$ in $\mathrm{ord}[\tau \dot{\cup} \{<\}]$ has a unique representative $\langle \mathfrak{A} \rangle$ in $\mathrm{stan}[\tau]$ determined by the requirement that $\mathfrak{A} \simeq \langle \mathfrak{A} \rangle$ and that the linear ordering $<^{\mathfrak{A}}$ translates into the natural ordering on the standard domain of $\langle \mathfrak{A} \rangle$ under the isomorphism. Obviously the mapping $(\mathfrak{A}, <^{\mathfrak{A}}) \mapsto \langle \mathfrak{A} \rangle$ induces a bijective correspondence between isomorphism classes of linearly ordered structures and structures over standard domains

$$\langle\,\rangle\colon \mathrm{ord}[\tau \dot{\cup} \{<\}] \,/\!\simeq \longrightarrow \mathrm{stan}[\tau].$$

1.1.2 Queries and Global Relations

A class Q of finite τ-structures may be identified with a boolean valued functor χ on $\mathrm{fin}[\tau]$ that maps structures to 1 or 0 according to membership in Q: $Q = \{\mathfrak{A} \in \mathrm{fin}[\tau] \mid \chi(\mathfrak{A}) = 1\}$. The term *boolean queries* for classes of structures stresses this functorial view. Since classes of structures are tacitly assumed to be closed under isomorphisms, their characteristic functions χ are *invariant under isomorphisms.*

Consider similarly an isomorphism-invariant boolean valued function on $\mathrm{fin}[\tau; r]$, $\chi\colon \mathrm{fin}[\tau; r] \to \{0, 1\}$. Such a functor constitutes an *r-ary query on* $\mathrm{fin}[\tau]$. An alternative view is that of a mapping from $\mathfrak{A} \in \mathrm{fin}[\tau]$ to a new r-ary predicate $R^{\mathfrak{A}}$ over $\mathfrak{A}$:

$$R^{\mathfrak{A}} = \left\{ \bar{a} \in A^r \;\middle|\; \chi(\mathfrak{A}, \bar{a}) = 1 \right\}.$$

The mapping $R\colon \mathfrak{A} \mapsto R^{\mathfrak{A}}$ is a *global relation of arity r*. At the level of this mapping isomorphism invariance of χ turns into *equivariance under isomorphisms*: if $\pi\colon \mathfrak{A} \to \mathfrak{B}$ is an isomorphism, then $\pi(R^{\mathfrak{A}}) = R^{\mathfrak{B}}$. This is in fact the standard defining condition on global relations or queries as introduced in [CH80]. We note in particular that the value of a global relation over $\mathfrak{A}$ must be invariant under all automorphisms of $\mathfrak{A}$.

Definition 1.1. *A* global relation or query R of arity r *over* $\mathrm{fin}[\tau]$ *is a mapping sending each structure* $\mathfrak{A} \in \mathrm{fin}[\tau]$ *to an* r*-ary predicate* $R^{\mathfrak{A}} \subseteq A^r$ *in* $\simeq$*-compatible fashion. Whenever* $\pi : \mathfrak{A} \to \mathfrak{B}$ *is an isomorphism, then* π *also preserves* R*:* $\pi(R^{\mathfrak{A}}) = R^{\mathfrak{B}}$*. The characteristic functor* χ_R *of* R *is the boolean valued mapping on* $\mathrm{fin}[\tau; r]$ *that sends* $(\mathfrak{A}, \bar{a})$ *to* 1 *if* $\bar{a} \in R^{\mathfrak{A}}$*. Compatibility*

[1] We apply the usual convention to identify the natural number $n \in \omega$ with the set of its predecessors $\{0, \ldots, n-1\} \subseteq \omega$.

of R with isomorphisms is equivalent with invariance of χ_R under isomorphisms.

It is often convenient to regard boolean queries (boolean global relations) as special, namely 0-ary cases of r-ary queries. To accommodate this view formally, we may naturally identify 0-ary predicates with boolean values and $\mathrm{fin}[\tau;0]$ with $\mathrm{fin}[\tau]$.

Some remark on our usage of the term *functor* is in order. We generally apply it to a mapping f whose domain is a class of structures (or of structures with parameters and the like) if f is required to be invariant under isomorphisms: $\mathfrak{A} \simeq \mathfrak{B} \Rightarrow f(\mathfrak{A}) = f(\mathfrak{B})$. If also the range of f consists of structures, for instance $f\colon \mathrm{fin}[\tau] \to \mathrm{fin}[\sigma]$ then the appropriate form of invariance is $\mathfrak{A} \simeq \mathfrak{B} \Rightarrow f(\mathfrak{A}) \simeq f(\mathfrak{B})$.

1.1.3 Logics

Let $\mathcal{L}$ be a logic. We do not require any formal general notion of *a logic*; the apparent generality here only serves to collect some notions, that we later apply to a few individual concrete logics, into common statements. $\mathcal{L}[\tau]$ denotes the class of all formulae of $\mathcal{L}$ in vocabulary τ. A formula $\varphi \in \mathcal{L}[\tau]$ without free variables (one that semantically evaluates to a boolean value over each τ-structure) is a *sentence*. Sentences define classes of structures, concentrating on finite structures we put

$$\mathrm{fmod}(\varphi) := \left\{ \mathfrak{A} \in \mathrm{fin}[\tau] \;\middle|\; \mathfrak{A} \models \varphi \right\}.$$

We mostly use letters $\varphi, \psi, \chi, \ldots$ to denote formulae. Let $\varphi \in \mathcal{L}[\tau]$. Variables displayed in brackets like the x_i in $\varphi(x_1, \ldots, x_r)$ indicate that semantically we consider φ as defining a global relation of arity r on $\mathrm{fin}[\tau]$. Over $\mathfrak{A}$, $\varphi(x_1, \ldots, x_r)$ evaluates to the predicate

$$\varphi[\mathfrak{A}] := \left\{ \overline{a} \in A^r \;\middle|\; \mathfrak{A} \models \varphi[\overline{a}] \right\}.$$

$\mathfrak{A} \models \varphi[\overline{a}]$ says that φ is satisfied over $\mathfrak{A}$ when the free variables are interpreted as indicated. In this usage the notation $\varphi(x_1, \ldots, x_k)$ does not imply that the displayed x_i must all be syntactically free in φ, but that the *free variables* of φ are among those displayed. We speak of a formula *in free variables* $x_1, \ldots, x_k$ with this meaning: $\mathrm{free}(\varphi) \subseteq \{x_1, \ldots, x_k\}$. For instance, we allow to regard the formula $x_1 = x_2$ also as a formula in free variables x_1, x_2, x_3, and write $\varphi(x_1, x_2, x_3) = x_1 = x_2$ if this view is intended.

Similar conventions apply to second-order variables (predicate variables) where such occur. In particular notation like $\varphi(\overline{X}, \overline{x}) \in \mathcal{L}[\tau]$ indicates that given a τ-structure plus additional interpretations for the second-order variables $\overline{X}$ by extra predicates and for the $\overline{x}$ by elements, φ evaluates to a boolean value. $\mathfrak{A} \models \varphi[\overline{P}, \overline{a}]$ expresses that φ is satisfied in $\mathfrak{A}$ with the indicated interpretations for $\overline{X}$ and $\overline{x}$.

It is sometimes convenient to consider interpretations for some free first- or second-order variables as momentarily fixed. The notation $(\mathfrak{A}, \Gamma)$ for some partial interpretation of free variables through Γ indicates this meaning.

Definition 1.2. *(i) The sentence* $\varphi \in \mathcal{L}[\tau]$ defines *the boolean query* $Q \subseteq \mathrm{fin}[\tau]$ *if* $Q = \mathrm{fmod}(\varphi)$.

(ii) The formula $\varphi(x_1, \ldots, x_r) \in \mathcal{L}[\tau]$ defines *the global relation* R *on* $\mathrm{fin}[\tau]$ *if* $R^{\mathfrak{A}} = \varphi[\mathfrak{A}]$ *for all* $\mathfrak{A} \in \mathrm{fin}[\tau]$.

The expressive power of a logic is determined in terms of those global relations that are definable in this logic.

Definition 1.3. *Two logics are* semantically equivalent *if they define exactly the same global relations on finite structures.*

$$\mathcal{L}_1 \equiv \mathcal{L}_2$$

denotes this semantic equivalence over finite relational structures. The possible weakening of this requirement, that the two logics define the same classes of finite structures is explicitly indicated. We write "$\mathcal{L}_1 \equiv \mathcal{L}_2$ for sentences*" or "*$\mathcal{L}_1 \equiv \mathcal{L}_2$ for boolean queries*".*

Observe that $\mathcal{L}_1 \equiv \mathcal{L}_2$ says that for every formula of $\mathcal{L}_1$ there is a formula of $\mathcal{L}_2$ that is equivalent over finite structures, and vice versa. The weaker notion of equivalence expresses the same requirement in restriction to sentences. The distinction between the two notions of equivalence is of a purely formal nature for our considerations. Most natural logics admit a faithful reduction from definable global relations to definable boolean global relations so that their expressive power is fully determined by their strength in defining classes, i.e. by their sentences.

The notation $\equiv$ for semantic equivalence extends with analogous meaning to classes of queries that are not specified by logics. For instance if $\mathcal{C}$ is a class of queries and $\mathcal{L}$ a logic then $\mathcal{C} \equiv \mathcal{L}$ says that every query in $\mathcal{C}$ is definable by a formula of $\mathcal{L}$ and that conversely all $\mathcal{L}$-definable queries are in $\mathcal{C}$.

1.1.4 Types

We are interested in $\mathcal{L}$-definable properties of element tuples. The $\mathcal{L}$-type of a tuple $\overline{a} = (a_1, \ldots, a_k)$ of elements of a τ-structure $\mathfrak{A}$ is the class of all $\mathcal{L}$-formulae in free variables $\overline{x} = (x_1, \ldots, x_k)$ that are satisfied by $\overline{a}$ in $\mathfrak{A}$:

$$\mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\overline{a}) = \left\{ \varphi(\overline{x}) \in \mathcal{L}[\tau] \;\middle|\; \mathfrak{A} \models \varphi[\overline{a}] \right\}.$$

$\mathrm{Tp}^{\mathcal{L}}(\tau; k)$ is the class of all $\mathcal{L}[\tau]$-types in variables $x_1, \ldots, x_k$:

$$\mathrm{Tp}^{\mathcal{L}}(\tau; k) = \left\{ \mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\overline{a}) \;\middle|\; \mathfrak{A} \in \mathrm{fin}[\tau], \overline{a} \in A^k \right\}.$$

$\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A}; k)$ denotes the set of all $\mathcal{L}[\tau]$-types of k-tuples over a particular $\mathfrak{A}$:

$$\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A};k) = \left\{ \mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\bar{a}) \;\middle|\; \bar{a} \in A^k \right\}.$$

We use Greek letters $\alpha, \beta, \ldots$ to denote types. If $\alpha \in \mathrm{Tp}^{\mathcal{L}}(\tau;k)$ then $\alpha \models \varphi$ means that $\mathfrak{A} \models \varphi[\bar{a}]$ whenever $\mathfrak{A}$ and $\bar{a}$ are such that $\mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\bar{a}) = \alpha$. In case φ is also in $\mathcal{L}[\tau]$ this is just to say that $\varphi \in \alpha$.

We often think of $\mathrm{Tp}^{\mathcal{L}}(\tau;r)$ for $1 \leqslant r < k$ as embedded into $\mathrm{Tp}^{\mathcal{L}}(\tau;k)$ via

$$\mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(a_1,\ldots,a_r) \longmapsto \mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\underbrace{a_1,\ldots,a_1}_{k-r},a_1,\ldots,a_r).$$

Some of the logics that play a central rôle in the following possess only a bounded supply of variable symbols. If $\mathcal{L}$ only has variables $x_1,\ldots,x_k$ we agree to apply the notion of $\mathcal{L}$-type only to tuples $\bar{a}$ of length at most k. We adopt the convention that $\mathrm{Tp}^{\mathcal{L}}(\tau;r) = \emptyset$ for $r > k$ in this context.

The most basic types considered are the *atomic* or *quantifier free* types. They are obtained in the above formalism if $\mathcal{L}$ is chosen to be the quantifier free fragment of first-order logic. We write $\mathrm{atp}_{\mathfrak{A}}(\bar{a})$ for the collection of all quantifier free formulae that hold true of $\bar{a}$ in $\mathfrak{A}$. Note that each such type can be fully represented by the set of atomic formulae contained in it. In this sense, and for finite relational vocabularies, each atomic type is finite and we may identify an atomic type θ with a single quantifier free formula: the conjunction over all atomic formulae contained in θ together with the negations of all those not contained in θ. Some such syntactic normal form is tacitly assumed when we deal with sets of atomic types. The set of all atomic τ-types in variables $x_1,\ldots,x_k$ is denoted by $\mathrm{Atp}(\tau;k)$:

$$\mathrm{Atp}(\tau;k) = \left\{ \mathrm{atp}_{\mathfrak{A}}(\bar{a}) \;\middle|\; \mathfrak{A} \in \mathrm{fin}[\tau], \bar{a} \in A^k \right\}.$$

Clearly only structures of size up to k need be considered since τ is purely relational. For finite τ therefore, $\mathrm{Atp}(\tau;k)$ is obviously finite. In fact a finite representation of $\mathrm{Atp}(\tau;k)$ in terms of the above syntactic normalization is immediately obtained.

Atomic types in vocabulary $\emptyset$ — in the language of pure sets, where only equality is available — are here called *equality types*. We write $\mathrm{eq}(\bar{a})$ for the equality type of $\bar{a}$ and $\mathrm{Eq}(k)$ for the finite set of all equality types in variables $x_1,\ldots,x_k$.

For indistinguishability of structures or structures with parameters in a logic we use the following notation.

Definition 1.4. *For the logic $\mathcal{L}$ we denote by $\equiv^{\mathcal{L}}$ the equivalence relation of indistinguishability in $\mathcal{L}$ or* $\mathcal{L}$-equivalence *both of structures and of structures with parameters:*

(i) $\mathfrak{A} \equiv^{\mathcal{L}} \mathfrak{A}'$ *if* $\mathfrak{A}$ *and* $\mathfrak{A}'$ *satisfy exactly the same $\mathcal{L}$-sentences.*

(ii) $(\mathfrak{A},\bar{a}) \equiv^{\mathcal{L}} (\mathfrak{A}',\bar{a}')$ *if* $\bar{a}$ *and* $\bar{a}'$ *satisfy exactly the same $\mathcal{L}$-formulae over* $\mathfrak{A}$ *and* $\mathfrak{A}'$ *respectively — equivalently if* $\mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\bar{a}) = \mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}'}(\bar{a}')$.

Note that $\mathrm{Tp}^{\mathcal{L}}(\tau;k)$ may be identified with $\mathrm{fin}[\tau;k]\,/{\equiv^{\mathcal{L}}}$.

1.2 Algorithms on Structures

The informal notion of *algorithms on structures* simply refers to algorithms that are intended to take finite structures as inputs. Algorithms do not deal with abstract structures directly but with *presentations* or *encodings* of these. In standard models of computation — we mainly think of Turing machines — algorithms directly deal with words, ordered strings of symbols over some fixed finite alphabet. Straightforward encoding schemes that faithfully map structures to words are available for structures over standard domains. The implicit ordering of the standard domain allows to enumerate all instantiations of atoms lexicographically. The entire structural information can thus be coded in a binary string that lists the boolean values of all instantiations of atoms in this ordering. Having fixed any such convention for the direct encoding of standard structures we can identify standard structures with their encodings as bit-strings. Without loss of generality we may thus pretend that algorithms for computations on finite τ-structures directly accept elements of $\mathrm{stan}[\tau]$ as inputs. We think of such an algorithm $\mathcal{A}$ as realizing a mapping

$$\begin{aligned} \mathcal{A}\colon \mathrm{stan}[\tau] &\longrightarrow \mathrm{range}(\mathcal{A}) \\ \mathfrak{A} &\longmapsto \mathcal{A}(\mathfrak{A}). \end{aligned}$$

The same considerations apply to algorithms that take structures with parameters as inputs; we replace $\mathrm{stan}[\tau]$ by $\mathrm{stan}[\tau; r]$ for some r. With respect to the range of $\mathcal{A}$ we may distinguish two cases (the distinction is purely intentional): either we regard $\mathrm{range}(\mathcal{A})$ simply as a set of words, or we similarly identify output words with standard objects they encode. In particular we adopt the latter view if we want $\mathcal{A}$ to realize a mapping from structures to structures. As for the input domain, we identify the output domain with some $\mathrm{stan}[\sigma]$ and pretend for instance that $\mathcal{A}$ directly realizes a mapping $\mathcal{A}\colon \mathrm{stan}[\tau] \to \mathrm{stan}[\sigma]$. Similar conventions can be employed to algorithms which are to output natural numbers: $\mathcal{A}\colon \mathrm{stan}[\tau] \to \omega$.

Algorithms are unproblematic as far as they realize mappings between certain domains of standard objects (objects with standard encodings). The picture becomes fundamentally different if we want to realize *functors* on structures. Consider the algorithmic evaluation of a boolean query on $\mathrm{fin}[\tau]$. No matter whether we restrict the domain to $\mathrm{stan}[\tau]$ or not, there remains the crucial *invariance condition* that $\mathcal{A}(\mathfrak{A}) = \mathcal{A}(\mathfrak{A}')$ whenever $\mathfrak{A}$ and $\mathfrak{A}'$ are isomorphic. Note that this condition really arises from two sources:

(a) when encoding even a single abstract $\mathfrak{A} \in \mathrm{fin}[\tau]$ through an element of $\mathrm{stan}[\tau]$ then a priori the representative in $\mathrm{stan}[\tau]$ is determined only up to isomorphism.

(b) since a boolean query by definition corresponds to an isomorphism invariant functor, its restriction to $\mathrm{stan}[\tau]$ still has to be invariant under isomorphisms.

Of course there are situations in which uniquely determined representatives of abstract input structures by elements of $\mathrm{stan}[\tau]$ are available. Most notably this applies to linearly ordered structures. As pointed out above we may identify $\mathrm{ord}[\tau \dot{\cup} \{<\}]/\simeq$ with $\mathrm{stan}[\tau]$. Linearly ordered structures, even when viewed only up to isomorphism, therefore are themselves objects with standard encodings — whence their notorious special status in considerations concerning logics for complexity classes arises. Somewhat more generally such exceptions occur wherever there is some adequate normalization or canonization procedure available. Variations on this issue will concern us in later chapters.

Entirely similar considerations apply of course to the evaluation of r-ary queries that we choose to realize in the boolean format

$$\mathcal{A}\colon \mathrm{stan}[\tau; r] \longrightarrow \{0,1\}$$

which is also subject to invariance under isomorphism. Finally, functors from structures to structures are realized as

$$\mathcal{A}\colon \mathrm{stan}[\tau] \longrightarrow \mathrm{stan}[\sigma]$$

with invariance condition $\mathfrak{A} \simeq \mathfrak{B} \Rightarrow \mathcal{A}(\mathfrak{A}) \simeq \mathcal{A}(\mathfrak{B})$.

Definition 1.5. *(i) The algorithm* $\mathcal{A}\colon \mathrm{stan}[\tau] \to \{0,1\}$ computes the boolean query $Q \subseteq \mathrm{fin}[\tau]$ *if for all* $\mathfrak{A} \in \mathrm{stan}[\tau]$*:* $\mathcal{A}(\mathfrak{A}) = 1$ *if and only if* $\mathfrak{A} \in Q$*. We also say that* $\mathcal{A}$ recognizes *the class* Q*.*

(ii) An algorithm $\mathcal{A}\colon \mathrm{stan}[\tau; r] \to \{0,1\}$ computes the r-ary query R *on* $\mathrm{fin}[\tau]$ *if for all* $(\mathfrak{A}, \bar{a}) \in \mathrm{stan}[\tau; r]$*:* $\mathcal{A}(\mathfrak{A}, \bar{a}) = 1$ *if and only if* $\bar{a} \in R^{\mathfrak{A}}$*.*

(iii) An algorithm $\mathcal{A}\colon \mathrm{stan}[\tau] \to \mathrm{stan}[\sigma]$ computes the functor $F\colon \mathrm{fin}[\tau] \to \mathrm{fin}[\sigma]$ *if for all* $\mathfrak{A} \in \mathrm{stan}[\tau]$*:* $\mathcal{A}(\mathfrak{A}) \simeq F(\mathfrak{A})$*.*

(iv) An algorithm $\mathcal{A}\colon \mathrm{stan}[\tau] \to S$ computes the functor $F\colon \mathrm{fin}[\tau] \to S$ *whose range is some domain of standard objects* S *if for all* $\mathfrak{A} \in \mathrm{stan}[\tau]$*:* $\mathcal{A}(\mathfrak{A}) = F(\mathfrak{A})$*.*

1.2.1 Complexity Classes and Presentations

We are interested in the complexity of problems that concern structures. Consider a structural problem, any of the several kinds of computational problems considered in Definition 1.5. The complexity of such problems is the complexity in the standard sense of its algorithmic realizations $\mathcal{A}$. When dealing with relational input structures we identify the *input size* with the size of the universe of the input structure. Although this parameter may differ from the length of an actual encoding of the input structure, the difference does not matter for our purposes because the complexity classes considered — mainly PTIME and PSPACE, but the same applies to all standard classes from LOGSPACE upward — are robust under polynomially bounded re-scalings of the input size.

A boolean query $Q \subseteq \text{fin}[\tau]$, for instance, is in PTIME if there is a PTIME algorithm $\mathcal{A}$ that computes Q in the sense of Definition 1.5 (i). More precisely, if there is an algorithm $\mathcal{A}$ for Q that terminates its computation on all $\mathfrak{A} = (n, \ldots) \in \text{stan}[\tau]$ in time polynomial in n. It is customary to denote the *class of all* PTIME *queries* again by PTIME, and similar conventions apply to all the usual complexity classes. It should always be clear from context whether we think of for instance PTIME either as the class of all polynomial time computable functions (on the natural numbers, or on some other domain of objects with standard encodings), or as the class of all polynomial time computable queries on finite relational structures. In order to emphasize the latter interpretation we shall sometimes speak of PTIME or other complexity classes *as complexity classes of queries*, a notion introduced by Chandra and Harel [CH80].

As pointed out in the introduction the issue of logics for complexity classes is closely related with the abstract notion of *recursive presentations* for complexity classes of queries.

Definition 1.6. *Let $\mathcal{C}$ be a complexity class of queries. $\mathcal{C}$ is* recursively presented *by a recursive or recursively enumerable set $\mathcal{M}$ of algorithms if each $\mathcal{A} \in \mathcal{M}$ is an algorithmic realization of a query in complexity $\mathcal{C}$, and if $\mathcal{M}$ is semantically complete for $\mathcal{C}$: $\mathcal{C} = \{Q \mid Q$ realized by some $\mathcal{A} \in \mathcal{M}\}$.*

We write $\mathcal{C} \equiv \mathcal{M}$ to stress the underlying semantic equivalence and speak of $\mathcal{M}$ as a recursive presentation *for $\mathcal{C}$. For short we also just call $\mathcal{C}$* recursively enumerable *if it admits a recursive presentation.*

This notion of a recursive presentation similarly applies to any class of problems $\mathcal{C}$ that is specified by algorithmic criteria. Recall from the introduction that ordinary PTIME, as the class of all polynomial time computable problems on natural numbers say, is recursively presentable through the subclass of polynomially clocked PTIME machines (compare Section 0.1.2). PTIME as a class of queries is a paradigmatic semantic class. As a subclass of ordinary PTIME, PTIME as a class of queries is characterized by the *semantic condition of invariance under isomorphism.* The problem whether there are logics for complexity classes of queries essentially is the problem of finding recursive presentations of these semantically defined classes.

1.2.2 Logics for Complexity Classes

This following notion was first presented in precise terms in [Gur88].

Definition 1.7. *Let $\mathcal{C}$ be a complexity class of queries. Assume that $\mathcal{L}$ is a logic with recursive syntax and semantics: for finite τ the set $\mathcal{L}[\tau]$ of τ-formulae of $\mathcal{L}$ is recursive*[2] *and there is a recursive mapping from $\mathcal{L}[\tau]$ to*

[2] Note that just as for recursive presentations it does not really matter whether we require recursive or recursively enumerable syntax. If $(\varphi_i)_{i \geqslant 1}$ is a recursive enumeration then the recursive set $\{(\varphi_i, i) | i \geqslant 1\}$ can replace the original syntax if necessary.

algorithms, $\varphi \mapsto \mathcal{A}_\varphi$ such that $\mathcal{A}_\varphi$ evaluates (the query defined by) φ over $\mathrm{fin}[\tau]$.

$\mathcal{L}$ is a logic for $\mathcal{C}$ *or* captures $\mathcal{C}$ *if* $\mathcal{C}$ *coincides with the class of queries that are definable in* $\mathcal{L}$, $\mathcal{C} \equiv \mathcal{L}$, *and if the recursive semantics* $\varphi \mapsto \mathcal{A}_\varphi$ *maps* $\mathcal{L}[\tau]$ *to algorithmic realizations within* $\mathcal{C}$.

Of course the same notion applies to other classes of queries that are defined in terms of algorithmic criteria, in particular to subclasses of complexity classes of queries. The well known theorem of Immerman and Vardi for instance says that the class of all PTIME queries on ordered structures is captured in exactly this sense by fixed-point logic.

Sometimes it is useful to strengthen these requirements so that other important data also become recursive in terms of the formulae of $\mathcal{L}$, compare [Gur88, EF95]. For instance one may require that data describing complexity bounds on $\mathcal{A}_\varphi$ be recursive in φ. While such strengthenings are crucial for certain arguments we can here stick to the basic notion.

It is worth to note the essential equivalence between the notions of capturing by some logic and that of a recursive presentation. It is clear that a logical representation as in the last definition provides a recursive presentation through $\{\mathcal{A}_\varphi \mid \varphi \in \mathcal{L}\}$. Conversely, any recursive set of algorithms may be regarded as a logic with recursive syntax in the abstract sense; for the semantics choose the obvious one embodied in the algorithms. In this way any recursive presentation of $\mathcal{C}$ can essentially be regarded as a logic that captures $\mathcal{C}$. There are some fine points to be considered if as usual we want abstract logics to satisfy some appropriate regularity criteria as outlined in [Ebb85]. For this it is obviously necessary that $\mathcal{C}$ itself as a set of queries satisfies corresponding closure criteria. At least for classes $\mathcal{C}$ that are natural in such respects it follows that indeed the two notions are equivalent. We do not here enlarge on this issue, in fact an informal concept of 'logics for complexity classes' will be quite sufficient for our purposes.

1.3 Some Particular Logics

1.3.1 First-Order Logic and Infinitary Logic

We write $L_{\omega\omega}$ for first-order logic. The expressive power of first-order logic over finite structures is very unsatisfactory in terms of computational complexity. While all $L_{\omega\omega}$-definable queries are LOGSPACE computable, first-order logic fails to define fundamental structural properties in LOGSPACE or even below. This was briefly discussed in the introduction. First-order equivalence $\equiv^{L_{\omega\omega}}$, however, turns out to be too strong a notion of equivalence over finite structures. Two finite structures are first-order equivalent if and only if they are isomorphic: a first-order sentence, that uses enough variables to enumerate all elements of a given structure and specify all basic relations between them, characterizes that structure up to isomorphism.

Full infinitary logic is the logic $L_{\infty\omega}$ that has the usual first-order rules for the formation of formulae and in addition is closed under *infinitary conjunctions* and *disjunctions*: if Ψ is any set (!) of formulae of $L_{\infty\omega}$ then $\bigwedge\Psi$, the conjunction over Ψ, and $\bigvee\Psi$, the disjunction over Ψ, are also formulae of $L_{\infty\omega}$. Their semantics is the obvious one: $\bigwedge\Psi$ evaluates to true if all formulae in Ψ evaluate to true and $\bigvee\Psi$ evaluates to true if at least one formula in Ψ does. Note that one often has to deal with families of formulae $(\psi_i)_{i\in I}$ and then writes for instance $\bigwedge_{i\in I}\psi_i$ instead of $\bigwedge\{\psi_i \mid i \in I\}$.

As mentioned in the introduction *any* query on finite structures is definable in $L_{\infty\omega}$. This follows from the observation that any finite structure $\mathfrak{A}$ is characterized up to isomorphism by some first-order sentence $\varphi_{\mathfrak{A}}$. If Q is a boolean query, for instance, then the infinite disjunction $\psi = \bigvee\varphi_{\mathfrak{A}}$ over all $\varphi_{\mathfrak{A}}$ for $\mathfrak{A} \in Q \cap \mathrm{stan}[\tau]$ clearly defines Q. Recall, however, that $\varphi_{\mathfrak{A}}$ typically requires $n+1$ variables if the size of $\mathfrak{A}$ is n. This motivates the introduction of the finite variable fragments of $L_{\infty\omega}$.

1.3.2 Fragments of Infinitary Logic

Definition 1.8. *$L^k_{\infty\omega}$ is the fragment of $L_{\infty\omega}$ that consists of formulae using only variable symbols from $\{x_1, \ldots, x_k\}$. The union of the $L^k_{\infty\omega}$ is denoted $L^\omega_{\infty\omega}$. It consists of all formulae of $L_{\infty\omega}$ that use finitely many variable symbols (from the standard supply $\{x_i | i \geqslant 1\}$).*

We also consider the corresponding bounded variable fragments of $L_{\omega\omega}$: let $L^k_{\omega\omega}$ denote first-order logic with variable symbols $\{x_1, \ldots, x_k\}$.

The union of the $L^k_{\omega\omega}$ is full first-order logic $L_{\omega\omega}$. In actual formalizations we often use variable symbols x, y, z, ... instead of the standardized x_i for the sake of easier readability. The official restriction to standard sets of variables is convenient, however, to have syntactic closure under conjunctions and disjunctions for each $L^k_{\infty\omega}$. We give some examples for the expressive power of the $L^k_{\infty\omega}$. Formalizations with few variable symbols typically require clever re-use of already quantified variables. Examples 1.9 and 1.11 are from [KVa92a], Example 1.16 plays an important rôle in [DLW95] in a context that will also concern us here later.

Example 1.9. Over linear orderings $(A, <)$ two different variable symbols suffice to produce first-order formulae $\varphi_i(x)$, for $i \geqslant 0$, which express that x is the i-th element with respect to $<$. Equivalently, for the standard linear orderings $(n, <)$:

$$(n, <) \models \varphi_i[m] \quad \text{exactly for } m = i.$$

To obtain these formulae put $\varphi_0(x) := \neg\exists y\, y < x$ to define the bottom element in any linear ordering $(A, <)$. Inductively let

$$\varphi_{i+1}(x) := \bigwedge_{j\leqslant i} \neg\varphi_j(x) \wedge \forall y\Big(y < x \to \bigvee_{j\leqslant i} \varphi_j(y)\Big),$$

where $\varphi_j(y)$ is the result of exchanging x and y throughout the formula $\varphi_j(x)$.

Example 1.10. The class of acyclic directed graphs (A, E) is definable by a sentence of $L^2_{\infty\omega}$. Observe that a finite graph is acyclic if it has no infinite E-paths, or equivalently if there is some finite bound on the length of E-paths. Put $\xi_0(x) := \neg\exists y Eyx$ to characterize those vertices that have no E-predecessors. Inductively let

$$\xi_{i+1}(x) := \forall y\big(Eyx \to \xi_i(y)\big).$$

Then $(A, E) \models \xi_i[v]$ if and only if there is no E-path of length greater than i reaching v. It follows that

$$\xi := \bigvee_{i\in\omega} \forall x \xi_i(x)$$

characterizes acyclic directed finite graphs as desired.

The sequence of formulae ξ_i from Example 1.10 can be extended to ordinal indices to form formulae $\xi_\alpha(x)$ asserting (over arbitrary structures) that the *E-rank* of x is at most α: inductively $\xi_\alpha(x) = \forall y\big(Eyx \to \bigvee_{\beta<\alpha} \xi_\beta(y)\big)$. E is *well-founded* if there are no infinite descending E-paths, which is equivalent with the existence of some λ such that $(A, E) \models \bigvee_{\alpha<\lambda} \forall x \xi_\alpha(x)$. It follows that the class of well-founded relations of rank less than λ is $L^2_{\infty\omega}$-definable over arbitrary structures, for each λ. We shall return to two variables, linear orderings, and well-foundedness in Example 1.12 and Corollaries 1.13 and 1.14 below.

Example 1.11. The reflexive transitive closure of a binary relation E is definable in $L^3_{\infty\omega}$. The formula $\psi_1(x, y) := x{=}y \vee Exy$ describes the pairs of E-distance at most 1. Inductively, $\psi_{i+1}(x, y) := \psi_i(x, y) \vee \exists z\big(\psi_i(x, z) \wedge Ezy\big)$ defines those pairs (x, y), whose E-distance is at most $i + 1$. Thus $\xi(x, y) := \bigvee_{i\geqslant 1} \psi_i(x, y)$ defines the reflexive transitive closure of E.

It is a well known fact (that will also be illustrated in Example 2.6 with a typical game argument) that two variables do not suffice to define transitive closures or to assert transitivity of a given binary relation. The following observation, which is also a direct consequence of the very first exercise in Poizat's [Poi82], is therefore quite intriguing. The way it is proved here is inspired by an argument from [GOR96a].

Example 1.12. The class of finite linear orderings is $L^2_{\infty\omega}$-definable (even over not necessarily finite structures). Let ξ' be an $L^2_{\infty\omega}[<]$-sentence asserting that $<$ is acyclic (obtained from ξ in Example 1.10 through replacing E by $<$). We claim that

$$\begin{aligned} \varphi_{\text{ord}} &:= \xi' \wedge \varphi_0\,, \\ \text{where } \varphi_0 &:= \forall x \forall y\big(x{=}y \vee x{<}y \vee y{<}x\big), \end{aligned}$$

defines the class of all finite linear orderings. It remains to argue that φ_{ord} enforces

(i) irreflexivity — $\forall x \neg x < x$: it does since ξ' forbids loops.
(ii) antisymmetry — $\forall x \forall y \neg\big(x < y \wedge y < x\big)$: it does since ξ' also forbids cycles of length two.
(iii) transitivity — $\forall x \forall y \forall z\big(x < y \wedge y < z \to x < z\big)$: if $x < y$ and $y < z$ then $x \neq z$ (forbidden cycle of length two) and $z \not< x$ (forbidden cycle of length three), whence φ_0 forces $x < z$.

In fact, over not necessarily finite structures and for any ordinal λ the class of all well-orderings of order type less than λ is $L^2_{\infty\omega}$-definable. It follows that for instance the class of all countable well-orderings is $L^2_{\infty\omega}$-definable over arbitrary structures. This claim is justified just as in the last example, if we replace ξ' by a sentence that expresses that $<$ is a well-founded relation of rank less than λ (compare the remark following Example 1.10). ξ' in particular forbids loops and cycles, so that irreflexivity, antisymmetry and transitivity follow as above.

Returning to finite structures, we have the following:

Corollary 1.13. *Let $<\in \tau$ and let τ have no relation symbols of arity greater than 2. Then any query of linearly ordered finite τ-structures $Q \subseteq \text{ord}[\tau]$ is $L^2_{\infty\omega}$-definable.*

Sketch of Proof. Consider without loss $\tau = \{<, E\}$ with one extra binary relation E besides $<$. Let $\mathfrak{A} = (n, <, E)$ have standard domain with the natural interpretation of $<$. Put, for formulae φ_i as in Example 1.9 that characterize the i-th element of the ordering,

$$\varphi_{\mathfrak{A}} := \varphi_{\text{ord}} \wedge \forall x \bigvee_{i<n} \varphi_i(x) \wedge \bigwedge_{(i,j)\in E} \exists x \exists y\big(\varphi_i(x) \wedge \varphi_j(y) \wedge Exy\big) \wedge \bigwedge_{(i,j)\notin E} \exists x \exists y\big(\varphi_i(x) \wedge \varphi_j(y) \wedge \neg Exy\big).$$

Then $\varphi_{\mathfrak{A}}$ characterizes $\mathfrak{A}$ up to isomorphism. The disjunction $\bigvee \varphi_{\mathfrak{A}}$ over those of these $\mathfrak{A}$ that are in Q defines Q. □

In particular, for any set $W \subseteq \omega \setminus \{0\}$, the class of those linear orderings whose size is in W is definable in $L^2_{\infty\omega}$. This immediately gives the following:

Corollary 1.14. *$L^2_{\infty\omega}$ is sufficiently expressive to define arbitrarily complex and even non-recursive queries.*

Example 1.15. The class of all finite trees is definable in $L^3_{\infty\omega}$. A structure (A, E) is a *tree* if E is acyclic and if

(i) there is exactly one element without E-predecessors (the root).
(ii) each element has at most one E-predecessor.

Note that connectedness is implied by (i) and (ii) if there cannot be cycles. Now cycles can be forbidden in $L^2_{\infty\omega}$ according to Example 1.10. (i) can even be formalized in $L^2_{\omega\omega}$ through

$$\exists x \forall y \neg Eyx \wedge \forall x \forall y \Big(\big(\forall y \neg Eyx \wedge \forall x \neg Exy \big) \longrightarrow x = y \Big).$$

(ii) actually needs a third variable for mere counting, as for instance in the formalization

$$\forall x \forall y \forall z \big(Eyx \wedge Ezx \rightarrow y = z \big).$$

A *binary tree* is a tree in which all nodes have out-degree 0 or 2. A *full binary tree* is a binary tree in which all leaves (nodes of out-degree 0) are at the same distance from the root (at the same height).

Example 1.16. The class of all full binary trees is definable in $L^3_{\infty\omega}$. The condition on equal height of all leaves is formalized in three variables using auxiliary formulae φ_i that define the set of vertices at height i. These are constructed inductively similar to the formulae used in Example 1.9. $\varphi_0(x) := \neg \exists y Eyx$ defines the root. Inductively $\varphi_{i+1}(x) := \exists y (\varphi_i(y) \wedge Eyx)$ is as desired. The $L^2_{\infty\omega}$-sentence $\bigvee_i \forall x \big(\neg \exists y Exy \rightarrow \varphi_i(x) \big)$ forces all leaves to be at the same height.

It might look surprising that the condition on out-degree 2 should also be expressible in just three variables. It is however, and quite simply, in the context of trees. The sentence

$$\neg \exists x_1 \exists x_2 \exists x_3 \Big(\bigwedge_{i \neq j} x_i \neq x_j \wedge \bigwedge_{i=1,2,3} \Big(\exists x_i \bigwedge_{j \neq i} Ex_i x_j \Big) \Big)$$

expresses that there are no three vertices with common direct predecessors for any two of them. If the in-degree can at most be 1 then this is equivalent with a bound of 2 on the out-degree. The condition that the out-degree of all vertices apart from the leaves must be at least 2 is trivially first-order expressible in three variables.

In Example 1.15 some variables had to be spent for mere counting in the conditions on the degree of vertices. In general the explicit formalization of $\exists^{\geqslant m} x \varphi(x)$ requires at least m variables as for instance in

$$\exists x_1 \ldots \exists x_m \Big(\bigwedge_{1 \leqslant i < j \leqslant m} x_i \neq x_j \wedge \bigwedge_{1 \leqslant i \leqslant m} \varphi(x_i) \Big).$$

Indeed, over the empty vocabulary it follows easily from game arguments that are treated in Chapter 2, that no sentence of $L^{m-1}_{\infty\omega}$ can express the existence of at least m distinct elements. It is therefore natural to consider fragments of first-order and infinitary logic with a bounded supply of variables that admit, however, arbitrary *counting quantifiers* $\exists^{\geqslant m}$.

Definition 1.17. *$C^k_{\infty\omega}$ is the fragment of $L_{\infty\omega}$ that consists of formulae using only variable symbols from $\{x_1, \ldots, x_k\}$ but allows arbitrary counting quantifiers $\exists^{\geqslant m}$, $m \geqslant 1$, instead of the standard existential quantifier $\exists$. The union of the $C^k_{\infty\omega}$ is denoted $C^\omega_{\infty\omega}$.*

Some simplifications in actual formalizations are achieved with the following agreements. We write $\exists^{=m} x\varphi(x)$ as an abbreviation for $\exists^{\geqslant m} x\varphi(x) \wedge \neg\exists^{\geqslant m+1} x\varphi(x)$. This notation may be extended to allow $\exists^{=0} x\varphi(x)$ as shorthand for $\neg\exists x\varphi(x)$. Quantifiers $\exists^{>m}$, $\exists^{\leqslant m}$ and $\exists^{<m}$ are similarly introduced.

We give a few simple examples for the expressive power of $C^2_{\infty\omega}$. A much more central example, the expressibility of the stable colouring of graphs in $C^2_{\infty\omega}$, is presented in detail in Chapter 2.

Example 1.18. The class of regular graphs is definable in $C^2_{\infty\omega}$. Apart from the standard axioms for graphs, which are in two variables, take

$$\chi := \forall x \forall y \bigwedge_{m\in\omega} \left(\exists^{\geqslant m} yExy \leftrightarrow \exists^{\geqslant m} xEyx\right)$$

to express regularity.

Example 1.19. Even $C^2_{\omega\omega}$ does not have the *finite model property*, i.e. there are sentences of $C^2_{\omega\omega}$ without finite models that are satisfiable in infinite structures. For instance it is expressible in $C^2_{\omega\omega}$ that a binary relation E is the graph of an injective total function that is not surjective:

$$\forall x \exists^{=1} yExy \wedge \forall y \exists^{\leqslant 1} xExy \wedge \exists y \forall x \neg Exy.$$

First-order logic with two variables $L^2_{\omega\omega}$ on the other hand is known to have the finite property by a result of Mortimer's, see Theorem 7.35.

Example 1.20. Any finite equivalence relation is characterized up to isomorphism among all finite equivalence relations by its $C^2_{\infty\omega}$-theory. The axiomatization of equivalence relations, however, needs at least three variables. Let $\mathfrak{A} = (A, \sim)$ where $\sim$ is an equivalence relation over A. A complete invariant that characterizes any finite equivalence relation up to isomorphism is its spectrum, the set of pairs (i, ν_i) such that there are exactly ν_i classes of size i in $(A, \sim)$. This information is expressed in two variables by

$$\bigwedge_i \exists^{=i\nu_i} x\, \exists^{=i} y\; x \sim y.$$

That two variables even with counting quantifiers do not suffice to express transitivity of a binary relation E is shown easily with games discussed in Chapter 2, see Example 2.6.

Example 1.21. The classes of all trees and of all full binary trees are definable in $C^2_{\infty\omega}$. In fact we observed in Example 1.15 and 1.16 that a third variable was only needed for counting purposes. For instance (ii) of Example 1.15 now simply becomes $\forall x \exists^{\leqslant 1} yEyx$.

1.3.3 Fixed-Point Logics

Fixed-point logics provide extensions of first-order logic that capture some *recursion on relations*. Consider a formula $\varphi(X, \overline{x})$ in free variables X and $\overline{x}$ of matching arities, X a second-order variable for a predicate of arity r and $\overline{x} = (x_1, \ldots, x_r)$. Over structures $\mathfrak{A}$ that interpret φ up to X and the $\overline{x}$ the formula φ induces a mapping on r-ary predicates:

$$\begin{aligned} F_\varphi^{\mathfrak{A}} \colon \mathcal{P}(A^r) &\longrightarrow \mathcal{P}(A^r) \\ P &\longmapsto \{\overline{a} \in A^r \mid \mathfrak{A} \models \varphi[P, \overline{a}]\}. \end{aligned}$$

$\mathcal{P}(A^r)$ denotes the power set of A^r. For any mapping $F \colon \mathcal{P}(A^r) \to \mathcal{P}(A^r)$ consider the following two recursive processes.

The partial fixed point of F. Inductively, a sequence of r-ary predicates is generated through iterated application of F to the empty predicate:

$$\begin{aligned} X_0 &:= \emptyset \\ X_{i+1} &:= F(X_i). \end{aligned}$$

With this sequence the *partial fixed point* of F is defined to be either the stationary value of this sequence if such exists or the empty set otherwise:

$$\mathrm{PFP}[F] := \begin{cases} X_i & \text{if } X_{i+1} = X_i \\ \emptyset & \text{if } X_{i+1} \neq X_i \text{ for all } i. \end{cases}$$

The inductive or inflationary fixed point associated with F. Inductively, the following *increasing* sequence of r-ary predicates is generated:

$$\begin{aligned} X_0 &:= \emptyset \\ X_{i+1} &:= X_i \cup F(X_i). \end{aligned}$$

Note that this sequence may alternatively be obtained through iterated application of a modified operation F_{infl} to the empty predicate. F_{infl} is the *inflationary* variant of F, defined by $F_{\mathrm{infl}}(P) := P \cup F(P)$; generally an operation G on sets is called inflationary if always $G(P) \supseteq P$.

Since the domain A is finite, the sequence of the X_i must become stationary within polynomially many steps of the iteration. The limit reached is the *inductive or inflationary fixed point*:

$$\mathrm{IFP}[F] := X_{i_0} \quad \text{where } i_0 = \min\{i \mid X_{i+1} = X_i\}.$$

Equivalently: $\mathrm{IFP}[F] = \bigcup_i X_i$,

or $\mathrm{IFP}[F] = \mathrm{PFP}[F_{\mathrm{infl}}]$.

The X_i in these definitions are referred to as the *stages* in the generation of the respective fixed points.

Fixed-point logics provide constructors for the definition of those predicates that are obtained as fixed points of operators $F = F_\varphi$ as above. For the

inductive definition of their syntax and semantics it is important to consider formulae φ that may have other free first- and second-order variables than X and $\overline{x}$. To make clear which variables are those involved in the fixed-point process, we write $\mathrm{PFP}_{X,\overline{x}}\varphi$ for the partial fixed point associated with the operator F_φ, where free variables other than X and $\overline{x}$ in φ are kept fixed as parameters with respect to the operation F_φ. The same applies to the inductive fixed point.

Syntax and semantics of partial fixed-point logic PFP are defined as follows. Syntactically, PFP is the closure of atomic formulae (where second-order variables are admitted) under the usual first-order constructions and the partial fixed-point constructor. The latter allows to construct a new formula $\psi = [\mathrm{PFP}_{X,\overline{x}}\varphi]\overline{x}$ from any formula $\varphi \in \mathrm{PFP}$, where X is a second-order variable of some arity r and $\overline{x}$ an r-tuple of distinct first-order variables. For the free occurrence of variables put $\mathrm{free}(\psi) := (\mathrm{free}(\varphi) \cup \{\overline{x}\}) \setminus \{X\}$.

The semantics for the PFP-constructor is that induced by the partial fixed point of the corresponding operator. If $(\mathfrak{A}, \Gamma)$ is a structure of the appropriate vocabulary together with interpretations for all variables in $\mathrm{free}(\varphi) \setminus \{X, \overline{x}\}$, then

$$(\mathfrak{A}, \Gamma) \models \psi[\overline{a}] \quad \text{if} \quad \overline{a} \in \mathrm{PFP}[F_\varphi^{\mathfrak{A},\Gamma}],$$

where $F_\varphi^{\mathfrak{A},\Gamma}\colon \mathcal{P}(A^r) \to \mathcal{P}(A^r)$ is the operation induced over $(\mathfrak{A}, \Gamma)$ by φ.

Definition 1.22. *Partial fixed-point logic* PFP *is the smallest logic closed under the usual first-order constructors and the* PFP*-constructor.*

Clearly $\mathrm{PFP} \subseteq \textsc{Pspace}$. Recall that first-order queries are in $\textsc{Logspace}$. Over structures of size n PFP-processes in arity r terminate within at most $2^m + 1$ steps, where $m = n^r$. A corresponding step counter can be run in $\textsc{Pspace}$ and only two stages of the fixed-point generation need be kept simultaneously.

For *fixed-point logic* FP one considers the inductive or inflationary fixed-point operator instead of the partial fixed-point operator. The syntax is exactly the same as for PFP, only that we write FP instead of PFP in fixed-point formulae and choose the semantics based on the inductive fixed point of the underlying operation: for φ and $(\mathfrak{A}, \Gamma)$ as above, and $\psi = [\mathrm{FP}_{X,\overline{x}}\varphi]\overline{x}$ put

$$(\mathfrak{A}, \Gamma) \models \psi[\overline{a}] \quad \text{if} \quad \overline{a} \in \mathrm{IFP}[F_\varphi^{\mathfrak{A},\Gamma}].$$

Definition 1.23. *Fixed-point logic* FP *is the smallest logic closed under first-order constructors and the* FP*-constructor.*

It is obvious that $\mathrm{FP} \subseteq \textsc{Ptime}$ as IFP-processes terminate within a polynomial number of iterations.

We regard FP as a sublogic of PFP in the sense that $[\mathrm{FP}_{X,\overline{x}}\varphi]\overline{x}$ may be identified with $[\mathrm{PFP}_{X,\overline{x}}(X\overline{x} \vee \varphi)]\overline{x}$, since the inflationary variant of the operation F_φ is the same as $F_{\varphi'}$ for $\varphi' = X\overline{x} \vee \varphi$.

Theorem 1.24 (Immerman, Vardi). *Over the class of linearly ordered finite structures* FP *captures* PTIME*: a class of linearly ordered structures is recognized by a* PTIME *algorithm if and only if it is the class of finite models of some sentence of* FP*; a global relation on ordered structures is* PTIME *computable if and only if it is definable in* FP.

Theorem 1.25 (Abiteboul, Vardi, Vianu). *Over linearly ordered finite structures* PFP *captures* PSPACE*: a class of linearly ordered structures is recognized by a* PSPACE *algorithm if and only if it is the class of finite models of some sentence of* PFP*; a global relation on ordered structures is* PSPACE *computable if and only if it is definable in* PFP.

Theorem 1.24 was obtained in [Imm86] and [Var82]. Theorem 1.25 combines results from [Var82] and [AV89] (via equivalences with relational While-queries).

Let us briefly indicate how the fixed-point processes available in FP and PFP can be used to code PTIME, respectively PSPACE, computations over linearly ordered structures — an observation that is the key to the preceding theorems.

Example 1.26. Polynomially space bounded configurations of a Turing machine can be encoded by predicates over the ordered domains of the input structures. The arity of the predicate depends on the degree of the space bound. A definable indexing of the tape cells is provided by the lexicographic ordering on an appropriate power of the universe. In terms of these encodings by predicates it is easily seen that the successor step from one configuration to the next becomes first-order definable.

For a PSPACE machine that computes a boolean query say, consider the partial fixed-point process based on this first-order formalization of the computational successor (with a corresponding definition of the initial configuration). Provided that we adapt the transition function of the machine in such a way that an eventual halting configuration is formally repeated indefinitely, the limit value of this partial fixed-point process is an encoding of the halting configuration if the machine halts on the given input structure.

For a PTIME machine we may pass from the encoding of individual configurations to a cumulative encoding of initial segments of the computation path. We use a fixed-point variable with additional entries for a lexicographic indexing also for a polynomial number of time steps. This leads to an inductive fixed-point process whose i-th level describes the computation path up to step i. The limit reached in this process is an encoding of the entire computation path on the given input structure.

In either case the actual result of the computation (acceptance or rejection in the case of a boolean query) is first-order definable in terms of the final configuration.

The original and intuitively also very appealing definition of fixed-point logic FP in terms of a *least fixed-point operator* LFP rather than the in-

ductive or inflationary operator IFP is equivalent in expressive power. This equivalence between the least fixed-point extension of first-order logic and the formalization using IFP is shown by Gurevich and Shelah [GS86], see also [Lei90]. The formalization using inductive fixed points will be more convenient in connection with the counting extensions to be introduced in Chapter 4.

As a further indication of the versatility of fixed-point constructs we briefly consider *systems of simultaneous fixed points.*

Example 1.27. Let $\overline{\varphi} = (\varphi_i(X_1, \ldots, X_m, \overline{x}^{(i)}))_{i=1,\ldots,m}$ be a tuple of formulae in which the arities r_i of the $\overline{x}^{(i)}$ match those of the X_i. With $\overline{\varphi}$ one may associate an operation $F_{\overline{\varphi}}$ on $\mathcal{P}(A^{r_1}) \times \cdots \times \mathcal{P}(A^{r_m})$, over structures that interpret the φ_i up to the X_i and $\overline{x}^{(i)}$, through:

$$\overline{P} \longmapsto \left(\{ \overline{a}^{(i)} \mid (\mathfrak{A}, \overline{P}) \models \varphi_i[\overline{a}^{(i)}] \} \right)_{i=1,\ldots,m} .$$

Iteration in the spirit of IFP or PFP yields inflationary or partial fixed points for such systems of interrelated relational transformations. Standard techniques for the encoding of tuples of relations through a single relation of higher arity can be employed to show that fixed-point logic FP and partial fixed-point logic PFP are closed under the formation of respective fixed points for systems.

1.3.4 Fixed-Point Logics and the $L^k_{\infty\omega}$

The fixed-point logics PFP and FP both are sublogics of $L^\omega_{\infty\omega}$. Since FP itself is a sublogic of PFP it suffices to review the argument that yields PFP $\subseteq L^\omega_{\infty\omega}$. Since we have already seen that $L^\omega_{\infty\omega}$ expresses queries of arbitrary complexity, whereas PFP-definable queries obviously are in PSPACE, it follows that PFP $\subsetneq L^\omega_{\infty\omega}$.

For a convenient statement of the following arguments it is useful to eliminate first-order parameters in fixed-point processes. This is easily done at the expense of an increase in the arity of the second-order fixed-point variable.

Lemma 1.28. *Any formula in* PFP *is equivalent with one in which fixed-point operators are applied only in the form* $\mathrm{PFP}_{X,\overline{x}}\varphi$*, where all free first-order variables of* φ *are among those in* $\overline{x}$*. The same holds of* FP.

Sketch of Proof. Consider φ with free first-order variables $\overline{x}$, $\overline{z}$. Assume that $\overline{x}$ and $\overline{z}$ together consist of pairwise distinct variables and that no variable in $\overline{z}$ is bound in φ. Let φ' be the result of replacing any atom $X\overline{u}$ in φ by the atom $Z\overline{z}\,\overline{u}$, where Z is a new second-order variable whose arity is that of X plus arity of $\overline{z}$. It is easily seen that $\{\overline{z}\,\overline{x} \mid [\mathrm{PFP}_{X,\overline{x}}\varphi]\overline{x}\} = \{\overline{z}\,\overline{x} \mid [\mathrm{PFP}_{Z,\overline{z}\,\overline{x}}\varphi']\overline{z}\,\overline{x}\}$. □

Lemma 1.29. *Let* $\varphi \in L^k_{\infty\omega}$ *be a formula, possibly with free second-order variables. Let* X *be a second-order variable of arity* r*, and* $\overline{x}$ *an* r*-tuple of*

pairwise distinct variables from $\{x_1, \ldots, x_k\}$. *Assume that all free first-order variables of* φ *are among those in* $\overline{x}$. *Then* $\psi := [\mathrm{PFP}_{X,\overline{x}}\varphi]\overline{x}$ *is equivalent with a formula in* $L^k_{\infty\omega}$.

Proof. Let us write $\varphi(X, \overline{x})$ for the given φ, second-order variables apart from X are irrelevant in the argument. Without loss of generality we assume that all X-atoms in φ are of the form $X\overline{y}$ for an r-tuple of mutually distinct variables $\overline{y}$. An atom $Xx_ix_i\overline{y}'$ for instance can be replaced by the formula $\exists x_j(x_j = x_i \wedge Xx_jx_i\overline{y}')$ for a variable x_j different from x_i and the $\overline{y}'$.

It is shown inductively that the stages X_i in the generation of the fixed point $\mathrm{PFP}_{X,\overline{x}}\varphi(X, \overline{x})$ are definable by $L^k_{\infty\omega}$-formulae $\varphi_i(\overline{x})$. $\varphi_0(\overline{x})$ is the result of replacing each X-atom in $\varphi(\overline{x})$ by some universally false expression like $\neg x = x$ in the same variables. Inductively assume that $\varphi_i(\overline{x})$ is as desired. Semantically $\varphi_{i+1}(\overline{x})$ has to be the result of substituting $\{\overline{x} \mid \varphi_i(\overline{x})\}$ for X in $\varphi(\overline{x})$. Consider a single atom $X\overline{y}$ in $\varphi(\overline{x})$, $\overline{y}$ a tuple of mutually distinct variables. Choose a permutation of the set of variables $\{x_1, \ldots, x_k\}$ that maps $\overline{x}$ to $\overline{y}$. An application of this permutation to all variables in $\varphi_i(\overline{x})$ yields a formula $\varphi_i(\overline{y})$ with $X_i = \{\overline{y} \mid \varphi_i(\overline{y})\}$ that can be substituted in place of $X\overline{y}$ in $\varphi(\overline{x})$ without any clashes with bound variables. $\varphi_{i+1}(\overline{x})$ is the result of corresponding substitutions for all X-atoms in $\varphi(\overline{x})$. It follows that $[\mathrm{PFP}_{X,\overline{x}}\varphi(X, \overline{x})]\overline{x}$ is equivalent with $\bigvee_{i\geqslant 0}\Big(\forall\overline{x}(\varphi_i(\overline{x}) \leftrightarrow \varphi_{i+1}(\overline{x})) \wedge \varphi_i(\overline{x})\Big)$. □

Corollary 1.30. $\mathrm{FP} \subseteq \mathrm{PFP} \subsetneq L^\omega_{\infty\omega}$.

The first semantic inclusion is strict if and only if PTIME $\subsetneq$ PSPACE by a theorem of Abiteboul and Vianu. We shall come back to this in Chapter 3.

Note that the $L^k_{\infty\omega}$ are indeed not closed with respect to PFP or FP, since fixed points may involve first-order parameters. The elimination of these according to Lemma 1.28 may need extra variables. An easy example to this effect is the following one in the language of a single binary relation E.

Example 1.31. The formula $\varphi(x, y) := [\mathrm{FP}_{X,x}(x{=}y \vee \exists y(Xy \wedge Eyx))]x$ defines the reflexive transitive closure of E. If φ were equivalent with a formula in $L^2_{\infty\omega}$ it would follow that transitivity of a symmetric reflexive relation E is expressible in $L^2_{\infty\omega}$ since it is expressed by $\forall x\forall y\varphi(x, y)$. As we shall see in Example 2.6 below, transitivity is not even definable in $C^2_{\infty\omega}$.

Another corollary to Lemma 1.29 concerns a frequently used collapsing argument over structures that realize few $L^k_{\infty\omega}$-types.

Corollary 1.32. *Let* $\mathcal{K} \subseteq \mathrm{fin}[\tau]$ *be a class such that for some* d *each* $\mathfrak{A} \in \mathcal{K}$ *realizes at most* d *different* $L^k_{\infty\omega}$*-types. Then any fixed point over an* $L^k_{\infty\omega}$*-formula without first-order parameters as considered in Lemma 1.29 is reached after at most* $2^d + 1$ *iterations. It follows in particular that* $L^k_{\omega\omega}$ *is closed with respect to fixed points without first-order parameters over* $\mathcal{K}$.

Sketch of Proof. Under the assumptions and for $\varphi(X,\overline{x})$ as in Lemma 1.29, there are no more than 2^d different $L^k_{\infty\omega}$-definable k-ary relations over any $\mathfrak{A} \in \mathcal{K}$ that can occur as stages in the fixed-point iteration. This is because each $L^k_{\infty\omega}$-definable relation over $\mathfrak{A}$ corresponds to a union of $L^k_{\infty\omega}$-types over A^k. Therefore, in the notation of the proof of Lemma 1.29, $[\mathrm{PFP}_{X,\overline{x}}\varphi]\overline{x}$ is equivalent over $\mathcal{K}$ with

$$\psi(\overline{x}) := \forall\overline{x}\big(\varphi_m(\overline{x}) \leftrightarrow \varphi_{m+1}(\overline{x})\big) \wedge \varphi_m(\overline{x}),$$

for $m = 2^d$. If φ is in $L^k_{\omega\omega}$, then so is ψ. □

1.4 Types and Definability in the $L^k_{\infty\omega}$ and $C^k_{\infty\omega}$

The following lemma applies to the fragments of infinitary logic considered above. Since they provide conjunctions and disjunctions over arbitrary sets of formulae, they are in particular *closed with respect to countable conjunctions and disjunctions.* Generally, a logic $\mathcal{L}$ is closed with respect to countable conjunctions and disjunctions if for any family $(\varphi_i)_{i\in\omega}$ of formulae in $\mathcal{L}$ there are $\mathcal{L}$-formulae with semantics corresponding to the conjunction and disjunction over the φ_i, respectively. As usual we write $\bigwedge_{i\in\omega}\varphi_i$ and $\bigvee_{i\in\omega}\varphi_i$ for these. *Closure under negation* is similarly defined. The lemma is a consequence of the fact that, for fixed finite vocabulary τ, $\mathrm{fin}[\tau]$ and $\mathrm{fin}[\tau;k]$ are countable up to isomorphism.

Lemma 1.33. *Let $\mathcal{L}$ be closed under negation and with respect to countable conjunctions and disjunctions, τ a finite vocabulary. Then*

(i) each $\mathfrak{A} \in \mathrm{fin}[\tau]$ is characterized up to $\equiv^{\mathcal{L}}$ by some sentence $\varphi_{\mathfrak{A}} \in \mathcal{L}[\tau]$, i.e. for all $\mathfrak{A}' \in \mathrm{fin}[\tau]$: $\mathfrak{A}' \equiv^{\mathcal{L}} \mathfrak{A} \iff \mathfrak{A}' \models \varphi_{\mathfrak{A}}$.

(ii) each $\mathcal{L}$-type over $\mathrm{fin}[\tau]$ is isolated *by a formula of $\mathcal{L}$, i.e. for all $\alpha \in \mathrm{Tp}^{\mathcal{L}}(\tau;r)$ there is a formula $\varphi_\alpha(x_1,\dots,x_r) \in \mathcal{L}[\tau]$ such that for all $(\mathfrak{A},\overline{a}) \in \mathrm{fin}[\tau;r]$: $\mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\overline{a}) = \alpha \iff \mathfrak{A} \models \varphi[\overline{a}]$.*

(iii) a global relation on $\mathrm{fin}[\tau]$ is definable *in $\mathcal{L}$ if and only if it is closed with respect to $\equiv^{\mathcal{L}}$.*
In the boolean case, $Q \subseteq \mathrm{fin}[\tau]$ is $\mathcal{L}$-definable if for all $\mathfrak{A}, \mathfrak{A}' \in \mathrm{fin}[\tau]$: $\mathfrak{A} \equiv^{\mathcal{L}} \mathfrak{A}' \Longrightarrow \big(\mathfrak{A} \in Q \Leftrightarrow \mathfrak{A}' \in Q\big)$.
For an r-ary global relation R on $\mathrm{fin}[\tau]$, R is $\mathcal{L}$-definable if for all $(\mathfrak{A},\overline{a})$, $(\mathfrak{A}',\overline{a}') \in \mathrm{fin}[\tau;r]$: $(\mathfrak{A},\overline{a}) \equiv^{\mathcal{L}} (\mathfrak{A}',\overline{a}') \Longrightarrow \big(\overline{a} \in R^{\mathfrak{A}} \Leftrightarrow \overline{a}' \in R^{\mathfrak{A}'}\big)$.

Recall that by convention for logics like $L^k_{\infty\omega}$ and $C^k_{\infty\omega}$, which only have variables $x_1,\dots,x_k$, we do not consider types in more than k variables: $\mathrm{Tp}^{\mathcal{L}}(\tau;r) = \emptyset$ for $r > k$. The statement in (iii) has to to be restricted to arities $r \leqslant k$ accordingly.

Proof. (i) Since $\mathcal{L}$, as any logic, cannot distinguish between isomorphic structures, and since $\mathrm{fin}[\tau]/\simeq$ is countable, it follows that also $\mathrm{fin}[\tau]/\equiv^{\mathcal{L}}$ is countable. Let $(\mathfrak{A}_i)_{i\in I}$, for I finite or $I=\omega$, be a system of representatives for $\mathrm{fin}[\tau]/\equiv^{\mathcal{L}}$. For $i,j\in I, i\neq j$ let $\varphi_{ij}\in\mathcal{L}[\tau]$ be such that $\mathfrak{A}_i\models\varphi_{ij}$ and $\mathfrak{A}_j\models\neg\varphi_{ij}$. It follows that each $\mathfrak{A}_i$ is characterized up to $\equiv^{\mathcal{L}}$ by the $\mathcal{L}$-sentence $\varphi_i:=\bigwedge_{j\neq i}\varphi_{ij}$. For the claim about boolean global relations in (iii) assume that Q is $\equiv^{\mathcal{L}}$-closed. Then the disjunction $\bigvee\varphi_i$ over those i, for which $\mathfrak{A}_i\in Q$, defines Q.
(ii) and that part of (iii) that concerns r-ary queries are proved in exactly the same way using a system of representatives for $\mathrm{Tp}^{\mathcal{L}}(\tau;r)=\mathrm{fin}[\tau;r]/\equiv^{\mathcal{L}}$ instead of $\mathrm{fin}[\tau]/\equiv^{\mathcal{L}}$. □

Corollary 1.34. *For $\mathcal{L}=L^k_{\infty\omega}$ or $C^k_{\infty\omega}$:*

$$\mathfrak{A}\equiv^{\mathcal{L}}\mathfrak{A}' \quad \textit{if and only if} \quad \mathrm{Tp}^{\mathcal{L}}(\mathfrak{A};k)=\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A}';k).$$

Sketch of Proof. Let $\varphi_{\mathfrak{A}}$ characterize $\mathfrak{A}$ up to $\equiv^{\mathcal{L}}$ as in (i) of Lemma 1.33. Then $\mathfrak{A}\equiv^{\mathcal{L}}\mathfrak{A}'$ iff $\alpha\models\varphi_{\mathfrak{A}}$ for some (any) type $\alpha\in\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A}';k)$. Therefore $\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A};k)=\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A}';k)$ implies $\mathfrak{A}\equiv^{\mathcal{L}}\mathfrak{A}'$.

Conversely assume that there is some $\alpha\in\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A};k)\setminus\mathrm{Tp}^{\mathcal{L}}(\mathfrak{A}';k)$. Let $\varphi_\alpha(\overline{x})$ be as in (ii) of Lemma 1.33. Then $\mathfrak{A}\models\exists\overline{x}\varphi_\alpha(\overline{x})$ whereas $\mathfrak{A}'\models\neg\exists\overline{x}\varphi_\alpha(\overline{x})$. □

We apply the observations of Lemma 1.33 to the $L^k_{\infty\omega}$, $C^k_{\infty\omega}$, and to their *fragments of bounded quantifier rank.* Note that these are all closed with respect to infinitary conjunctions and disjunctions (as well as under negation) while $L^\omega_{\infty\omega}$ and $C^\omega_{\infty\omega}$ are not!

Definition 1.35. *The* quantifier rank *of a formula in $L_{\infty\omega}$ is given by an ordinal-valued function* qr *that is inductively defined by:*

$$\mathrm{qr}(\varphi)=0 \quad \textit{for atomic } \varphi, \qquad \mathrm{qr}(\textstyle\bigwedge\Psi)=\mathrm{qr}(\bigvee\Psi)=\sup\{\mathrm{qr}(\psi)\mid\psi\in\Psi\},$$
$$\mathrm{qr}(\neg\varphi)=\mathrm{qr}(\varphi), \qquad \mathrm{qr}(\exists x\varphi)=\mathrm{qr}(\forall x\varphi)=\mathrm{qr}(\varphi)+1.$$

The quantifier rank of formulae in $C^\omega_{\infty\omega}$ is defined similarly, with the last clause replaced by $\mathrm{qr}(\exists^{\geqslant m}x\varphi)=\mathrm{qr}(\varphi)+1$ for all m.

Definition 1.36. *For $\mathcal{L}=L^k_{\infty\omega},L^k_{\omega\omega},C^k_{\infty\omega}$ or $C^k_{\omega\omega}$ and $m\in\omega$ let $\mathcal{L}_{;m}$ denote the fragment defined through restriction to those formulae that have quantifier rank at most m.*

In the absence of counting quantifiers, infinitary logic of finitely bounded quantifier rank collapses to the corresponding fragment of first-order logic as follows.

Lemma 1.37. *Let τ be finite and relational, $m\in\omega$. Any $L^k_{\infty\omega;m}[\tau]$-formula is equivalent with a formula in $L^k_{\omega\omega;m}[\tau]$. There are only finitely many formulae in $L^k_{\omega\omega;m}[\tau]$ up to logical equivalence. It follows that for $\mathcal{L}:=L^k_{\infty\omega;m}$*

also $\mathrm{fin}[\tau]\ /\!\equiv^{\mathcal{L}}$ *and* $\mathrm{fin}[\tau;k]\ /\!\equiv^{\mathcal{L}}$ *are finite. Hence each structure in* $\mathrm{fin}[\tau]$ *is characterized up to* $\equiv^{\mathcal{L}}$ *even by a sentence of* $L^k_{\omega\omega;m}$ *and all* $\mathcal{L}$*-types are isolated by formulae in* $L^k_{\omega\omega;m}$.

Sketch of Proof. The proof is by induction on m. The case of quantifier free formulae is clear. For the induction let $\Psi_m \subseteq L^k_{\omega\omega;m}[\tau]$ be a finite system of representatives for all of $L^k_{\infty\omega;m}[\tau]$. Then up to logical equivalence all formulae of quantifier rank at most $m+1$ are boolean combinations over the finite set $\Psi \cup \{\exists x_i \psi \mid \psi \in \Psi, 1 \leqslant i \leqslant k\}$. □

The same claim cannot be made for the $C^k_{\infty\omega;m}$, because there are infinitely many counting quantifiers. Over structures of fixed finite size n, however, only quantifiers $\exists^{\geqslant s}$ for $s \leqslant n$ are non-trivial. This fact is used in the following lemma.

Lemma 1.38. *Let* $\mathcal{L} := C^k_{\infty\omega;m}$, $m \in \omega$, τ *finite and relational. Then each* $\alpha \in \mathrm{Tp}^{\mathcal{L}}(\tau;k)$ *is isolated by a formula* $\varphi_\alpha(x_1,\ldots,x_k) \in C^k_{\omega\omega;m}$ *and each* $\mathfrak{A} \in \mathrm{fin}[\tau]$ *is characterized up to* $\equiv^{\mathcal{L}}$ *by a sentence of* $C^k_{\omega\omega;m}$.

Proof. Consider the claim for types. The claim is trivial for $m = 0$; so assume $m \geqslant 1$. Let $\alpha = \mathrm{tp}^{\mathcal{L}}_{\mathfrak{A}}(\bar{a})$, $(\mathfrak{A},\bar{a}) \in \mathrm{fin}[\tau;k]$ with $|A| = n$. Let $\mathrm{fin}_n[\tau]$ and $\mathrm{fin}_n[\tau;k]$ denote the restrictions of $\mathrm{fin}[\tau]$ and $\mathrm{fin}[\tau;k]$ to structures of size n. It is obvious that in restriction to $\mathrm{fin}_n[\tau]$, each formula in $C^k_{\infty\omega;m}[\tau]$ is equivalent with one that only uses quantifiers $\exists^{\geqslant s}$ for $s \leqslant n$, since any $\exists^{\geqslant s} x_i \varphi$ with $s > n$ is universally false on $\mathrm{fin}_n[\tau]$. An adaptation of the argument in the proof of Lemma 1.37 shows that up to equivalence there are only finitely many formulae in $C^k_{\infty\omega;m}[\tau]$ of this kind, and that these can all be represented up to equivalence by formulae in $C^k_{\omega\omega;m}[\tau]$. Therefore again, $\mathrm{fin}_n[\tau;k]\ /\!\equiv^{\mathcal{L}}$ is finite, and each such type is isolated by a formula of $C^k_{\omega\omega;m}[\tau]$ within $\mathrm{fin}_n[\tau;k]\ /\!\equiv^{\mathcal{L}}$. Let $\psi_\alpha \in C^k_{\omega\omega;m}[\tau]$ isolate α in $\mathrm{fin}_n[\tau;k]\ /\!\equiv^{\mathcal{L}}$. Then $\varphi_\alpha := \exists^{=n} x\, x = x \wedge \psi_\alpha$ is in $C^k_{\omega\omega;m}[\tau]$ and isolates α in $\mathrm{Tp}^{\mathcal{L}}(\tau;k) = \mathrm{fin}[\tau;k]\ /\!\equiv^{\mathcal{L}}$. □

We sum up these observations as follows.

Lemma 1.39. *Let* τ *be a finite relational vocabulary.*

(i) For $\mathcal{L} = C^k_{\infty\omega}$ *or* $L^k_{\infty\omega}$, *each type in* $\mathrm{Tp}^{\mathcal{L}}(\tau;k)$ *is isolated by some formula of* $\mathcal{L}[\tau]$.

(ii) If $\mathcal{L} = C^k_{\infty\omega;m}$ *or* $\mathcal{L} = L^k_{\infty\omega;m}$, $m \in \omega$, *then furthermore each type in* $\mathrm{Tp}^{\mathcal{L}}(\tau;k)$ *is isolated by a formula of* $C^k_{\omega\omega;m}[\tau]$ *or* $L^k_{\omega\omega;m}[\tau]$, *respectively.*

As an immediate consequence of (ii) in the lemma we get the following.

Corollary 1.40. *Let* τ *be finite and relational,* $m \in \omega$. *Then over* $\mathrm{fin}[\tau]$ *and over all* $\mathrm{fin}[\tau;r]$ *for* $r \leqslant k$, $C^k_{\infty\omega;m}$*-equivalence coincides with* $C^k_{\omega\omega;m}$*-equivalence and* $L^k_{\infty\omega;m}$*-equivalence coincides with* $L^k_{\omega\omega;m}$*-equivalence.*

The analysis of the games for $C^k_{\infty\omega}$ and $L^k_{\infty\omega}$ in the next chapter will extend this observation from the bounded quantifier fragments to the logics $C^k_{\infty\omega}$, $C^k_{\omega\omega}$ and $L^k_{\infty\omega}$, $L^k_{\omega\omega}$ themselves. See Corollaries 2.3 and 2.4.

1.5 Interpretations

Interpretations concern the definition of one structure within another. The basic form is that of a *direct interpretation.* An $\mathcal{L}[\tau]$-formula $\varphi(x_1,\ldots,x_r)$ in free variables $x_1,\ldots,x_r$ defines an r-ary global relation on $\mathrm{fin}[\tau]$: the value of this relation over $\mathfrak{A} \in \mathrm{fin}[\tau]$ is

$$\varphi[\mathfrak{A}] = \left\{ \overline{a} \in A^r \;\middle|\; \mathfrak{A} \models \varphi[\overline{a}] \right\}.$$

Alternatively, φ may be viewed as defining a structure of vocabulary $\{R\}$, R an r-ary relation symbol, over each $\mathfrak{A} \in \mathrm{fin}[\tau]$, namely the structure $\left(A, \varphi[\mathfrak{A}]\right)$. To obtain defined structures in an arbitrary finite relational vocabulary σ we use tuples of formulae, one for each relation symbol in σ.

Definition 1.41. *Let* $\sigma = \{R_1,\ldots,R_l\}$, R_i *of arity* r_i. *An* $\mathcal{L}$*-definable* (σ,τ)-interpretation *is given by a tuple* $\overline{\varphi} = \left(\varphi_i(\overline{x}^{(i)})\right)_{1\leqslant i\leqslant l}$ *of formulae in* $\mathcal{L}[\tau]$, *with* $\overline{x}^{(i)} = (x_1,\ldots,x_{r_i})$. *The* σ*-structure interpreted by* $\overline{\varphi}$ *over* $\mathfrak{A} \in \mathrm{fin}[\tau]$ *is*

$$\left(A, \overline{\varphi}[\mathfrak{A}]\right) := \left(A, \varphi_1[\mathfrak{A}],\ldots,\varphi_l[\mathfrak{A}]\right).$$

1.5.1 Variants of Interpretations

One may first of all allow other free variables in the φ_i to obtain interpretations with parameters (first- and second-order). This is also true of all the variants considered in the following.

Relativized interpretations. Relativizations serve to restrict the universe of the interpreted structure to a definable subset of the parent structure. Let σ and $\overline{\varphi}$ be as above, $\varphi_0(x)$ an extra τ-formula in one free variable. The σ-structure interpreted by $\overline{\varphi}$ over φ_0 on $\mathfrak{A}$ is defined if $\varphi_0[\mathfrak{A}] \neq \emptyset$. It is the restriction of the above to $\varphi_0[\mathfrak{A}]$:

$$\left(A, \overline{\varphi}[\mathfrak{A}]\right) \restriction \varphi_0[\mathfrak{A}] = \left(\varphi_0[\mathfrak{A}], \varphi_1[\mathfrak{A}] \cap \left(\varphi_0[\mathfrak{A}]\right)^{r_1},\ldots,\varphi_l[\mathfrak{A}] \cap \left(\varphi_0[\mathfrak{A}]\right)^{r_l}\right).$$

Interpretations in powers. It is often natural to regard not the given universe but some power of it as the domain for the interpreted structure. This leads to the concept of an *interpretation over some power of the universe.* Let σ be as above. Let $\overline{\varphi} = \left(\varphi_i(\overline{x}^{(i)})\right)_{1 \leqslant i \leqslant l}$ be a tuple of formulae in $\mathcal{L}[\tau]$, where now φ_i has to be in sr_i distinct free variables. We index these so as to indicate a natural identification of the sr_i-tuple with an r_i-tuple of s-tuples:

$$\overline{x}^{(i)} = \left(x_{(1,1)}, \ldots, x_{(1,s)}, \ldots, x_{(r_i,1)}, \ldots, x_{(r_i,s)}\right).$$

Definition 1.42. *The σ-structure* interpreted by $\overline{\varphi}$ over the s-th power of $\mathfrak{A} \in \mathrm{fin}[\tau]$ *is*

$$\left(A^s, \overline{\varphi}[\mathfrak{A}]\right) = \left(A^s, \varphi_1[\mathfrak{A}], \ldots, \varphi_l[\mathfrak{A}]\right),$$

where $\varphi_i[\mathfrak{A}] \subseteq A^{sr_i}$ is regarded as an r_i-ary predicate over A^s.

Interpretations in quotients. Another variant that occurs in many natural applications further admits that the universe of the interpreted structure is represented as the quotient with respect to a definable equivalence relation. In order to yield a well defined σ-structure, the given equivalence relation must be compatible with the defined σ-predicates. An equivalence relation $\sim$ is a *congruence* for some r-ary predicate R if the following is satisfied for all $\overline{x} = (x_1, \ldots, x_r)$ and $\overline{x}' = (x'_1, \ldots x'_r)$: $\bigwedge_i x_i \sim x'_i \longrightarrow \left(R\overline{x} \leftrightarrow R\overline{x}'\right)$.

Let σ be as above, $\overline{\varphi}$ in a format appropriate for the direct interpretation of a σ-structure. Let in addition $\psi(x, x')$ be in the format for the interpretation of a binary relation $\sim$.

Definition 1.43. *The σ-structure* interpreted as a quotient with respect to ψ by $\overline{\varphi}$ over $\mathfrak{A} \in \mathrm{fin}[\tau]$ *is defined if the binary relation $\psi[\mathfrak{A}]$ is a congruence with respect to the predicates $\overline{\varphi}[\mathfrak{A}]$. In this case it is the quotient structure* $(A, \overline{\varphi}[\mathfrak{A}]) \,/\, \psi[\mathfrak{A}]$.

It is instructive to think of the congruence defined by ψ as a definition of the equality relation for the interpreted structure.

Note that these variants are not mutually exclusive. Quite to the contrary all combinations are possible and in fact occur naturally, see Example 1.45 below. One may speak for instance of an interpretation as a quotient over the s-th power of the universe, meaning that what is interpreted in the s-th power is itself the interpretation of a σ-structure as a quotient. The most general notion of interpretation we want to consider is that of a relativized interpretation as a quotient in some power. It subsumes the others as special cases.

Definition 1.44. *A* generalized (σ, τ)-interpretation *is an interpretation of σ-structures as relativizations in a quotient over some s-th power of τ-structures.*

For σ as above such an interpretation is specified by formulae φ_0, $\overline{\varphi}$ and ψ where the φ_i in $\overline{\varphi}$ are of arities sr_i, φ_0 and ψ of arities s and $2s$, respectively. Let $\mathfrak{i} = (\varphi_0; \overline{\varphi}; \psi)$ denote this interpretation itself and $\mathfrak{i}(\mathfrak{A})$ the interpreted structure over $\mathfrak{A}$:

$$\mathfrak{i}(\mathfrak{A}) = \Big(\big(A^s, \overline{\varphi}[\mathfrak{A}]\big) \restriction \varphi_0[\mathfrak{A}] \Big) \Big/ \psi[\mathfrak{A}].$$

$\mathfrak{i}(\mathfrak{A})$ is defined if $\varphi_0[\mathfrak{A}]$ is non-empty and if $\psi[\mathfrak{A}]$ interprets a congruence with respect to the $\overline{\varphi}[\mathfrak{A}] \restriction \varphi_0[\mathfrak{A}]$. It is useful to note that as a (partially defined) mapping

$$\begin{array}{rcl} \mathfrak{i}\colon \mathrm{fin}[\tau] & \longrightarrow & \mathrm{fin}[\sigma] \\ \mathfrak{A} & \longmapsto & \mathfrak{i}(\mathfrak{A}) \end{array}$$

an interpretation $\mathfrak{i}$ is a functor that preserves isomorphism. In particular the interpreted structure must be invariant under all automorphisms of the parent structure.

1.5.2 Examples

Example 1.45. The dual of a symmetric graph is interpretable in first-order logic as the relativization of a quotient in the second power. Let $\mathfrak{G} = (V, E)$ be a symmetric graph. Its dual is the graph $\overline{\mathfrak{G}}$, whose vertices are the edges of $\mathfrak{G}$, with an edge connecting two different ones of these if they share a common vertex of $\mathfrak{G}$. An edge of $\mathfrak{G}$ is an element of $E/\sim$ where two different pairs (v_1, v_2) and (v_1', v_2') in E represent the same edge of $\mathfrak{G}$, if $\{v_1, v_2\} = \{v_1', v_2'\}$. Two different edges represented by (v_1, v_2) and (v_1', v_2') share a common vertex if $\{v_1, v_2\} \cap \{v_1', v_2'\} \neq \emptyset$. Thus the dual of $\mathfrak{G}$ is the quotient

$$\Big(\big(V^2, \varphi_1[\mathfrak{G}]\big) \restriction \varphi_0[\mathfrak{G}] \Big) \Big/ \psi[\mathfrak{G}], \text{ where}$$

$$\begin{aligned} &\varphi_0(v_1, v_2) = Ev_1v_2\,, \\ &\varphi_1(v_1, v_2, v_1', v_2') = \text{“}\{v_1, v_2\} \neq \{v_1', v_2'\}\text{”} \wedge \text{“}\{v_1, v_2\} \cap \{v_1', v_2'\} \neq \emptyset\text{”}, \\ &\psi(v_1, v_2, v_1', v_2') = \text{“}\{v_1, v_2\} = \{v_1', v_2'\}\text{”}. \end{aligned}$$

With $\mathfrak{i} = (\varphi_0; \varphi_1; \psi)$ this is an interpretation of the dual. Explicit first-order formulae for the expressions in quotes are immediately supplied.

Example 1.46. A pre-ordering is a binary relation $\preccurlyeq$ that is reflexive, transitive, and connex, see Definition 1.62 below. It is easily checked, that $(A, \preccurlyeq)$ satisfies these axioms if and only if (i) – (iii):

(i) $\psi(x, y) = x \preccurlyeq y \wedge y \preccurlyeq x$ interprets an equivalence relation $\sim$ on A.
(ii) $\sim$ is a congruence with respect to $\preccurlyeq$.
(iii) $\varphi(x, y) = x \preccurlyeq y$ interprets a linear ordering in the sense of $\leqslant$ in the quotient of A with respect to $\sim$.

The following example restates the theorem of Immerman and Vardi, Theorem 1.24, in the terminology of interpretations.

Example 1.47. Let σ_1 and σ_2 both contain a binary predicate $<$ for a linear ordering. Recall that $\mathrm{ord}[\sigma]$ stands for the class of all finite σ-structures that are linearly ordered by $<$. Let f be a PTIME functor

$$f\colon \mathrm{ord}[\sigma_1] \longrightarrow \mathrm{ord}[\sigma_2].$$

Then there is a FP-definable (σ_2, σ_1)-interpretation i, more precisely a relativized interpretation in some power, such that for all sufficiently large $\mathfrak{A} \in \mathrm{ord}[\sigma_1]$:

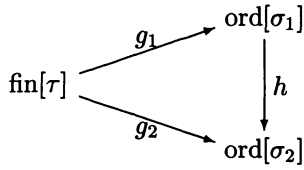
$$f(\mathfrak{A}) \simeq \mathrm{i}(\mathfrak{A}).$$

The proof is a standard application of the Immerman-Vardi Theorem.

In a situation where g_1 and g_2 are two functors with the same domain and with classes of ordered structures for their ranges

$$g_i\colon \mathrm{fin}[\tau] \longrightarrow \mathrm{ord}[\sigma_i]$$

we shall say that g_2 is FP-*interpretable in terms of* g_1, or that $g_2(\mathfrak{A})$ is *uniformly* FP-*interpretable* over $g_1(\mathfrak{A})$, if there is a (σ_2, σ_1)-interpretation i in FP such that $\mathrm{i}\big(g_1(\mathfrak{A})\big) = g_2(\mathfrak{A})$ for all $\mathfrak{A} \in \mathrm{fin}[\tau]$. By the above this is the case if and only if there is a PTIME computable functor h which makes the following diagram commute:



All these considerations apply analogously to PSPACE computable functors and PFP-interpretability, by the theorem of Abiteboul, Vianu and Vardi, Theorem 1.25.

1.5.3 Interpretations and Definability

Natural logics are often semantically closed with respect to definable properties of definably interpreted structures. For an example think of first-order properties of the dual of a graph. These are first-order definable on the graph itself, since the dual is interpretable over the original graph by first-order means.

Closure under direct interpretations and under relativized interpretations are standard regularity requirements on logics in abstract model theory. They correspond to the *substitution property* and *relativization property*, compare [Ebb85]. Closure properties related to interpretations in quotients are also sometimes considered as an abstract criterion under the name of *congruence closure*.

Let $\mathfrak{i}$ be a (σ,τ)-interpretation, R a global relation on $\mathrm{fin}[\sigma]$. With R we may associate a global relation $\mathfrak{i}(R)$ defined on all those structures $\mathfrak{A} \in \mathrm{fin}[\tau]$ for which $\mathfrak{i}(\mathfrak{A})$ is defined. The value of $\mathfrak{i}(R)$ on $\mathfrak{A}$ is the interpretation over $\mathfrak{A}$ of the value of R on $\mathfrak{i}(\mathfrak{A})$.

Formally, for a generalized interpretation $\mathfrak{i} = (\varphi_0;\overline{\varphi};\psi)$ in the sense of Definition 1.44 and for $\mathfrak{A}$ such that $\mathfrak{i}(\mathfrak{A})$ is defined, let $\pi : \varphi_0[\mathfrak{A}] \to \varphi_0[\mathfrak{A}]/\psi[\mathfrak{A}]$ be the natural projection with respect to the equivalence relation interpreted by ψ. Then $\mathfrak{i}(R)^{\mathfrak{A}} := \pi^{-1}\big(R^{\mathfrak{i}(\mathfrak{A})}\big)$. We may put $\mathfrak{i}(R)^{\mathfrak{A}} = \emptyset$ for those $\mathfrak{A}$ for which $\mathfrak{i}(\mathfrak{A})$ is not defined. That $\mathfrak{i}(R)^{\mathfrak{A}}$ is well defined as a value for a global relation follows from the fact that $R^{\mathfrak{i}(\mathfrak{A})}$ is closed under isomorphisms of $\mathfrak{i}(\mathfrak{A})$.

Definition 1.48. *Let σ and τ be finite relational. We say that $\mathcal{L}$ is* closed under (σ,τ)-interpretations *(of a certain kind) if the following is satisfied.*

If $\mathfrak{i}$ is an $\mathcal{L}$-definable (σ,τ)-interpretation (of the respective kind) and R is an $\mathcal{L}$-definable global relation on $\mathrm{fin}[\sigma]$*, then $\mathfrak{i}(R)$ is $\mathcal{L}$-definable as a global relation over* $\mathrm{fin}[\tau]$.

Consider the boolean case: for any $\mathcal{L}$-definable class Q of σ-structures, the class of those τ-structures for which the interpreted σ-structure is in Q is $\mathcal{L}$-definable itself.

Lemma 1.49. *First-order logic $L_{\omega\omega}$, the infinitary logics $L_{\infty\omega}$,$L^{\omega}_{\infty\omega}$,$C^{\omega}_{\infty\omega}$, and the fixed-point logics* FP, PFP *are each closed under generalized interpretations.*

The proofs by syntactic induction are technically tedious though not difficult at all. In the case of interpretations in some power s one replaces all first-order variables by s-tuples of variables. For atomic expressions involving predicates from σ the corresponding defining formulae are substituted. Equality is replaced by the defined equivalence relation. Quantification translates to higher arity quantification (relativized where necessary) in an obvious manner.

Consider for instance a relativized first-order interpretation, of structures in the vocabulary E of graphs, as a quotient in power 2. Let $\mathfrak{i} = \big(\varphi_0(v_1,v_2);\varphi_1(v_1,v_2,v_1',v_2');\psi(v_1,v_2,v_1',v_2')\big)$. The graph axiom $\forall x\forall y(Exy \to \neg x = y)$ then translates into

$$\forall v_1\forall v_2\forall v_1'\forall v_2'\left(\begin{array}{l}(\varphi_0(v_1,v_2)\wedge\varphi_0(v_1',v_2')\wedge\varphi_1(v_1,v_2,v_1',v_2'))\\ \to\neg\psi(v_1,v_2,v_1',v_2')\end{array}\right).$$

Second-order variables of arity r are accordingly replaced by second-order variables of arity sr over interpretations in power s. Fixed-point processes are also modelled in correspondingly higher arity in a natural way.

It is obvious that the $L^{k}_{\infty\omega}$ and $C^{k}_{\infty\omega}$ on the other hand cannot be robust with respect to interpretations in powers, owing to the bounded supply of variables. This is the only restriction, however, so that the straightforward arguments give the following.

Lemma 1.50. *Let $\mathcal{L}^k = L^k_{\infty\omega}, C^k_{\infty\omega}, L^k_{\omega\omega}$ or $C^k_{\omega\omega}$, respectively. Let accordingly $\mathcal{L}^{sk}$ stand for the respective logic with sk variables instead of k. For any $\mathcal{L}^{sk}$-definable generalized (σ,τ)-interpretation* i *in the s-th power and any $\mathcal{L}^k$-definable global relation R on $\mathrm{fin}[\sigma]$, the global relation* $\mathrm{i}(R)$ *on $\mathrm{fin}[\tau]$ — whose value on $\mathfrak{A}$ is the interpretation of $R^{\mathrm{i}(\mathfrak{A})}$ over $\mathfrak{A}$ — is $\mathcal{L}^{sk}$-definable in restriction to all those $\mathfrak{A} \in \mathrm{fin}[\tau]$ for which* $\mathrm{i}(\mathfrak{A})$ *is defined.*[3]

1.6 Lindström Quantifiers and Extensions

Lindström quantifiers provide the means to assert structural properties of definably interpreted structures. Let Q be any isomorphism-closed class of structures of type $\sigma = \{R_1, \ldots, R_l\}$, R_i of arity r_i. With Q we associate a Lindström quantifier of type σ, for which we also write Q. The quantifier Q binds a tuple $\overline{\varphi}$ of formulae apt for a direct interpretation of σ-structures, possibly with parameters. For a logic $\mathcal{L}$ the syntax is extended to allow the construction of a formula

$$\psi := Q\Big(\overline{x}^{(i)}; \varphi_i(\overline{x}^{(i)})\Big)_{i=1\ldots l}$$

from formulae φ_i. Put $\mathrm{free}(\psi) = \bigcup_i (\mathrm{free}(\varphi_i) \setminus \{\overline{x}^{(i)}\})$. The semantics is defined such that

$$\mathfrak{A} \models \psi \quad \text{if} \quad (A, \overline{\varphi}[\mathfrak{A}]) \in Q.$$

Here we have suppressed parameters and assumed that $\mathfrak{A}$ itself interprets the φ_i up to the free variables $\overline{x}^{(i)}$. The "closure" of $\mathcal{L}$ under this new rule of formula formation is denoted $\mathcal{L}(Q)$. As we are only interested in such extensions of first-order and fixed-point logics it suffices to give the following precise definition.

Definition 1.51. *If $\mathcal{Q}$ is a class of Lindström quantifiers, we denote by* $\mathrm{FP}(\mathcal{Q})$ *and* $\mathrm{PFP}(\mathcal{Q})$ *the logics obtained as the simultaneous closure of first-order logic under the respective fixed-point constructor, the usual first-order constructors, and Q-quantification for all quantifiers $Q \in \mathcal{Q}$. $L_{\omega\omega}(\mathcal{Q})$ similarly is obtained from first-order constructors together with Q-quantification for $Q \in \mathcal{Q}$.*

1.6.1 Cardinality Lindström Quantifiers

The class of cardinality Lindström quantifiers is an example of a semantically defined class of quantifiers with natural closure properties. Cardinality Lindström quantifiers express purely numerical relations about the cardinalities of definable predicates.

[3] In non-trivial quotient interpretations more variables may be necessary to express that the interpreted structure is well defined.

Definition 1.52. *Let $S \subseteq \omega^{l+1}$ be a numerical predicate, $\bar{r} = (r_1, \ldots, r_l)$ a tuple of arities. Let $\sigma = \{R_1, \ldots, R_l\}$, R_i of arity r_i. With S and $\bar{r}$ associate a cardinality Lindström quantifier $Q_{S,\bar{r}}$ of type σ whose defining class is*

$$Q_{S,\bar{r}} = \Big\{(B, R_1, \ldots, R_l) \in \mathrm{fin}[\sigma] \;\Big|\; (|B|, |R_1|, \ldots, |R_l|) \in S\Big\}.$$

Let $\mathcal{Q}_{\mathrm{card}}$ be the family of all cardinality Lindström quantifiers.

Note that there is no restriction with respect to the number of formulae bound, their arities, or even recursiveness of the underlying numerical relation. Two important cardinality properties which are naturally rendered as Lindström quantifiers are those that express equality of two cardinalities and comparison in the sense of $<$, respectively. For the original sources see [Här65] and [Res62], respectively.

Definition 1.53. *Q_{H} and Q_{R} are the Lindström quantifiers of type $\sigma = \{U_1, U_2\}$, U_1 and U_2 unary, with the following defining classes.*

(i) For the Härtig quantifier*:* $Q_{\mathrm{H}} = \Big\{(A, U_1, U_2) \;\Big|\; |U_1| = |U_2|\Big\}$.

(ii) For the Rescher quantifier*:* $Q_{\mathrm{R}} = \Big\{(A, U_1, U_2) \;\Big|\; |U_1| < |U_2|\Big\}$.

It is natural to extend these quantifiers to higher arities and to introduce for instance a variant of the Härtig quantifier that expresses equicardinality for two definable predicates of arity k. All these natural variants are cardinality quantifiers themselves. A further extension that goes beyond the power of ordinary cardinality Lindström quantifiers replaces the counting of tuples in a relation by the counting of equivalence classes within a relation, relative to a given congruence. Let us call the quantifiers thus obtained *quotient cardinality Lindström quantifiers*. We give an ad-hoc definition here and indicate a more systematic treatment as an aside below.

Definition 1.54. *Let $S \subseteq \omega^{l+1}$, $\bar{r} = (r_1, \ldots, r_l)$ a tuple of arities. Let σ consist of r_i-ary relation symbols R_i and $2r_i$-ary relation symbols $\sim_i$ for $i = 1, \ldots, l$. With S and $\bar{r}$ associate a cardinality Lindström quantifier $Q^{\sim}_{S,\bar{r}}$ of type σ whose defining class is*

$$\Big\{(B, R_1, \ldots, R_l, \sim_1, \ldots, \sim_l) \;\Big|\; \begin{array}{c} \sim_i \text{ a congruence of } (B^{r_i}, R_i) \text{ and} \\ (|B|, |R_1/\sim_1|, \ldots, |R_l/\sim_l|) \in S \end{array}\Big\}.$$

$\mathcal{Q}^{\sim}_{\mathrm{card}}$ is the family of all quotient cardinality Lindström quantifiers.

1.6.2 Aside on Uniform Families of Quantifiers

The material presented in this aside will not be used explicitly in the sequel. According to the definitions a Lindström quantifier Q can express the structural property of belonging to the class Q of structures that are *directly*

interpreted over the structure at hand. It is often reasonable to make this same property available in application to structures interpreted according to one of the natural variants of interpretations considered above. Formally this can be achieved with derived quantifiers. Suppose for instance that Q is of type $\sigma = \{R_1, \ldots, R_l\}$ and that we want to capture the property of belonging to Q for relativized interpreted σ-structures. The derived quantifier that does exactly this is one of type $\sigma \dot\cup \{U\}$ for a new unary relation symbol U and with defining class

$$Q^{\mathrm{rel}} = \Big\{(\mathfrak{B}, U) \;\Big|\; \mathfrak{B} \restriction U \in Q\Big\}.$$

The other variants of interpretations are treated similarly. Thus, for example a quantifier that corresponds to Q in interpretations in the s-th power is one of type $\{R_1^{(s)}, \ldots, R_l^{(s)}\}$, where $R_i^{(s)}$ is of arity sr_i if r_i is the arity of R_i. Its defining class is

$$Q^{(s)} = \Big\{(B, R_1^{(s)}, \ldots, R_l^{(s)}) \;\Big|\; (B^s, R_1^{(s)}, \ldots, R_l^{(s)}) \in Q\Big\}.$$

$Q^{(s)}$ is called the s-th power of Q. The countable set of all quantifiers $Q^{(s)}$ for $s \geqslant 1$ is called the *uniform sequence* generated by Q in [Daw95a]. Let Q^ω stand for this uniform sequence generated by Q and $\mathcal{Q}^\omega$ for the union of the Q^ω for $Q \in \mathcal{Q}$.

To deal with interpretations as quotients we can further pass to type $\sigma \dot\cup \{\sim\}$ for a new binary relation symbol $\sim$ and consider the quantifier with defining class

$$Q^{\sim} = \{(\mathfrak{B}, \sim) \mid \sim \text{ a congruence of } \mathfrak{B} \text{ and } \mathfrak{B}/\!\sim \in Q\}.$$

If Q is any quantifier, let $\overline{Q}$ stand for the class of all quantifiers obtained by translating Q to generalized interpretations so that $\overline{Q}$ consists of all powers of $(Q^{\mathrm{rel}})^{\sim}$. Similarly, let for a class $\mathcal{Q}$ of quantifiers $\overline{\mathcal{Q}}$ denote all quantifiers obtained in this manner from quantifiers $Q \in \mathcal{Q}$.

A very weak and fundamental notion of reducibility between quantifiers is that of *quantifier free reducibility*. Q is said to be quantifier free reducible to Q' if Q is quantifier free definable from Q' in the sense that

$$Q = \{\mathfrak{B} \mid (B, \overline{\varphi}[\mathfrak{B}]) \in Q'\}$$

for quantifier free formulae $\overline{\varphi}$ in the vocabulary of Q. Write $Q \lhd Q'$ for this reducibility, and $\mathcal{Q} \lhd \mathcal{Q}'$ if each $Q \in \mathcal{Q}$ is quantifier free reducible to some $Q' \in \mathcal{Q}'$. It is instructive to check that $\lhd$ is preserved in the passage to $\overline{\mathcal{Q}}$: $\mathcal{Q} \lhd \mathcal{Q}'$ implies $\overline{\mathcal{Q}} \lhd \overline{\mathcal{Q}'}$.

The class of all cardinality Lindström quantifiers has nice closure properties. If $Q \in \mathcal{Q}_{\mathrm{card}}$, then Q^{rel} and all $Q^{(s)}$ are quantifier free reducible to $\mathcal{Q}_{\mathrm{card}}$. Consider the relativization $Q^{\mathrm{rel}}_{S,\overline{r}}$ of a cardinality quantifier $Q_{S,\overline{r}}$ to find that

$$Q^{\mathrm{rel}}_{S,\overline{r}} = \left\{ (B, R_1, \ldots, R_l, U) \;\middle|\; (B, U, R_1 \cap U, \ldots, R_l \cap U) \in Q_{S',\overline{r}'} \right\}$$

with $\overline{r}' = (1, \overline{r})$ and $S' = \{(n, m_0, \overline{m}) \mid (m_0, \overline{m}) \in S\}$.

Let $\mathcal{Q}_{\mathrm{mon}}$ be the class of Lindström quantifiers of monadic type . $Q \in \mathcal{Q}_{\mathrm{mon}}$ asserts some property of a tuple of unary predicates. $\mathcal{Q}_{\mathrm{mon}}$ and $\mathcal{Q}_{\mathrm{card}}$ are closely related.

Lemma 1.55. *Any quantifier in $\mathcal{Q}_{\mathrm{mon}}$ is quantifier free reducible to a cardinality quantifier, in fact to one in $\mathcal{Q}_{\mathrm{card}} \cap \mathcal{Q}_{\mathrm{mon}}$. Conversely, any quantifier in $\mathcal{Q}_{\mathrm{card}}$ is quantifier free reducible to some power of a monadic quantifier:*

$$\mathcal{Q}_{\mathrm{mon}} \lhd \mathcal{Q}_{\mathrm{card}} \lhd \mathcal{Q}^{\omega}_{\mathrm{mon}}.$$

Sketch of Proof. Both claims are obvious. For the first claim consider Q of type $\{U_1, \ldots, U_l\}$, all U_i unary. Any monadic structure $(B, U_1, \ldots, U_l)$ is characterized up to isomorphism by the cardinalities of B and all boolean combinations of the U_i. The corresponding cardinality quantifier is in a type that has one unary predicate for each boolean combination over the U_i. In its numerical predicate collect all tuples of characteristic cardinalities of structures in Q.

For the second claim observe that first of all a cardinality quantifier $Q_{S,\overline{r}}$ is quantifier free reducible to one of homogeneous type $(r, \ldots, r)$, r the maximum of the arities in $\overline{r}$. For instance if $\overline{r} = (1, 2)$, then $(B, R_1, R_2) \in Q_{S,(1,2)}$ if $\big(B, \{(x, y) | x = y \wedge R_1 x\}, R_2\big) \in Q_{S,(2,2)}$. Further $Q_{S,(r,\ldots,r)} = Q^{(r)}_{S',(1,\ldots,1)}$ for $S' = \{(n^r, \overline{m}) | (n, \overline{m}) \in S\}$. □

It is easy to show that the quotient variant of a cardinality quantifier or of a unary quantifier is not in general reducible to a cardinality quantifier.

The above definition of quotient cardinality quantifiers, Definition 1.54, may seem to be more general even than the extension of $\mathcal{Q}_{\mathrm{card}}$ to generalized interpretations, $\overline{\mathcal{Q}}_{\mathrm{card}}$. Up to quantifier free reducibility, however, these classes coincide. We only give a brief sketch of the argument. Note first, that by arguments as in the proof of Lemma 1.55, any quotient cardinality quantifier reduces to one of homogeneous type. In a further reduction process one may also achieve reduction to applications to disjoint predicates R_i. For this the arities are further increased, and some components are used to attach labels consisting of different equality types to the individual predicates. An example would be the passage from R_1 and R_2 to $R'_1 = \{(x_1, x_2, \overline{x}) \mid x_1 = x_2 \wedge R_1 \overline{x}\}$ and $R'_2 = \{(x_1, x_2, \overline{x}) \mid x_1 \neq x_2 \wedge R_2 \overline{x}\}$. In this situation the several $\sim_i$ may be reduced to a single $\sim$ by piecewise definition over the individual R'_i. The combined reduction leads to a quantifier in $\overline{\mathcal{Q}}_{\mathrm{card}}$.

Remark 1.56. *Up to quantifier free reducibility the following classes of quantifiers coincide: $\overline{\mathcal{Q}}_{\mathrm{card}}$, $\mathcal{Q}^{\sim}_{\mathrm{card}}$, and $\overline{\mathcal{Q}}_{\mathrm{mon}}$.*

1.7 Miscellaneous

1.7.1 Canonization and Invariants

Generally the *canonization problem* for an equivalence relation $\sim$ is the problem of assigning unique representatives to each $\sim$-class.

Definition 1.57. *A function $H: X \longrightarrow X$ is called a* canonization *with respect to the equivalence relation $\sim$ over X if it satisfies the following two conditions:*

$$\forall x \quad H(x) \sim x,$$
$$\forall x \forall x' \quad x \sim x' \to H(x) = H(x').$$

Note that the converse implication in the second condition is implied by the first condition, so that H satisfies $x \sim x' \Leftrightarrow H(x) = H(x')$. In particular therefore, a canonization function H for $\sim$ classifies objects in X exactly up to $\sim$. In the sense of the following definition it is a complete invariant for $\sim$ as well.

Definition 1.58. *A function $I: X \longrightarrow S$ is a* complete invariant for $\sim$ *if it satisfies*

$$\forall x \forall x' \quad x \sim x' \leftrightarrow I(x) = I(x').$$

The difference between canonizations and complete invariants is that canonizations must map elements to representatives of their class.

Definition 1.59. *Let $I: X \longrightarrow S$ be a complete invariant for $\sim$. A mapping $F: \mathrm{image}(I) \longrightarrow X$ is regarded as an* inverse to I *if $I \circ F$ is the identity on* $\mathrm{image}(I)$*, equivalently if*

$$\forall x \quad F\big(I(x)\big) \sim x.$$

Assume that $I: X \to S$ and $I': X \to S'$ are both complete invariants for $\sim$ and that both mappings are surjective. It follows that there is a bijection $\sigma: S \to S'$ between the ranges such that $I' = \sigma \circ I$. Let now H be a canonization and I any complete invariant for $\sim$. Since H also is a complete invariant, there is a bijection from $\mathrm{image}(I)$ to $\mathrm{image}(H)$ that relates the two. It follows directly from the definition that this bijection is an inverse to I. Conversely, given any complete invariant I and any inverse F to this invariant it is immediate that their composition $F \circ I$ is a canonization. We thus have the following little lemma.

Lemma 1.60. *Given any canonization H and any complete invariant I for $\sim$, there is a uniquely determined inverse F of I such that $H = F \circ I$. If F is an inverse to any complete invariant I for $\sim$ then $H := F \circ I$ is a canonization function with respect to $\sim$.*

It is important to note that under complexity considerations some invariants might be easier to invert than others. It is obvious that the problem of

computing some complete invariant reduces to that of computing a canonization function, and that the decision problem for $\sim$ reduces to the computation of any invariant. That the converse reductions are not to be expected for instance as regards PTIME computability and polynomial time Turing reductions in general is shown in [BG84]. General PTIME equivalence of the canonization problem and the problem of the computation of a complete invariant for instance is equivalent with a "shrinking principle" for NP sets that is introduced in [BG84]. Blass and Gurevich also construct an equivalence relation and an oracle relative to which the canonization problem and the problem of computing a complete invariant are not PTIME equivalent. These general results do not have any immediate implications, however, in the case of particular individual equivalence relations.

We shall consider the notions of complete invariants and of canonization for classes of finite relational structures with respect to equivalence relations $\equiv^{\mathcal{L}}$ on these. As any $\equiv^{\mathcal{L}}$ is compatible with isomorphisms, it is clear that canonizations as well as invariants have to be functors that are compatible with isomorphisms.

For computable invariants we also require these to take values in some domain of canonically encoded objects. With respect to computability it is moreover natural to require canonization functors on $\mathrm{fin}[\tau]$ to take canonically encoded structures as their values: the point of canonization is that we get *unique representatives* and not just representatives up to isomorphism.

A complete invariant for $\equiv^{\mathcal{L}}$ on $\mathrm{stan}[\tau]$ or a canonization with respect to $\equiv^{\mathcal{L}}$ on $\mathrm{stan}[\tau]$ immediately extend to corresponding functors on all of $\mathrm{fin}[\tau]$. Algorithmically a computable invariant on $\mathrm{stan}[\tau]$ *is* an invariant on $\mathrm{fin}[\tau]$ and the same is true of computable canonization. This is simply because algorithmic realizations of functors on $\mathrm{fin}[\tau]$ take (encodings of) structures in $\mathrm{stan}[\tau]$ as inputs anyway.

Definition 1.61. *Let $\sim$ be an equivalence relation on $\mathrm{fin}[\tau]$ that is invariant under isomorphism.*

(i) A computable complete invariant *for $\sim$ is a computable function I from $\mathrm{stan}[\tau]$ to some domain S of standard objects such that*

$$\mathfrak{A} \sim \mathfrak{A}' \iff I(\mathfrak{A}) = I(\mathfrak{A}').$$

(ii) A computable canonization functor *with respect to $\sim$ is a computable function H from $\mathrm{stan}[\tau]$ to $\mathrm{stan}[\tau]$ such that*

$$H(\mathfrak{A}) \sim \mathfrak{A} \quad \textit{and} \quad \mathfrak{A} \sim \mathfrak{A}' \Rightarrow H(\mathfrak{A}) = H(\mathfrak{A}').$$

Extensions to domains $\mathrm{fin}[\tau; k]$ are straightforward.

1.7.2 Orderings and Pre-Orderings

Usually we reserve the binary symbol $<$ for linear orderings. $\leqslant$ then stands for the corresponding weak ordering $x \leqslant y \leftrightarrow x < y \vee x = y$, which we call a *linear ordering in the sense of* $\leqslant$.

Definition 1.62. *A* pre-ordering $\preccurlyeq$ *is a binary relation that is transitive, reflexive and connex:*

$$\forall xyz(x \preccurlyeq y \wedge y \preccurlyeq z \to x \preccurlyeq z) \wedge \forall x(x \preccurlyeq x) \wedge \forall xy(x \preccurlyeq y \vee y \preccurlyeq x).$$

We always write $\prec$ *for the associated strict pre-ordering and* $\sim$ *for the induced equivalence relation:*

$$\begin{aligned} x \prec y &:\leftrightarrow x \preccurlyeq y \wedge \neg y \preccurlyeq x, \\ x \sim y &:\leftrightarrow x \preccurlyeq y \wedge y \preccurlyeq x. \end{aligned}$$

It is readily checked that the axioms for $\preccurlyeq$ are equivalent with the statement "$\sim$ is an equivalence relation and $\preccurlyeq/\sim$ is an ordering in the sense of $\leqslant$". Therefore, pre-orderings exactly are the interpretations of linear orderings as quotients, cf. Example 1.46 above.

Obviously $\prec$ and $\preccurlyeq$ are quantifier free definable in terms of each other. $\sim$ is quantifier free definable form both, but contains strictly less information than $\prec$ and $\preccurlyeq$ (unless $\sim$, $\prec$ and $\preccurlyeq$ are trivial).

1.7.3 Lexicographic Orderings

The standard way to construct new orderings in products from given ones in the factors is by *lexicographic orderings*. It is useful to fix one definite convention regarding these. Consider first the case of the product set $D_1 \times D_2$, where D_i is linearly ordered by $<_i$. We write $<_{\text{lex}}$ for the lexicographic ordering on $D_1 \times D_2$ with *dominant first component*: $(d_1, d_2) <_{\text{lex}} (d_1', d_2')$ if $d_1 <_1 d_1'$ or if $d_1 = d_2$ and $d_2 <_2 d_2'$. We employ similar conventions to products with any number of components: entries further to the left always dominate those to the right. The lexicographic ordering is always understood if we are dealing with multiply indexed objects. For instance matrices $(d_{ij}) = (d_{ij})_{1 \leqslant i \leqslant t, 1 \leqslant j \leqslant s}$ are interpreted as tuples where the ordering of the components is the lexicographic one on $\{1, \ldots, t\} \times \{1, \ldots, s\}$. If the entries d_{ij} themselves are from an ordered domain $(D, <)$ we further obtain the lexicographic ordering on $\{(d_{ij}) \mid d_{ij} \in D \text{ for } 1 \leqslant i \leqslant t, 1 \leqslant j \leqslant s\}$ according to $(d_{ij}) <_{\text{lex}} (d_{ij}')$ if $d_{ij} < d_{ij}'$ for the least index pair (i, j) such that $d_{ij} \neq d_{ij}'$.

Note that lexicographic orderings are always first-order definable from the constituent orderings in the components and the ordering of the components.