Chapter 4

Inductive Definitions

First-order logic is not rich enough to express most interesting computations. A useful and natural way to increase the expressive power of first-order logic is to add the power to define new relations by induction. In this chapter we formalize the notion of inductive definitions via the least fixed point operator (LFP)*. We prove that first-order logic extended by the least fixed point operator captures exactly polynomial-time.*

4.1 Least Fixed Point

A useful way to increase the power of first-order logic without jumping all the way up to second order logic is to add the power to define new relations by induction. Many such inductively defined relations are not first-order expressible.

One useful example of a relation that is not first-order expressible, but can be defined inductively, is transitive closure. Recall the vocabulary $\tau_g = \langle E^2, s, t\rangle$ of graphs. We can define the reflexive, transitive closure E^* of E as follows. Let R be a binary relation variable and consider the formula,

$$\varphi_{1.1}(R,x,y) \quad\equiv\quad x = y \vee \exists z(E(x,z) \wedge R(z,y)) \tag{1.1}$$

The formula $\varphi_{1.1}$ formalizes an inductive definition of E^*. This may be more suggestively written as follows,

$$E^*(x,y) \quad\equiv\quad x = y \vee \exists z(E(x,z) \wedge E^*(z,y)) \tag{1.2}$$

For any structure $\mathcal{A}$ with vocabulary τ_g, $\varphi_{1.1}$ induces a map from binary relations on the universe of $\mathcal{A}$ to binary relations on the universe of $\mathcal{A}$,

$$\varphi_{1.1}^{\mathcal{A}}(R) \;=\; \bigl\{\langle a,b\rangle \;\bigm|\; \mathcal{A} \models \varphi_{1.1}(R,a,b)\bigr\}$$

Such a map is called *monotone* if for all R, S,

$$R \subseteq S \;\Rightarrow\; \varphi^{\mathcal{A}}(R) \subseteq \varphi^{\mathcal{A}}(S).$$

The relation symbol R appears only positively in $\varphi_{1.1}$, i.e., within an even number of negation symbols. It follows that $\varphi_{1.1}^{\mathcal{A}}$ is monotone. Let $(\varphi_{1.1}^{\mathcal{A}})^r$ denote $\varphi_{1.1}^{\mathcal{A}}$ iterated r times. With $\varphi_{1.1}$ defined as in Equation (1.1), $\mathcal{A}$ any graph, and $r \geq 0$, observe that,

$$\begin{aligned}
(\varphi_{1.1}^{\mathcal{A}})(\emptyset) &= \bigl\{\langle a,b\rangle \in |\mathcal{A}|^2 \;\bigm|\; \text{distance}(a,b) \leq 0\bigr\}\\
(\varphi_{1.1}^{\mathcal{A}})^2(\emptyset) &= \bigl\{\langle a,b\rangle \in |\mathcal{A}|^2 \;\bigm|\; \text{distance}(a,b) \leq 1\bigr\}
\end{aligned}$$

and in general,

$$(\varphi_{1.1}^{\mathcal{A}})^r(\emptyset) \;=\; \bigl\{\langle a,b\rangle \in |\mathcal{A}|^2 \;\bigm|\; \text{distance}(a,b) \leq r-1\bigr\}$$

Thus, for $n = \|\mathcal{A}\|$, then $(\varphi_{1.1}^{\mathcal{A}})^n(\emptyset) = E^* =$ the least fixed point of $\varphi_{1.1}^{\mathcal{A}}$, i.e., the minimal relation T such that $\varphi_{1.1}^{\mathcal{A}}(T) = T$. This is a general situation, as we now show in the finite version of the Knaster-Tarski Theorem.

Theorem 4.3 *Let R be a new relation symbol of arity k, and let $\varphi(R, x_1, \ldots, x_k)$ be a monotone first-order formula. Then for any finite structure $\mathcal{A}$, the least fixed point of $\varphi^{\mathcal{A}}$ exists. It is equal to $(\varphi^{\mathcal{A}})^r(\emptyset)$ where r is minimal so that $(\varphi^{\mathcal{A}})^r(\emptyset) = (\varphi^{\mathcal{A}})^{r+1}(\emptyset)$. Furthermore, letting $n = \|\mathcal{A}\|$, we have $r \leq n^k$.*

Proof Consider the sequence

$$\emptyset \;\subseteq\; (\varphi^{\mathcal{A}})(\emptyset) \;\subseteq\; (\varphi^{\mathcal{A}})^2(\emptyset) \;\subseteq\; (\varphi^{\mathcal{A}})^3(\emptyset) \;\subseteq\; \cdots \tag{1.4}$$

The containment follows because $\varphi^{\mathcal{A}}$ is monotone. If $(\varphi^{\mathcal{A}})^{i+1}(\emptyset)$ strictly contains $(\varphi^{\mathcal{A}})^i(\emptyset)$, then it must contain at least one new k-tuple from $|\mathcal{A}|$. Since there are at most n^k such k-tuples, for some $r \leq n^k$, $(\varphi^{\mathcal{A}})^r(\emptyset) = (\varphi^{\mathcal{A}})^{r+1}(\emptyset)$, i.e, $(\varphi^{\mathcal{A}})^r(\emptyset)$ is a fixed point of $\varphi^{\mathcal{A}}$.

Let S be any other fixed point of $\varphi^{\mathcal{A}}$. We show by induction that $(\varphi^{\mathcal{A}})^i(\emptyset) \subseteq S$ for all i. The base case is that,

$$(\varphi^{\mathcal{A}})^0(\emptyset) = \emptyset \subseteq S \; .$$

Inductively, suppose that $(\varphi^{\mathcal{A}})^i(\emptyset) \subseteq S$. Since $\varphi^{\mathcal{A}}$ is monotone,

$$(\varphi^{\mathcal{A}})^{i+1}(\emptyset) \;=\; \varphi^{\mathcal{A}}((\varphi^{\mathcal{A}})^i(\emptyset)) \;\subseteq\; \varphi^{\mathcal{A}}(S) \;=\; S \; .$$

Thus, $(\varphi^{\mathcal{A}})^r(\emptyset) \subseteq S$ and $(\varphi^{\mathcal{A}})^r(\emptyset)$ is the least fixed point of $\varphi^{\mathcal{A}}$ as claimed. $\square$

If R occurs only positively in φ, i.e., within an even number of negation symbols, then φ is monotone. Theorem 1.3 tells us that any R-positive formula $\varphi(R^k, x_1, \ldots, x_k)$ determines a least fixed point relation. We write $(\text{LFP}_{R^k x_1 \ldots x_k}\varphi)$ to denote this least fixed point. The least fixed point operator (LFP) thus formalizes the definition of new relations by induction. The subscript "$R^k x_1 \ldots x_k$" explicitly tells us which relation and domain variables we are taking the fixed point with respect to. When the choice of variables is clear, these subscripts may be omitted.

As an example, $(\text{LFP}_{Rxy}\varphi_{1.1})$ denotes the reflexive, transitive closure of the edge relation E. Thus boolean query REACH is expressible as:

$$\text{REACH} \quad \equiv \quad (\text{LFP}_{Rxy}\varphi_{1.1})(s,t)$$

Definition 4.5 Define FO(LFP), the language of first-order inductive definitions, by adding a least fixed point operator (LFP) to first-order logic. If $\varphi(R^k, x_1, \ldots, x_k)$ is an R^k-positive formula in FO(LFP), then $(\text{LFP}_{R^k x_1 \ldots x_k}\,\varphi)$ may be used as a new k-ary relation symbol denoting the least fixed point of φ. $\square$

Example 4.6 In Definition 3.23 we gave an inductive definition of alternating paths. We then defined boolean query REACH_a to be the set of graphs having an alternating path from s to t. (See Figure 1.7 which shows a graph that satisfies REACH_a.)

We now give a first-order inductive definition of the alternating path property, P_a,

$$\begin{aligned}\varphi_{ap}(P,x,y) \quad \equiv \quad & x = y \;\vee\; [(\exists z)(E(x,z) \wedge P(z,y)) \;\wedge \\ & \qquad (A(x) \;\rightarrow\; (\forall z)(E(x,z) \rightarrow P(z,y)))]\end{aligned}$$

Thus,

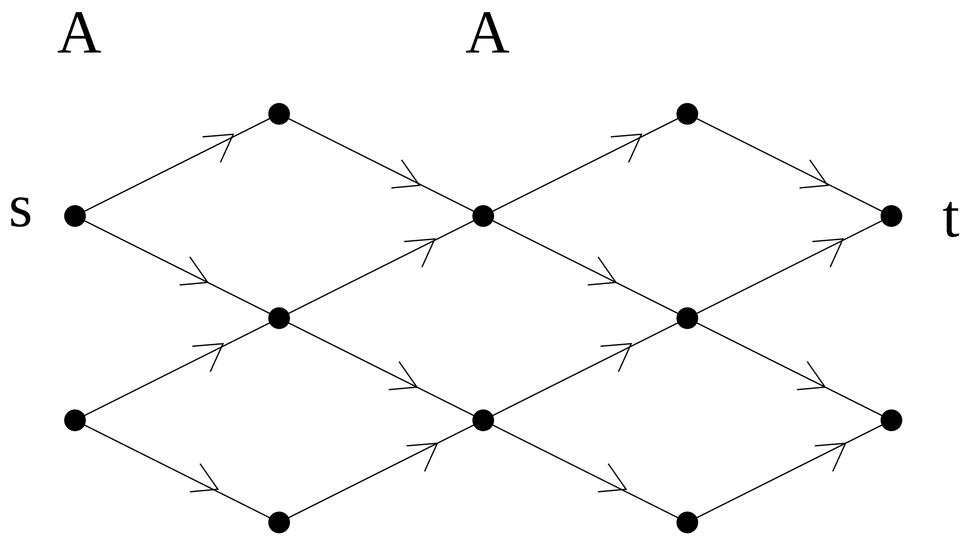

Figure 1.7: A graph satisfying the boolean query REACH$_a$.

$$P_a \;=\; (\text{LFP}_{Pxy}\varphi_{ap}) \quad \text{and} \quad \text{REACH}_a \;=\; (\text{LFP}_{Pxy}\varphi_{ap})(s,t) \qquad \square$$

Recall that REACH$_a$ is complete for P via first-order reductions, Theorem 3.26. It follows from Example 1.6 and from the following exercise that FO(LFP) contains all the polynomial-time boolean queries.

Exercise 4.8 Show that FO(LFP) is closed under first-order reductions.

Hint: this is a special case of Meta-Proposition 3.11. You have to show that if Q is a k-ary first-order query and $\Phi \in$ FO(LFP), then $\widehat{Q}(\Phi) \in$ FO(LFP). This is clear once you observe that

$$\widehat{Q}(\text{LFP}_{R^a,x_1,\ldots,x_a}\alpha) \quad \equiv \quad (\text{LFP}_{R^{ka},x_1^1\ldots x_1^k,\ldots,x_a^1\ldots x_a^k}\widehat{Q}(\alpha)) \qquad \square$$

Now that we have formalized inductive definitions, we feel free to write them in the intuitive form of Equation (1.2) rather than as $(\text{LFP}_{Rx,y}\,\varphi_{1.1})$. It is often convenient to define several relations by simultaneous induction. The following exercise shows that there is no harm in doing this.

Exercise 4.9 Suppose $\psi(\bar{y}, S, T)$ and $\varphi(\bar{x}, S, T)$ are first order formulas that are positive in S and T. Let r_0 = arity(S) = $|\bar{y}|$ and r_1 = arity(T) = $|\bar{x}|$. For any structure $\mathcal{A}$ define the relations I_0^ω and I_1^ω by simultaneous induction:

$$\begin{array}{lcl} I_0^0 & = & I_1^0 \quad = \quad \emptyset \\ \bar{a} \in I_0^n & \Leftrightarrow & \mathcal{A} \models \psi(\bar{a}, I_0^{n-1}, I_1^{n-1}) \\ \bar{b} \in I_1^n & \Leftrightarrow & \mathcal{A} \models \varphi(\bar{b}, I_0^{n-1}, I_1^{n-1}) \\ I_b^\omega & = & \bigcup_{n=1}^{\infty} I_b^n, \quad b = 0, 1 \end{array}$$

Show that both I_0^ω and I_1^ω are expressible in FO(LFP).

Hint: Assume that there are distinct constants $c_0 \neq c_1$.[1] Assume that $r_0 \leq r_1$. Define a single new relation U of arity $1 + r_1$ so that $U(c_0, \bar{y}, \bar{u})$ refers to $S(\bar{y})$, with $\bar{u}$ an $(r_1 - r_0)$-tuple of dummy variables and $U(c_1, \bar{x})$ refers to $T(\bar{x})$. $\square$

We next show that FO(LFP) — the closure of first-order logic under the power to inductively define new relations — describes exactly the set of all polynomial-time computable boolean queries.

Theorem 4.10 *Over finite, ordered structures,*

$$\text{FO(LFP)} = \text{P}$$

Proof ($\subseteq$): Let $\mathcal{A}$ be an input structure, let $n = \|\mathcal{A}\|$, and let $(\text{LFP}_{Rx_1 \ldots x_k}\, \varphi)$ be a fixed-point formula. By Theorem 1.3, we know that this fixed point evaluated on $\mathcal{A}$ is $(\varphi^{\mathcal{A}})^{n^k}(\emptyset)$. This amounts to evaluating the first-order query φ at most n^k times. We have seen in Theorem 3.1 that first-order queries may be evaluated in L, thus easily in P.

($\supseteq$): Since FO(LFP) includes query REACH_a, which is complete for P via first-order reductions, and FO(LFP) is closed under first-order reductions, FO(LFP) includes all polynomial-time queries. $\square$

Theorem 1.10 equates polynomial time — one of the most important complexity classes — with FO(LFP) — the closure of first-order logic under the power to make inductive definitions. The latter is very natural from a descriptive point of view. This theorem thus increases our intuition that polynomial time is a class

[1]For example, 0 and *max* would do for ordered structures of size greater than one. If no constants are available, then one can quantify two distinct elements u, v and use u in place of c_0 and v in place of c_1. If the universe has size only one, then any formula is trivial, and that case can be dealt with separately; see Proviso 1.15.

whose fundamental nature goes beyond the machine models with which it is usually defined.

The use of ordering in Theorem 1.10 is required in the proof that REACH_a is complete via $\leq_{\mathrm{fo}}$. Stripped of its numeric relations including ordering, $\mathrm{FO(LFP)}$ does not describe all polynomial-time properties. For example, we will see that it cannot even express the parity of its universe. (The rôle of ordering is described extensively in Chapter 12.)

To conclude this section we note that the above proof leads to the following normal form theorem for language $\mathrm{FO(LFP)}$ over ordered structures. The language $\mathrm{FO(LFP)}$ allows the application of a complicated series of nested fixed points including extra quantifications and negations. The normal form theorem says that any such formula is equivalent to a single fixed point applied to a first-order formula. Such a normal form theorem makes it easier to understand exactly what can be expressed in language $\mathrm{FO(LFP)}$. The same result holds without ordering, but the proof is more subtle (Theorem 9.6).

Corollary 4.11 *Let φ be any formula in language* $\mathrm{FO(LFP)}$. *There exists a first-order formula ψ and a tuple of constants $\bar{c}$ such that over finite, ordered structures,*

$$\varphi \;\equiv\; (\mathrm{LFP}\,\psi)(\bar{c})$$

Proof The completeness of REACH_a for P means that every polynomial-time query is expressible as $\widehat{Q}(\mathrm{REACH}_a)$ for some first-order query Q. Now, in Example 1.6 we saw that,

$$\mathrm{REACH}_a = (\mathrm{LFP}\varphi_{ap})(s,t)$$

Thus, an arbitrary polynomial-time query is expressible as,

$$\widehat{Q}(\mathrm{REACH}_a) = (\mathrm{LFP}\,\widehat{Q}(\varphi_{ap}))\widehat{Q}(s,t) \tag{1.12}$$

Since the first-order reductions used in Theorem 3.26 replace the constants s and t by k-tuples of the constants 0 and *max* — see Equation (3.19) — the form of Equation (1.12) is as desired. □

4.2 The Depth of Inductive Definitions

The number of iterations until an inductive definition closes is called its *depth*[2]. We will see that inductive depth is an important complexity measure, corresponding to parallel time (Theorem 5.2).

Definition 4.13 Let $\varphi(R, x_1, \ldots, x_k)$ be an R-positive formula, where R is a relation symbol of arity k, and let $\mathcal{A}$ be a structure of size n. Define the *depth* of φ in $\mathcal{A}$, in symbols $|\varphi^{\mathcal{A}}|$, to be the minimum r such that

$$\mathcal{A} \models \Big(\varphi^r(\emptyset) \leftrightarrow \varphi^{r+1}(\emptyset)\Big)$$

As we saw in the proof of Theorem 1.3, $|\varphi^{\mathcal{A}}| \le n^k$. Define the depth of φ as a function of n equal to the maximum depth of φ in $\mathcal{A}$ for any structure $\mathcal{A}$ of size n:

$$|\varphi|(n) \;=\; \max_{\|\mathcal{A}\|=n} \{|\varphi^{\mathcal{A}}|\} \qquad \Box$$

Remark 4.14 *The inductive definition $\varphi_{1.1}$ given in Equation (1.1) has depth $|\varphi|(n) = n$. However, the following alternate inductive definition of E^* has depth $|\varphi_{tc}|(n) = \lceil \log n \rceil + 1$.*

$$\varphi_{tc}(R, x, y) \quad \equiv \quad x = y \;\vee\; E(x,y) \;\vee\; \exists z(R(x,z) \wedge R(z,y)) \qquad (1.15)$$

In computer science, the depth of an inductive definition corresponds to the depth of the stack needed to evaluate a recursive definition, that is, the depth of nesting of recursive calls. We will see in Theorem 5.2 that this also corresponds to the parallel time needed to evaluate such a recursive definition.

Definition 4.16 Let IND$[f(n)]$ be the sublanguage of FO(LFP) in which only fixed points of first-order formulas φ for which $|\varphi|$ is $O[f(n)]$ are included. For example, REACH is expressible as (LFP$_{Rxy}\,\varphi_{tc}$) and is thus in IND$[\log n]$. Note also that,

$$\text{FO(LFP)} \;=\; \bigcup_{k=1}^{\infty} \text{IND}[n^k] \;. \qquad \Box$$

[2]In the logic literature where structures are usually infinite this is called the *closure ordinal*.

The facts that REACH $\in$ IND$[\log n]$, REACH is complete for NL via first-order reductions (Theorem 3.16), and IND$[\log n]$ is closed under first-order reductions imply

Proposition 4.17 NL $\subseteq$ IND$[\log n]$

We will see in Theorem 5.22 that IND$[\log n]$ is equal to AC^1, a well-studied complexity class. As another example, the inductive definition of REACH$_a$ in Example 1.6 has depth equal to the length of the longest path from s to t in the graph.

In the following exercise you are asked to show that the numeric relations BIT, PLUS, and TIMES, are all definable in IND$(wo\text{BIT})[\log n]$, that is, via first-order inductive definitions that use only the numeric relation $\leq$. This shows that the descriptive class IND$[\log n]$ is somewhat more robust than IND$[1]$ $=$ FO and has a more general definition. This is a general pattern: the more powerful the language, the less important exactly which numeric relations are included.

Exercise 4.18 Show that BIT $\in$ IND$(wo\text{BIT})[\log n]$, i.e., relation BIT is definable by a depth $\log n$ induction just from $\leq$.

[Hint: by successive inductive definitions, show that PLUS and then TIMES are definable in IND$(wo\text{BIT})[\log n]$. The result will then follow by Theorem 1.17.] $\square$

Exercise 4.19 Recall that PARITY $\subset$ STRUC$[\tau_s]$ is the set of binary strings with an odd number of 1's (Example 2.12).

1. Show that PARITY $\in$ IND$[\log n]$.

2. For a more challenging problem, show that PARITY $\in$ IND$[\log n/\log\log n]$. This requires BIT. [Hint: divide the n bit input string into $\lfloor \log n \rfloor$ pieces. The string has odd parity iff an odd number of the pieces have odd parity. You may use Lemma 1.18 for the fact that the parity of a $\log n$ bit string is first-order.]

We see in Chapter 6 that PARITY $\notin$ IND$(wo\text{BIT})[o(\log n)]$. We also see in Corollary 13.8 that with BIT, $[\log n/\log\log n]$ is optimal. $\square$

4.3 Iterating First-Order Formulas

Theorem 1.3 shows that for any first-order inductive definition $\varphi(R, x_1, \ldots, x_k)$, the least fixed point of φ amounts to iterating φ at most n^k times. Thus, in some sense, LFP is a polynomial iteration operator. This is even more apparent when we put the inductive definitions into the following normal form. Then the effect of the least-fixed-point operator is to iterate a certain block of restricted quantifiers a polynomial number of times. (Recall our notation for restricted quantifiers: $(\forall x.M)(\psi)$ means $(\forall x)(M \to \psi)$ and $(\exists x.M)(\psi)$ means $(\exists x)(M \wedge \psi)$.)

Lemma 4.20 *Let φ be an R-positive first-order formula. Then φ can be written in the following form,*

$$\varphi(R, x_1, \ldots, x_k) \;\equiv\; (Q_1 z_1.M_1) \ldots (Q_s z_s.M_s)(\exists x_1 \ldots x_k.M_{s+1})R(x_1, \ldots, x_k)\ . \tag{1.21}$$

where the M_i's are quantifier-free formulas in which R does not occur.

Proof By induction on the complexity of φ. We assume that all negations have been pushed all the way inside. There are two base cases: If $\varphi \equiv R(v_1, \ldots, v_k)$, then,

$$\begin{aligned} \varphi &\equiv (\exists z_1, \ldots, z_k.M_1)(\exists x_1, \ldots, x_k.M_2)R(x_1, \ldots, x_k) \\ M_1 &\equiv z_1 = v_1 \ \wedge \ \cdots \ \wedge \ z_k = v_k \\ M_2 &\equiv x_1 = z_1 \ \wedge \ \cdots \ \wedge \ x_k = z_k \end{aligned}$$

If φ is quantifier free and R does not occur in φ, then,

$$\varphi \quad\equiv\quad (\forall z.\neg\varphi)(\exists x_1, \ldots, x_k.x_1 \neq x_1)R(x_1, \ldots, x_k)\ .$$

In the inductive cases $\varphi = (\exists v)\psi$ and $\varphi = (\forall v)\psi$, we simply put the new quantifier $(\exists v)$ or $(\forall v)$ in front of the quantifier block for ψ.

The remaining cases for $\wedge$ and $\vee$ are similar to each other. Suppose that $\varphi = \alpha \wedge \beta$ and

$$\begin{aligned} \alpha &\equiv (Q_1 y_1.N_1) \ldots (Q_t y_t.N_t)(\exists x_1 \ldots x_k.N_{t+1})R(x_1, \ldots, x_k) \\ \beta &\equiv (Q_1 z_1.M_1) \ldots (Q_s z_s.M_s)(\exists x_1 \ldots x_k.M_{s+1})R(x_1, \ldots, x_k) \end{aligned}$$

where we may assume that the y's and z's are disjoint, and we can add dummy variables and quantifiers to assure that the form of the two quantifier blocks are identical. Let

$$\begin{aligned}\text{QB}_1 &\equiv (Q_1 y_1.N'_1)\dots(Q_t y_t.N'_t)\ ,\\ \text{QB}_2 &\equiv (Q_1 z_1.M'_1)\dots(Q_s z_s.M'_s)\ .\end{aligned}$$

where,

$$N'_i \ \equiv\ N_i \ \vee\ b=1; \quad \text{and } M'_i \ \equiv\ M_i \ \vee\ b=0$$

Let $\psi(\bar{u}/\bar{x})$ denote the formula ψ with variables $u_1,\dots,u_k$ substituted for $x_1,\dots,x_k$ and define the quantifier-free formulas,

$$\begin{aligned}S &\equiv (b=0 \wedge N_{t+1}(\bar{u}/\bar{x})) \vee (b=1 \wedge M_{s+1}(\bar{u}/\bar{x}))\ ,\\ T &\equiv (u_1 = x_1 \wedge \dots \wedge u_k = x_k)\ .\end{aligned}$$

Recall that $\text{bool}(b)$ means that $b=0$ or $b=1$, that is, b is a boolean variable (Definition 1.16). We can now write φ in the desired form,

$$\varphi \ \equiv\ (\forall b.\text{bool}(b))(\text{QB}_1)(\text{QB}_2)(\exists\bar{u}.S)(\exists\bar{x}.T)R(x_1,\dots,x_k) \qquad \square$$

Note that in Equation (1.21), the requantification of the x_i's means that these variables may occur free in $M_1 \dots M_s$, but they are bound in M_{s+1} and $R(x_1,\dots,x_k)$. The same variables may now be requantified. Let us write QB to denote the quantifier block $(Q_1 z_1.M_1)\dots(Q_s z_s.M_s)(\exists x_1\dots x_k.M_{s+1})$. Thus, in particular, for any structure $\mathcal{A}$, and any $r \in \mathbf{N}$,

$$\mathcal{A} \models \big((\varphi^{\mathcal{A}})^r(\emptyset)\big) \leftrightarrow \big([\text{QB}]^r\textbf{false}\big)\ .$$

Here $[\text{QB}]^r$ means QB literally repeated r times. It follows immediately that if $t = |\varphi|(n)$ and $\mathcal{A}$ is any structure of size n then

$$\mathcal{A} \models \big(\text{LFP}\,\varphi\big) \leftrightarrow \big([\text{QB}]^t\textbf{false}\big)\ .$$

Example 4.22 We show how to write the inductive definition of transitive closure in the normal form of Lemma 1.20.

Recall the definition of transitive closure from Equation (1.15),

$$\varphi_{tc}(R,x,y) \quad\equiv\quad x=y \ \vee\ E(x,y) \ \vee\ (\exists z)(R(x,z)\wedge R(z,y))$$

First, code the base case using a dummy universal quantification,

$$\begin{aligned}\varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(R(x,z)\wedge R(z,y))\\ M_1 &\equiv \neg(x=y\vee E(x,y))\end{aligned}$$

Note that there are no free occurrences of z within the scope of the $(\forall z.M_1)$ quantifier. Next, use universal quantification to replace the two occurrences of R with a single one:

$$\begin{aligned}\varphi_{tc}(R,x,y) &\equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)R(u,v)\\ M_2 &\equiv (u=x\wedge v=z)\ \vee\ (u=z\wedge v=y)\ .\end{aligned}$$

Finally, requantify x and y. We have transformed Equation (1.15), into the normal form of Lemma 1.20,

$$M_3 \equiv (x=u\wedge y=v)$$

$$\varphi_{tc}(R,x,y) \equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)(\exists xy.M_3)R(x,y) \tag{1.23}$$

$\square$

Define the quantifier block,

$$\mathrm{QB}_{tc} \equiv (\forall z.M_1)(\exists z)(\forall uv.M_2)(\forall xy.M_3)\ .$$

Equation (1.23) tells us that an application of the operator φ_{tc} corresponds exactly to the writing of QB$_{tc}$,

$$\varphi_{tc}(R,x,y) \equiv [\mathrm{QB}_{tc}]R(x,y)$$

It follows that for any r,

$$\varphi_{tc}^r(\emptyset) \equiv [\mathrm{QB}_{tc}]^r(\textbf{false})\ .$$

We have thus demonstrated a syntactic uniformity for the inductive definition of REACH. For any structure $\mathcal{A} \in \text{STRUC}[\tau_g]$,

$$\begin{aligned} \mathcal{A} \in \text{REACH} \quad &\Leftrightarrow \quad \mathcal{A} \models (\text{LFP}\varphi_{tc})(s,t) \\ &\Leftrightarrow \quad \mathcal{A} \models \Big([\text{QB}_{tc}]^{\lceil 1+\log \|\mathcal{A}\| \rceil}\Big)\textbf{false}(s/x, t/y) \end{aligned}$$

We now define FO$[t(n)]$ to be the set of properties defined by quantifier blocks iterated $t(n)$ times. (This is the same as being iterated $O(t)$ times since a quantifier block may be any constant size.) Even though such expressions grow as a function of the size of their inputs, they use only the variables in the quantifier block. That is, the number of variables is a fixed constant independent of the size of the input.

Definition 4.24 A set $S \subseteq \text{STRUC}[\tau]$ is a member of FO$[t(n)]$ iff there exist quantifier free formulas M_i, $0 \le i \le k$, from $\mathcal{L}(\tau)$, a tuple $\bar{c}$ of constants and a quantifier block,

$$\text{QB} \;=\; \big[(Q_1 x_1.M_1)\ldots(Q_k x_k.M_k)\big]$$

such that for all $\mathcal{A} \in \text{STRUC}[\tau]$,

$$\mathcal{A} \in S \quad \Leftrightarrow \quad \mathcal{A} \models \Big([\text{QB}]^{t(\|\mathcal{A}\|)} M_0\Big)(\bar{c}/\bar{x})$$

□

The reason for the substitution of constants is that the quantifier block QB may contain some free variables that must be substituted to build a sentence. See Example 1.22 which shows that REACH $\in$ FO$[\log n]$.

Combining Lemma 1.20 and Definition 1.24, we see that,

Lemma 4.25 *For all $t(n)$ and all classes of finite structures,*

$$\text{IND}[t(n)] \;\subseteq\; \text{FO}[t(n)]\ .$$

A converse of Lemma 1.25 also holds, but we put off its proof until the next chapter.

Exercise 4.26 Write an inductive definition showing that CVP, the circuit value problem (Definition 2.27) is describable in FO(LFP). The depth of your inductive definition should be equal to the depth of the circuit, i.e., the length of the longest path from root to leaf. □

Exercise 4.27 Write a sentence in FO(wo≤)(LFP) meaning that a graph is two-colorable, i.e., each vertex may be colored red or blue in such a way that no two adjacent vertices are the same color. [Hint: one way to do this is to first simultaneously define the relations OPath(x, y) and EPath(x, y) meaning that there is a path of odd length, respectively even length, from x to y. A graph is two-colorable iff it has no cycles of odd length.] □

Historical Notes and Suggestions for Further Reading

Moschovakis has written a thorough and excellent book on inductive definitions, [Mos74]. Although its main focus is infinite structures, our treatment of inductive definitions follows the approach set out in that book. In particular, Exercise 1.9 and Lemma 1.20 are from [Mos74].

The Knaster-Tarski Theorem (Theorem 1.3) originally appeared in [Tar55]. Ajtai and Gurevich proved that over finite structures, not every monotone formula has an equivalent positive formula [AG87]. Theorem 1.10 is due to Vardi and Immerman [I82, Var82].