Chapter 2
General Theory: Subcomputations

This chapter adds a new notion to the general theory, viz. the notion of *subcomputation*. We develop the elementary theory including a general version of the first recursion theorem, and ending up with a representation theorem which is "faithful" in the sense that it preserves the full structure of subcomputations.

2.1 Subcomputations

In Chapter 1 we took as our basic relation

$$\{a\}(\sigma) \simeq z,$$

asserting that the computing device a acting on the input sequence σ gives z as output. We wrote down for the set Θ of all computation tuples (a, σ, z) a set of axioms and were able to derive within this framework a number of results of elementary recursion theory, leading up to a *simple representation theorem* for any such Θ.

However, many arguments from the more advanced parts of recursion theory seem to require an analysis not only of the computation tuple, but of the whole structure of "subcomputations" of a given computation tuple. In fact, such an analysis was involved in the proof of the first recursion Theorem 1.7.9 *via* the representation Theorem 1.6.3 (see Definition 1.5.9).

In his paper *Axioms for computation theories—first draft* [113] Moschovakis emphasized the fact that whatever computations may be, they have assigned a well-defined *length*, which is always an ordinal, finite or infinite. Thus he proposed to add as a further primitive a map from the set Θ of computation tuples to the ordinals, denoting by $|a, \sigma, z|_\Theta$ the ordinal associated with the tuple $(a, \sigma, z) \in \Theta$.

We shall, in addition, abstract another but related aspect of the notion of computation and add as a further primitive a relation between computation tuples

$$(a', \sigma', z') < (a, \sigma, z),$$

which is intended to express that (a', σ', z') is a "subcomputation" of (a, σ, z), i.e. the computation (a, σ, z) depends upon the previous computation (a', σ', z').

2.1.1 Definition. Let $\mathfrak{A} = \langle A, C, N; s, M, K, L\rangle$ be a computation domain. A *computation structure* $\langle \Theta, <_\Theta\rangle$ over $\mathfrak{A}$ is a pair where Θ is a computation set and $<_\Theta$ is a transitive and well-founded relation on Θ.

Remark. The notation $\langle \Theta, <_\Theta\rangle$ is clumsy. Whenever the context permits we shall write $<$ rather than $<_\Theta$ and use the shorter Θ for $\langle \Theta, <_\Theta\rangle$ or $\langle \Theta, <\rangle$.

Note that if $(a, \sigma, z) \in \Theta$, the set

$$\mathbf{S}_{(a,\sigma,z)} = \{(a', \sigma', z') : (a', \sigma', z') < (a, \sigma, z)\},$$

is a well-founded transitive set, the set of "subcomputations" of (a, σ, z). $\mathbf{S}_{(a,\sigma,z)}$ has an associated ordinal $|a, \sigma, z|_\Theta$, which may be called the "length" of the computation (a, σ, z).

The notion of computable function carries over unchanged to the present setting. In the definition of Θ-computable functional we make an addition.

2.1.2 Definition. Let $\langle \Theta, <\rangle$ be a computation structure on a domain $\mathfrak{A}$. A consistent functional φ is called *weakly Θ-computable* if there exists a $\hat{\varphi} \in C$ such that for all $e_1, \ldots, e_l \in C$ and all sequences $\sigma = (x_1, \ldots, x_n)$ from A we have

(a) $\varphi(\{e_1\}^{n_1}_\Theta, \ldots, \{e_l\}^{n_l}_\Theta, \sigma) \simeq z$ iff $\{\hat{\varphi}\}^{l+n}_\Theta(e_1, \ldots, e_l, \sigma) \simeq z$.

(b) If $\varphi(\{e_1\}^{n_1}_\Theta, \ldots, \{e_l\}^{n_l}_\Theta, \sigma) \simeq z$, then there exist functions $g_1, \ldots, g_l$ such that

(i) $g_1 \subseteq \{e_1\}^{n_1}_\Theta, \ldots, g_l \subseteq \{e_l\}^{n_l}_\Theta$ and $\varphi(\mathbf{g}, \sigma) \simeq z$;

(ii) for all $i = 1, \ldots, l$, if $g_i(t_1, \ldots, t_{n_i}) \simeq u_i$, then $(e_i, t_1, \ldots, t_{n_i}, u_i) < (\hat{\varphi}, e_1, \ldots, e_l, \sigma, z)$.

2.1.3 Remark. We may motivate clause (b) of the above definition by reflecting on how we compute in a theory PR[**f**], where $\mathbf{f} = f_1, \ldots, f_l$ is a list of partial functions, see Definitions 1.5.6–1.5.9. Let $(a, \sigma, z) \in \mathrm{PR}[\mathbf{f}]$ and consider its subcomputation tree. Among the subcomputations of (a, σ, z) are various tuples $(\hat{f}_i, t_1, \ldots, t_{n_i}, u_i)$, where $\hat{f}_i$ is the code in PR[**f**] of f_i and $f_i(t_1, \ldots, t_{n_i}) \simeq u_i$. For each $i = 1, \ldots, l$ let $g_i = \{(t_1, \ldots, t_{n_i}, u_i) : (\hat{f}_i, t_1, \ldots, t_{n_i}, u_i) <_{\mathrm{PR}[\mathbf{f}]} (a, \sigma, z)\}$. Then $g_1, \ldots, g_l$ are subfunctions of $f_1, \ldots, f_l$ encoding all the information necessary to compute $\{a\}(\sigma) \simeq z$, and, moreover, $(a, \sigma, z) \in \mathrm{PR}[\mathbf{g}]$.

We are now ready for the definition of a computation theory.

2.1.4 Definition. A computation structure $\langle \Theta, <_\Theta\rangle$ on the domain $\mathfrak{A}$ is called a *computation theory* on $\mathfrak{A}$ if there exist Θ-computable mappings p_1, p_2, and p_3 such that

(a) the functions s, M, K, L, DC are Θ-computable with Θ-codes $\hat{s}$, m, k, l, d, respectively;

(b) the functionals $\mathbf{C}^n$ and $\mathbf{P}^m_{n,j}$ are weakly Θ-computable with Θ-codes $c_n = p_1(n)$ and $p_{n,j,m} = p_2(n, j, m)$;

(c) Θ satisfies the following iteration property: For all n, m $p_3(n, m)$ is a Θ-code for a mapping $S_m^n(a, x_1, \ldots, x_n)$ such that for all a, σ in C and τ in A, where $\mathrm{lh}(\sigma) = n$ and $\mathrm{lh}(\tau) = m$,

(i) $\{a\}_\Theta^{n+m}(\sigma, \tau) = \{S_m^n(a, \sigma)\}_\Theta^m(\tau)$;
(ii) if $\{a\}_\Theta^{n+m}(\sigma, \tau) \simeq z$, then $(a, \sigma, \tau, z) < (S_m^n(a, \sigma), \tau, z)$.

Note that (c), (ii) is the only point where the definition reads differently from the precomputation case (Definition 1.5.3). And further that the condition enforced axiomatically in (c), (ii) is one which is naturally satisfied for constructed recursion theories, in particular, theories of the type PR[**f**] where $\{a\}(\sigma, \tau) \simeq z$ occurs as a "subcomputation" of $\{S_m^n(a, \sigma)\}(\tau) \simeq z$.

2.1.5 Remark. Elementary results about precomputation theories extend in an obvious fashion to computation theories, in particular, the results of Section 1.2 do so.

2.2 *Inductively Defined Theories*

In Section 1.5 we constructed from a given list of functions $\mathbf{f} = f_1, \ldots, f_l$ a theory PR[**f**], the prime computation theory generated by the list **f**. Using the Definition 1.5.9 of subcomputation for PR[**f**] it is obvious that the theory there constructed is a computation theory in the sense of Definition 2.1.4.

We will now extend the construction to cover also the case of functionals. *In this section we restrict ourselves to functionals defined only on total functions.* The general case leads into several knotty problems concerning subcomputations in partial objects of higher types.

2.2.1 Construction of $\Gamma_{f,\varphi}(\Theta)$. Let $\mathfrak{A}$ be a computation domain, f a partial function on A, and φ a functional on A defined only on total functions. A monotone operator $\Gamma_{f,\varphi}$ is introduced by the following set of clauses:

1. $(\langle 1, 0\rangle, x, s(x)) \in \Gamma_{f,\varphi}(\Theta)$.
2. $(\langle 2, 0\rangle, x, y, M(x, y)) \in \Gamma_{f,\varphi}(\Theta)$.
3. $(\langle 3, 0\rangle, x, K(x)) \in \Gamma_{f,\varphi}(\Theta)$.
4. $(\langle 4, 0\rangle, x, L(x)) \in \Gamma_{f,\varphi}(\Theta)$.
5. $(\langle 5, 0\rangle\, x, a, b, c, DC(x, a, b, c)) \in \Gamma_{f,\varphi}(\Theta)$.
6. If $\exists u[(\hat{g}, \sigma, u) \in \Theta$ and $(\hat{f}, u, \sigma, z) \in \Theta]$, then $(\langle 6, 0\rangle, \hat{f}, \hat{g}, \sigma, z) \in \Gamma_{f,\varphi}(\Theta)$.
7. Let $0 \leqslant j < n$ and τ any m-tuple from A, if $(\hat{f}, x_{j+1}, x_1, \ldots, x_j, x_{j+2}, \ldots, x_n, z) \in \Theta$, then $(\langle 7, j\rangle, \hat{f}, x_1, \ldots, x_n, \tau, z) \in \Gamma_{f,\varphi}(\Theta)$.
8. If $a, x_1, \ldots, x_n \in C$ and $(a, x_1, \ldots, x_n, y_1, \ldots, y_m, z) \in \Theta$, then $(\langle 8, a, x_1, \ldots, x_n\rangle, y_1, \ldots, y_m, z) \in \Gamma_{f,\varphi}(\Theta)$.
9. If $f(t_1, \ldots, t_n) \simeq z$, then $(\langle 9, 0\rangle, t_1, \ldots, t_n, z) \in \Gamma_{f,\varphi}(\Theta)$.

10. If for all $u \in A$ there is a $v \in A$ such that $(\hat{g}, u, v) \in \Theta$ and $\varphi(\{\hat{g}\}) \simeq z$, then $(\langle 10, 0\rangle, \hat{g}, z) \in \Gamma_{f,\varphi}(\Theta)$.

We note that clauses 1–9 are exactly the same as those in Definition 1.5.6. Clause 10 introduces the functional φ. For partial φ we would instead have a clause

10′. If there exists a partial function g on A such that $\forall u \forall v[g(u) \simeq v \Rightarrow (\hat{g}, u, v) \in \Theta]$ and $\varphi(g) \simeq z$, then $(\langle 10, 0\rangle, \hat{g}, z) \in \Gamma_{f,\varphi}(\Theta)$.

This is actually the format of clauses 6 and 7. For 6 the subfunctions asserted to exist can simply be taken as $\{\langle \sigma, u\rangle\}$ and $\{\langle u, \sigma, z\rangle\}$, respectively.

2.2.2 Remark. As Moschovakis has pointed out in [113], in many connections it may be more natural to use a stronger notion of computable functional. Let φ be a functional acting on one unary function and which is weakly Θ-computable for some theory Θ. Let $g(x, y)$ be a Θ-computable function. Consider the function

$$f(x) = \varphi(\lambda y \cdot g(x, y)).$$

If $A = C$, this is Θ-computable; we have $f(x) = \{\hat{\varphi}\}(S^1_1(\hat{g}, x))$. If $A \neq C$ we may need the following stronger notion: Let $\varphi(f)$ be a consistent unary functional on unary functions. For each n we define φ^n by

$$\varphi^n(f, \sigma) = \varphi(\lambda y \cdot f(\sigma, y)).$$

φ is called *uniformly* (*weakly*) Θ-*computable* if there is a Θ-computable mapping $p(n)$ such that for each n, φ^n is weakly Θ-computable with Θ-code $p(n)$. It is immediate how to change clause 10 to accommodate the stronger notion.

2.2.3 Definition. The computation set generated by f and φ over $\mathfrak{A}$, which will be called the *prime computation set* in f and φ and be denoted by $\mathrm{PR}[f, \varphi]$, is defined as the least fixed-point of the operator $\Gamma_{f,\varphi}$.

For each $(a, \sigma, z) \in \mathrm{PR}[f, \varphi]$ we set

$$|a, \sigma, z|_{\mathrm{PR}[f,\varphi]} = \text{least } \xi \text{ such that } (a, \sigma, z) \in \Theta^{\xi}_{f,\varphi}.$$

The ordinal number $|a, \sigma, z|_{\mathrm{PR}[f,\varphi]}$ is called *the length of the computation* (a, σ, z).

Note that Definition 2.2.3 makes sense whether we restrict ourselves to φ acting on total functions, i.e. clause 10 of 2.2.1 or allow partial φ as in clause 10′. In either case the notion of length is well defined, even if there is no unique choice of subfunction (as in clauses 6 and 7).

In the next definition we restrict ourselves to functionals acting only on total functions.

2.2.4 Definition. For each $(a, \sigma, z) \in \mathrm{PR}[f, \varphi]$ we introduce the set of *immediate subcomputations.*

(i) If $a = \langle 1, 0\rangle$, $\langle 2, 0\rangle$, $\langle 3, 0\rangle$, $\langle 4, 0\rangle$, $\langle 5, 0\rangle$, or $\langle 9, 0\rangle$, then (a, σ, z) has no immediate subcomputations.

(ii) $(\langle 6, 0\rangle, \hat{f}, \hat{g}, \sigma, z)$ has $(\hat{g}, \sigma, u)$ and $(\hat{f}, u, \sigma, z)$ as immediate subcomputations. (Note that u is uniquely determined.)

(iii) $(\langle 7, j\rangle, \hat{f}, x_1, \ldots, x_n, \tau, z)$ has $(\hat{f}, x_{j+1}, x_1, \ldots, x_j, x_{j+2}, \ldots, x_n, z)$ as immediate subcomputation.

(iv) $(\langle 8, a, x_1, \ldots, x_n\rangle, y_1, \ldots, y_m, z)$ has $(a, x_1, \ldots, x_n, y_1, \ldots, y_m, z)$ as immediate subcomputation.

(v) $(\langle 10, 0\rangle, \hat{g}, z)$ has as immediate subcomputations the set of all tuples $(\hat{g}, u, v)$, where u runs over the whole domain A. (Note that by clause 10 in 2.2.1 $|\hat{g}, u, v| < |\langle 10, 0\rangle, \hat{g}, z|$ for all u, v.)

The *subcomputation* relation is the transitive closure of the immediate subcomputation relation, and is denoted by $<_{\mathrm{PR}[f,\varphi]}$.

We note that if $(a, \sigma, z) <_{\mathrm{PR}[f,\varphi]} (b, \tau, w)$, then $|a, \sigma, z|_{\mathrm{PR}[f,\varphi]} < |b, \tau, w|_{\mathrm{PR}[f,\varphi]}$ but not conversely.

2.2.5 Proposition. *The prime computation set* $\mathrm{PR}[f, \varphi]$ *is a computation theory on* $\mathfrak{A}$ *in which* f *is computable and* φ *is weakly computable.*

We verify that φ is weakly $\mathrm{PR}[f, \varphi]$-computable with code $\langle 10, 0\rangle$. First we must show that

$$\varphi(\{\hat{g}\}) \simeq z \quad \text{iff} \quad (\langle 10, 0\rangle, \hat{g}, z) \in \mathrm{PR}[f, \varphi].$$

Assume $\varphi(\{\hat{g}\}) \simeq z$. Then $\{\hat{g}\}$ satisfies the premiss of clause 10 with $\Theta = \mathrm{PR}[f, \varphi]$. Thus $(\langle 10, 0\rangle, \hat{g}, z) \in \Gamma_{f,\varphi}(\mathrm{PR}[f, \varphi]) = \mathrm{PR}[f, \varphi]$, as $\mathrm{PR}[f, \varphi]$ is a fixed-point for $\Gamma_{f,\varphi}$. For the converse assume that $(\langle 10, 0\rangle, \hat{g}, z) \in \mathrm{PR}[f, \varphi]$. But this is the case only if $\varphi(\{\hat{g}\}) \simeq z$.

The condition on subcomputations follows immediately from clause (v) in Definition 2.2.4.

We make one final construction.

2.2.6 Definition. Let H be computation theory on a domain $\mathfrak{A}$ and let $\mathbf{f}$ and $\boldsymbol{\varphi}$ be lists of functions and functionals over A. We define a theory $H[\mathbf{f}, \boldsymbol{\varphi}]$ on $\mathfrak{A}$ in the following way. Let $\Gamma_{\mathbf{f},\boldsymbol{\varphi}}$ be the operator of Definition 2.2.1. Set

$$\Theta^\xi = \Gamma_{\mathbf{f},\boldsymbol{\varphi}}\left(\bigcup_{\eta<\xi} \Theta^\eta\right) \cup \{(\langle 11, a\rangle, \sigma, z) : (a, \sigma, z) \in H \text{ and } |a, \sigma, z|_H \leqslant \xi\}$$

We now define

$$H[\mathbf{f}, \boldsymbol{\varphi}] = \bigcup_\xi \Theta^\xi.$$

The length function for $H[\mathbf{f}, \boldsymbol{\varphi}]$ is defined as in 2.2.3. In defining the subcomputation relation we add the following clause to Definition 2.2.4.

$$(b, \tau, w) <_{H[\mathbf{f},\boldsymbol{\varphi}]} (\langle 11, a\rangle, \sigma, z) \quad \text{iff} \quad b \text{ is of the form } \langle 11, b'\rangle \text{ and } (b', \tau, w) <_H (a, \sigma, z).$$

We should expect that $H[f, \varphi]$ is in some suitable sense the "least extension" of H in which f and φ are computable. A first step is to introduce the appropriate modification of Definition 1.6.1.

2.2.7 Definition. Let $\langle \Theta, <_\Theta\rangle$ and $\langle H, <_H\rangle$ be two computation theories on a common domain $\mathfrak{A}$. We say that $\langle H, <_H\rangle$ *extends* $\langle \Theta, <_\Theta\rangle$, in symbols,

$$\langle \Theta, <_\Theta\rangle \leqslant \langle H, <_H\rangle,$$

if there is an H-computable mapping $p(a, n)$ such that

1. $(a, \sigma, z) \in \Theta$ iff $(p(a, n), \sigma, z) \in H$,

where $n = \mathrm{lh}(\sigma)$, and

2. if $(a, \sigma, z) \in \Theta$ and $(b, \tau, w) <_\Theta (a, \sigma, z)$, then $(p(b, m), \tau, w) <_H (p(a, n), \sigma, z)$.

If $\langle \Theta, <_\Theta\rangle \leqslant \langle H, <_H\rangle$ and $\langle H, <_H\rangle \leqslant \langle \Theta, <_\Theta\rangle$, we say that the theories are equivalent, $\langle \Theta, <_H\rangle \sim \langle H, <_\Theta\rangle$.

Definitions 2.2.6 and 2.2.7 combine to give the following elementary proposition.

2.2.8 Proposition. *Let* $\langle \Theta, <_\Theta\rangle$ *be a computation theory on a domain* $\mathfrak{A}$, *and let the lists* $\mathbf{f}$ *and* $\boldsymbol{\varphi}$ *be given. Then* $\langle \Theta[\mathbf{f}, \boldsymbol{\varphi}], <_{\Theta[\mathbf{f},\boldsymbol{\varphi}]}\rangle$ *is an extension of* $\langle \Theta, <_\Theta\rangle$.

The imbedding map is simply $p(a, n) = \langle 11, a\rangle$. That the extension is the *least* one in which $\mathbf{f}$ and $\boldsymbol{\varphi}$ are computable will be proved in Section 2.6.

2.3 The First Recursion Theorem

The following recursion theorem will play a central role in our theory. In its present form it is due to Moschovakis [113]. The reader may find it instructive to compare the present version with the corresponding result 1.7.9 for precomputation theories.

2.3.1 First Recursion Theorem. *Let* $\langle \Theta, <\rangle$ *be a computation theory on* $\mathfrak{A}$. *Let*

$\varphi(f, x)$ be a consistent weakly Θ-computable functional. Let f^* be the least solution of the equation

$$\varphi(f, x) = f(x), \quad \textit{all} \quad x \in A.$$

Then f^* is Θ-computable.

For the proof we first remark that a least solution exists and can be defined inductively as $f^* = \bigcup f^\xi$, where

$$f^\xi(x) = \varphi\left(\bigcup_{\eta<\xi} f^\eta, x\right),$$

and $(\bigcup_\eta f^\eta)(x) \simeq z$ iff $\exists\eta[f^\eta(x) \simeq z]$. We will show that $f^*(x)$ is Θ-computable.

First construct a Θ-computable function $g(a, b, x)$ with Θ-code $\hat{g}$ such that for all $a, b \in C$ and all $x \in A$

(i) $g(a, b, x) = \{a\}_\Theta(\{b\}_\Theta(0), x)$,

(ii) if $g(a, b, x) \simeq z$, then $(a, \{b\}_\Theta(0), x, z) <_\Theta (\hat{g}, a, b, x, z)$.

Let $\hat{\varphi}$ be a Θ-code for φ and choose by the second recursion theorem a code $\hat{p}$ such that for all t

$$\{\hat{p}\}_\Theta(t) = S_1^2(\hat{g}, \hat{\varphi}, \hat{p}).$$

Finally, set

$$c = \{\hat{p}\}_\Theta(0) = S_1^2(\hat{g}, \hat{\varphi}, \hat{p}).$$

We will show that

$$f^*(x) \simeq z \quad \text{iff} \quad \{c\}_\Theta(x) \simeq z.$$

We do this by showing: (a) $\{c\}_\Theta$ is a fixed-point for φ, and (b) $\{c\}_\Theta \subseteq f^*$.

(a) Let $\{\hat{\varphi}\}_\Theta(c, x) \simeq z$; by (i) above, noting that $c = \{\hat{p}\}_\Theta(0)$, we get $g(\hat{\varphi}, \hat{p}, x) \simeq z$. Thus $\{S_1^2(\hat{g}, \hat{\varphi}, \hat{p})\}_\Theta(x) \simeq z$, i.e. $\{c\}_\Theta(x) \simeq z$. Running the argument in reverse, we see that $\{c\}_\Theta$ is a fixed-point for φ.

(b) The proof that $\{c\}_\Theta \subseteq f^*$ proceeds by induction on subcomputations. Let $\{c\}_\Theta(x) \simeq z$. This means that $g(\hat{\varphi}, \hat{p}, x) \simeq z$, and hence $\{\hat{\varphi}\}_\Theta(c, x) \simeq z$. By (ii) above

$$(\hat{\varphi}, c, x, z) <_\Theta (\hat{g}, \hat{\varphi}, \hat{p}, x, z).$$

Since $(\hat{g}, \hat{\varphi}, \hat{p}, x, z) <_\Theta (S_1^2(\hat{g}, \hat{\varphi}, \hat{p}), x, z)$ and $c = S_1^2(\hat{g}, \hat{\varphi}, \hat{p})$, we get

$$(\hat{\varphi}, c, x, z) <_\Theta (c, x, z).$$

φ is Θ-computable. Hence there is a partial function $h \subseteq \{c\}_\Theta$ such that $\varphi(h, x) \simeq z$ and $(c, u, v) <_\Theta (\hat{\varphi}, c, x, z)$ for all pairs u, v such that $h(u) \simeq v$. This means that

$$\text{if } h(u) \simeq v, \quad \text{then} \quad \{c\}_\Theta(u) \simeq v \quad \text{and} \quad (c, u, v) <_\Theta (c, x, z).$$

By the induction hypothesis we thus get:

$$\text{if } h(u) \simeq v, \quad \text{then} \quad f^*(u) \simeq v,$$

i.e. $h \subseteq f^*$. The consistency of φ then gives $\varphi(f^*, x) \simeq z$, which by the fixed-point property of f^* further entails that $f^*(x) \simeq z$. This proves that $\{c\}_\Theta \subseteq f^*$.

2.4 *Semicomputable Relations*

At this point it may be appropriate to include a brief discussion of Θ-computable and Θ-semicomputable relations. For comparison the reader should refer back to the discussion in Section 1.3. We will in later parts deal with the topic in greater depth. Here are a few rather superficial preliminary remarks.

2.4.1 Definition. Let $\langle \Theta, < \rangle$ be a computation theory on a domain $\mathfrak{A}$. A relation $R(\sigma)$ is called Θ-*semicomputable* if there exists a Θ-computable (partial) function f such that

$$R(\sigma) \quad \text{iff} \quad f(\sigma) \simeq 0.$$

A relation $R(\sigma)$ is called Θ-*computable* if there is a Θ-computable mapping f such that

$$R(\sigma) \quad \text{iff} \quad f(\sigma) = 0.$$

We note that the condition $f(\sigma) \simeq 0$ in the definition of semicomputable could have been replaced by the condition $f(\sigma)\downarrow$, i.e. "is defined".

There are (at least) three basic properties a decent notion of semicomputability and computability for relations should satisfy.

A. If $R(x, \sigma)$ is Θ-semicomputable, then so is $\exists x R(x, \sigma)$.
B. If $R(x, \sigma)$ and $S(x, \sigma)$ are Θ-semicomputable, then so is $R(x, \sigma) \vee S(x, \sigma)$.
C. A relation R is Θ-computable iff R and $\neg R$ are both Θ-semicomputable.

In Section 1.3 we showed in Theorem 1.3.4 that these properties hold if the theory in question has *selection operators*. The discussion there carries over without any significant change to the present context.

As we remarked in 1.3 it would be extremely restrictive to require a single-valued selection operator. And we are not inclined to save the situation by making our theories multiple-valued as in 1.3. This would lead to unwanted complications in analyzing the subcomputation relation.

Short of introducing property A axiomatically, there seems to be no way of saving it without a selection operator. The $\exists$-quantifier ranges over the whole domain A.

B can be saved if the Θ-semicomputable relations are closed under $\exists$-quantification over the natural numbers N. The proof is essentially the same as in 1.3.4.

2.4.2 Proposition. *Let Θ be a computation theory on $\mathfrak{A}$ and assume that the Θ-semicomputable relations are closed under existential quantification over N. If both $R(\sigma)$ and $S(\sigma)$ are Θ-semicomputable, then so is $R(\sigma) \vee S(\sigma)$.*

For the proof, let r and s be Θ-codes for R and S, respectively. Let $f(r, s)$ be a Θ-code, Θ-computable in r, s, for a Θ-computable mapping such that $\{f(r, s)\}_\Theta(0) = r$ and $\{f(r, s)\}_\Theta(x) = s$, $x \neq 0$. Then $R(\sigma) \vee S(\sigma)$ iff $\exists n \in N[\{\{f(r, s)\}_\Theta(n)\}(\sigma) \simeq 0]$.

In a similar way we see that C will be saved if Θ admits a selection operator over N.

2.4.3 Definition. Let Θ be a computation theory on $\mathfrak{A}$. An n-ary *selection operator for Θ over N* is an $n + 1$-ary Θ-computable function $q(a, \sigma)$ with Θ-code $\hat{q}$ such that

(i) if $q(a, \sigma)\downarrow$, then $q(a, \sigma) \in N$;
(ii) if $\exists n \in N\{a\}_\Theta(n, \sigma) \simeq 0$, then $q(a, \sigma)\downarrow$ and $\{a\}_\Theta(q(a, \sigma), \sigma) \simeq 0$.
(iii) if $\{a\}_\Theta(n, \sigma) \simeq 0$ and $q(a, \sigma) \simeq n$, then $(a, n, \sigma, 0) <_\Theta (\hat{q}, a, \sigma, n)$.

2.4.4 Proposition. *Let Θ be a computation theory on $\mathfrak{A}$ admitting a selection operator over N. A relation R is Θ-computable iff R and $\neg R$ are Θ-semicomputable.*

The proof is exactly as in 1.3.4. Let r, s be codes for $R, \neg R$, respectively. Let $m(r, s)$ be a code (Θ-computable in r, s) for the Θ-semicomputable relation

$$(\{r\}_\Theta(\sigma) \simeq 0 \wedge t = 0) \vee (\{s\}_\Theta(\sigma) \simeq 0 \wedge t = 1).$$

Using the selection operator we get a Θ-computable *mapping* $f(\sigma) = q(m(r, s), \sigma)$ such that

$$R(\sigma) \quad \text{iff} \quad f(\sigma) = 0.$$

2.4.5 Remark. The hypothesis of Proposition 2.4.4 implies the hypothesis of Proposition 2.4.2 (see 2.4.3 (ii)). In the next chapter we shall study an important class of theories which admit selection operators over N.

2.5 *Finiteness*

The importance of the notion of *finiteness* in general recursion theory was strongly emphasized by G. Kreisel (see e.g. [89, 91]), and many of our further developments will testify to his insight. In this section we shall, following Moschovakis [113], introduce a notion of finiteness for general computation theories and prove a few basic properties.

In order that our notion of finiteness shall be well behaved we must restrict the class of computation theories somewhat.

2.5.1 Definition. A computation theory $\langle \Theta, < \rangle$ on $\mathfrak{A}$ is called *regular* if

(a) $C = A$.
(b) Θ has selection operators over N.

2.5.2 Remarks. First observe that 2.5.1 (a), *via* definition by cases, implies that the equality relation on A is Θ-computable. Finite theories on two types will not satisfy this condition, but will still admit a reasonable theory of finiteness, see Chapter 4. The infinite theories of Chapter 5 will be regular in the above sense, as will be the finite theories of Chapter 3.

2.5.3 Definition. Let Θ be a computation theory on $\mathfrak{A}$. Let $B \subseteq A$, by the *B-quantifier* on $\mathfrak{A}$ we understand the following consistent functional $\mathbf{E}_B(f)$ defined as

$$\mathbf{E}_B(f) \simeq \begin{cases} 0, & \text{if } \exists x \in B[f(x) \simeq 0] \\ 1, & \text{if } \forall x \in B[f(x) \simeq 1]. \end{cases}$$

The set $B \subseteq A$ is called Θ-*finite* with Θ-canonical code e, if the B-quantifier $\mathbf{E}_B$ is weakly Θ-computable with Θ-code e.

Some of the basic properties of finiteness are given in the next theorem.

2.5.4 Theorem. *Let Θ be a regular computation theory on $\mathfrak{A}$:*

1. *If B is a finite subset of A, then B is Θ-finite.*
2. *If B is Θ-finite, then B is Θ-computable.*
3. *If B is Θ-finite, $D \subseteq B$ and D is Θ-computable, then D is Θ-finite.*
4. *If B is Θ-finite and f is a Θ-computable mapping, then $f[B]$ is Θ-finite.*
5. *If B is Θ-finite and f is a Θ-computable mapping such that for all $x \in B$, $f(x)$ is a Θ-canonical code for a Θ-finite set B_x, then $\bigcup_{x \in B} B_x$ and $\bigcap_{x \in B} B_x$ are Θ-finite.*

We shall briefly indicate the proofs.

1. For simplicity assume that $B = \{y_1, y_2\} \subseteq A$. First find a Θ-code a^* (as a Θ-computable mapping of y_1, y_2) of the Θ-semicomputable relation

$$\{a^*(y_1, y_2)\}_\Theta(\hat{f}, z) \simeq 0 \quad \text{iff} \quad (\{\hat{f}\}(y_1) \simeq 0 \vee \{\hat{f}\}(y_2) \simeq 0) \wedge z = 0 \cdot \vee \cdot (\{\hat{f}\}(y_1) \simeq 1 \wedge \{\hat{f}\}(y_2) \simeq 1) \wedge z = 1.$$

(We here use the results of Section 2.4.) Let q be the selection operator on N. We set $\{e\}_\Theta(\hat{f}) = q(a^*(y_1, y_2), \hat{f})$. e will be a Θ-canonical code for $B = \{y_1, y_2\}$ as a Θ-finite set. We first see that

$$\begin{aligned} \mathbf{E}_B(\{\hat{f}\}) \simeq 0 \quad &\text{iff} \quad \exists y \in B[\{\hat{f}\}(y) \simeq 0]. \\ &\text{iff} \quad q(a^*(y_1, y_2), \hat{f}) \simeq 0 \\ &\text{iff} \quad \{e\}_\Theta(\hat{f}) \simeq 0 \end{aligned}$$

Similarly we see that $\mathbf{E}_B(\{\hat{f}\}) \simeq 1$ iff $\{e\}_\Theta(\hat{f}) \simeq 1$.

It remains to verify the subcomputation condition on $\mathbf{E}_B$: Suppose $\mathbf{E}_B(\{\hat{f}\}) \simeq z$, where $z = 0$ or $z = 1$. Let g be the subfunction of $\{\hat{f}\}$ which gives the values of $\{\hat{f}\}$ at the arguments y_1, y_2. By construction $(a^*, \hat{f}, z, 0) <_\Theta (\hat{q}, a^*, \hat{f}, z)$ and $(\hat{q}, a^*, \hat{f}, z) <_\Theta (e, \hat{f}, z)$. By construction of a^* we also see that $(\hat{f}, y_i, v) <_\Theta (a^*, \hat{f}, z, 0)$, whenever $g(y_i) \simeq v$. Combining the inequalities, the result follows.

2. Construct a Θ-computable function $g(x)$ such that

$$\{g(x)\}_\Theta(u) = \begin{cases} 0, & \text{if } u = x \\ 1, & \text{if } u \neq x. \end{cases}$$

Then $f(x) = \mathbf{E}_B(\{g(x)\})$ is a Θ-computable mapping such that $x \in B$ iff $f(x) = 0$.

3. Let f_D be defined as follows

$$f_D(x) \simeq \begin{cases} f(x), & \text{if } x \in D \\ 1, & \text{if } x \notin D. \end{cases}$$

We then observe that $\mathbf{E}_D(f) \simeq \mathbf{E}_B(f_D)$.

4. In this case let

$$\mathbf{E}_{f[B]}(g) = \mathbf{E}_B(g \circ f),$$

where $(g \circ f)(x) \simeq z$ iff $\exists y[f(x) \simeq y \wedge g(y) \simeq z]$.

5. We consider the case $B_0 = \bigcup_{x \in B} B_x$. We have the following equivalences:

$$\begin{aligned} \mathbf{E}_{B_0}(g) \simeq 0 \quad &\text{iff} \quad \exists y \in B_0[g(y) \simeq 0] \\ &\text{iff} \quad \exists x \in B[\exists y \in B_x[g(y) \simeq 0]] \\ &\text{iff} \quad \exists x \in B[\mathbf{E}_{B_x}(g) \simeq 0] \\ &\text{iff} \quad \mathbf{E}_B(\lambda x \cdot \mathbf{E}_{B_x}(g)) \simeq 0. \\ \mathbf{E}_{B_0}(g) \simeq 1 \quad &\text{iff} \quad \forall y \in B_0[g(y) \simeq 1] \\ &\text{iff} \quad \forall x \in B[\forall y \in B_x[g(y) \simeq 1]] \end{aligned}$$

$$\text{iff} \quad \forall x \in B[\mathbf{E}_{B_x}(g) \simeq 1]$$
$$\text{iff} \quad \mathbf{E}_B(\lambda x \cdot \mathbf{E}_{B_x}(g)) \simeq 1.$$

The assumption that we have a Θ-computable mapping f such that $f(x)$ is a Θ-canonical code for B_x, for all $x \in B$, shows that $\mathbf{E}_{B_0}$ is Θ-computable.

2.6 *Extension of Theories*

In Definition 2.2.7 we introduced a notion of Extension, $\Theta \leqslant H$, for computation theories. We remarked in Proposition 2.2.8 that if $\mathbf{f}$ and $\boldsymbol{\varphi}$ are Θ-computable, then $\Theta[\mathbf{f}, \boldsymbol{\varphi}]$, as constructed in 2.2.6, is an extension of Θ. We left open the problem whether $\Theta[\mathbf{f}, \boldsymbol{\varphi}]$ is the least extension of Θ in which $\mathbf{f}$ and $\boldsymbol{\varphi}$ are computable, i.e. if whenever $\mathbf{f}$ and $\boldsymbol{\varphi}$ are H-computable, and H extends Θ, then $\Theta[\mathbf{f}, \boldsymbol{\varphi}] \leqslant H$.

This question has no simple answer for arbitrary consistent partial functionals due to the fact that there is no canonical choice of *subcomputations* for partial objects of higher types. The situation is not problematic if we compute relative to functionals acting only on total objects.

2.6.1 Proposition. *Let* $\langle \Theta, <_\Theta \rangle$ *be a computation theory on a domain* $\mathfrak{A}$ *and let* $\mathbf{f}$ *and* $\boldsymbol{\varphi}$ *be given lists of functions and* total *functionals on* $\mathfrak{A}$. *Let* $\langle H, <_H \rangle$ *be any extension of* Θ *in which* $\mathbf{f}$ *and* $\boldsymbol{\varphi}$ *are computable. Let* H *further satisfy the following condition*

(*) *If* φ_i *is in the list* $\boldsymbol{\varphi}$ *and if* $\varphi_i(\{\hat{g}\}) \simeq z$ *is an* H*-computation, then* $\{\hat{g}\}$ *is total and* $(\hat{g}, u, v)$ *is an* H*-subcomputation of* $(\hat{\varphi}_i, \hat{g}, z)$ *for all* $u, v \in A$.

Then there exists an imbedding of $\Theta[\mathbf{f}, \boldsymbol{\varphi}]$ *into* H.

We shall give the proof in some detail. It is a typical and reasonably complex example of an index transfer theorem based on the second recursion theorem. We have given:

1. $\Theta \leqslant H$ *via* an H-computable mapping $q(a, n)$.
2. $\mathbf{f} = f$ (for simplicity) is H-computable with code $\hat{f}$.
3. $\boldsymbol{\varphi} = \varphi$ is Θ-computable with code $\hat{\varphi}$, and assume for simplicity that φ is unary, i.e. of the form $\varphi(g)$.

By 2.2.6 $\Theta[f, \varphi]$ is inductively constructed using an operator $\Gamma_{f,\varphi}$. We construct the reduction map $p(a, n)$ of $\Theta[f, \varphi]$ to H by cases according to the form of the index a:

(i) If $a = \langle 1, 0 \rangle$, $\langle 2, 0 \rangle$, $\langle 3, 0 \rangle$, $\langle 4, 0 \rangle$, or $\langle 5, 0 \rangle$, then we set $p(a, n) = a'$, where a' is the corresponding code in H.

(ii) Let $a = \langle 6, 0 \rangle$. This is the case of substitution $\varphi(g, h, \sigma) = g(h(\sigma), \sigma)$. We want to have

$$(p(a, n + 2), \hat{g}, \hat{h}, \sigma, z) \in H \quad \text{iff} \quad (a, \hat{g}, \hat{h}, \sigma, z) \in \Theta[f, \varphi].$$

In order to do this we must go through an intermediate stage $(a', p(\hat{g}, n+1), p(\hat{h}, n), \sigma, z) \in H$, where a' is the fixed H-code for the substitution functional in H.

We first construct in H a mapping $t(a')$ such that

$$\{t(a')\}_H(\hat{p}, \hat{g}, \hat{h}, \sigma) \simeq z \quad \text{iff} \quad \{a'\}_H(\{\hat{p}\}(\hat{g}, n+1), \{\hat{p}\}(\hat{h}, n), \sigma) \simeq z$$

where $\hat{p}$ is a H-code for the map $p(a, n)$. We can then take

$$p(a, n+2) = S^1_{n+2}(t(a'), \hat{p}).$$

The construction of $t(a')$ must be such that whenever u is an element such that $\{\{\hat{p}\}(\hat{h}, n)\}_H(\sigma) \simeq u$ and $\{\{\hat{p}\}(\hat{g}, n+1)\}_H(u, \sigma) \simeq z$, then

$$(\{\hat{p}\}(\hat{h}, n), \sigma, u) \text{ and } (\{\hat{p}\}(\hat{g}, n+1), u, \sigma, z) <_H (t(a'), \hat{p}, \hat{g}, \hat{h}, \sigma, z).$$

(iii) If $a = \langle 7, 0\rangle$, we proceed as in case (ii).

(iv) The case $a = \langle 8, 0\rangle$ is similar to case (i).

(v) If $a = \langle 9, 0\rangle$, we set $p(a, n) = \hat{f}$, the given H-code for f.

(vi) Let $a = \langle 10, 0\rangle$, here we are in the case of introducing the functional φ. If we have a tuple $(\langle 10, 0\rangle, \hat{g}, z) \in \Theta[f, \varphi]$, this means that for some total function $\{\hat{g}\}$, $\varphi(\{\hat{g}\}) \simeq z$. This is a statement in the theory $\Theta[f, \varphi]$, where the function $\{\hat{g}\}$ occurs "extensionally".

Considered as a statement inside H, we must have $\varphi(\{p(\hat{g}, 1)\}) \simeq z$, since extensionally $\{\hat{g}\}_\Theta = \{p(\hat{g}, 1)\}_H$. This means that $(\hat{\varphi}, p(\hat{g}, 1), z) \in H$ where $\hat{\varphi}$ is the given H-code for φ. We must define $p(\langle 10, 0\rangle, 1)$ such that

$$\begin{aligned}(p(\langle 10, 0\rangle, 1), \hat{g}, z) \in H \quad &\text{iff} \quad (\varphi, p(\hat{g}, 1), z) \in H \\ &\text{iff} \quad (\langle 10, 0\rangle, \hat{g}, z) \in \Theta[f, \varphi].\end{aligned}$$

It is indeed possible to construct $p(\langle 10, 0\rangle, 1)$ in such a fashion, and moreover, such that

$$(\hat{\varphi}, p(\hat{g}, 1), z) <_H (p(\langle 10, 0\rangle, 1), \hat{g}, z).$$

(This is completely analogous to the construction in case (ii).)

(vii) If $a = \langle 11, a_1\rangle$, we simply set $p(a, n) = q(a_1, n)$, q being the mapping which imbeds Θ into H.

If a is not of one of the forms considered in (i)–(vii), we set $p(a, n) = 0$. An application of the second recursion theorem gives us $p(a, n)$ as an H-computable mapping. It remains to verify that $\Theta[f, \varphi] \leqslant H$ *via* $p(a, n)$.

We have to prove:

(1) If $(a, \sigma, z) \in \Theta[f, \varphi]$, then $(p(a, n), \sigma, z) \in H$.

(2) If $(a_0, \sigma, z) \in H$ and $a_0 = p(a, n)$, then $(a, \sigma, z) \in \Theta[f, \varphi]$.

(3) The subcomputation condition of Definition 2.2.7.

We take case (ii) in the construction of $p(a, n)$ as representative for (1) and (2).

(1) Let $a = \langle 6, 0\rangle$: If some $(\langle 6, 0\rangle, \hat{g}, \hat{h}, \sigma, z) \in \Theta[f, \varphi]$, then there exists a unique u such that $(\hat{h}, \sigma, u), (\hat{g}, u, \sigma, z) \in \Theta[f, \varphi]$. By the induction hypothesis $(p(\hat{h}, n), \sigma, u), (p(\hat{g}, n + 1)\, u, \sigma, z) \in H$. Hence $(a', p(\hat{g}, n + 1), p(\hat{h}, n), \sigma, z) \in H$, which by construction of p gives that $(p(a, n + 2), \hat{g}, \hat{h}, \sigma, z) \in H$.

(2) Conversely, we proceed by induction on the subcomputation relation $<_H$: Assume that we have the premiss of (2) above with $a = \langle 6, 0\rangle$. From the construction of t and the transitivity of $<_H$ we conclude that

$$(p(\hat{h}, n), \sigma, u) \quad \text{and} \quad (p(\hat{g}, n + 1), u, \sigma, z) <_H (a_0, \sigma, z),$$

for a suitable u. The induction hypothesis yields that $(\hat{g}, u, \sigma, z), (\hat{h}, \sigma, u) \in \Theta[f, \varphi]$, hence $(a, \hat{g}, \hat{h}, \sigma, z) \in \Theta[f, \varphi]$.

In order to verify (3) we have to show that if (b, τ, w) and (a, σ, z) are $\Theta[f, \varphi]$-computations and $(b, \tau, u) <_{\Theta[f,\varphi]} (a, \sigma, z)$, then

$$(p(b, m), \tau, w) <_H (p(a, n), \sigma, z).$$

This is proved by induction on the subcomputation relation $<_{\Theta[f,\varphi]}$. We treat case (vi) in the construction of $p(a, n)$ as an example.

Thus let $(\langle 10, 0\rangle, \hat{g}, z) \in \Theta[f, \varphi]$. $\{\hat{g}\}$ is then total and has H-code $p(\hat{g}, 1)$. In this case it suffices to verify that

$$(+) \qquad (p(\hat{g}, 1), u, v) <_H (\hat{\varphi}, p(\hat{g}, 1), z),$$

for all u, v. The conclusion then follows from the way in which $p(\langle 10, 0\rangle, 1)$ was constructed.

Since $(\langle 10, 0\rangle, \hat{g}, z) \in \Theta[f, \varphi]$, the induction hypothesis tells us that $(\hat{\varphi}, p(\hat{g}, 1), z) \in H$. We now invoke condition (*) to conclude that $(p(\hat{g}, 1)\, u, v)$, for all $u \in A$, is an H-subcomputation of $(\hat{\varphi}, p(\hat{g}, 1), z)$, which is the desired conclusion (+).

Remark. There is a stronger notion of imbedding which requires in Definition 2.2.7 that $(b, \tau, w) <_\Theta (a, \sigma, z)$ iff $(p(b, m), \tau, w) <_H (p(a, n), \sigma, z)$. If H were a well-behaved theory, e.g. of the type PR[**f**, **φ**], we could prove Proposition 2.6.1 using this strengthened notion of imbedding. In a general H we lack enough information about $<_H$ to do so.

An immediate corollary is that PR[**f**, **φ**] $\leqslant H$ if every item in the lists **f** and **φ** are H-computable in the appropriate sense, in particular, every φ_i in **φ** is total. We further note the following:

2.6.2 Proposition. *Let* $\langle \Theta, <_\Theta\rangle$ *and* $\langle H, <_H\rangle$ *be computation theories on the same domain* $\mathfrak{A}$. *Let* **f** *and* **φ** *be given lists, where every* φ_i *in* **φ** *is total. If* $\Theta \leqslant H$, *then*

$$\Theta[\mathbf{f}, \boldsymbol{\varphi}] \leqslant H[\mathbf{f}, \boldsymbol{\varphi}].$$

Finally we state the following transitivity (or cut-elimination) lemma.

2.6.3 Proposition. *Let G, H be either partial functions or total functionals on the domain $\mathfrak{A}$, and $\boldsymbol{\varphi}$ a list of total functionals on the same domain. If G is prime computable in $H, \mathbf{f}, \boldsymbol{\varphi}$ (i.e. G is* PR$[H, \mathbf{f}, \boldsymbol{\varphi}]$*-computable in the appropriate sense) and H is prime computable in $\mathbf{f}, \boldsymbol{\varphi}$, then G is prime computable in $\mathbf{f}, \boldsymbol{\varphi}$.*

The reader is invited to construct a proof for him or herself proceeding along the lines of the proof of 2.6.1. Let us also repeat in order to avoid a possible ambiguity that by "total" we mean the same as in the introduction to this section, i.e. we compute relative to functionals acting only on total objects.

2.7 Faithful Representation

We have so far restricted ourselves to functionals acting only on total functions over the domain. The definitions in Section 2.1 do, however, make sense in the more general situation of consistent partial functionals. And the construction in Definition 2.2.1 with clause 10′ replacing 10, as well as Definition 2.2.3 work equally well in the partial case.

It is with the notion of *subcomputations* that difficulties arise. And this is a problem we cannot gloss over by suppressing subcomputations in favor of *length of computations*. In computing in partial objects of higher types there is in general no canonical choice of subcomputations—incompatible subcomputations may serve the same purpose.

The difference between total and partial functionals is brought out clearly by contrasting clauses 10 and 10′ in Definition 2.2.1. At the same time we note that in clauses 6 and 7 there are canonical choices.

One situation where a partial φ leads to problems is when we want to compare its "computability" in different theories. Proposition 2.6.1 is a case in point. Let Θ and H be two theories on a common domain and assume that $\Theta \leqslant H$. From this we would like to conclude that if φ is H-computable, then $\Theta[\varphi] \leqslant H$. But there are difficulties, the subfunction g_H "picked" by the relation $<_H$ to witness a computation $\varphi(g) \simeq z$ in H need not be an extension of the subfunction $g_{\Theta[\varphi]}$ used to secure the computation $\varphi(g) \simeq z$ inside $\Theta[\varphi]$. (Referring to the proof of 2.6.1, it is not necessary that $(\hat{g}, u, v) <_{\Theta[f,\varphi]} (\langle 10, 0\rangle, \hat{g}, z)$ implies that $(p(\hat{g}, 1), u, v) <_H (p(\langle 10, 0\rangle, 1), \hat{g}, z)$.)

Computing in total objects is in an obvious sense *deterministic*, computing in partial objects need not be. In the latter case there is no unique subcomputation tree to verify a computation $\{a\}_\Theta(\sigma) \simeq z$. There is in general a family of subcomputation trees which act as possible *derivations* in Θ of the "theorem" $\{a\}_\Theta(\sigma) \simeq z$.

As this section and other parts of this work will show (see e.g. (4) of Example 3.1.3), there are many cases where we can handle the more general situation, but we have no abstract axiomatic analysis to offer.

2.7.1 Remark. It should be clear that we do not expect a general version of Proposition 2.6.1 for partial functionals. However, a suitable version of the cut-elimination Lemma 2.6.3 is valid for partial functionals and their "derivations".

We now turn to the main topic of this section, a representation theorem for computation theories which is faithful to the notion of subcomputation, i.e. which preserves the complexity of subcomputations.

As always let $\mathfrak{A}$ be a computation domain and $\langle \Theta, < \rangle$ a computation theory on $\mathfrak{A}$. Let $x \in \Theta$. In connection with Definition 2.1.1 we introduced the set of subcomputations

$$\mathbf{S}_x = \{(a, \sigma, z) : (a, \sigma, z) < x\}.$$

We impose the following regularity condition on Θ.

2.7.2 Definition. The theory Θ is called *s-normal* (subcomputation normal) if the sets $\mathbf{S}_x$ are uniformly Θ-finite for $x \in \Theta$.

We spell this out in a bit more detail. Let $\langle x \rangle$ denote the usual code of the finite sequence x. The sets $\mathbf{S}_x$ are uniformly Θ-finite if there exists a Θ-computable mapping p such that whenever $x \in \Theta$, then $p(\langle x \rangle)$ is a Θ-canonical code for $\mathbf{S}_x$ as a Θ-finite set.

We can now state the main result of this section.

2.7.3 Theorem. Faithful Representation. *Let $\langle \Theta, < \rangle$ be an s-normal theory on the domain $\mathfrak{A}$. There exists a partial weakly Θ-computable functional φ such that Θ is equivalent to* PR[φ].

The idea of the proof is rather straightforward. We want to construct φ such that

a. $\varphi(f, \langle a, \sigma \rangle) \simeq z$ iff $(a, \sigma, z) \in \Theta$ and
$\forall (b, \tau, w) \in \mathbf{S}_{(a,\sigma,z)}[f(\langle b, \tau \rangle) \simeq w]$.

This φ is seen to be consistent, and if it is weakly Θ-computable, it has, by Theorem 2.3.1 a least fixed-point f_0. We would like to show that f_0 reduces Θ to PR[φ]. If we construct PR[φ] in the correct way, i.e. are careful in introducing the notion of subcomputation for φ, the converse reduction will also follow.

Before going into the details of the proof, we indicate the following part of the reduction $\Theta \leqslant \text{PR}[\varphi]$ by f_0.

1. $(a, \sigma, z) \in \Theta$ iff $(\hat{f_0}, \langle a, \sigma \rangle, z) \in \text{PR}[\varphi]$.

First, assume that $(a, \sigma, z) \in \Theta$ and that **1** is true for all $(b, \tau, w) < (a, \sigma, z)$. From **a** it then follows that $\varphi(f_0, \langle a, \sigma \rangle) \simeq z$, which by the fixed-point property of f_0, gives $(\hat{f_0}, \langle a, \sigma \rangle, z) \in \text{PR}[\varphi]$. Conversely, if $(\hat{f_0}, \langle a, \sigma \rangle, z) \in \text{PR}[\varphi]$, then $\varphi(f_0, \langle a, \sigma \rangle) \simeq z$, hence $(a, \sigma, z) \in \Theta$.

For the proof we need the following auxiliary construction.

2.7.4 WDC (**Weak Definition by Cases**). Let f and g be functions of the same number of arguments.

$$\mathrm{WDC}(f, g, x, y, \sigma) \simeq \begin{cases} f(\sigma) & \text{if } x = y \\ g(\sigma) & \text{if } x \neq y. \end{cases}$$

This is to be understood in the following sense: If $x = y$, then $\mathrm{WDC}(f, g, x, y, \sigma)\downarrow$ and equal to $f(\sigma)$ iff $f(\sigma)\downarrow$; the question whether $g(\sigma)\downarrow$ is irrelevant. Similarly if $x \neq y$, then $\mathrm{WDC}(f, g, x, y, \sigma)\downarrow$ iff $g(\sigma)\downarrow$.

We construct an index for WDC uniformly in indices for f and g. First let

$$\pi(\hat{f}, \hat{g}, i) = \begin{cases} \hat{f} & \text{if } i = 0 \\ \hat{g} & \text{if } i \neq 0. \end{cases}$$

Next, let $\mathrm{eq}(x, y)$ test equality on the code set C, i.e.

$$\mathrm{eq}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y. \end{cases}$$

We now see that

$$\mathrm{WDC}(\{\hat{f}\}, \{\hat{g}\}, x, y, \sigma) \simeq \{\pi(\hat{f}, \hat{g}, \mathrm{eq}(x, y))\}(\sigma),$$

and hence we have an index $\hat{w}$ for WDC as a weakly Θ-computable functional.

We now proceed to the proof of Theorem 2.7.3. First let $\hat{g}_0$ be an index for the totally undefined function, i.e. $\{\hat{g}_0\}(\sigma)\uparrow$ for all σ. Next, introduce the following functional

$$(*) \qquad \kappa(f, \langle a, \sigma\rangle) \simeq \begin{cases} 0, & \text{if } \forall \langle b, \tau, z\rangle \in \mathbf{S}_{\langle a,\sigma,-\rangle}(f(\langle b, \tau\rangle) \simeq z) \\ \uparrow, & \text{ow.} \end{cases}$$

This definition requires a comment. We compute $\kappa(\{\hat{f}\}, \langle a, \sigma\rangle)$ by starting to compute $\{a\}(\sigma)$. If $\{a\}(\sigma)\downarrow$ and $\simeq z$, we then decide, using s-normality, for each $(b, \tau, w) < (a, \sigma, z)$ whether $\{\hat{f}\}(\langle b, \tau\rangle) \simeq w$.

We construct a code in the following way. First recall that if $(a, \sigma, z) \in \Theta$, then $p(\langle a, \sigma, z\rangle)$ is a Θ-canonical code for the set $\mathbf{S}_{\langle a,\sigma,z\rangle}$. Next let $\hat{f}_1$ be a code computable in $\hat{f}$ such that $\{\hat{f}_1\}(\langle b, \tau\rangle, w) \simeq 1$ iff $\{\hat{f}\}(\langle b, \tau\rangle) \simeq w$. Then we see that

$$\kappa(\{\hat{f}\}, \langle a, \sigma\rangle) \simeq 0 \quad \text{iff} \quad \{p(\langle a, \sigma, \{a\}(\sigma)\rangle)\}(\hat{f}_1) \simeq 1,$$

from which an appropriate code $\hat{\kappa}$ for κ is easily extracted. But $\hat{\kappa}$ constructed above may also make κ defined in cases other than those indicated in (*). This we get rid of by using the totally undefined $\{\hat{g}_0\}$ in WDC as follows,

(**) $\varphi(f, \langle a, \sigma\rangle) \simeq \mathrm{WDC}(\{a\}, \{\hat{g}_0\}, \kappa(f, \langle a, \sigma\rangle), 0, \sigma)$.

Using the codes $\hat{w}$ and $\hat{\kappa}$ it is not difficult to construct a code $\hat{\varphi}_1$ such that if f is Θ-computable with code $\hat{f}$, then

(***) $\varphi(\{\hat{f}\}, \langle a, \sigma\rangle) \simeq \{\hat{\varphi}_1\}(\hat{f}, \langle a, \sigma\rangle)$.

from which we can conclude that φ is a weakly Θ-computable functional. We may also now prove **a** above. Then apply the first recursion theorem to get a least Θ-computable fixed-point f_0 for φ.

It remains to construct the "correct" version of PR[φ] for proving that $\Theta \sim$ PR[φ], i.e. to define for the set PR[φ] a "correct" notion of immediate subcomputation. Let us use $\hat{\varphi}$ as PR[φ]-code for φ and $\hat{f}_0$ as the PR-code for the least fixed-point for φ.

The notion of subcomputation for φ in PR[φ] can then be determined from condition **a** above. We want an interaction of $\hat{f}_0$ and $\hat{\varphi}$ such that if $(f_0, \langle a, \sigma\rangle, z) \in$ PR[φ], then $(\hat{\varphi}, \hat{f}_0, \langle a, \sigma\rangle, z) <_{\mathrm{PR}[\varphi]} (\hat{f}_0, \langle a, \sigma\rangle, z)$. Further, the subfunction h used in the calculation $\varphi(h, \langle a, \sigma\rangle) \simeq z$ shall consist of all tuples $(\hat{f}_0, \langle b, \tau\rangle, w)$ such that $(b, \tau, w) <_\Theta (a, \sigma, z)$.

Remark. The precise clause corresponding to clause 10 of Definition 2.2.1 is

$$\text{If } (\hat{f}, \langle b, \tau\rangle, w) \in \Theta \quad \text{for all} \quad (b, \tau, w) <_\Theta (a, \sigma, z) \quad \text{and} \quad (a, \sigma, z) \in \Theta, \quad \text{then} \quad (\hat{\varphi}, \hat{f}, \langle a, \sigma\rangle, z) \in \Gamma(\Theta^*).$$

From this we determine the "correct" notion of immediate subcomputation replacing clause (v) of Definition 2.2.4.

With this version of PR[φ] we can prove the reduction $\Theta \leqslant$ PR[φ] by f_0.

2. $(b, \tau, w) <_\Theta (a, \sigma, z)$ iff $(\hat{f}_0, \langle b, \tau\rangle, w) <_{\mathrm{PR}[\varphi]} (\hat{f}_0, \langle a, \sigma\rangle, z)$.

Assume that $(b, \tau, w) <_\Theta (a, \sigma, z)$. Using the subfunction h above we see that $h(\langle b, \tau\rangle) \simeq w$, hence

$$(\hat{f}_0, \langle b, \tau\rangle, w) <_{\mathrm{PR}[\varphi]} (\hat{\varphi}, \hat{f}_0, \langle a, \sigma\rangle, z) <_{\mathrm{PR}[\varphi]} (\hat{f}_0, \langle a, \sigma\rangle, z).$$

Conversely, we note that if $(\hat{f}_0, \langle b, \tau\rangle, w) <_{\mathrm{PR}[\varphi]} (\hat{f}_0, \langle a, \sigma\rangle, z)$, then also $(\hat{f}_0, \langle b, \tau\rangle, w) <_{\mathrm{PR}[\varphi]} (\hat{\varphi}, \hat{f}_0, \langle a, \sigma\rangle, w)$ by the way things are constructed. Again, by property **a** of φ and the minimality of the fixed-point f_0, this means that $(a, \sigma, z) \in \Theta$ and $(b, \tau, w) <_\Theta (a, \sigma, z)$.

2.7.5 Remark. A weak version of Theorem 2.7.3 was proved by J. Moldestad (unpublished), viz. he obtained a reduction

1′ $(a, \sigma, z) \in \Theta$ iff $(\hat{f}_0, \langle a, \sigma, z\rangle, 0) \in \mathrm{PR}[\varphi]$.

The difference between this result and Theorem 2.7.3 is a bit subtle and merits a comment, it is the difference between a function and its graph. Going from the graph to the function we need some sort of selection operator, and, indeed, in the presence of a selection operator we immediately get 1 from 1′. We were able in 2.7.3 to prove the strong version without assuming a selection operator in the theory.

2.7.6 Remark. A computation theory $\langle \Theta, < \rangle$ on a domain $\mathfrak{A}$ is also a precomputation theory on $\mathfrak{A}$. If we are only interested in the Θ-semicomputable relations on $\mathfrak{A}$, this is adequately represented by the semicomputable relations in some theory PR[**f**], where **f** is a list of partial functions on $\mathfrak{A}$, see Theorem 1.6.3. But this reduction trivializes the computation structure, everything is coded into the functions **f**. What is added in the representation theorem of this section is that by moving up one type, i.e. by considering suitable theories of the type PR[φ], where φ is a consistent partial functional on $\mathfrak{A}$, then we can preserve the structure of subcomputations, in particular, the length function associated with subcomputations.