Part I

Introduction

Chapter 1

Motivation

Making use of highly effective computation performed in parallel, modern computer science technologies are designed to work with information of enormous size. During a single computational step, however, a computational process usually needs only some selected elements of the information. Typically, it selects only a finite number of scattered, but often mutually related elements of the information, intuitively referred to as *scattered information* in this book, and simultaneously derives new pieces of information from these elements. By substituting the new pieces of information for the selected elements, it produces new information and, thereby, completes such a computational step. The newly obtained information is subsequently processed in the same selective way during the next computational step. This computational process successfully ends when some terminating conditions concerning the information are fulfilled; as a rule, these conditions simply require that the information consists solely of elements from a prescribed terminating set. Although this way of scattered computation processing frequently underlies modern information technologies, computer science has not monographically formalized this processing so far, so it lacks any systematic body of knowledge regarding this important kind of information processing. The present book fills this gap.

In general, to discuss some kind of information processing in a rigorous way in computer science, we often formalize the corresponding information processors by formal language models, such as suitable grammars. Then, by studying these grammars and the way they yield their languages in strictly mathematical terms of formal language theory, we actually rigorously investigate the processors that work with the information under discussion. Following this mathematical approach, we formalize the scattered information processors by scattered context grammars, introduced by formal language theory several decades ago (see [2]). We have chosen the theory of formal languages because it straightforwardly and naturally allows us to formalize the scattered information within information i by *scattered context* of symbols, A_1 through A_n, occurring in i as $i = x_0A_1x_1A_2x_2 \ldots A_nx_n$, where the x's represent parts of i irrelevant to the current computational step. A computational rule according to which the computational step, referred to as *derivation step* in formal language theory, is made, has the form $(A_1, A_2, \ldots, A_n) \rightarrow (y_1, y_2, \ldots, y_n)$, where y_1 through y_n represent the new piece of information derived from A_1 through A_n. In this way, we formalize all the computational

rules according to which the computational process performs its steps. As these *scattered context rules* actually define how the words are changed, the set containing these rules forms a *scattered context grammar* as a whole. These rules operate over two sets of symbols. The grammar contains a finite set of *terminal symbols*. Apart from the terminal symbols, it also contains finitely many *nonterminal symbols*, one of which is defined as the *start symbol*. Beginning from the start symbol, the formalized computational process consisting of a sequence of derivation steps successfully ends when the derived words consist solely of terminal symbols. A terminal word derived in this way is included into the language of this grammar, which contains all words derived in this way. As proved later in this book, scattered context grammars are equivalent to Turing machines in the sense that both define the family of recursively enumerable languages. As a result, these grammars represent not only an elegant formalization of scattered information processing, but also a powerful generator of languages.

To illustrate the formalization of scattered information processing, consider a file f in its binary form. Double f by attaching f to its duplicate as ff, which is a common file operation in practice. Suppose we want to specify the generative process that produces the set of all files doubled in this way. More formally speaking, we want to generate the language $L = \{ff : f \in \{0,1\}^*\}$. Generating this set requires to guarantee that every member of this set is formed by a binary word followed by an equally long binary word that coincides with the preceding word. In greater detail, we need to make sure that both words consist of the same number of binary digits and, in addition, the ith binary digit in the first word coincides with the ith binary digit in the second word. In this case, scattered information processing consists in verifying the identity of the ith binary digit in the first f and the ith binary digit in the second f. The start rule derives FF from the start symbol S by a rule of the form $(S) \to (FF)$. Then, to guarantee that the ith binary digit in the left word coincides with the ith binary digit in the right word, we simultaneously derive two identical binary digits from the left and from the right F by the scattered context rules $(F,F) \to (0F,0F)$ and $(F,F) \to (1F,1F)$. To complete this process, we simultaneously erase both F's by $(F,F) \to (\varepsilon,\varepsilon)$, where ε is the empty word, which consists of no symbols at all. For instance, 0101 is derived by a sequence of derivation steps S, FF, $0F0F$, $01F01F$, 0101.

To enhance the reason why we have chosen scattered context grammars as a proper grammatical formalization of scattered information processing, we now consider how some other fundamental formal grammars describe context dependencies and perform their derivation steps in order to see why these grammars are less appropriate for this formalization than scattered context grammars. Regarding the context description, we can divide grammatical rules into context-dependent and context-independent rules in the theory of formal languages. Accordingly, we distinguish between context-dependent grammars, such as phrase-structure grammars or interactive Lindenmayer systems, and context-independent grammars, such as context-free grammars. Context-independent rules are applied quite independently of other symbols, so they are obviously incapable of formalizing the mutually related elements of scattered information. Besides,

they are significantly less powerful than scattered context grammars; in fact, they cannot even generate the trivial language L above. Context-dependent grammars are more powerful; however, their context-dependent rules depend on the context surrounding the rewritten symbol, so they fail to appropriately formalize widely scattered elements of information. Concerning the description of scattered context, all these grammars are thus less suitable than scattered context grammars. The performance of derivation steps in grammars should obviously reflect the way scattered information is processed in reality. Therefore, sequential grammars, such as context-free grammars, can hardly serve as a proper formalization of scattered information processing because they rewrite only a single symbol during a derivation step. Although totally parallel grammars, such as L systems, reflect scattered information processing more appropriately, this reflection is still inadequate from a realistic point of view. Indeed, these grammars work in a completely parallel way because they rewrite all symbols of a word during a single derivation step. However, due to hardware-related limitations, only a finite number of symbols can be processed during one computational step in practice. As scattered context grammars simultaneously rewrite finitely many selected symbols during a single derivation step while leaving the other symbols unchanged, they formalize scattered information processing most appropriately.