EATCS
Monographs on Theoretical Computer Science
Volume 15
Editors: W. Brauer G. Rozenberg A. Salomaa

Seppo Sippu
Eljas Soisalon-Soininen

Parsing Theory

Volume I
Languages and Parsing

Springer-Verlag
世界图书出版公司

Seppo Sippu
Eljas Soisalon-Soininen

Parsing Theory

Volume I
Languages and Parsing

With 55 Figures

Springer-Verlag Berlin Heidelberg New York
London Paris Tokyo
世界图书出版公司
北京・广州・上海・西安

Authors
Professor S. Sippu
Department of Computer Science, University of Jyväskylä
Seminaarinkatu 15, SF-40100 Jyväskylä, Finland

Professor E. Soisalon-Soininen
Department of Computer Science, University of Helsinki
Teollisuuskatu 23, SF-00510 Helsinki, Finland

Editors
Prof. Dr. Wilfried Brauer
Institut für Informatik, Technische Universität München
Arcisstr. 21, D-8000 München 2, Germany

Prof. Dr. Grzegorz Rozenberg
Institute of Applied Mathematics and Computer Science
University of Leiden, Niels-Bohr-Weg 1, P.O. Box 9512
NL-2300 RA Leiden, The Netherlands

Prof. Dr. Arto Salomaa
Department of Mathematics, University of Turku
SF-20500 Turku 50, Finland

ISBN 3-540-13720-3 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-13720-3 Springer-Verlag New York Heidelberg Berlin

Library of Congress Cataloging-in-Publication Data
Sippu, Seppo, 1950–
Parsing theory / Seppo Sippu, Eljas Soisalon-Soininen.
p. cm. – (EATCS monographs on theoretical computer science ; v. 15)
Bibliography: p.
Included index.
Contents: v. 1. Languages and parsing.
ISBN 0-387-13720-3 (U.S.: v. 1)
1. Parsing (Computer grammar) 2. Formal languages. I. Soisalon-Soininen, Eljas, 1949– II. Title. III. Series
QA267.3.S59 1988 511.3–dc 19 88-20091 CIP

Reprinted by World Publishing Corporation, Beijing, 1993
for distribution and sale in The People's Republic of China only
ISBN 7 5062 1529 2

EATCS
Monographs on Theoretical Computer Science
Volume 15

Preface

The theory of parsing is an important application area of the theory of formal languages and automata. The evolution of modern high-level programming languages created a need for a general and theoretically clean methodology for writing compilers for these languages. It was perceived that the compilation process had to be "syntax-directed", that is, the functioning of a programming language compiler had to be defined completely by the underlying formal syntax of the language. A program text to be compiled is "parsed" according to the syntax of the language, and the object code for the program is generated according to the semantics attached to the parsed syntactic entities.

Context-free grammars were soon found to be the most convenient formalism for describing the syntax of programming languages, and accordingly methods for parsing context-free languages were developed. Practical considerations led to the definition of various kinds of restricted context-free grammars that are parsable by means of efficient deterministic linear-time algorithms.

Today, the theory of parsing is a well-established area of computer science. The most notable individual achievements in the area date from as early as the 1960s. These include the two major deterministic parsing methods now used in programming language compilers: LR(k) parsing and LL(k) parsing. However, since the invention of these methods, a great deal of research has been done on their analysis and on the practical issues involved in implementing parsers. As a result of this research, constructing a parser for a programming language is no longer an ad hoc task but a completely automatic process executed by a compiler writing system.

This monograph is intended as an up-to-date reference work on the theory of deterministic parsing of context-free grammars. The material included is treated in depth, with emphasis on the LR(k) and LL(k) methods, which are developed in a uniform way. Special attention is paid to the efficient implementation of LR(k) and LL(k) parsers. Construction algorithms for parsers are derived from general graph-theoretic methods, and complexity questions about parsable grammars are analyzed.

The treatment is mathematical in spirit, and contributes to the analysis of algorithms. The work tries to be self-contained in that relevant results from the general theory of formal languages and computational complexity are cited explicitly in the text (usually as propositions). For some of these results, a proof is also provided.

"Parsing Theory" appears in two volumes, "Volume I: Languages and Parsing" (Chapters 1 to 5) and "Volume II: LR(k) and LL(k) Parsing" (Chapters 6 to 10). The two volumes form an integrated work, with chapters, theorems, lemmas, etc. numbered consecutively.

Volume I provides an introduction to the basic concepts of languages and parsing. It also contains the relevant mathematical and computer science background needed in the development of the theory of deterministic parsing. In Chapter 1, concepts from discrete mathematics, formal languages and computational complexity are reviewed. Chapter 2 contains the basic algorithms on relations and graphs needed later in constructing parsers. In Chapter 3, the main classical results on regular languages are reviewed, with emphasis on the complexity of algorithms. Chapter 4 is a short introduction to context-free grammars and related concepts. Volume I ends with Chapter 5, which introduces the concepts of a pushdown automaton, pushdown transducer, left parser, right parser, strong LL(k) parser, and simple precedence parser. In this chapter, the emphasis is on the analysis of strong LL(k) parsing and on the efficient construction and implementation of strong LL(1) parsers.

Volume II contains a thorough treatment of the theory of LR(k) and LL(k) parsing. The topics covered are: LR(k), LALR(k), and SLR(k) parsers and grammars (Chapter 6), construction and implementation of LR(1) parsers (Chapter 7), LL(k) parsers and grammars, and non-left-recursive grammatical covers (Chapter 8), syntax error recovery and reporting (Chapter 9), and the complexity of testing grammars for parsability (Chapter 10).

This work is intended to be used as a textbook at graduate and senior undergraduate levels. A suitable background for a student would be an elementary knowledge of formal language theory, complexity, data structures and analysis of algorithms.

Some of the material has been used in a one-semester course on parsing theory at the University of Helsinki. A one-semester course on the basic theory of languages and parsing can be taught from Volume I. The whole material in both volumes can perhaps most conveniently be covered in an advanced two-semester course on parsing theory.

Numerous exercises are provided at the end of each chapter. The bibliographic notes attempt to point to a published source for exercises that are more difficult than average or that cover topics not discussed in the text.

Jyväskylä and Helsinki, March 1988 Seppo Sippu
Eljas Soisalon-Soininen

Acknowledgements
The work was supported by the Academy of Finland, the Finnish Cultural Foundation, and the Ministry of Education of Finland.

Contents

1. Elements of Language Theory 1

1.1 Mathematical Preliminaries 1
1.2 Languages 6
1.3 Random Access Machines 12
1.4 Decision Problems. 14
1.5 Computational Complexity 19
1.6 Rewriting Systems. 24
Exercises 30
Bibliographic Notes 35

2. Algorithms on Graphs 37

2.1 Basic Algorithms 37
2.2 Finding Strongly Connected Components. 39
2.3 Computing Functions Defined on Graphs 48
2.4 Computing Relational Expressions. 53
Exercises 60
Bibliographic Notes 64

3. Regular Languages 65

3.1 Regular Expressions 65
3.2 Finite Automata 72
3.3 Regular Grammars 83
3.4 Deterministic Finite Automata 87
3.5 Decision Problems on Regular Languages 91
3.6 Applications to Lexical Analysis. 97
Exercises 106
Bibliographic Notes 113

4. Context-free Languages 115

4.1 Context-free Grammars 115
4.2 Leftmost and Rightmost Derivations. 118
4.3 Ambiguity of Grammars 122
4.4 Useless and Nullable Symbols. 129
4.5 Canonical Two-form Grammars. 133
4.6 Derivational Complexity 136

4.7 Context-free Language Recognition 144
Exercises 147
Bibliographic Notes 151

5. Parsing 153

5.1 Pushdown Automata 154
5.2 Left Parsers and Right Parsers 160
5.3 Strong LL(k) Parsing 170
5.4 Strong LL(k) Grammars 178
5.5 Construction of Strong LL(1) Parsers 185
5.6 Implementation of Strong LL(1) Parsers 193
5.7 Simple Precedence Parsing 199
Exercises 206
Bibliographic Notes 214

Bibliography to Volume I 217

Index to Volume I 221

1. Elements of Language Theory

In this chapter we shall review the mathematical and computer science background on which the presentation in this book is based. We shall discuss the elements of discrete mathematics and formal language theory, emphasizing those issues that are of importance from the point of view of context-free parsing. We shall devote a considerable part of this chapter to matters such as random access machines and computational complexity. These will be relevant later when we derive efficient algorithms for parsing theoretic problems or prove lower bounds for the complexity of these problems. In this chapter we shall also discuss a general class of formal language descriptors called "rewriting systems" or "semi-Thue systems". Later in the book we shall consider various language descriptors and language recognizers as special cases of a general rewriting system. As this approach is somewhat unconventional, we advise even the experienced reader to go through the definitions given in this chapter if he or she wishes to appreciate fully the presentation in this book.

The first two sections of this chapter contain a brief introduction to relations, directed graphs, trees, functions, countable sets, monoids, strings, homomorphisms and languages. Section 1.3 deals with the abstract model of a computer on which the algorithms presented in this book are intended to run. This model coincides with the conventional random access machine model except that we allow nondeterministic programs. Section 1.4 deals with decision problems and solvability, and Section 1.5 discusses the complexity of programs in our model of computation. Finally, Section 1.6 defines a general rewriting system and related concepts such as derivations, time complexity, and space complexity in rewriting systems.

1.1 Mathematical Preliminaries

Let A and B be sets. A *relation R from A to B*, denoted by $R\colon A \to B$, is any subset of the Cartesian product of A and B, i.e. $R \subseteq A \times B$. A is the *domain* and B the *range* of R. R is a relation *on* A if $A = B$. If a pair (a, b) is in R, we say that a is *R-related to b*, and write $a\ R\ b$.

If A' is a subset of A, we call the set

$$R(A') = \{b \in B \mid a\ R\ b \text{ for some } a \in A'\}$$

the *image of A′ under R*. In the case of a singleton set $\{a\}$ we may write $R(a)$ for $R(\{a\})$.

The relation R^{-1} from B to A defined by

$$R^{-1} = \{(b,a) \in B \times A \mid a\,R\,b\}$$

is called the *inverse* of R.

The *(relational) product* of relations $R_1 : A \to B$ and $R_2 : B \to C$, denoted by $R_1 R_2$, is the relation from A to C defined by

$$R_1 R_2 = \{(a, c) \in A \times C \mid a\,R_1\,b \text{ and } b\,R_2\,c \text{ for some } b \in B\} \ .$$

The relational product $R_1 R_2$ is sometimes called the *composition* of R_1 and R_2 and may also be denoted by $R_2 \circ R_1$ (note the reversed order).

Fact 1.1 Multiplication of relations is an associative binary operation on the set of all relations. That is, for any relations $R_1 : A \to B$, $R_2 : B \to C$ and $R_3 : C \to D$, we have

$$R_1(R_2 R_3) = (R_1 R_2) R_3 \ .$$

Thus we may omit the parentheses and write $R_1 R_2 R_3$. □

We say that a relation R on a set A is

(1) *reflexive*, if $a\,R\,a$ for all $a \in A$—in other words, R includes the *identity relation* $\mathrm{id}_A = \{(a, a) \mid a \in A\}$ on A;

(2) *symmetric*, if $a\,R\,b$ always implies $b\,R\,a$—in other words, $R^{-1} = R$;

(3) *antisymmetric*, if $a\,R\,b$ and $b\,R\,a$ always imply $a = b$—in other words, $R^{-1} \cap R \subseteq \mathrm{id}_A$;

(4) *transitive*, if $a\,R\,b$ and $b\,R\,c$ always imply $a\,R\,c$—in other words, $RR \subseteq R$.

Let R be a relation on A and n a natural number. The n^{th} *power of R* (or *n-fold product of R*), denoted by R^n, is defined inductively by

(1) $R^0 = \mathrm{id}_A$;
(2) $R^n = RR^{n-1}$, for $n > 0$.

Fact 1.2 $a\,R^n\,b$ if and only if for some $a_0, \ldots, a_n \in A$, $a = a_0$, $a_n = b$ and $a_i\,R\,a_{i+1}$ for all $i = 0, \ldots, n-1$. □

The *transitive closure* of R, denoted by R^+, is the relation on A defined by

$$R^+ = \bigcup_{n=1}^{\infty} R^n \ .$$

The *reflexive transitive closure* of R, denoted by R^*, is the relation on A defined by

$$R^* = \bigcup_{n=0}^{\infty} R^n .$$

Thus $R^* = R^0 \cup R^+ = \mathrm{id}_A \cup R^+$.

Lemma 1.3 *Let R be a relation on a set A. Then R^+ is the smallest transitive relation on A that includes R, and R^* is the smallest reflexive and transitive relation on A that includes R. In other words, the following statements hold:*

(1) *R^+ is transitive and $R \subseteq R^+$.*
(2) *$R^+ \subseteq R'$ whenever R' is a transitive relation on A such that $R \subseteq R'$.*
(3) *R^* is reflexive and transitive and $R \subseteq R^*$.*
(4) *$R^* \subseteq R'$ whenever R' is a reflexive and transitive relation on A such that $R \subseteq R'$.* □

Let A' be a subset of A. Then $R^+(A')$, the image of A' under R^+, is called the *positive closure of A' under R*, and $R^*(A')$, the image of A' under R^*, is called the *closure of A' under R*.

A pair $G=(A, R)$ is a *directed graph* (or *graph* for short) if A is a set and R is a relation on A. The elements of A are called *nodes* (or *vertices*) of G and the elements of R *edges* (or *arcs*) of G. An edge (a, b) is said to *leave* node a and to *enter* node b. If (a, b) is an edge, node a is called a *predecessor* of node b, and node b a *successor* of node a. In the figures in this book we usually represent an edge (a, b) by an arrow that goes from a to b (see Figure 1.1).

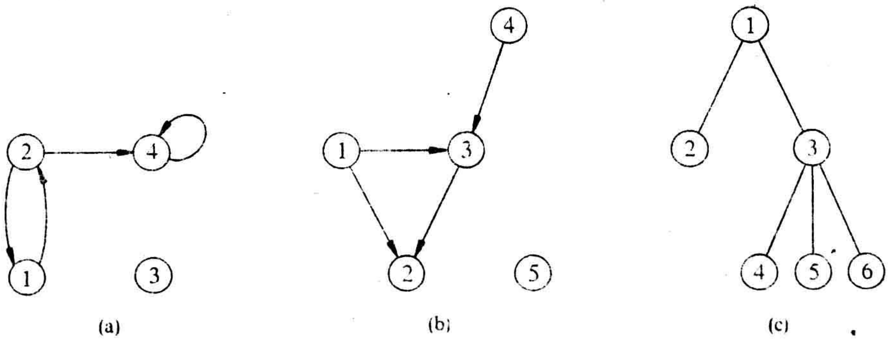

Figure 1.1 Examples of graphs. **(a)** A cyclic directed graph $(\{1, 2, 3, 4\}, \{(1, 2), (2, 1), (2, 4), (4, 4)\})$. **(b)** An acyclic directed graph $(\{1, 2, 3, 4, 5\}, \{(1, 2), (1, 3), (3, 2), (4, 3)\})$. (c) A tree $(\{1, 2, 3, 4, 5, 6\}, \{(1, 2), (1, 3), (3, 4), (3, 5), (3, 6)\})$

A sequence of nodes $(a_0, a_1, \ldots, a_n)$, $n \geq 0$, is a *path of length n from a_0 to a_n* in a graph G if for all $i=0, \ldots, n-1$ (a_i, a_{i+1}) is an edge of G.

Fact 1.4 The following statements hold for all $n \geq 0$ and nodes a, b of a graph $G=(A, R)$:

(1) $a R^n b$ if and only if there is a path of length n from a to b in G.
(2) $a R^* b$ if and only if there is a path from a to b in G.
(3) $a R^+ b$ if and only if there is a path of positive length from a to b in G. □

Let $G = (A, R)$ be a graph. The graph (A, R^+) is called the *transitive closure* of G and is denoted by G^+, and the graph (A, R^*) is called the *reflexive transitive closure* of G and is denoted by G^*. A *subgraph* of G is any graph (A', R') where $A' \subseteq A$ and $R' = (A' \times A') \cap R$.

A *cycle* is a path of positive length from a node to itself. A graph is said to be *cyclic* if it contains a cycle, otherwise it is *acyclic*.

An acyclic graph is a *tree* if there is a node r, called the *root*, such that for any other node a there is exactly one path from r to a. If (a, b) is an edge of a tree, a is called the *father* of b, and b a *son* of a. If there is a path of positive length from node a to node b, we say that a is an *ancestor* of b, and b is a *descendant* of a. A node having no sons is called a *leaf*. A *subtree* of a tree (A, R) is any tree (A', R') which is a subgraph of (A, R) and in which no node is an ancestor of any node in $A \setminus A'$. In the figures, we usually represent a tree so that the root is at the top and the leaves at the bottom. Sons are connected to their fathers by plain lines (without arrowheads: see Figure 1.1c).

A relation R on a set A is an *equivalence* if it is reflexive, symmetric and transitive. The image $R(a)$ of a singleton set $\{a\}$ under an equivalence R on A is called the *equivalence class of a under R* and is denoted by $[a]_R$. (We may drop the subscript R and write $[a]$ if there is no ambiguity.)

Fact 1.5 For any equivalence R on a set A, the set $\{[a]_R \mid a \in A\}$ is a *partition* of A, that is, A is the union of the sets $[a]_R$, $a \in A$, and the intersection of any two distinct equivalence classes is empty. Conversely, if $\mathbb{P}$ is a partition of a set A, the relation R defined by

$$R = \{(a, b) \mid a, b \in B \text{ for some } B \text{ in } \mathbb{P}\}$$

is an equivalence on A with $\{[a]_R \mid a \in A\} = \mathbb{P}$. □

A relation R on A is a *(reflexive) partial order* if it is reflexive, antisymmetric and transitive. A partial order R on A is a *total order* (or *linear order*) if for all $a, b \in A$ either $a R b$ or $b R a$. If R is a partial order, the relation $R \setminus R^0$ is called an *irreflexive partial order*. We often use $\leqslant$ to denote a partial order. Then $<$ denotes the corresponding irreflexive partial order $\leqslant \setminus \leqslant^0$. Thus $a < b$ if and only if $a \leqslant b$ and $a \neq b$. Furthermore, we may denote $\leqslant^{-1}$ by $\geqslant$ and $<^{-1}$ by $>$.

If $\leqslant$ is a partial order on a set A, the pair $(A, \leqslant)$ is called a *partially ordered set*. If $\leqslant$ is a total order, $(A, \leqslant)$ is a *totally ordered set*.

An element $a \in A$ is *maximal* with respect to a partial order $\leqslant$ on A if $a < b$ is false for all $b \in A$, and *minimal* if $b < a$ is false for all $b \in A$. If $\leqslant$ is a total order, A can have at most one maximal element and at most one minimal element. When these exist, we call them the *maximum* and *minimum* of A with respect to $\leqslant$ and denote them by $\max_{\leqslant} A$ and $\min_{\leqslant} A$ (or max A and min A for short).

A relation f from a set A to a set B is a *partial function* (or *partial mapping*) if for all $a \in A$, $f(a)$ contains at most one element. If in addition f is *defined* for all $a \in A$, i.e., if $f(a)$ is nonempty for all $a \in A$, then f is called a (*total*) *function* (or *mapping*).

If f is a partial function and $f(a) = \{b\}$ then we write $f(a) = b$.

Let f be a function from A to B and A' a subset of A. The *restriction of f to A'* is the function f' from A' to B that agrees with f on A', i.e., $f'(a) = f(a)$ for all $a \in A'$. The restriction f' is sometimes denoted by $f|A'$.

A function f from A to B is an *injection* (or *one-to-one*) if its inverse f^{-1} is a partial function from B to A, or, equivalently, if $f(a) = f(b)$ always implies $a = b$. f is a *surjection* (or *onto*) if $f(A) = B$. A function that is both an injection and a surjection is called a *bijection*.

Fact 1.6 The following statements hold for all sets A, B and C:

(1) id_A is a bijection from A to A.

(2) If f is a bijection from A to B, then f^{-1} is a bijection from B to A.

(3) If f is a bijection from A to B and g is a bijection from B to C, then fg is a bijection from A to C. □

Let $\mathbb{U}$ be a collection of sets. ($\mathbb{U}$, the "universe", is assumed to contain all sets under discussion.) We say that a set A in $\mathbb{U}$ is *isomorphic with* a set B in $\mathbb{U}$, written $A \cong B$, if there is a bijection from A to B.

Fact 1.6 immediately implies

Fact 1.7 The set isomorphism $\cong$ is an equivalence relation on $\mathbb{U}$. □

The equivalence classes under set isomorphism are called *cardinal numbers*. A cardinal number $[A]_{\cong}$ is denoted by $|A|$ and is called the *size* (or *cardinality*) of set A.

A set is *finite* if it has the same size as the set $\{1, 2, \ldots, n\}$ for some natural number n. (We take $\{1, 2, \ldots, n\}$ to mean the empty set $\emptyset$ if $n = 0$.) A set is *infinite* if it is not finite.

Since $|\{1, 2, \ldots, m\}| = |\{1, 2, \ldots, n\}|$ if and only if $m = n$, we can denote $|\{1, 2, \ldots, n\}|$ by n. Thus the size of a finite set is the number of elements in the set.

We say that a set A is *countable* (or *denumerable*) if it has the same size as some subset of $\mathbb{N}$, the set of all natural numbers. A set is *uncountable* (or *nondenumerable*) if it is not countable. A countable infinite set is called *countably infinite*.

Proposition 1.8 *Any countably infinite set is of size* $|\mathbb{N}|$. □

If a set is countable, we can enumerate its elements and write $\{a_0, a_1, \ldots\}$; if it is finite, we can write it as $\{a_0, a_1, \ldots, a_n\}$ for some n.

Lemma 1.9 Let $\{A_n | n = 0, 1, \ldots\}$ *be a collection of pairwise disjoint finite sets. Then the set*

$$\bigcup_{n=0}^{\infty} A_n$$

is countable. □

If A and B are sets, we define

$$A^B = \{f \mid f \text{ is a function from } B \text{ to } A\} \ .$$

The elements in $A^{\mathbb{N}}$ are called *infinite strings* (or *sequences*) *over* A (or *of* elements in A). If $f \in A^{\mathbb{N}}$ and $f(i) = a_i$, $i = 0, 1, \ldots$, we write

$$f = (a_0, a_1, \ldots) \ .$$

We now show that $\{0, 1\}^{\mathbb{N}}$, the set of all infinite strings over $\{0, 1\}$, is uncountable. We use a method of proof known as *Cantor's Diagonal Argument*. This involves assuming that the set is countable and deriving a contradiction. So, assuming that $\{0, 1\}^{\mathbb{N}}$ is countable, we can write

$$\{0, 1\}^{\mathbb{N}} = \{f_0, f_1, \ldots\}$$

for some infinite strings f_i over $\{0, 1\}$, $i \in \mathbb{N}$. Each f_i can be written as

$$f_i = (a_{i0}, a_{i1}, \ldots) \ ,$$

where $a_{ij} \in \{0, 1\}$ for all $j \in \mathbb{N}$. Using the "diagonal elements" a_{ii}, $i \in \mathbb{N}$, we can then construct f, another infinite string over $\{0, 1\}$:

$$f = (b_0, b_1, \ldots) \ ,$$

where the elements b_i are defined by

$$b_i = \begin{cases} 0, \text{ if } a_{ii} = 1 \ ; \\ 1, \text{ if } a_{ii} = 0 \ ; \end{cases}$$

Now f is not equal to f_i for any $i \in \mathbb{N}$, because

$$f(i) = b_i \neq a_{ii} = f_i(i) \ .$$

Thus f is not in the set $\{f_0, f_1, \ldots\}$, which is a contradiction.

We therefore have

Theorem 1.10 $\{0, 1\}^{\mathbb{N}}$, *the set of all infinite strings over* $\{0, 1\}$, *is uncountable.* □

1.2 Languages

A language whose sentences are written using letters from an alphabet V is defined mathematically as a subset of an algebraic structure called the "free monoid generated by V". In what follows we shall define this structure and other algebraic concepts needed in the formal treatment of languages.

A pair $(M, \cdot)$ is a *semigroup* if M is a set and $\cdot$ is an associative binary operation on M. That is, $\cdot$ is a function from $M \times M$ to M that satisfies

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

for all $x, y, z \in M$. Here we have used the infix notation $x \cdot y$ for the image $\cdot(x, y)$. If no ambiguity arises, we may even abbreviate this to xy.

An element $e \in M$ is an *identity* of a semigroup $(M, \cdot)$ if for all $x \in M$

$$ex = xe = x \ .$$

Lemma 1.11 *A semigroup has at most one identity.* □

A triple $(M, \cdot, e)$ is a *monoid* (or *semigroup with identity*) if $(M, \cdot)$ is a semigroup and e its identity. If no ambiguity arises, we may denote a semigroup $(M, \cdot)$ or a monoid $(M, \cdot, e)$ simply by M.

Fact 1.12 Let A be a set and let $\cdot$ be the multiplication of relations on A. Then $(2^{A \times A}, \cdot, \mathrm{id}_A)$ is a monoid. (Here $2^{A \times A}$ denotes the set of all subsets of $A \times A$, i.e., the set of all relations on A.) □

Let M be a monoid, x an element of M and n a natural number. The n^{th} *power of* x, denoted by x^n, is defined inductively by

(1) $x^0 = e$;

(2) $x^n = xx^{n-1}$, for $n > 0$.

Let A and B be subsets of a monoid $(M, \cdot, e)$. The operation $\cdot$ induces in a natural way a binary operation $\cdot$ on 2^M, the set of all subsets of M. This binary operation is defined by

$$A \cdot B = \{x \cdot y \mid x \in A \text{ and } y \in B\}$$

for all subsets A and B of M.

Fact 1.13 Let $(M, \cdot, e)$ be a monoid. Then $(2^M, \cdot, \{e\})$, where $\cdot$ is the induced operation, is also a monoid. □

The monoid $(2^M, \cdot, \{e\})$ is called the monoid *induced by* $(M, \cdot, e)$ *on* 2^M.

If A is a subset of M and x is an element of M, we may write (in the induced monoid) xA in place of $\{x\}A$ and Ax in place of $A\{x\}$.

A subset A of a monoid M is *closed* if for all natural numbers n

$$x_1, \ldots, x_n \in A \text{ always implies } x_1 \ldots x_n \in A \ .$$

We take $x_1 \ldots x_n$ to mean the identity e if $n = 0$. A is *positively closed* if the above implication is true for all positive n.

Fact 1.14 Let A be a subset of a monoid. Then

(1) A is positively closed if and only if $x, y \in A$ always implies $xy \in A$.

(2) A is closed if and only if A is positively closed and contains the identity e. □

Fact 1.15 Let A be a closed subset of a monoid $(M, \cdot, e)$. Then $(A, \cdot, e)$, where $\cdot$ is the restriction of the operation of M to $A \times A$, is also a monoid. □

Such a monoid $(A, \cdot, e)$ is called a *submonoid* of $(M, \cdot, e)$.

Let A be any subset of a monoid $(M, \cdot, e)$. The *positive closure* of A, denoted by A^+, is defined by

$$A^+ = \bigcup_{n=1}^{\infty} A^n \ .$$

The *closure* of A, denoted by A^*, is defined by

$$A^* = \bigcup_{n=0}^{\infty} A^n \ .$$

Here A^n means the n^{th} power of A in the induced monoid $(2^M, \cdot, \{e\})$. We have

$$A^* = A^0 \cup A^+ = \{e\} \cup A^+ \ .$$

Lemma 1.16 *Let A be a subset of a monoid M. Then A^+ is the smallest positively closed subset of M that includes A, and A^* is the smallest closed subset of M that includes A. In other words, the following statements hold:*

(1) *A^+ is positively closed and $A \subseteq A^+$.*
(2) *$A^+ \subseteq B$ whenever B is a positively closed subset of M such that $A \subseteq B$.*
(3) *A^* is closed and $A \subseteq A^*$.*
(4) *$A^* \subseteq B$ whenever B is a closed subset of M such that $A \subseteq B$.* □

Note the analogy between Lemmas 1.3 and 1.16.

A subset B of a monoid M *generates* (or *spans*) M if $B^* = M$. B is then called a *basis* (or *generator*) of M.

If B generates M then, by definition, any $x \in M$ has a representation as a product $x_1 \ldots x_n$ of elements $x_1, \ldots, x_n$ of B for some $n \geq 0$. We say that B generates M *freely* if this representation is always unique, i.e., for all $x \in M$ there is exactly one natural number n and exactly one sequence of elements $x_1, \ldots, x_n$ of B such that $x = x_1 \ldots x_n$. M is called a *free monoid* if it contains a subset B which freely generates it.

Lemma 1.17 *Let M be a free monoid. Then M has left and right cancellation, i.e., for all $x, y, z \in M$*

(1) *$zx = zy$ implies $x = y$.*
(2) *$xz = yz$ implies $x = y$.* □