MEMBERSHIP TESTING IN TRANSFORMATION MONOIDS

by

Martin Beaudry

THESIS

submitted to the faculty of graduate studies and research
in partial fulfillment of the requirements for the degree of

PHILOSOPHIAE DOCTOR

School of Computer Science

McGill University

Montreal

Résumé

Etant donnés un ensemble fini X de points, un ensemble de transformations de X (générateurs) et une autre transformation f de X, on analyse la complexité du problème de l'appartenance, qui consiste à décider si f peut être obtenu par compositions successives des générateurs. Ce problème est étudié pour diverses classes (pseudovariétés) de monoides. Sa complexité est NP-difficile pour les monoides de seuil 2 ou plus, en particulier NP-complet pour les monoides commutatifs, J- et R-triviaux. Pour les monoides idempotents (apériodiques de seuil 1), le problème de l'appartenance est NP-complet dans le cas général. On identifie également la plus grande classe de monoides apériodiques pour laquelle le problème appartient à FL, de même que la plus grande classe pour laquelle le problème de l'appartenance n'est pas NP-difficile.

Le problème qui consiste à caractériser un monoide idempotent est également étudié: étant donné un ensemble de transformations, on peut décider en NC^2 si le monoide qu'il engendre est idempotent. Des tests similaires sont donnés pour les classes de monoides idempotents $\mathbf{R}_1$ et $\mathbf{L}_1$, dont la complexité est NC^1.

Une borne supérieure dans le modèle séquentiel classique est fournie pour chacune des complexités en parallèle susmentionnées.

Abstract

Given a finite set X of states, a finite set of transformations of X (generators), and another transformation f of X, we analyze the complexity of the membership problem, which consists in deciding whether f can be obtained by composition of the generators. This problem is studied for various classes (pseudovarieties) of monoids. It is shown that the complexity is NP-hard for monoids of threshold 2 or more, and NP-complete in commutative, J- and R-trivial monoids. For idempotent monoids (aperiodic of threshold one), the problem is NP-complete in the general case; subcases are analyzed, and a largest class of aperiodic monoids is identified for which the problem is in FL, as well as a largest class for which the problem is not NP-hard.

The problem which consists in characterizing an idempotent monoid is also addressed: given a set of transformations, it can be decided in NC^2 whether the monoid they generate is idempotent. Similar tests are given for the classes of idempotent monoids $\mathbf{R}_1$, and $\mathbf{L}_1$. In these cases, the complexity is NC^1.

A sequential upper bound is also given for each of the parallel complexities given above.

Remerciements

J'adresse mes principaux remerciements à Denis Thérien, le mérite de cette thèse lui revenant tout autant qu'à moi. Il y a contribué avec de nombreuses suggestions et critiques, et aussi avec tout l'enthousiasme, la confiance et l'érudition qui ont pu être nécessaires.
Je tiens aussi à remercier Pierre McKenzie, dont l'apport en idées et en commentaires fut également inestimable.
Les remarques et commentaires des membres du comité d'examen, MM. Avis, Barrington, De Mori, Friedman et Styan, ont également été appréciés.
Je remercie enfin ma famille et mes collègues, dont John Kirkpatrick et Ryan Hayward, qui, au cours de ces longues années, m'ont encouragé, aidé, et aussi enduré.

Au cours de mes études à McGill, j'ai bénéficié de l'aide financière des organismes suivants: le Conseil de recherches en sciences naturelles et génie, le Fonds de formation de chercheurs et d'aide à la recherche et la Fondation McConnell.

Contents

Résumé ii
Abstract iii
Remerciements iv
Contents v
List of figures vi
1. Introduction 1
2. Background 4
2.1. Monoids and their representations 4
2.2. Complexity Theory 8
2.3. Evaluating computational complexities 10
2.4. Dual classes 12
2.5. Pseudovarieties discussed in the thesis 14
3. Commutative monoids 16
3.1. A general algorithm for membership testing 16
3.2. Membership in aperiodic commutative monoids 20
3.3. Commutative monoids of threshold one 21
3.4. Restrictions on the monoids 22
4. Idempotent monoids 24
4.1. Testing for idempotency 24
4.2. The Membership Problem belongs to *NP* 25
4.3. Alphabets, J-classes and strongly connected components 27
4.4. An intuitive look at idempotent transformation monoids 29
4.5. The lattice of idempotent pseudovarieties 32
4.6. NP-completeness of the Membership Problem 39
5. Other pseudovarieties 43
5.1. Other aperiodic pseudovarieties 43
5.2. Non-aperiodic pseudovarieties 46
6. Transformation semigroups 48
6.1. Idempotent semigroups 49
6.2. Non-idempotent aperiodic semigroups 52
7. Related problems 58
7.1. Intersection of regular languages 58
7.2. Reduction of finite automata 60
Bibliography 63
Appendix: Abelian Group Membership in sequential 67

List of figures

4.1. Pseudovarieties of idempotent monoids 33
4.2. Connected components for the proof of Theorem 4.18 35
4.3. Connected component for the proof of Theorem 4.25 40
5.1. Some pseudovarieties of monoids 43
6.1. Pseudovarieties of idempotent semigroups 51
6.2. Connected component for the proof of Theorem 6.5 54
7.1. Automaton built for the proof of Fact 7.1 59
7.2. Automaton built for the proof of Fact 7.2 60

I Introduction

This thesis analyses the computational complexity of the Membership Problem, defined as follows:
Given a set A of transformations of a set X (both finite), and another transformation f, decide whether f belongs to the monoid generated by A.
The input for an instance of this problem is a description of the action of the n transformations of A on the m elements of X, plus a similar description for f. The size of the input and the computational complexity of the problem are measured in terms of the two parameters m and n.
This complexity ranges from next-to-trivial (a monoid can have as little as one element), up to complete for *PSpace* in the general case, a result proved by Kozen [Koz]. The Problem can be restricted to a given class of monoids, in which case its complexity may end up somewhere between these two extremes.

The Membership Problem was first studied in the context of permutation groups (Group Membership), as a part of the already well-established field of computational group theory; its study was particularly motivated by the links between problem on permutation groups and the Graph Isomorphism Problem (see [Hof]). A first result was Sims' algorithm [Sims], which provided a polynomial-time sequential membership test [FHL,Jer]. More recent research has been directed at finding the parallel computational complexity of the Membership Problem, which was shown to be feasible in polylog time on a polynomial number of processors, first for Abelian groups by McKenzie and Cook, who provided an algorithm later completed by Mulmuley [McC,Mul], then for solvable groups [LMK], and recently for the general case of arbitrary permutation groups [BLS].
The purpose of this thesis is to partially fill the void which remained between this successful work on groups and Kozen's result. It deals primarily with aperiodic monoids, and studies how the complexity of the Problem rises when ascending the hierarchy of monoids classes.

The above-mentioned research on groups has not been restricted to the sole Membership Problem: problems such as finding the order of the group, computing generators for certain subgroups (pointwise and setwise stabilizers, intersections), or testing for isomorphism have also been studied [FHL,McC,BLS,Hof]. Similarly, other problems on monoids have been looked at, such as the complexity of computing the monoid element corresponding to a given expression [CFL,BP], and characterizing semiautomata by their transformation monoids (and conversely) [GBB,Stei,Ste2].

The theory of monoids is closely linked to the theories of languages and of automata

[Eil,Lall], so that classes of automata can be defined and described in terms of their transformation monoids or of the type of languages they recognize. Transformation monoids can therefore be related to models of computation and, beyond, to classes of computational complexity. Work is currently being done with the intention of establishing more precise links between these theories, and possibly gain more knowledge on the theory of computational complexity. For instance, recent research has been concerned with classes of non-uniform Boolean circuits, such as NC^1, defined by circuits of polynomial size and logarithmic depth with gates of constant fan-in and unbounded fan-out, and its subclass AC^0, where the circuits have constant depth and the gates unbounded fan-in. The results obtained so far include the proof that the languages belonging to non-uniform NC^1 are exactly those recognized by polynomial-sized branching programs of width five [Bar], similar relationships established between AC^0 and its subclasses and polynomial-sized non-uniform deterministic finite automata on various classes of monoids [BT], and a characterization of non-uniform AC^0 in terms of the complexity of computing the product of a list of elements of a monoid [CFL].

These results actually establish strict inclusions between these complexity classes. A natural extension of this work is therefore to look for a similar kind of characterization for other models of computational complexity and their associated classes, in an attempt to solve open questions such as the well-known NP-problem, or the relationship between NC and FP.

The Membership Problem has advantages which could make it useful for research in this vein: it is defined for all classes of monoids, including groups, and its computational complexity covers a wide range of complexity classes, while not rising as rapidly as that of other problems on monoids, to which the Membership Problem reduces (see chapter Seven).

The data for the Membership Problem include a list of generators for a transformation monoid; verifying whether this monoid belongs to a given class is an important problem by itself [Ste1,Ste2]. In this thesis, it is identified as the Class Test, and will be discussed for most of the classes in which the Membership Problem is considered.

It has been chosen in this thesis to work within those models of computation already used for the study of Group Membership, that is, the usual one-processor sequential model and the synchronous uniform Boolean circuit, introduced by Borodin [Bor]. Given this choice, the approach preferred has been to explore the lattice of classes of monoids, looking for upper bounds or completeness results, with a particular emphasis on those classes where polynomial-time sequential complexities can be obtained. In such cases, both a parallel and a sequential algorithm were sought. The upper bounds thus obtained are not known to be tight. Techniques to find lower bounds are still to be developed; furthermore, in the context of a first look at a yet

unexplored area, it seemed reasonable to leave aside a search for lower bounds, until research done on a wide enough range of monoid classes would allow to decide where a more detailed analysis is worth doing.

Overview of the thesis: The rest of this thesis is divided into six chapters, the three central ones dealing with different classes of monoids. With the exception of more specialized data, all the necessary background and definitions have been gathered in chapter Two. In chapter Three, it is shown that testing membership in arbitrary commutative transformation monoids can be reduced to testing in aperiodic commutative monoids and Abelian permutation groups. The other results of this chapter open the way for the discussion which follows in the rest of the thesis: it is proved that membership testing in monoids of threshold 2 or more is NP-hard, and in NC^3 for commutative monoids of threshold one. Chapter Four explores the direction of idempotent monoids (aperiodic of threshold one); its main results are that the Membership Problem, for such monoids in general, is NP-complete, and that there is a unique largest class of aperiodic monoids for which the Membership Problem is in FL, and another largest class for which it is feasible in polynomial sequential time. The thesis is concluded with comments on other classes of monoids, in which the Membership Problem is intractable, but in some cases possibly not in NP (chapter Five), a discussion of the consequences of considering transformation semigroups instead of monoids (chapter Six), and an application of the methods used in this thesis to other problems in the theory of automata (chapter Seven).

The standard nomenclature for statements ('Theorem', 'Lemma', 'Proposition', 'Corollary') will be reserved for new results, although the reader will observe that some propositions and corollaries have been obtained by merely looking at known facts from a different perspective; the name 'Fact' will be used to designate data taken from the literature, with the reference given, or to results for which no originality is claimed.

II Background

2.1 Monoids and their representations

Most of this section is based on the standard references in the field of semigroup theory [CP,Lall,Eil] and in the closely related theories of finite automata [LP,HU] and of formal languages [Pin]. It is assumed that the reader is familiar with all three fields. More specialized background will be introduced in the other chapters, immediately before it is needed.

A *semigroup* is a set equipped with an associative binary operation. Given a set X, a *transformation* of X is a mapping from X to X. A set of transformations, with the ordinary composition of functions as the binary operation, will be a semigroup whenever it is stable under this operation. In particular, given a set A of transformations of X *(generators)*, the smallest semigroup containing A *(the semigroup generated by* A, denoted $\langle A \rangle$*)*, will coincide with the set of all transformations of X expressible as composition of elements of A.

A *monoid* is a semigroup S containing a neutral element, that is, an element 1 such that $g1 = 1g = g$ for every $g \in S$. The identity transformation 1, which maps every element of X on itself, will be the neutral element in all the monoids considered in this thesis.

Throughout the thesis, the words 'semigroup' and 'monoid' will always be understood to mean 'transformation semigroup' and 'monoid'; whenever abstract monoids are considered, this will be specified explicitly.

Here are some definitions related to a semigroup $\langle A \rangle$ of transformations of X:

- the elements of X shall be called *states;*
- the image of state x by transformation g is denoted xg ;
- for $Y \subseteq X$ and $g \in \langle A \rangle$, define $Yg = \{ yg \mid y \in Y \}$;
- for $x \in X$ and $g \in \langle A \rangle$, define $xg^{-1} = \{ y \in X \mid yg = x \}$;
- similarly, with $Y \subseteq X$, define $Yg^{-1} = \{ x \in X \mid xg \in Y \}$.

All sets considered in this thesis are finite, with the exception of the free semigroup on A, denoted A^+, and the free monoid $A^* = A^+ \cup \{1\}$.

In a finite semigroup, if an element g is composed with itself, then eventually $g^{t+q} = g^t$ for some $t \geq 0$ and $q \geq 1$; the smallest such integers are called the *threshold* and the *period* of g. An element with period $q = 1$ is called *aperiodic,* while a transformation with threshold $t = 0$ is a *permutation.* Furthermore, an element g for which $t = q = 1$, that is, such that $g^2 = g$, is called *idempotent.*

These definitions extend to the semigroup as a whole: a (transformation) semigroup whose elements all have threshold zero is a (permutation) *group;* a semigroup whose

elements all have period one is called *aperiodic,* or *group-free.* A semigroup is said to be of threshold t if the largest threshold its elements can have is t. An idempotent semigroup is therefore defined as being aperiodic of threshold one.

As far as the Membership Problem is concerned, there is no difference between working in a given semigroup S or in the monoid $S \cup \{1\}$: whether the data given represents a semigroup or a monoid can be easily tested.

Fact 2.1: Semigroup $\langle A \rangle$ contains the identity transformation **iff** at least one element of A is a permutation.

Proof: For any permutation g of m states, one has $g^{m!} = 1$. Conversely, permutations are the only transformations for which the sets Xf and X have the same cardinality, so that 1 can be generated by nothing but permutations. □

In the rest of this thesis, with the exception of chapter Six, it will exclusively be question of monoids; given a set A of generating transformations, the notation $\langle A \rangle$ will be understood to represent the union of the semigroup generated by A and of the singleton {1}, this addition being redundant if A contains a permutation. The definitions and facts which follow in this chapter are given in terms of monoids; however, they can be applied to semigroups as well, with but minor modifications in some cases.

Green's relations [Gre] are equivalence relations inside monoids, defined as follows: for every elements f and g of a monoid S, one has

$f \; L \; g$ **iff** there are $u, v \in S$ such that $uf = g$ and $f = vg$;
$f \; R \; g$ **iff** there are $u, v \in S$ such that $fu = g$ and $f = gv$;
$f \; H \; g$ **iff** $f \; R \; g$ and $f \; L \; g$;
$f \; J \; g$ **iff** there are $u, v, u', v' \in S$ such that $ufu' = g$ and $f = vgv'$;
$f \; D \; g$ **iff** there is an $h \in S$ such that $f \; L \; h$ and $h \; R \; g$, or conversely.

Relations D and J coincide in finite monoids; the label J will therefore be used to represent both. Equivalence classes for these relations are called L-, R-, J-, and H-classes. Notice that, in an aperiodic monoid, the H-classes are trivial, that is, they all consist of a unique element [Pin].
Green's relations induce the following conditions on transformations.

Fact 2.2: [CP,Lall] In any monoid of transformations of a finite set X, one has:
$f \; J \; g$ only if $|Xf| = |Xg|$;
$f \; L \; g$ only if $Xf = Xg$.
$f \; R \; g$ only if for all $y \in Xf$, there is an $x \in Xg$ such that $yf^{-1} = xg^{-1}$.

A monoid of transformations of a set X given by a set A of generators can be

represented as a *semiautomaton,* denoted (X,A), which is a directed graph with a vertex for every state of X, and an edge labelled a from vertex x to vertex xa, for every $x \in X$ and $a \in A$. The monoid $<A>$ of transformations of X is the *transformation monoid* of the semiautomaton (X,A). Definitions related to this representation are:
- a *source*: a state is a source for $<A>$ if the corresponding vertex in (X,A) is of indegree zero, disregarding trivial loops;
- a *connected component*, or CC: semiautomaton (X,A) is partitioned into CCs by the following relation of equivalence: two states belong to the same CC iff there is a path between them in the semiautomaton, disregarding the direction of the edges;
- a *strongly connected component*, or SCC: semiautomaton (X,A) is partitioned into SCCs by the following relation of equivalence: two states x and y belong to the same SCC iff there are an f and a g in $<A>$ such that $xf = y$ and $yg = x$;
- a *sub-SCC* of a strongly connected component K of (X,A) is an SCC of the semiautomaton (K,B), where B is a subset of A;
- a *maximal state* (resp. SCC): state x will be maximal for a subset B of A if, for every $a \in B$, $xa = x$ (resp. an SCC K is maximal for B when $Ka \subseteq K$ for all $a \in B$);
- a state of (X,A) is called a *sink* if it is maximal for A;
- a SCC is said to *dominate* the connected component of (X,A) it is included in, if it is the only SCC of this CC to be maximal for A.

In the context of the theory of languages, the generators of A can be seen as *characters,* and an expression can be regarded as a *word,* that is, an element of the free monoid A^*, with the empty word denoted ϵ. More formally, an abstract monoid S, defined as $S = (E, \bullet, 1)$, where E is the set of monoid elements, $\bullet$ the operation, and 1 the identity, is said to be generated by a set A of characters under the mapping $\phi : A \rightarrow E$ if ϕ can be extended to the free monoid A^* in order to have $\phi(a_1 \cdots a_n) = \phi(a_1) \bullet \cdots \bullet \phi(a_n)$ for every word $a_1 \cdots a_n \in A^+$, and $\phi(\epsilon) = 1$, so that ϕ is a surjective homomorphism.
This mapping defines a relation of equivalence on A^* (the kernel of ϕ, denoted $Ker\,\phi$, see [Lall], chapter One, Proposition 4.2): words w_1 and w_2 are equivalent iff $\phi(w_1) = \phi(w_2)$. This relation is a *congruence,* in that it satisfies the property that, if words v_1 and v_2 are equivalent by the relation, then so are uv_1w and uv_2w, for every $u, w \in A^*$.
In the rest of this thesis, given a monoid generated by a set A, the notation ϕ will be reserved for the homomorphism which maps A^* onto the abstract monoid isomorphic to $<A>$, and by extension, onto $<A>$ itself *(canonical homomorphism).*

The *maximal alphabet* of a transformation $f \in <A>$ is defined as the set of all those characters which can appear in an expression of f :

$$\alpha(f) = \{\, a \in A \mid f = gah \text{ for some } g,h \in \langle A \rangle \,\}.$$

This shall not be confused with the *alphabet of a word* $w \in A^*$, denoted $\alpha(w)$, which is the set of those characters which appear in w. An equivalent definition for $\alpha(f)$ is therefore: $\alpha(f) = \bigcup_{f = \phi(w)} \alpha(w)$.

The expression 'class of monoids' used up to now means the set of all those monoids sharing some set of properties. These 'classes' are formally defined as *pseudovarieties,* that is, collections of finite monoids, closed under homomorphism, taking of submonoids, and under finite direct products [Eil,Lall] (a *submonoid* is a subset of a monoid which is itself a monoid). Throughout this thesis, the words 'class' and 'pseudovariety' will be considered synonymous.

A monoid belongs to a given class iff it satisfies to a set of conditions, which define the pseudovariety. These conditions can be given in terms of *defining identities,* a set of equations which all the elements in the monoid must satisfy to. For instance, the class of the aperiodic monoids of threshold t is defined by the two identities $fg = gf$ and $g^{t+1} = g^t$.

Pseudovarieties can also be defined in terms of congruences. A monoid generated by an alphabet A defines a relation of equivalence between the words of A^*, as explained above. A pseudovariety can also be defined by a relation on A^*, which every monoid belonging to it must satisfy: if the class **P** defines relation R on A^*, then the monoid S, defined by the congruence T, will belong to **P** iff $R \subseteq T$. It can happen that $R = T$, in which case S is called the *free monoid on* A *for class* **P**. It is worth noticing that the cardinality of the free monoid coincides with the index (number of equivalence classes) of the congruence R, which is not always finite.

This section is completed by two examples which illustrate the notion of pseudovariety.

Fact 2.3: Let the semiautomaton (X,A), with transformation monoid S, be partitioned into connected components $K_1, \cdots, K_j$, and let S_i be the transformation monoid of (K_i, A). Then S belongs to a given pseudovariety iff each S_i belongs to this same class, so that the Class Test can be performed on each connected component separately.

Proof: Each monoid S_i is generated by the alphabet A, through its individual canonical homomorphism, denoted ϕ_i. Assume that each S_i belongs to the pseudovariety **P**, and consider the direct product $S_1 \times \cdots \times S_j$, whose elements are of the form $(\phi_1(w_1), \cdots, \phi_j(w_j))$. Observe that S is isomorphic to $\{\, (\phi_1(w), \cdots, \phi_j(w)) \mid w \in A^* \,\}$, a submonoid of $S_1 \times \cdots \times S_j$. Therefore, if $S_1, \ldots, S_j$ belong to **P**, then so does S. In the other direction, it suffices to observe that if every element of S satisfies to the defining identities of **P**, then so do the elements of every S_i. □

Example: In the pseudovariety $\mathbf{A}_1$ of all finite idempotent monoids, which defines relation $\approx$ on A^*, the defining identity $g^2 = g$ (idempotency) implies that any word of A^* containing a square, that is, of the shape $uvvw$, will be equivalent to the same word where the square vv has been reduced, that is, to uvw. In every idempotent monoid generated by the alphabet A, the words $uvvw$ and uvw will remain equivalent; this is an example of the inclusion of $\approx$ in the relation R defined by the monoid. The canonical homomorphism ϕ by which the free idempotent monoid generated by alphabet A is obtained from A^* is such that $Ker\,\phi = \approx$; this congruence is of finite index [GrR].

2.2 Complexity Theory

As was mentioned in chapter One, this thesis analyses problems whose computational complexity covers quite a wide range, and is aimed at discovering where these problems are located on the lattice of complexity classes. In what follows, it is expected that the reader is familiar with the theories of sequential complexity and of intractability [AHU1,HU,GJ]. In particular, the reader is referred to [GJ] for definitions of the notions of *NP*-completeness and *NP*-hardness, which will be used in this thesis, and for a comprehensive survey of the open problem concerning the relationship between the complexity classes *FP* and *NP*.

The Membership Problem and the Class Test satisfy to the definition of a ***decision problem,*** which can be characterized by a total function from $\{0,1\}^n$ (the input) onto $\{0,1\}$ (one-bit answer). However, other problems used in this thesis require a description in terms of a ***search problem,*** described by partial, multiple-valued functions from $\{0,1\}^n$ (the input) to $\{0,1\}^m$; a circuit is said to compute this function if it returns one of the possibly many m-bit long solutions in the case there is at least one, and reports the non-existence of solutions otherwise. Classes of sequential complexity can be defined in terms of search problems: for example, the class *FP* is the set of those search problems solvable by a sequential processor in polynomial time. Similar definitions apply for *FL* (search problems feasible in deterministic logarithmic space) and for *NL* (non-deterministic log-space).

For parallel complexity, the model used in this thesis is the Synchronous Boolean Circuit; it is discussed in Cook's survey [Cook], on which the rest of this section is based. A Boolean circuit consists of a directed acyclic graph whose edges are wires and whose nodes are gates with fan-in zero (data gates), one (NOT gates) or two (other Boolean gates). Of the data gates, n of them actually provide the input, while the others give constant values (0 or 1). Fan-out is unrestricted. With m gates labelled output gates,

the circuit will be computing a function from $\{0,1\}^n$ to $\{0,1\}^m$. The *size* of the circuit is the number of gates it contains, and its *depth* is the longest path from an input node to an output node.

The Boolean circuits designed to solve a given problem are defined in terms of *circuit families* $<\alpha_n>$, of which the n^{th} member α_n computes the same function for all inputs of size at most $g(n)$, with g a polynomially-bounded, monotonic increasing function. Circuit families are required to be *uniform*, so that the n^{th} circuit can be constructed in a realistic fashion. The definition of uniformity adopted in this thesis is that of *log-space uniformity*, for which circuit α_n must be constructible by a deterministic Turing machine with value n as an input (in binary notation), and $O(\log n)$ storage space at its disposal.

Complexity classes for parallel computation are defined in the following fashion. The class NC^k is the class of all problems solvable by a (log-space) uniform circuit family $<\alpha_n>$, with $Size(\alpha_n) = n^{O(1)}$ and $Depth(\alpha_n) = O(\log^k n)$. The class NC is the union of all classes NC^k, and is included in FP.
The class NC can be defined in terms of several other models of parallel computation; a parallel algorithm written within the above model can be translated in terms of, for example, Parallel Random Access Machines (with protocols such as Concurrent-Read, Exclusive-Write), at a cost of possibly increasing the exponent of $\log n$ in the depth, but still retaining polynomial-size and polylog-depth.

Proving upper bounds on parallel boolean circuits can be done using *NC^1-reducibility:* a problem f is NC^1-reducible to a set S of problems, denoted $f \leq S$, iff f can be computed by a uniform family of circuits of size $n^{O(1)}$ and depth $O(\log n)$ using oracle gates for problems of S. In counting the depth of this circuit, an oracle node g counts as depth $\lceil \log(r+s) \rceil$, where r and s are the size of the input and output of g.
In all cases considered in this thesis, oracle gates will count for depth $O(\log n)$, and the NC^1 reductions will follow a similar pattern: they will consist in rearranging the data in order to put it in a form acceptable as an input for the oracle gates, an operation done by a log-depth circuit; the output of the oracle will then be taken as such or fed into a circuit for some boolean operation, so that there is no more than one oracle gate along any path from input to output. In the case of an algorithm consisting of several steps, the output from one step is fed into the next step, at the cost of possibly some NC^1 data reorganization, so that the overall depth of the circuit for the whole algorithm remains logarithmic.
Here follow some facts from the theory of NC^1-reducibility.

Fact 2.4: The relation of NC^1 reducibility is transitive. If f is a problem and S a set of problems, if $S \subseteq NC^k$, and $f \leq S$, then $f \in NC^k$. The class NC^k is closed

under $\leq$ for any k.

The class FL^* (resp. NL^*) is the set of all search problems NC^1-reducible to problems in FL (resp. NL).

Fact 2.5: Given below are the classes of computational complexity considered in this thesis. There are problems in FP which are complete for NC^1-reducibility ('inherently sequential' problems [Cook,JL]); there are also problems in NL^* and FL^* which are complete for NC^1-reducibility [Jon].

$$NC^1 \subseteq FL^* \subseteq NL^* \subseteq NC^2 \subseteq NC^3 \subseteq \cdots \subseteq NC \subseteq FP \subseteq NP \subseteq Pspace$$

2.3 Evaluating computational complexities

Whenever a problem is not intractable, one looks for both a parallel and a sequential algorithm to solve it. In only one case (Section 4.6) is the sequential algorithm the only one available. The complexity analysis of these algorithms is based on considerations developed in this section.

Transformations on a set X of m states can be represented as $2 \times m$ matrices, with the first row enumerating the states, and their image given underneath on the second row. If the states are always given in the same order, the first row becomes redundant, as seen in this example:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 1 & 4 & 3 \end{pmatrix} \text{ or } (3\ 4\ 1\ 4\ 3)$$

In either case, this means $O(m \log m)$ bits to represent a transformation in binary notation, and $O(nm \log m)$ bits to encode the input for the Membership Problem, which consists of n generators plus one test-transformation.

The basic operations on these transformations consist of comparing and composing them. In sequential computation, the choice is made to count as taking $O(1)$ steps basic operations such as comparison, data transfer, addition and substraction of indices. Therefore, comparing or composing two transformations takes $O(m)$ time. In parallel, this can be done with m-fold Boolean operations, hence NC^1.

Whenever upper bounds for the complexity of a problem belonging to NC are given, the results in parallel and in sequential are stated together, separated by a slash. Example: NC^1 / $O(mn)$. Parallel complexities do not take into account the size of the circuit, otherwise than specifying that it is polynomial. Sequential complexities

are measured in terms of m and n, the cardinalities of the sets X and A.

Fact 2.6: The algorithms discussed in this thesis take the following problems as building blocks:

-taking the product of n transformations of m states: $NC^2/O(mn)$ [McK]; more precisely, this belongs to FL;

- accessibility problem in a directed graph with m vertices and e edges: $NC^2/O(\max\{m,e\})$ [Bor,Jon,AHU2];

- decomposition of a directed graph with m vertices and e edges into connected or strongly connected components: $NC^2/O(\max\{m,e\})$ [Bor,Jon,AHU2];

- membership testing in an Abelian group of permutations of m states, given by n generators: $NC^3 / O(SLC(mn))$ [McC,Mul]. This is the complexity of solving a system of linear congruences modulo an integer, as explained in the appendix.

Example: Some of the problems encountered in this thesis reduce to testing the inclusion of a regular language in another, with the following restrictions: the languages are accepted by finite automata constructed on the same semiautomaton (X,A), with the same set of final states, and different initial states. In this example, a routine to perform this test is presented.

For $x \in X$ and $F \subseteq X$, define the set $L_x = \{ w \in A^* \mid x\phi(w) \in F \}$. This is a regular language, accepted by the deterministic finite automaton $M = (X, A, x, \delta, F)$, where x is the initial state, F the set of final states, and the transition function $\delta \subseteq X \times A \times X$ is given by $\delta(p,a) = q$ iff $p\phi(a) = q$. The language L_z is defined similarly.

To decide whether $L_x \subseteq L_z$, the equivalent condition $L_x \cap \overline{L_z} = \varnothing$ will be tested, where $\overline{L_z} = A^* - L_z$ is the complement of L_z, and is accepted by the automaton $M' = (X, A, z, \delta, X-F)$.

By a classical construction ([LP], exercise 2.4.3), the intersection of $L_x \cap \overline{L_z}$ is accepted by an automaton M, with set of states $X \times X$, initial state (x,z), and $F \times (X-F)$ as a set of final states. Its transition function Δ is given by

$$\Delta((p,r),a) = (q,s) \text{ iff } \delta(p,a) = q \text{ and } \delta(r,a) = s.$$

The language accepted by M will be empty iff none of its final states is accessible from the initial state (x,z). The test for inclusion is therefore reduced to a set of accessibility tests in a deterministic finite automaton, which can be regarded, for this purpose, as a directed graph with labelled edges.

Parallel algorithm

1. Build M.
2. For every state (p,q) of M, $p \in F$, $q \notin F$, in parallel, test whether (p,q) is

accessible from (x,z). If not, let $\gamma(p,q) = True$, else $False$.

3. Let $\beta(x,z)$ be the AND of all $\gamma(p,q)$. One has $L_x \subseteq L_z$ iff $\beta(x,z) = True$.

Analysis: Let (X,A) have m states and n generators. The automaton **M** is built in a straightforward fashion (depth $O(1)$). Step 2 consists in performing $m-1$ accessibility tests (NC^2) in parallel, the results of which are fed into an $O(m^2)$-fold AND operation (NC^1) at Step 3. Therefore, testing whether $L_x \subseteq L_z$ is NC^1-reduced to problems belonging to NC^2.

Sequential algorithm

1. Build M.
2. If there are $p \in F$ and $q \notin F$ for which (p,q) can be reached from (x,z) in the semiautomaton, then $L_x \not\subseteq L_z$.

Analysis: Step 1 is $O(m^2n)$, while Step 2 can be executed by doing a depth-first search from the state (x,z), which is linear in the number of edges in the graph formed by the semiautomaton, that is, $O(m^2n)$.

Fact 2.7: Let languages L_x and L_z be defined as above.
The inclusion of L_x in L_z can be tested in NC^2 / $O(m^2n)$. □

2.4 Dual classes

An abstract monoid has been defined as a set equipped with an associative binary operation and containing a neutral element for this operation: $S = (E,\bullet,1)$. The *dual* of S is the monoid $S^R = (E,*,1)$, where $*$ is defined by $r*s = s \bullet r$ for every $r,s \in E$. Let A be a set of generators of S, and ϕ the canonical homomorphism which maps the free monoid A^* onto S. Then S^R can also be obtained from A^*. Let the mapping ψ be defined as follows: $\psi(e) = 1$, $\psi(a) = \phi(a)$ for every $a \in A$, and

$$\psi(a_1 \cdots a_n) = \psi(a_1) * \cdots * \psi(a_n)$$

for every word $a_1 \cdots a_n \in A^*$. It can be verified that this implies

$$\psi(a_1 \cdots a_n) = \phi(a_n) \bullet \cdots \bullet \phi(a_1) = \phi(a_n \cdots a_1)$$

Fact 2.8: Mapping ψ is surjective, and the image of A^* by ψ is the monoid S^R. □

For $s \in E$, let $\phi^{-1}(s) = \{w \in A^* \mid \phi(w) = s\}$, and $\psi^{-1}(s) = \{w \in A^* \mid \psi(w) = s\}$; as a consequence of the definition of ψ, one has $\psi^{-1}(s) = \{w \in A^* \mid \phi(w^R) = s\}$.

Denote R the kernel of ϕ, and R^R the kernel of ψ. For all $u, v \in A^*$, $u\ R\ v$ iff $\phi(u) = \phi(v)$. By the above, this is equivalent to having $\psi(u^R) = \psi(v^R)$, that is, $u^R\ R^R\ v^R$. Equivalently, $u\ R^R\ v$ iff $u^R\ R\ v^R$.

The definition of dual monoids can be extended to pseudovarieties: if class P is defined by relation T, let $\mathbf{P}^R$ be defined by T^R, such that $u\ T^R\ v$ iff $u^R\ T\ v^R$: it can be verified that $\mathbf{P}^R = \{ S^R \mid S \in \mathbf{P} \}$. Notice that a class or a monoid can be its own dual. For instance, a commutative monoid defines a relation R on A^* which does not take into account the order of characters in a word, so that $w\ R\ w^R$ for every $w \in A^*$.

A way of representing the dual of a transformation monoid is to use inverse transformations. Let (X,A) be a semiautomaton, and $S = \langle A \rangle$ its transformation monoid; denote 2^X the power set of X, and for $f \in \langle A \rangle$, define f^{-1} as in section 2.1. Then $S^{-1} = \langle A^{-1} \rangle = \langle \{a^{-1} \mid a \in A\} \rangle$ is the transformation monoid of $(2^X, A^{-1})$.

Fact 2.9: Monoids S and S^{-1} are dual. □

It is tempting to start from (X,A) and work with its dual $(2^X, A^{-1})$. This, however, means an exponential increase in the size of the semiautomaton, so that some argument must be used which allows to restrict the work on a polynomial-sized subgraph of $(2^X, A^{-1})$. Fact 2.10 and Theorem 2.11 are an example of this.

Fact 2.10: Let (X,A) be partitioned into j connected components: $K_1, \cdots, K_j$. Then the transformation monoid of $(2^X, A^{-1})$ is isomorphic to the monoid of (Y, A^{-1}), where $Y = \bigcup_{1 \le i \le j} 2^{K_i}$.

Proof: It will suffice to prove that $f^{-1} \neq g^{-1}$ in $(2^X, A^{-1})$ iff $f^{-1} \neq g^{-1}$ in (Y, A^{-1}). The *(if)* part is trivial, since $Y \subseteq 2^X$. For the *(only if)* part, notice that $f^{-1} \neq g^{-1}$ in $(2^X, A^{-1})$ iff there is an $E \subseteq X$ such that $Ef^{-1} \neq Eg^{-1}$. This implies the existence of an $x \in E$ for which there is a $y \in X$ such that $yf = x$ and $yg \neq x$. Since the states x, y and yg must belong to the same connected component K_i of (X,A), one gets $(E \cap K_i)f^{-1} \neq (E \cap K_i)g^{-1}$. □

In the case where every connected component has a number of states bounded by some constant k, an instance of the Membership Problem in (X,A) can therefore be transformed into a polynomial-sized instance in its dual monoid, by building semiautomaton (Y, A^{-1}), which has $O(|X|)$ states, and taking f^{-1} as the test-transformation. This reduction can be done in parallel in a straightforward fashion.

Theorem 2.11: If every connected component of (X,A) contains at most k states, with k a constant, then an instance of the Membership Problem in (X,A) can be NC^1-reduced to an instance in (Y,A^{-1}). □

Most of the proofs of 'completeness' and 'hardness' appearing in this thesis consist in building an instance of the Membership Problem in a semiautomaton which satisfies to the conditions of this theorem; in such cases, the result obtained for the appropriate class can be directly transferred to its dual. It is worth noticing that no similar method exists yet, which would allow to transfer upper bounds between dual classes.

2.5 Pseudovarieties discussed in the thesis

In order to give the reader a global overview of the pseudovarieties treated in the next three chapters, their definitions are given here, more or less in the same order as the one in which they will later be discussed. Their inclusion relationships are represented in Figures 4.1 and 5.1.

$\mathbf{J_1}$: Commutative and idempotent monoids.
Defining equations: $fg = gf$ and $g^2 = g$. Congruence: $u \equiv v$ iff $\alpha(u) = \alpha(v)$.
Self-dual.

AC: all aperiodic and commutative monoids.
Defining equations: $fg = gf$ and $g^{t+1} = g^t$ for some $t \geq 0$.
Subclass: $\mathbf{J_1}$. Self-dual.

$\mathbf{R_1}$: idempotent R-trivial monoids.
Defining equations: $fgf = fg$ and $g^2 = g$.
Subclass: $\mathbf{J_1}$. Dual class: $\mathbf{L_1}$.

$\mathbf{M_3}$: a class of idempotent monoids.
Defining equations: $fghf = fgfhf$ [Fen] and $g^2 = g$.
Subclasses: $\mathbf{R_1}$ and $\mathbf{L_1}$. Self-dual.

$\mathbf{N_4}$: a class of idempotent monoids.
Defining equations: $fghkhgf = fghfhkhfhgf$ [Fen] and $g^2 = g$. Congruence: $\approx_4$ (see section 4.5).
Subclass: $\mathbf{M_3}$. Self-dual.

XR: a class of idempotent monoids.
Defining equations: $fgh = fghfhgh$ [Fen] and $g^2 = g$.
Subclass: $\mathbf{N}_4$. Not self-dual.

$\mathbf{A}_1$: all idempotent monoids.
Defining equation: $g^2 = g$. Congruence: $\approx$ (see Fact 4.15).
Subclasses: $\mathbf{N}_4$, **XR**. Self-dual.

J: J-trivial monoids.
Defining equations: $f^{n+1} = f^n$ and $(fg)^n = (gf)^n$ for some $n \geq 0$.
Subclass: **AC**. Self-dual.

R: R-trivial monoids.
Defining equation: $(fg)^n f = (fg)^n$ for some $n \geq 0$ [Fich].
Subclasses: **J** and $\mathbf{R}_1$. Dual class: **L**.

A: all aperiodic monoids.
Defining equation: $f^{t+1} = f^t$ for some $t \geq 0$.
Subclasses: **R**, **L**, $\mathbf{A}_1$. Self-dual.

$\mathbf{Th}_1$: all monoids of threshold one.
Defining equation: $f^r = f$ for some $r \geq 0$.
Subclass: $\mathbf{A}_1$. Self-dual.

III Commutative monoids

The class of commutative transformation monoids was the first to be considered [Be], as it was rightfully expected that it would be the easiest to deal with, and that its study would provide information to set the orientation for further work. Indeed, the main result of this chapter, the *NP*-completeness result of Theorem 3.10, gave the demonstration of how rapidly the Membership Problem could become computationally hard when one left the realm of groups, as well as a first case for which the complexity of this problem lies between *NC* and *PSpace*.

Identifying a monoid as commutative is easy, as well as finding its threshold and period, as shown by the following.

Fact 3.1: Let A be a set of transformations on X. The elements of A commute with one another iff $\langle A \rangle$ is commutative. Further, if element a_i of A has threshold t_i and period p_i, and if $\langle A \rangle$ is commutative, then every element of $\langle A \rangle$ has a threshold $t \leq \max(t_i)$ and a period p which divides $\mathrm{lcm}(p_i)$. □

It can be shown directly that the Membership Problem in commutative monoids belongs to *NP*. The argument goes as follows. Guess an expression for the test-transformation f: by commutativity, all the occurences of a generator can be grouped together, so that an expression can be given as a list of the number of occurrences (exponent) of each generator. Although these numbers can be exponential, bounded above by m^m, each can be represented with a polynomial number of bits. Then, for every generator g with exponent a, compute g^a, by first computing g^2, g^4, g^8 ..., and then taking the appropriate product of these powers of g, a method which takes a polynomial amount of time. Finally, compute the transformation corresponding to the whole expression, and compare it with f.

3.1 A general algorithm for membership testing

In this section, an algorithm is built, which decomposes an instance of the Problem in an arbitrary commutative monoid into a test in an aperiodic transformation monoid and a test in an Abelian group of permutations. This algorithm is based on simple observations on the structure of semiautomata having commutative transformation monoids. These observations have to do with the SCCs of semiautomaton (X,A), and are valid for any transformation which commutes with the monoid.

Let S be the transformation monoid of semiautomaton (X,A). For $x \in X$, denote $\bar{x}$

the SCC containing x, and let $\overline{X}$ be the set of all SCCs of (X,A). Next, let transformation g commute with S, and define $\overline{x}g = \{ yg \mid y \in \overline{x} \}$.

Proposition 3.2: In a commutative monoid S, $\overline{xg} \subseteq \overline{x}\,\overline{g}$. Therefore, for any g which commutes with S, a well-defined transformation $\overline{g}$ on $\overline{X}$ can be built, such that $\overline{x}\ \overline{g} = \overline{xg}$; the function which maps g to $\overline{g}$ can also be defined; it is a homomorphism.

Proof: In order to prove the first statement, let states x and y belong to the same SCC, with $xa = y$ and $yb = x$, for some $a, b \in S$. Then, for any g which commutes with S, one has $xg = (xab)g = (xa)gb = ygb$, so that $xg = ygb$. Similarly, $yg = xga$. To show that the mapping from g to $\overline{g}$ is a homomorphism, let g and h be two transformations commuting with S. For any x, one has $\overline{x}(\overline{g}\ \overline{h}) = (\overline{x}\ \overline{g})\overline{h} = \overline{xg}\ \overline{h}$ by definition of $\overline{g}$; since $\overline{xg}$ can be regarded as the SCC of state xg, one gets $\overline{xg}\ \overline{h} = \overline{(xg)h} = \overline{x(gh)}$, which is $\overline{x}\ \overline{gh}$, by definition of $\overline{gh}$ □

Fact 3.3: If a transformation monoid is commutative, then it will be aperiodic **iff** its SCCs are trivial, that is, they all consist of a single state.

This fact is a restriction of a result by I.Simon [Simon] to the commutative monoids, which is stated in full as Fact 5.1. For the purpose of this discussion, a proof for the commutative case is given here.

Proof: The *(if)* part is trivial. For the other part, suppose that S is aperiodic and has a non-trivial SCC containing states $x \neq y$, so that there exist $a, b \in S$ such that $xa = y$ and $yb = x$. Notice that aperiodicity implies the existence of a smallest k for which $xa^{k+1} = xa^k \neq xa^{k-1}$. If $k = 0$, so that $xa^1 = xa^0 = x$, then $x = y$; therefore, $k \geq 1$. Next, observe that $(xa^k)ab = (xa^{k+1})b = (xa^k)b = (xa^{k-1})ab$. Using commutativity and $xab = x$, this gives $xa^k = xa^{k-1}$, a contradiction with the hypothesis on k. □

Corollary 3.4: The set $\overline{S} = \{ \overline{g} \mid g \in S \}$ is an aperiodic commutative monoid, generated by $\overline{A} = \{ \overline{a} \mid a \in A \}$. □

An aperiodic monoid $\overline{S}$ is therefore obtained out of S, by collapsing the SCCs of X into states of $\overline{X}$. Call $\overline{S}$ the aperiodic part of S, and it will soon appear how appropriate this name is. Now look inside the SCCs of S. Given an SCC $\overline{x}$, say that a transformation $g \in S$ *stabilizes* $\overline{x}$ if $\overline{x}\ \overline{g} = \overline{x}$, and call the stabilizer of $\overline{x}$ the set $S(\overline{x})$ of all such transformations.

Proposition 3.5: The restriction of $S(\bar{x})$ to $\bar{x}$ is a transitive Abelian group generated by the restriction of $S(\bar{x}) \cap A$ to $\bar{x}$.

Proof: $S(\bar{x})$ is obviously a set closed under composition. Further, let y and z be elements of $\bar{x}$, with $g, h, a, b \in S(\bar{x})$ such that $y = xg$, $x = yh$, $z = xa$, $x = zb$. One has $zgh = xagh = xgha = xa = z = zhg$, hence gh acts as the identity on $\bar{x}$, and therefore is a neutral element in $S(\bar{x})$, and every transformation g in $S(\bar{x})$ has an inverse h. A permutation group is transitive **iff** its semiautomaton consists of a unique connected component [Wie], which is the case here.
The second part of the proposition is shown by contradiction. Let a transformation g in $S(\bar{x})$ be expressed as

$$g = a_1^{\lambda_1} \cdots a_i^{\lambda_i} b_1^{\mu_1} \cdots b_j^{\mu_j}$$

so that the generators labelled a are in $S(\bar{x})$ and those labelled b aren't. Let $h = b_1^{\mu_1 - 1} \cdots b_j^{\mu_j}$. Making the a_i's act first, one gets $\bar{x}\ \bar{g} = \bar{x}\ \bar{b}_1 \bar{h}$. But $\bar{x}\ \bar{b}_1 \neq \bar{x}$ while $\bar{x}\ \bar{g} = \bar{x}$, which means a non-trivial SCC in $\bar{S}$, hence a contradiction with Corollary 3.4. □
As a result, if g stabilizes $\bar{x}$, then g acts as a permutation on $\bar{x}$, and

Corollary 3.6: For any $g \in S$, the thresholds of g and $\bar{g}$ are equal. □

Notice that, as a consequence of Fact 3.3, an aperiodic commutative transformation monoid over a finite set of states has at least one source; if the monoid is non-aperiodic, it has at least one source-SCC (source of $\bar{S}$). The next proposition can be shown by simple use of commutativity, and implies that membership testing can be restricted to a subset of X, the sources of the monoid.

Fact 3.7: Let S be commutative and let f and g commute with S. If every state in every source-SCC has the same image by f and by g, then $f = g$. □

For the remainder of this chapter, f will denote the transformation on which the membership test is applied. Let Y be the union of all source-SCCs. Observe that the restriction of $\bigcap_{x \in Yf} S(\bar{x})$ on $\bar{Y}\bar{f}$, that is, the restriction on $\bar{Y}\,\bar{f}$ of the set of all those transformations $g \in S$ such that $\bar{x}\ \bar{g} = \bar{x}$ for every SCC $\bar{x}$ in $\bar{Y}\bar{f}$, is a group generated by those $a_i \in A$ which individually stabilize every SCC in $\bar{Y}\bar{f}$.
For any transformation g which commutes with S, let $\hat{g}$ be a transformation of X defined as follows: $x\hat{g} = xg$ if $\bar{x} \subseteq \bar{Y}\bar{f}$ and g stabilizes every SCC of $\bar{Y}\bar{f}$, and $x\hat{g} = x$ otherwise. Further, define $\hat{S} = \{ \hat{g} \mid g \in \bigcap_{x \in Yf} S(\bar{x}) \}$. This set is a group,

isomorphic to $\bigcap_{x \in Yf} S(\bar{x})$, and generated by the set $\hat{A} = \{ \hat{a} \mid a \in A \}$. It can now be shown how the test for membership is split into a test in $\bar{S}$ and a test in $\hat{S}$.

Lemma 3.8: Let f commute with commutative transformation monoid S. Then $f \in S$ iff there exist $g_a \in S$ and $\hat{g}_c \in \hat{S}$ such that $\bar{f} = \bar{g}_a$ and $xf = xg_a\hat{g}_c$ for every state $x \in Y$..

Proof: (only if) Trivial: take $g_a = f$ and $\hat{g}_c = 1$.

(if) (Notice that, in general, $\hat{g}_c$ will not commute with g_a). For any $x \in Y$, one has $\bar{x}\bar{f} = \bar{x}\,\bar{g}_a$, and $(\bar{x}\bar{f})\bar{g}_c = \bar{x}\bar{f}$, since g_c has been taken among those transformations of S which stabilize every SCC of $\bar{Y}\bar{f}$. Hence $xg_a g_c = xg_a\hat{g}_c = xf$. This is true for all states in the source-SCCs, and can be extended to every state of X by Fact 3.7. And since g_a and g_c are elements of S, it follows that $f \in S$. □

This lemma gives the proof of correctness for the following algorithm, which tests membership in arbitrary commutative monoids of transformations.

Algorithm

0. Test whether $fa = af$ for every generator $a \in A$. If not so, $f \notin S$.
1. Build $\bar{X}$, Y, $\bar{A}$ and $\hat{A}$; the latter two sets generate the monoids $\bar{S}$ and $\hat{S}$.
2. Test whether $\bar{f} \in \bar{S}$. If so, find an expression for $\bar{f}$, that is, build $g_a \in S$ such that $\bar{f} = \bar{g}_a$. Else, $f \notin S$.
3. Build $\hat{g}_c$ so that for every x in Y, $(xg_a)\hat{g}_c = xf$.
4. Test whether $\hat{g}_c \in \hat{S}$. If so, then $f \in S$.

Analysis and remarks: Step 0 belongs to NC^1 / $O(mn)$. Step 1, through the construction of the SCCs, is NC^2 / $O(mn)$. Step 2 is an instance of the Membership Problem in aperiodic commutative monoids; the complexity of this step is the topic of the next sections. Step 3 has built-in the condition $f = g_a\hat{g}_c$. Further, since the restriction of $\hat{S}$ to any SCC of $\bar{Y}\bar{f}$ is a transitive Abelian group, it suffices to define the action of $\hat{g}_c$ on but one state of the SCC, such as some xg_a, $x \in Y$, to totally determine its action over the whole SCC ([Wie], chapter One, Proposition 4.4). This step is NC^1 / $O(m)$. Step 4 is an instance of the Membership Problem in an Abelian permutation group: NC^3 / $O(SLC(mn))$. If this test fails, f does not belong to S.

Some facts can be observed from this algorithm. For instance, since aperiodic commutative monoids have the property that every J-class consists of a unique element (it suffices to verify that, by commutativity, $f\ J\ g$ implies $f\ H\ g$, which means $f = g$ in an aperiodic monoid), and since the transformation $\hat{g}_c$ of the decomposition $f = g_a\hat{g}_c$ acts as a permutation on the SCCs in which Xf is included, it follows

that $f\ R\ g_e$, hence $f\ J\ g_e$, so that Step 2 of the algorithm consists in finding the J-class of f. Furthermore, it can be seen that the *NP*-completeness of the Membership Problem in commutative monoids resides entirely in the aperiodic part (Step 2).

3.2 Membership in aperiodic commutative monoids

In this section, the work is restricted to aperiodic monoids. The complexity of the Membership Problem in this case is shown to depend on the threshold of the monoid, that is, the largest threshold a transformation of this monoid can have. In the case of monoids of threshold 2 or more, it is related to the problem of solving a system of linear equations under a threshold, which recalls the NC^1-equivalence established by McKenzie and Cook [McC] between testing membership in Abelian permutation groups and solving systems of linear equations over the ring of integers modulo q.

Lemma 3.9: Testing membership in an aperiodic commutative monoid of threshold two is *NP*-complete.

Theorem 3.10: Testing membership in arbitrary commutative transformation monoids is *NP*-complete.

Proof: By the argument which followed Fact 3.1, the Problem belongs to *NP*. This reasoning can actually be simplified, since every element of an aperiodic commutative monoid has at least one expression of linear length, due to the fact that no transformation of m states can have a threshold greater than $m-1$, so that no more than that number of instances of a generator in an expression is non-redundant.

The rest of the proof is done by reducing to the Membership Problem a variant of the zero-one Integer Programming Problem [GJ], with only equalities involved.
The reduction consists in building a transformation monoid out of the following system, where all a_j's, b_i's and c_{ij}'s are either 0 or 1:

$$\text{(I)} \quad \begin{cases} c_{11}a_1 + \cdots + c_{1n}a_n = b_1 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ c_{m1}a_1 + \cdots + c_{mn}a_n = b_m \end{cases}$$

The construction is as follows:
-For each line i, $1 \leq i \leq m$, define states x_{i0}, x_{i1}, x_{i2}. The set X contains $3m$ states.
-For each column j, $1 \leq j \leq n$, define a generator g_j, whose action is $x_{ik}\, g_j = x_{il}$, with $l = k + c_{ij}$ for $k = 0,1$ and $l = 2$ for $k = 2$, for all i. Generator g_j is therefore

either the identity transformation, in the trivial case where all the c_{ij}'s are zero, or aperiodic of threshold 2. Let $A = \{ g_1, \ldots, g_n \}$.
-Finally, define the test-transformation f, such that $x_{ik} f = x_{il}$ with $l = 2$ for $k = 2$, and $l = k + b_i$ for $k = 0,1$. This transformation is aperiodic of threshold 2.

Proposition 3.11: The monoid $\langle A \rangle$ is commutative, aperiodic of threshold 2, with sources $x_{10}, \ldots, x_{m0}$. □

Lemma 3.12: System (I) has a solution **iff** test-transformation f belongs to $\langle A \rangle$.

Proof: Without loss of generality, assume that $b_i = 1$ for all i.
(only if) Let the system (I) have a solution $a_1, \ldots, a_n$. Construct the product $g = g_1^{a_1} \cdots g_n^{a_n}$; its action on the source x_{i0} will be $x_{i0} g = x_{il}$, where $l = 1$ if $a_1 c_{i1} + \ldots + a_n c_{in} = 1$, and $l = 2$ if $a_1 c_{i1} + \ldots + a_n c_{in} \geq 2$. By the hypothesis, $a_1 c_{i1} + \ldots + a_n c_{in} = b_i$, with $b_i = 1$, so that $l = 1$ for all i, which means $x_{i0} g = x_{i0} f$. And since f and g have the same action on the sources of $\langle A \rangle$, by Fact 3.7 they are equal.
(if) That $f \in \langle A \rangle$ implies the existence of an expression for f of the form $f = g_1^{a_1} \cdots g_n^{a_n}$; this means that for every source x_{i0}, one has $x_{i0} f = x_{il}$ with $l = b_i$, and $a_1 c_{i1} + \ldots + a_n c_{in} = b_i = 1$. Source x_{i0} defines therefore a condition on the coefficients a_j, $1 \leq j \leq n$, which corresponds to the i^{th} line of system (I). □

Lemma 3.9 generalizes to all pseudovarieties of aperiodic commutative monoids of threshold $t \geq 2$. This lemma has consequences on arbitrary aperiodic monoids as well, since every pseudovariety of monoids of threshold t contains the class of commutative aperiodic monoids of threshold t.

Corollary 3.13: The Membership Problem in transformation monoids of threshold 2 or more is *NP*-hard. □

3.3 Commutative monoids of threshold one

The above discussion leaves open the threshold-1 case, which is now shown to belong to *NC* by use of the following algorithm, which tests membership in a commutative monoid, aperiodic of threshold one (idempotent).

Algorithm

1. Compute the maximal alphabet $a(f)$;
   generator a belongs to $a(f)$ iff $af = fa = f$;
2. compute the product g of all generators of $a(f)$, in any order;
3. $f \in \langle A \rangle$ iff $f = g$.

The algorithm can be seen to be valid as follows. First, notice that $a \in a(f)$ implies that $af = f$, since commutativity allows to bring the extra a at the left of af next to the nearest occurence of a in the expression of f; it can then be eliminated by idempotency ($a^2 = a$). The converse is trivial, and so is the rest of the algorithm. In sequential, this can be done in $O(mn)$; in parallel, Steps 1 and 3 are NC^1, while Step 2 is in FL (NC^2).

Proposition 3.14: The Membership Problem in idempotent commutative transformation monoids has complexity NC^2 / $O(mn)$. □

As a consequence, the complexity of the Membership Problem in arbitrary commutative monoids of threshold one is dominated by Step 4, that is, membership in Abelian permutation groups.

Theorem 3.15: The Membership Problem in commutative transformation monoids of threshold one has complexity NC^3 / $O(SLC(mn))$. □

This result opens the possibility of finding other types of aperiodic monoids for which the Membership Problem is not NP-hard, namely cases where the constraint of having threshold one is retained, while the monoid no longer has to be commutative. This is the subject of the next chapter.

3.4 Restrictions on the monoid

Finally, restrictions of another kind are mentioned. The first one consists in setting a constant upper bound on the number of sources in the aperiodic part of S. For example, let a commutative aperiodic monoid S have but one source. Since every state of X can be reached from it, using a transformation of S, it suffices by Fact 3.7 that a transformation commute with S and map this source to a state of X to make it belong to the monoid. This implies that the only transformations of X that can commute with such a monoid are its own elements.
This reasoning can be extended to the case where S is aperiodic and has k sources, k

a constant. Build the Cartesian product X^k and let transformation g act on X^k as

$$(y_1, \ldots, y_k)g = (y_1 g, \ldots, y_k g).$$

This brings the problem down to the one-source case, by testing whether $(x_1, \ldots, x_k)f$ can be reached from $(x_1, \ldots, x_k)$, with $|X|^k$ states in total. The sequential complexity of the Membership Problem is therefore dominated by the accessibility test in (X^k, A), which is $O(m^k n)$, and by the test for Abelian Group Membership, which is $O(SLC(mn))$. In parallel, this is an instance of the Accessibility Problem in a directed graph, which lies in NC^2, so that the complexity in the general case is dominated by Step 4 of the general algorithm of section 3.1 (NC^3).

Theorem 3.16: Testing membership in a commutative monoid whose aperiodic part has a number of sources bounded by a constant k can be done in $NC^3/O(\max\{m^k n, SLC(mn)\})$. □

Another restriction worth mentioning is derived from the observation that every element of an aperiodic commutative monoid has an expression of length at most $n(m-1)$ (see the proof of Lemma 3.9). There are m^n such expressions, taking into account that commutativity allows to specify only the number of occurences for every generator. If there is a constant upper bound on n, then all these expressions can be tried by a brute-force search, leading to a polynomial-time sequential complexity, dominated by the search, which is $O(m^{k+1})$, and by Abelian Group Membership. In parallel, the test in the aperiodic part consists in first generating simultaneously all the m^n possible expressions, and then, for each expression, computing the corresponding transformation and comparing it with f. All three steps are NC^1, so that the complexity is dominated by step 4 of the general algorithm (NC^3).

Theorem 3.17: Testing membership in a commutative transformation monoid with a number of generators bounded by a constant k can be done in $NC^3/O(\max\{m^{k+1}, SLC(m)\})$. □

Notice that the first restriction corresponds to an instance (I) where the number of lines in the system is bounded by a constant; the second means a constant upper bound on the number of columns.

It is also worth observing that a result analogous to Theorem 3.17 can be proved in any pseudovariety which defines a congruence of finite index. Indeed, in this case, the number of equivalence classes, which is the cardinality of the monoid, depends only on the number of generators. If the latter is bounded by a constant, then so is the former.

IV Idempotent monoids

As was proved in the previous chapter, the Membership Problem is intractable for all monoids of threshold 2 or more, excluding special restrictions such as the ones mentioned in section 3.4. An obvious action to take is to stay with threshold one, and to relax instead the condition that the monoid be commutative, in order to look for possibilities of other pseudovarieties where membership could be tested in FP or NC. This chapter reports therefore on the research done on idempotent monoids, that is, aperiodic monoids of threshold one, which can be described as satisfying to the defining identity $g^2 = g$.

The chapter is divided into two main parts. In sections 4.1, 4.2 and 4.3, results are described which hold for any idempotent monoid, namely an NC^2 test for idempotency, an NC^1 test for J-class, and the main result of this chapter, the fact that the Membership Problem is NP-complete in idempotent monoids. The remaining sections can be seen as a more detailed look at the lattice of pseudovarieties of idempotent monoids, whose structure is known precisely [Ger,Fen], with proofs that, in some of them, the Membership Problem is not intractable, as well as an attempt to understand how the Problem evolves from feasible in NC^1, in the case of the trivial monoid, up to NP-complete.

Remark: As was mentioned in chapter Two, the generators of the monoid can be regarded as characters (elements of A), and expressions as words (elements of A^*); there is also a canonical homomorphism ϕ which maps A^* onto $<A>$. In order to facilitate reading, the distinction between an expression (word w of A^*) and its corresponding transformation (element $\phi(w)$ of $<A>$) will be made by a systematic use of the canonical homomorphism, whenever the word consists of more than one character.

4.1 Testing for idempotency

Quite remarkably, idempotency can be tested with little knowledge of the inner structure and properties of idempotent monoids. The algorithm takes the definition of idempotency and translates it into a question on regular languages.

Theorem 4.1: Testing whether the transformation monoid of a semiautomaton is idempotent can be done in $NC^2/O(m^4n)$.

Let x and y be two states of X. Define $L_{xy} = \{ w \in A^* \mid x\phi(w) = y \}$, where obvi-

ously $L_{xy} = \varnothing$ if there is no $g \in \langle A \rangle$ such that $xg = y$.

Lemma 4.2: Monoid $S = \langle A \rangle$ is idempotent **iff** $L_{xy} \subseteq L_{yy}$ for every $x, y \in X$.

Proof: The *(only if)* part is trivial. For the *(if)* part, let $w \in A^*$. For any $x, y \in X$, one has $w \in L_{xy}$ iff $x\phi(w) = y$. That $L_{xy} \subseteq L_{yy}$ implies $y\phi(w) = y$, and thus $x\phi(w)\phi(w) = y = x\phi(w)$; this has to be true for every x, y and w. Hence $(\phi(w))^2 = \phi(w)$ for all $\phi(w)$, that is, for all elements of the monoid. □

This lemma gives the proof of validity of the following algorithm.

Algorithm

1. For every pair $\{ x, y \}$, test whether $L_{xy} \subseteq L_{yy}$ and $L_{yx} \subseteq L_{xx}$.
2. $\langle A \rangle$ is idempotent iff the above tests are successful for every pair $\{ x, y \}$.

Analysis: Let (X,A) have m states and n generators. Step 1 consists in $O(m^2)$ calls to the routine presented in Fact 2.7, which tests inclusion of regular languages, and has complexity NC^2 / $O(m^2 n)$. In sequential, the loop on all pairs increases the complexity of this step to $O(m^4 n)$, while Step 2 consists merely in testing the value of the answer-bit, whose value has been updated at each iteration of the loop of Step 1. In parallel, Step 2 consists in an $O(m^2)$-fold AND operation, which is NC^1, so that the problem as a whole belongs to NC^2. □

4.2 The Membership Problem belongs to *NP*

To set the context for the discussion which follows in the next sections, the main result of this chapter is stated here. The reduction for the proof of *NP*-completeness requires a more detailed knowledge of idempotent monoids, and will be done in a later section (Theorem 4.25). However, the proof that the Problem belongs to *NP* can be given now; it consists in demonstrating the existence of a polynomial upper bound on the length of an expression, for any element of an idempotent monoid.

Theorem 4.3: The Membership Problem in arbitrary idempotent transformation monoids is *NP*-complete.

Lemma 4.4: Every element of a monoid of idempotent transformations over m states has at least one expression of length at most $m-1$.

The proof of this lemma requires the following result.

Proposition 4.5: Let the transformation monoid of (X,A) be idempotent. For any $f, g, h, k \in \langle A \rangle$, with $f = ghk$ and $y \in Xf$, one has $(yf^{-1})gh \subseteq yf^{-1} \cap y(hk)^{-1}$.

Proof: The inclusion of $(yf^{-1})gh$ in yf^{-1} is demonstrated by observing that

$$(yf^{-1})ghf = (yf^{-1})gh(ghk) = (yf^{-1})ghk = (yf^{-1})f = \{y\}.$$

The other inclusion is proved in a similar fashion. □

Proof of Lemma 4.4: Given a transformation $f \in \langle A \rangle$, with expression $w \in A^+$, the proof consists in eliminating enough redundant characters in w to obtain an expression of length at most $m-1$. Let $y \in Xf$; for each decomposition of w into $w = uv$, with $u, v \in A^*$, define the sets $(yf^{-1})\phi(u) = \{ x\phi(u) \mid xf = y \}$ and

$$F_y(u) = \bigcup_{v = v_1 v_2} (yf^{-1})\phi(uv_1),$$

where the union is on every possible way of cutting v into two factors $v_1, v_2 \in A^*$. By Proposition 4.5, $F_y(u) \subseteq y\phi(v)^{-1}$.

Isolate the last character of u, so that $u = u_1 a$, $a \in A$, and consider the set $(yf^{-1})\phi(u_1)$. If $(yf^{-1})\phi(u_1) \subseteq F_y(u)$, then, since $F_y(u) \subseteq y\phi(v)^{-1}$, one gets

$$(yf^{-1})\phi(u_1 v) \subseteq (yf^{-1})f = \{y\}.$$

Therefore, with $w = u_1 av$, having $(yf^{-1})\phi(u_1) \subseteq F_y(u_1 a)$ is a sufficient condition for character a to be superfluous in bringing the states of yf^{-1} to their image y. If $(yf^{-1})\phi(u_1) \subseteq F_y(u_1 a)$ for every $y \in Xf$, then a is redundant in w and can be deleted from the expression.

To complete the proof, consider an algorithm, which reads an expression w from right to left, looking first at $w = u_1 a$, and starting with $F_y(w) = \{y\}$ for every $y \in Xf$. At every iteration, it is verified whether $(yf^{-1})\phi(u_1) \not\subseteq F_y(u_1 a)$ for some $y \in Xf$. If so, then character a is retained, and the sets $F_y(u_1 a)$ are updated into $F_y(u_1) = F_y(u_1 a) \cup (yf^{-1})\phi(u_1)$. Otherwise, a is redundant and is deleted from w.

Notice that, for a character to be retained, at least one $F_y(u_1 a)$ must be enlarged by the union with $(yf^{-1})\phi(u_1)$. As $F_y(u_1 a) \subseteq yf^{-1}$ for every $y \in Xf$ and every $u_1 a$, and since the sets yf^{-1} partition X, a state of X can contribute to enlarge only one set $F_y(u_1 a)$, and this only once, which means that there cannot be more characters retained than there are states available in $X - Xf$. With $|X| = m$ and $|Xf| \geq 1$, this means an upper bound of $m-1$. □

4.3 Alphabets, J-classes and strongly connected components

In this section, it is shown that the intractability of the Membership Problem in idempotent monoids does not lie in the identification of the J-class of the test-transformation, as is the case in commutative monoids of threshold two or more. Indeed, the results of this section give a method to identify this J-class, which is a first step in the algorithms for membership testing developped in this chapter. The discussion in this section is based on two facts taken from [GrR].

Fact 4.6: Let f and g belong to the same J-class of an idempotent monoid. Then, for any u, v, w, x belonging to the same J-class as f and g, and such that $f = uv$ and $g = wx$, one has $fg = ux$ and $gf = wv$.

Fact 4.7: For any two elements f and g in an idempotent monoid,

$$f \; J \; g \quad \textbf{iff} \quad \boldsymbol{a}(f) = \boldsymbol{a}(g).$$

As a consequence, one has

Proposition 4.8: For all $a \in A$, $f \in \langle A \rangle$, these three conditions are equivalent:

$$f \; J \; fa \quad \textbf{iff} \quad |Xf| = |Xfa| \quad \textbf{iff} \quad a \in \boldsymbol{a}(f).$$

Proof: The first condition implies the second by Fact 2.2.
The second implies the third, by observing first that $Xfaf \subseteq Xf$, and that $|Xfafa| = |Xfa| = |Xf|$, so that $Xfaf = Xf$. Therefore, af acts as a permutation on Xf, and aperiodicity forces it to act as the identity, so that $xaf = x$ for every $x \in Xf$, hence $faf = f$. In terms of the maximal alphabet of f, this means $a \in \boldsymbol{a}(f)$.
The third implies the first: that $a \in \boldsymbol{a}(f)$ implies that there are $u, v \in \langle A \rangle$ such that $f = uav$. Now, $fuaf = fua(uav) = f(uava)v = fuav = f^2 = f$ by idempotency. This means $af \; L \; f$, since $f = (fu)af$, and $af \; L \; f$ implies $af \; J \; f$. □

Corollary 4.9: For every $f, g \in \langle A \rangle$, if $\boldsymbol{a}(g) \subseteq \boldsymbol{a}(f)$, then $f \; J \; fg$. □

Proposition 4.10: The maximal alphabet $\boldsymbol{a}(f)$ which a transformation f would have if it were element of an idempotent monoid $\langle A \rangle$ can be computed in $NC^1/O(mn)$.

Proof (algorithm):
0. Set $\boldsymbol{a}(f) = \emptyset$ and compute $|Xf|$.
1. For each $a \in A$, add a to $\boldsymbol{a}(f)$ if $|Xfa| = |Xf|$. □

The next two theorems are direct consequences of Proposition 4.8. They show that, as is the case in commutative monoids, the J-classes are closely linked to the decomposition of the semiautomaton into strongly connected components.

Theorem 4.11: Let (X,A) be idempotent. For every SCC K of (X,A), there is a subset B of A such that, for every $g \in <A>$:

a/ if $\alpha(g) \subseteq B$, then $Kg \subseteq K$;

b/ if $\alpha(g) \not\subseteq B$, then $Kg \cap K = \emptyset$;

c/ if $\alpha(g) = B$, then $|Kg| = 1$.

Define the alphabet of a state $x \in X$ to be

$$\alpha(x) = \{ a \in A \mid a \in \alpha(g) \text{ for some } g : xg = x \}.$$

There is at least one $f \in <A>$ such that $\alpha(f) = \alpha(x)$ and $xf = x$: just compose together all the $g \in <A>$ for which $xg = x$. The following proposition allows to define the alphabet $\alpha(K)$ of an SCC K as the alphabet of its states.

Proposition 4.12: All the states inside an SCC have the same alphabet. □

Proof of Theorem 4.11: It is first shown that, if K is an SCC, $x \in K$, and $g \in <A>$ such that $\alpha(g) \subseteq \alpha(x)$, then $xg \in K$. To see this, notice that, by Corollary 4.9, for all $f \in <A>$ such that $\alpha(f) = \alpha(x)$, one has $fgf = f$. In particular, there is a f such that $xf = x$, which means $xfgf = x = xgf$; therefore xg and x belong to the same SCC. This proves part a/ of the Theorem.

Part b/ is demonstrated *ab absurdo* with a similar argument.

For part c/, assume the existence of two distinct states x and y in Kf, with $\alpha(f) = \alpha(K)$. There is an $h \in <\alpha(K)>$ such that $yh = x$; then one has $yfhf = (yf)hf = yhf = xf = x$. But $\alpha(h) \subseteq \alpha(f)$ implies $fhf = f$ by Corollary 4.9, hence $yfhf = yf = y$, a contradiction. □

Notice that the set B of Theorem 4.11 is the alphabet of the SCC, $\alpha(K)$.

Recall Fact 4.6, the decomposition of an f into $f = uv$, with $f \; J \; u \; J \; v$. Theorem 4.11 allows to decompose the action of transformation f into two parts, the action of a *prefix* (a transformation u satisfying to the conditions of Fact 4.6), and that of a *suffix* (a v defined as in Fact 4.6). The reader will notice that this definition of prefix and suffix allows for a vast choice, up to taking $u = v = f$.

Theorem 4.13: Any prefix u and suffix v of $f \in \langle A \rangle$ with $\alpha(f) = A$, act on a state $x \in X$ as follows: the prefix selects the SCC into which x will be mapped; the suffix then maps this whole SCC onto the image-state xf. Therefore, v coincides with f inside the image-SCC.

Notice that the condition $\alpha(f) = A$ does not mean a loss of generality: if $\alpha(f) \subset A$, then work with the SCCs of the semiautomaton $(X, \alpha(f))$.

Proof: The second part is a direct consequence of part c/ of theorem 4.11, since for an SCC K with $\alpha(K) = \alpha(f)$ one has $|Kf| = |Kv| = 1$ from $f \; J \; v$ (by Fact 2.2), and $Kf \subseteq Kv$ from $f = uv$.
For the first part, it suffices to show that for any $x \in X$, xu and xf are in the same SCC. This is obtained by observing that $\alpha(f) = \alpha(u) \subseteq \alpha(xu)$, and applying Theorem 4.11, part a/. □

This result suggests a decomposition of the membership test into three main steps:

1. Find the maximal alphabet $\alpha(f)$, using Proposition 4.10; from then on, work in $(X, \alpha(f))$.
2. Test the suffix of f in the semiautomaton $(X_1, \alpha(f))$, where X_1 is the union of all SCCs in $(X, \alpha(f))$ having alphabet $\alpha(f)$.
3. Test the prefix in $(X_2, \alpha(f))$, where $X_2 = X - X_1 \cup \{k_j \mid K_j \subseteq X_1\}$, that is, where each one of the SCCs retained at Step 2 has been replaced by a single sink state, for which $k_j a = k_j$ for every $a \in \alpha(f)$.

4.4 An intuitive look at idempotent transformation monoids

The aim of this section is to introduce definitions which will be used in the rest of this chapter, and to describe the action of a transformation in a semiautomaton with an idempotent monoid, in order to give the reader an intuitive idea of what makes the Membership Problem in such monoids intractable.

These definitions apply to a word $w \in A^*$, and are as follows
A *left pivot* of a word is the first occurence of a given character in this word, when reading it from left to right; pivots are ordered: first pivot, second, etc...
To a left pivot is associated a *left factor,* which is merely that part of the word which lies to the left of the pivot.
This vocabulary is local to this thesis. However, it can be related to the nomenclature used in [GrR] as follows: the last left pivot is the *initial mark,* and the last left factor,

that is, that part of the word which lies to the left of the initial mark, is the *initial* of the word.
Dual definitions apply for right pivots, right factors, terminal and terminal mark.

Example: Expression $\dot{a}\dot{b}\dot{c}acdacbdac\ddot{d}ab\ddot{a}\ddot{b}$ has its left pivots a, b, c, d marked with dots, and its right pivots b, a, d, c marked with double dots. Its left factors are: e, a, ab, $abcac$, and its right factors e, b, $abab$, $dabab$. Its initial is $abcac$, and its terminal $dabab$.

These definitions are used in the description of the congruence $\approx$ defined on A^* by the pseudovariety A_1 of idempotent monoids on the alphabet A.

Fact 4.15: [GrR] For any two words $w_1, w_2 \in A^*$, $w_1 \approx w_2$ **iff** they satisfy either of a/ or b/:

a/ $w_1 = w_2 = e$

b/ $-\alpha(w_1) = \alpha(w_2)$, and
- w_1 and w_2 have the same initial and terminal marks, and
- the initials of w_1 and w_2 are equivalent under $\approx$, and
- the terminals of w_1 and w_2 are equivalent under $\approx$.

Notice that this description ignores whatever lies between the initial mark and the terminal mark, provided that the former lies to the left of the latter. Furthermore, since $w \approx ww$, an expression can always be squared in order to make the initial mark appear to the left of the terminal mark. A word w can therefore be decomposed canonically into an initial u, an initial mark a, a terminal mark b, and a terminal v, with the initial and the terminal themselves decomposed canonically, so that $w \approx uabv$. Notice also that the relation $\approx$ implies that w_1 and w_2 must have the same sequences of left and right pivots.
With these definitions, observe that the image $\phi(ua)$ of the concatenation of the initial and of the initial mark of a word w by the canonical homomorphism is a prefix of $\phi(w)$, as defined by Fact 4.6. A similar observation applies for the suffix.

Example: The expression $\dot{a}\dot{b}\dot{c}acdacbdac\ddot{d}ab\ddot{a}\ddot{b}$ from the previous example can be reduced to an equivalent word by first taking away whatever lies between the initial and terminal marks, to give $w \approx \dot{a}\dot{b}\dot{c}acdc\ddot{d}ab\ddot{a}\ddot{b}$. Work can then be done on the initial and the terminal, in order to obtain a canonical decomposition for each of them. For the initial $abcac$, this gives $abcbcac$, and its own initial and terminal could then be transformed similarly.
One can also try to obtain a shortest equivalent expression for w, by directly applying the defining identity $f^2 = f$, which translates here into $x^2 \approx x$ for all $x \in A^*$. For this, start from the already simplified expression $w \approx abcacdcdabab$, obtained

above, and notice that the last four characters form a square, which can be reduced to give $w \approx abcacdcdab$. The central $cdcd$ can also be simplified, to give $abcacdab$. Monoid elements $\phi(abcacd)$ and $\phi(cdab)$ are valid prefix and suffix of $\phi(w)$.

The recursive decomposition of expressions given by Fact 4.15 leads to a 'recursive' way of analyzing how a transformation f, whose expression w admits a canonical decomposition $uabv$, maps a state x to its image xf. In what follows, work is done in (X,A), with $f = \phi(w)$; it is assumed, without loss of generality, that $w = uabv$, and that $\alpha(w) = A$.

By Theorem 4.13, the prefix $\phi(ua)$ maps x into the SCC K of alphabet $\mathcal{A}(K) = A$ which contains xf. With $\alpha(u) = A-\{a\}$, one has $A-\{a\} \subseteq \mathcal{A}(x\,\phi(u))$; so that either $x\,\phi(u) \in K$, and then the action of character a on x is redundant, as the whole SCC K will be mapped anyways onto xf by the suffix, so that $x\,\phi(ubv) = xf$, or $x\,\phi(u) \in K_1 \neq K$, with $\mathcal{A}(K_1) = A-\{a\}$. In this case, the passage from K_1 to K is done through an edge labelled a, going from state $x\,\phi(u) \in K_1$ to $x\,\phi(ua)$ in K.
Still analyzing this second case, observe that u can itself be canonically decomposed into $u = u_1a_1b_1v_1$, with $x\,\phi(u_1a_1) \in K_1$, and $\phi(b_1v_1)$ mapping K_1 onto $x\,\phi(u)$, that is, selecting that state $q = x\,\phi(u)$ of K_1 for which $qa = x\,\phi(ua) \in K$.
Meanwhile, the action of suffix $\phi(bv)$ on SCC K can be described as follows: generator b maps K onto a subset $Kb \subseteq K$, and $\phi(v)$ then maps Kb onto $K\phi(bv) = \{y\}$. Since $b \notin \alpha(v)$, the action of $\phi(v)$ can be seen as taking place in the semiautomaton $(K,A-\{b\})$. With v being canonically decomposed into $v = u_2a_2b_2v_2$, observe that $\phi(u_2a_2)$ maps Kb into the correct maximal SCC K_2 of $(K,A-\{b\})$ which contains xf, that is, a sub-SCC of K in (X,A) with alphabet $A-\{b\}$, and that $\phi(b_2v_2)$ then maps this sub-SCC onto the image-state xf.

From this description, a number of possible causes for intractability of the Membership Problem become apparent. The reader is reminded, at this point, that the Problem consists in giving a one-bit answer to a question about membership of f in $\langle A \rangle$, and that, theoretically, this could be done without having to explicitly compute an expression for the test-transformation. The following considerations should be therefore regarded as pertaining to a test to verify whether the action of f in the semiautomaton is 'compatible' with membership in $\langle A \rangle$.

First, recall that $\phi(u)$ maps SCC K_1 onto one of its states q, which is itself mapped by a into the maximal SCC K. For every such SCC K_1, there can be several states $q \in K_1$ such that $qa \in K$; to a given q corresponds a set of possible suffixes $\phi(b_1v_1)$ of $\phi(u)$, and the number of possible choices of q, when combined between all these

SCCs of alphabet $A-\{a\}$, can become exponential in the size of the semiautomaton. Inside a maximal SCC K (where the suffix is tested), the decomposition of the suffix into $\phi(bv)$ can also lead to an exponential number of possibilities to look at, as follows. Assume first that the terminal mark b is known, so that only the terminal v has to be tested for. With the canonical decomposition $v = u_2 a_2 b_2 v_2$, the action of $\phi(b_2 v_2)$ is already defined: it maps the sub-SCC K_2 onto the image-state xf. Meanwhile, the action of $\phi(u_2 a_2)$ is known for certain subsets of K: it maps K_2 into itself, and similarly for every sub-SCC of alphabet $\alpha(v) = A-\{b\}$, and it maps Kb into K_2. However, there remain states of K which belong to none of these subsets of K, and for which the action of $\phi(u_2 a_2)$ is undefined. Again, the number of possible choices, deciding which sub-SCC each such state is mapped into, can become exponential.

Finding the initial and terminal marks seems to be less of a problem: just take one among the n generators and verify that it is the right choice; but this choice cannot quite be verified until it has been decided whether there is a valid corresponding initial (or terminal), which brings the problem back to the above considerations.

4.5 The lattice of idempotent pseudovarieties

In section 4.6, the proof of Theorem 4.3 is completed, which will provide a formal support for the intuitive argument of the previous section. However, the lattice of classes of idempotent monoids will first be described and explored, in order to show where the passage from NC^1 to NP-completeness occurs. It will be demonstrated that, upon climbing this lattice up to $\mathbf{A}_1$, the Problem first belongs to FL, then becomes hard for NL, and therefore at least NC^2, while still feasible in polynomial sequential time (this section), and then reaches NP-completeness (section 4.6).

The lattice of pseudovarieties of idempotent monoids is represented in Figure 4.1; it is based on the work of Gerhard [Ger] and Fennemore [Fen], who gave a description of the whole lattice in terms of semigroup classes. In this figure, each arrow represents a relation of strict inclusion. The lattice starts at the left with a class consisting only of the trivial monoid; it is dominated at infinity (to the right) by the class $\mathbf{A}_1$ of all idempotent monoids. All pseudovarieties lying on the middle row are self-dual. Each class on the top row is dual to its counterpart on the bottom row. The data used in the following sections is taken from [Ger] and [Fen], although some notations and vocabulary are local to this thesis. In particular, the labels $\mathbf{M}_3$, $\mathbf{N}_4$ and $\mathbf{XR}$ are non-standard. In Gerhard's notation, these classes are identified as $\mathbf{R}_3$ $\mathbf{R}_3' \mathcal{J}_3 \mathcal{J}_3'$, $\mathbf{R}_4 \mathbf{R}_4'$, and $\mathbf{T}_3$ $\mathbf{R}_3' \mathcal{J}_3'$.

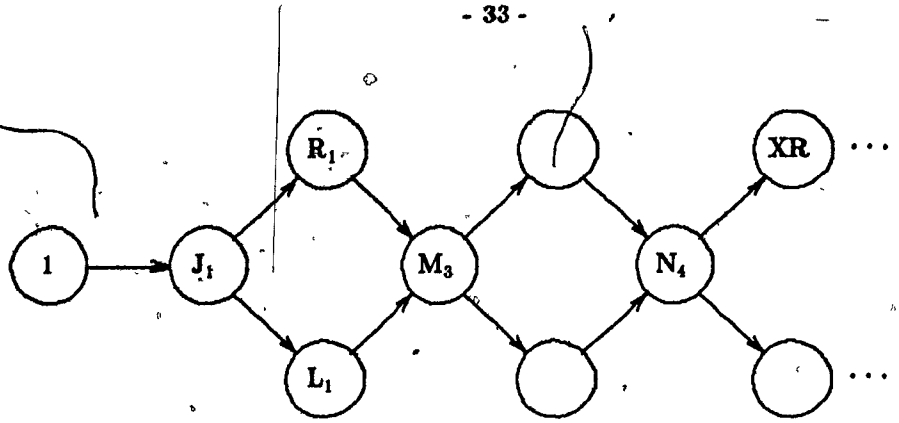

FIGURE 4.1: Pseudovarieties of idempotent monoids

Already seen in section 3.3 is the class $\mathbf{J}_1$ of commutative idempotent monoids. It will now be demonstrated that this is the unique largest class of idempotent, and therefore aperiodic, monoids for which the Membership Problem belongs to *FL*, provided that the complexity classes *FL* and *NL* differ. To do so, the first two classes not included in $\mathbf{J}_1$ are looked at: these are $\mathbf{R}_1$ and $\mathbf{L}_1$.

Definition: A monoid S belongs to $\mathbf{R}_1$ **iff** every $g, h \in S$ satisfy the equations $ghg = gh$ and $g^2 = g$. Similarly, the class $\mathbf{L}_1$ is defined by the identities $ghg = hg$ and $g^2 = g$.

Fact 4.16: Two words of A^+ are expressions of the same element of the free $\mathbf{R}_1$ monoid on A **iff** they have the same sequence of left pivots (right pivots for $\mathbf{L}_1$).

Proposition 4.17: The class test in $\mathbf{R}_1$ and $\mathbf{L}_1$ belongs to $NC^1 / O(mn^2)$.

Proof: The algorithm for $\mathbf{R}_1$ consists in testing that the conditions $g^2 = g$ and $ghg = gh$ are satisfied by the generators: $NC^1/O(mn^2)$. That this ensures that $ghg = gh$ for every $g, h \in \langle A \rangle$ is proved by induction on the length of an expression for g. With length 1, which means $g \in A$, let $h = a_1 a_2 a_3 \cdots a_p$, and do

$$\begin{aligned} ghg = ga_1a_2a_3\cdots a_p\, g &= (ga_1)a_2a_3\cdots a_p\, g = ga_1ga_2a_3\cdots a_p\, g \\ &\cdots = ga_1ga_2ga_3\cdots ga_p\, g = ga_1ga_2ga_3\cdots ga_p \\ &\cdots = ga_1a_2a_3\cdots a_p = gh. \end{aligned}$$

With length greater that 1, let $g = kb$, with $b \in A$. Then

$$kbhkb = kbkhkb = kbkhb = kbhb = kbh.$$

That $g^2 = g$ is demonstrated in a similar fashion. The proof for $\mathbf{L}_1$ is identical. Notice that this Class Test bypasses the test for idempotency of section 4.1, which has complexity $NC^2 / O(m^4 n)$. □

The existence of a jump in the complexity of the Membership Problem is now

demonstrated, by proving a 'hardness' result for the first two pseudovarieties of idempotent monoids not included in $\mathbf{J}_1$.

Theorem 4.18: The Membership Problem in $\mathbf{R}_1$ is non-deterministic log-space hard, relative to NC^1-reducibility.

Proof: The demonstration consists in doing a NC^1 reduction from the Accessibility Problem in a directed graph (DGAP).

Fact 4.19: [Jon] . DGAP is complete for NL via NC^1 reducibility.

Reduction: Let an instance of DGAP be given as a list of nodes and edges: sets V of vertices and $E \subseteq V \times V$ of edges, plus the name of the start and arrival nodes. Assume that these are v_1 and v_m, with the graph having m vertices, $m > 1$.
First, weed out trivial loops (NC^1), since they have no influence on the outcome of DGAP. Then, for each node v_i, construct the sets B_i of edges incoming to v_i and C_i of edges leaving v_i. This is feasible in NC^1, and consists merely in enlarging the input for the same instance of DGAP.
An instance of the Membership Problem in a semiautomaton (X,A) is now built by defining a connected component K_i for every vertex v_i of V, a generator of A for every edge of E, plus one extra generator d, and by redefining the sets B_i and C_i as subsets of A. The connected components of (X,A) are constructed as follows: (The one-line matrix notation is used for transformations, with states of origin given in alphabetic order, so that the identity in $\{p, q, r, x\}$ is denoted $(p \ q \ r \ x)$)

$K_1 = \{ p_1, q_1, x_1 \}$, with every generator $a \in C_1$ acting as $(q_1 \ q_1 \ x_1)$, and every $a \notin C_1$ as $(x_1 \ q_1 \ x_1)$;
$K_i = \{ p_i, q_i, r_i, x_i \}$, for $1 < i < m$, with every $a \in B_i$ acting as $(r_i \ q_i \ r_i \ x_i)$, every $a \in C_i$ acting as $(x_i \ q_i \ q_i \ x_i)$, generator d as $(q_i \ q_i \ q_i \ x_i)$, and every other generator as the identity;
$K_m = \{ p_m, q_m, r_m, x_m \}$, with every $a \in B_m$ acting as $(r_m \ q_m \ r_m \ x_m)$, every $a \in C_m$ acting as $(x_m \ q_m \ x_m \ x_m)$, generator d as $(x_m \ q_m \ q_m \ x_m)$, and every other generator as the identity.

Define test-transformation f as follows: for all i, $1 \leq i \leq m$, let $x_i f = x_i$, and $y_i f = q_i$ for all $y_i \neq x_i$.

Connected components K_1, K_i (for some i, $1 < i < m$), and K_m are drawn in Figure 4.2. In each of these connected components, only those generators not acting as the identity are represented. The arrows are labelled with the set of those generators whose action they correspond to.

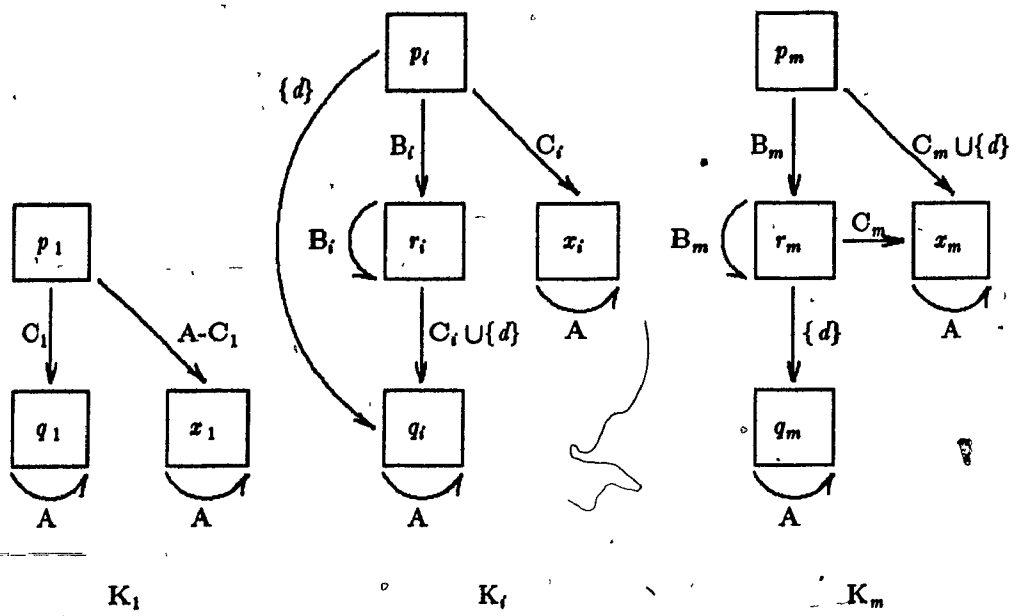

FIGURE 4.2: Connected components for the proof of Theorem 4.18

Lemma 4.20: There is a path from v_1 to v_m **iff** $f \in \langle A \rangle$

Proof: (only if) Let the vertices encountered along a path from v_1 to v_m be $u_1, \cdots, u_k$, with $v_1 = u_1$ and $v_m = u_k$; without loss of generality, the path can be assumed to be cycle-free. It can be described by a word of A^+ built from the characters corresponding to the edges traversed, listed in the same left-to-right order, so that $w = (u_1,u_2)(u_2,u_3)\cdots(u_{k-1},u_k)$. It is claimed that $\phi(wd)$ acts in (X,A) exactly as f does. This is verified separately on every connected component.

In K_1, since w starts with $(v_1,u_2) \in C_1$, this is immediate.

In every K_i corresponding to a node u_i along the path, one has $(u_{i-1},u_i) \in B_i$, and $(u_i,u_{i+1}) \in C_i$, with those edges appearing earlier on the path belonging to neither, and therefore acting as the identity on this component, so that the sequence of an element of B_i followed by an element of C_i acts as f on K_i.

In K_m, one has $(u_{k-1},u_k) \in B_m$, followed by d.

In those K_i corresponding to a node not traversed by the path, every character of w acts as the identity; then, d acts as f on K_i.

Therefore, $\phi(wd)$ acts exactly as f on every connected component of (X,A).

(if) Let $f \in \langle A \rangle$ with expression w. Connected component K_m imposes three conditions: 1/ that w contains at least one d; 2/ that at least one of the left pivots which precede the first occurence of d be a character from B_m, corresponding to an edge entering vertex v_m, and 3/ that no element of C_m (edges leaving v_m) appear before the first occurence of d.

It is now claimed that the sequence of left pivots, up to the one which is the first occurence of d, gives a list of edges, the endpoints of which are vertices accessible from v_1 without passing through v_m. The proof is by induction on the pivots encountered while reading the expression from the left.
Component K_1 imposes that the first character of w, that is, its first left pivot, belong to C_1, which means an edge leaving from v_1.
Assume that the sequence $(v_1,u_2)\cdot\cdots(u_{k-1},u_k)$ is such that every node u_i, $2 \leq i \leq k$, is accessible from vertex v_1. Then either the next pivot is d, which implies $u_i = v_m$ for some $i \leq k$, a constraint imposed by component K_m, or the next pivot is another edge (u_i,u_j). If vertex u_i has not been encountered so far, then all the characters to the left of the one considered now have acted as the identity on the connected component K_i associated with vertex u_i; since the new edge (u_i,u_j) belongs to C_i, the corresponding generator maps state p_i on x_i, which is not compatible with the action of f on K_i. Therefore, vertex u_i must have already been encountered, and, by the induction hypothesis, is accessible from v_1, which implies that the endpoint of the new edge, vertex u_j, can also be reached from v_1.

Semiautomaton (X,A) is of size linear in the size of the original graph, it has a transformation monoid belonging to $\mathbf{R}_1$, which can be tested using Proposition 4.17, and its construction can be done in NC^1. □

Corollary 4.21: The Membership Problem in $\mathbf{L}_1$ is also hard for NL.

Proof: Every connected component of the $\mathbf{R}_1$ semiautomaton built for the proof of Theorem 4.18 has at most four states. By Theorem 2.11, this instance of the Membership Problem can therefore be NC^1-reduced to an instance in an $\mathbf{L}_1$ monoid. □

It is shown below that membership in $\mathbf{R}_1$ and $\mathbf{L}_1$ monoids can be tested in polynomial time. However, the question as to whether this problem belongs to NC or is complete for FP ('inherently sequential') is still open. Yet, this represents a jump in complexity, therefore an intermediate step between FL ($\mathbf{J}_1$ monoids) and NP-completeness.

The pattern of having a unique maximal class of aperiodic monoids in which the Membership Problem belongs to some complexity class is repeated once more, this time with the class FP of problems solvable in polynomial time. The pseudovarieties $\mathbf{R}_1$ and $\mathbf{L}_1$ are contained in the class $\mathbf{N}_4$, defined by the relation $\approx_4$.

Definition: [Ger] For $p \geq 4$, for all $w_1,w_2 \in A^*$, $w_1 \approx_p w_2$ **iff** they satisfy all of
1/ w_1 and w_2 have the same sequence of left pivots
2/ w_1 and w_2 have the same sequence of right pivots
3/ every left factor u_1 of w_1 satisfies relation $\approx_{p-1}$ with the corresponding left

factor u_2 of w_2: $u_1 \approx_{p-1} u_2$;
4/ the same is true for right factors.

In the case $p = 4$, this means: for all $w_1, w_2 \in A^*$, $w_1 \approx_4 w_2$ **iff** they satisfy all of
1/ w_1 and w_2 have the same sequence of left pivots
2/ w_1 and w_2 have the same sequence of right pivots
3/ every left factor u_1 of w_1 satisfies relation $\approx_3$ with the corresponding left factor u_2 of w_2: $u_1 \approx_3 u_2$; two words are equivalent under the relation $\approx_3$ **iff** they have the same alphabet and the same initial and final characters;
4/ the same is true for right factors.

Example: The following two words are equivalent under $\approx_4$. In these words, the left pivots have been marked with dots, and the right pivots with double dots.

$$\dot{a}\ \dot{b}\ a\ \dot{c}\ a\ b\ c\ \dot{d}\ \ddot{c}\ d\ b\ a\ \ddot{b}\ a\ \ddot{d}\ a\ \ddot{a} \quad \approx_4 \quad \dot{a}\ \dot{b}\ a\ \dot{c}\ \dot{d}\ \ddot{c}\ d\ \ddot{b}\ a\ \ddot{d}\ \ddot{a}$$

Testing membership in an N_4 monoid can be done as follows.

Sequential algorithm

0. Find the maximal alphabet $A = \alpha(f)$; from then on, work in (X,A).
1. Compute a prefix of f :
  1.1. Find an $a \in A$ such that $f = af$. Let $B = \{a\}$ and $u = a$.
  1.2. While $B \neq A$ do
    Look for $b \in B$ and $c \notin B$ such that $\phi(ubc)f = f$. If none is found, then HALT: $f \notin \langle A \rangle$. Else, do $u = ubc$ and $B = B \cup \{c\}$.
2. Compute a suffix of f :
  2.1. Find an $a \in A$ such that $f = fa$. Let $B = \{a\}$ and $v = a$.
  2.2. While $B \neq A$ do
    Look for $b \in B$ and $c \notin B$ such that $f\,\phi(cbv) = f$. If none is found, then HALT: $f \notin \langle A \rangle$. Else, do $v = cbv$ and $B = B \cup \{c\}$.
3. $f = \phi(uv)$ **iff** $f \in \langle A \rangle$.

Proposition 4.22: $f = \phi(uv)$ **iff** $f \in \langle A \rangle$.

Proof: The *(only if)* part is trivial. The *(if)* part is demonstrated by showing that $f \in \langle A \rangle$ implies that the algorithm goes through Step 1 without halting; the same can be proved for Step 2 in a similar fashion.
The proof is by induction, showing that every iteration in the loop of Step 1.2 gives as a result a u which is the concatenation of a valid left factor of f and of its associated left pivot. The basis is clear: start with $u = a$, then $b = a$ and $f = acf$ for some $c \in A$. At the induction step, one has in hands a u, with $\alpha(u) = B$. If

$f \in \langle A \rangle$ and $\phi(u)f = f$, then the sequences of left pivots in the expressions of f and $\phi(u)f$ are the same, and their initial factors have the same alphabet and last character, by relation $\approx_3$. If $f \in \langle A \rangle$, then its next left factor u' will be such that $u' = ut$, for some $t \in \langle \alpha(u) \rangle$. Since u contains at least one instance of every character in $\alpha(u)$, however, all the characters of t are redundant, except the very last, so that $u' \approx_3 ub$ for some b. The next left pivot is looked at (the c of Step 1.2), and this completes the induction.
Remark that it is important to look for b and c simultaneously, since $\phi(ub)f = f$ is true for any $b \in \alpha(u)$, while this is not the case for $\phi(ubc)f = f$. □

Theorem 4.23: Membership in an $\mathbf{N}_4$ transformation monoid can be tested in $O(mn^3)$ sequential time.

Proof: The algorithm, shown to be valid in Proposition 4.22, has its complexity dominated by Steps 1 and 2, each consisting of a loop over A in which pairs of generators are tested on, hence $O(n)$ iterations at a cost of $O(mn^2)$ each. □

Corollary 4.24: Membership in an $\mathbf{R}_1$ or an $\mathbf{L}_1$ transformation monoid can be tested in $O(mn^2)$ sequential time.

Proof: In an $\mathbf{R}_1$ monoid, all that is looked for is the sequence of left pivots, so that Step 2 is ignored, and Step 1.2 simplifies to

1.2. While $B \neq A$ do
look for $c \notin B$ such that $\phi(uc)f = f$. If none is found, then HALT; else do $u = uc$ and $B = B \cup \{c\}$.

This lowers the complexity down to $O(mn^2)$. A similar argument applies for $\mathbf{L}_1$. □

The class $\mathbf{M}_3$ is the first idempotent pseudovariety to contain both $\mathbf{R}_1$ and $\mathbf{L}_1$ (Figure 4.1); in the relation defined by $\mathbf{M}_3$, two words are equivalent iff they have the same sequences of left and right pivots. It can be observed that Corollary 4.24 extends to this class, so that membership testing in $\mathbf{M}_3$ can be done in $O(mn^2)$.
At this point, it might be of interest to verify whether the upper bounds obtained for $\mathbf{M}_3$ and $\mathbf{N}_4$ are tight. If so, then $\mathbf{M}_3$ would be the unique largest class for which the Membership Problem would be feasible in $O(mn^2)$, that is, yet another step as the complexity of the Problem rises to *NP*-completeness.

4.6 NP-completeness of the Membership Problem

In this section, the proof of Theorem 4.3 is completed: it is shown that the pseudovariety $\mathbf{N}_4$ is the unique largest class of aperiodic monoids for which the Membership Problem is not intractable.

Theorem 4.25: The Membership Problem in aperiodic transformation monoids not belonging to $\mathbf{N}_4$ is *NP*-hard.

Proof: The demonstration is a reduction of the problem 'one-in-three 3SAT' with no negated variable [GJ] to an instance of the Membership Problem in a monoid belonging to the pseudovariety **XR**, a subclass of $\mathbf{A}_1$ defined by the identities $g^2 = g$ and $ghk = ghkgkhk$ [Fen]. Along with its dual, **XR** is the smallest class of idempotent monoids containing $\mathbf{N}_4$ (see Figure 4.1).

Reduction: Let an instance of 'one-in-three 3SAT' with no negated variables be given as a set U of Boolean variables and a set of clauses of the form (u_1,u_2,u_3), with $u_1,u_2,u_3 \in U$; a clause C_j will be satisfied **iff** exactly one of the three variables it involves has value *True*.
Define two generators a_i and b_i for every variable u_i, plus an extra generator called d. For every clause $C_j = (u_1,u_2,u_3)$, a connected component K_j of (X,A) is built as follows.
The connected component consists of ten states: two sinks, designated x and y, and an SCC whose eight states are labelled with the eight possible combinations of one generator of index 1, 2, and 3: $a_1a_2a_3$, $b_1a_2a_3$, $a_1b_2a_3$, etc...
The action of generator β_q on a state $\alpha_r\alpha_s\alpha_t$ of the component K_j, where α_r is one of a_r or b_r, and $\beta_q \in \{a_q, b_q\}$, is defined as follows.

-If $q \notin \{r,s,t\}$, then β_q acts on K_j as the identity;
-if $\beta_q = \alpha_r$, then β_q maps the state $\alpha_r\alpha_s\alpha_t$ onto itself;
-if $q = r$ and $\beta_q \neq \alpha_r$, that is, if α_r is one of a_r or b_r and β_q is the other, then the state $\alpha_r\alpha_s\alpha_t$ is mapped to $\beta_q\alpha_s\alpha_t$.

A connected component K_j is represented in Figure 4.3. In this figure, those generators acting as the identity on K_j are ignored, the action of generator d is represented with dotted arrows, and trivial loops are not drawn: instead, for every state of the SCC, represented by a box, the generators which map the state onto itself are listed inside the box; this list coincides with the label of the state.

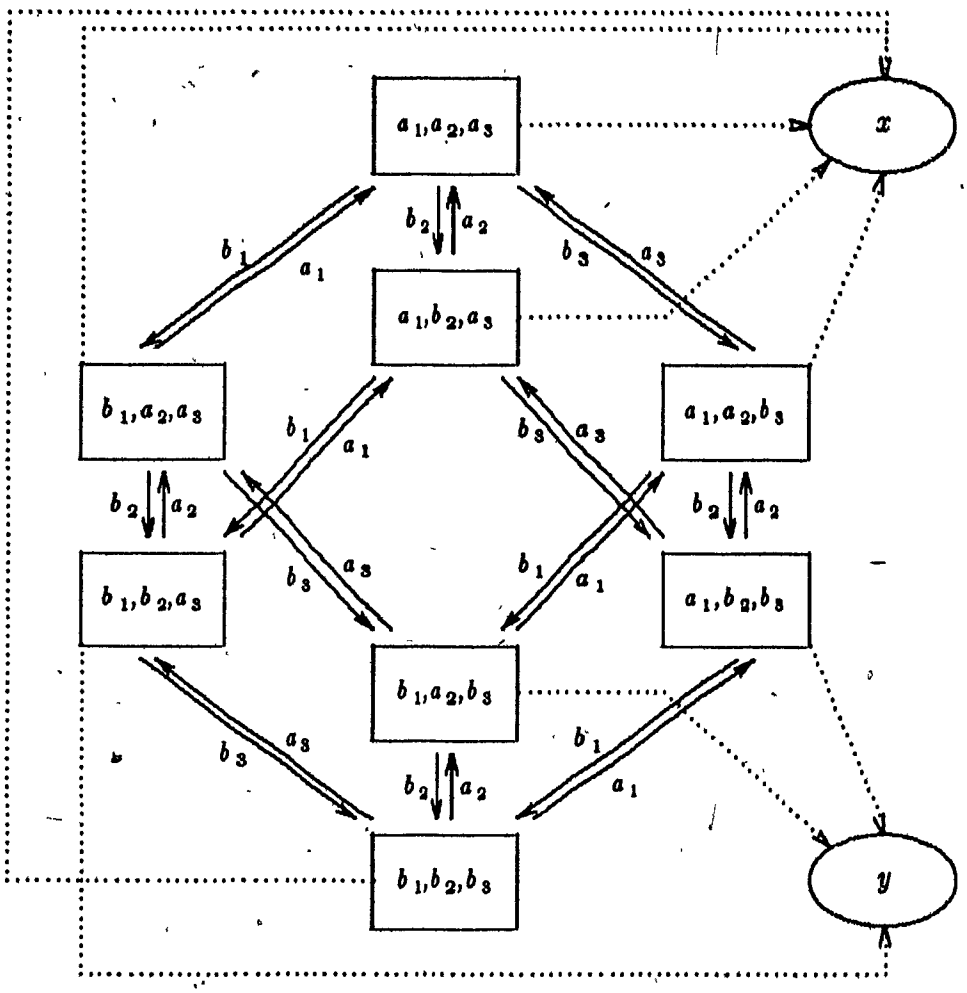

FIGURE 4.3: Connected component for the proof of Theorem 4.25

A transformation with expression udv, where $d \notin \alpha(u)$, will act on K_f as follows. Assume that u contains at least one instance of a character of index 1 (that is, $\alpha(u) \cap \{a_1, b_1\} \neq \emptyset$), and at least one of index 2 and 3, and consider the sequence of right pivots in u: the last occurences of characters of index 1, 2 and 3 will give the label of the state onto which the SCC will be mapped by $\phi(u)$. Then, generator d maps this image-state onto one of the two sinks: to y if the label contains exactly one a, and to x otherwise. The last part of the word, v, acts trivially on the sinks.

Example: Consider the words

$$w_1 = a_1\, a_2\, a_3\, b_1\, b_2\, b_3\, a_1\, b_1\, b_2\, b_3\, d$$
$$w_2 = a_1\, a_2\, a_3\, b_1\, b_2\, b_3\, b_1\, a_1\, b_2\, b_3\, d.$$

The last characters of same indices to occur are b_1, b_2, b_3 for w_1, and a_1, b_2, b_3 for w_2, so that the states of the SCC are mapped by $\phi(w_1)$ onto x, and onto y by $\phi(w_2)$.

Proposition 4.26: The transformation monoid of the semiautomaton thus built belongs to the class **XR**, and does not belong to $\mathbf{N}_4$.

Proof: To show that the monoid belongs to **XR**, it suffices to prove that the transformation monoid of every connected component belongs to this class. This is demonstrated by showing that the condition $ghk = ghkgkhk$ holds for every three elements g, h, k of the monoid.
First, let at least one of g, h or k have an expression containing an instance of d: $d \in a(g) \cup a(h) \cup a(k)$. This means that the SCC is mapped onto the sinks by ghk, and $gkhk$ then acts as the identity on them, so that the condition is satisfied. In the other case where $d \notin a(g) \cup a(h) \cup a(k)$, since the SCC takes into account only the last occurences of characters of index 1, 2, and 3, and since these last occurences are the same in ghk and $ghkgkhk$, the condition is also satisfied.
To show that the monoid does not belong to $\mathbf{N}_4$, consider the following two expressions:

$$w_1 = a_1\, a_2\, a_3\, b_1\, b_2\, b_3\, a_1\, b_1\, b_2\, b_3\, d\; a_1\, a_2\, a_3\, b_1\, b_2\, b_3$$
$$w_2 = a_1\, a_2\, a_3\, b_1\, b_2\, b_3\, b_1\, a_1\, b_2\, b_3\, d\; a_1\, a_2\, a_3\, b_1\, b_2\, b_3$$

First, verify that $w_1 \approx_4 w_2$. Then, by looking at the initials (everything up to the d), verify that transformation $\phi(w_1)$ will map the SCC of K_j onto sink x, while $\phi(w_2)$ will map it onto y, thus distinguishing between the two expressions, which shouldn't be in an $\mathbf{N}_4$ monoid. □

Define now the action of the test-transformation f on K_j as follows: $xf = x$, and $zf = y$ for every state $z \neq x$. Transformation f maps every connected component onto sink states, so that $fa = f$ for any $a \in A$; therefore, $a(f) = A$.
With the interpretation that the value of the affectation to variable u_i is determined by the last occurence of a character of index i in an expression before the first occurence of d, with a_i meaning $u_i = True$, and b_i meaning $u_i = False$, one gets the following.

Lemma 4.27: Transformation f belongs to the monoid **iff** the instance of 'one-in-three 3SAT' admits a solution.

Proof: (if) If there is a solution to the instance of 'one-in-three 3SAT', then one can build an expression of the form $\alpha_1 \cdots \alpha_n \beta_1 \cdots \beta_n d = wd$, where β_i is the generator of index i which corresponds to the value affected to variable u_i, and α_i is the other generator of index i. It is claimed that $\phi(wd)$ acts as f.
For every state p in the SCC of K_j, the state $p\phi(w)$ will also be inside the SCC, and its label will correspond to the last occurences of characters of each index in w. By the hypothesis, the label consists of one a and two b's, so that this image-state is

then mapped by d onto the sink y.

(only if) If f belongs to the monoid, then there is an expression $f = wdv$ with $\alpha(wdv) \subseteq A$ and $d \notin \alpha(w)$. It is claimed that $\alpha(w) \cap \{a_1, b_1\} \neq \emptyset$, and similarly for the indices 2 and 3. To see this, consider that in the connected component built for clause C_j, every state p of the SCC is mapped onto the sink y: this implies that p is mapped by $\phi(w)$ on a state q of the SCC whose label contains exactly one a. If p is the state $a_1a_2a_3$, then w must contain at least two of the characters b_1, b_2 and b_3. Meanwhile, for state $b_1b_2b_3$, the condition is that w contains at least one of the a's. If these two b's and one a are such that all three indices are not represented, for example, if $\alpha(w) \cap \{a_3, b_3\} = \emptyset$, then observe that no transformation of $<\{ a_1, a_2, b_1, b_2 \}>$ can simultaneously map states $a_1a_2a_3$ and $b_1b_2b_3$ onto states whose image by d is y. Therefore, all three indices are represented in w. This means that every variable u_i is assigned a value, and the condition on the last occurences of characters of a given index make this assignment consistent with the satisfaction of clause C_j. □

Proposition 4.28: The Membership Problem is *NP*-complete for the pseudovariety of monoids **XR** and for its dual.

Proof: The Problem for **XR** belongs to *NP* by Lemma 4.4, while the above reduction and Lemma 4.27 prove *NP*-hardness. For the dual class of **XR**, it suffices to observe that the number of states in every connected component built in the above reduction is a constant, so that Theorem 2.11 can be applied. □

This result completes the proof of Theorem 4.3 (*NP*-completeness of the Membership Problem in arbitrary idempotent monoids); the proof of Theorem 4.25 is also completed, recalling that the Problem is *NP*-hard in all monoids of threshold two or more, by Corollary 3.13. □

There is no Class Test available yet for the classes $\mathbf{M}_3$ and $\mathbf{N}_4$.

V Other pseudovarieties

In this thesis, the emphasis has been put on the classes of commutative and idempotent monoids, since it was there that polynomial-time membership tests could be found. Other pseudovarieties are now discussed, in order to give an idea of what can be expected from further research in this field. The main relationships between these classes are depicted in Figure 5.1.

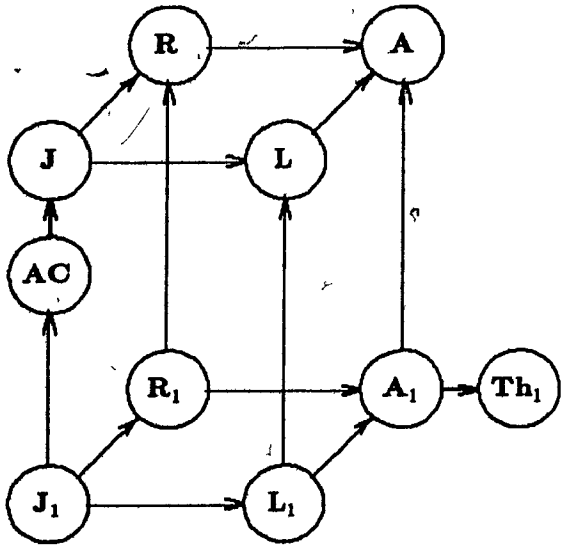

FIGURE 5.1: Some pseudovarieties of monoids

5.1 Other aperiodic pseudovarieties

A first direction in which further research on the Membership Problem can be oriented is the study of pseudovarieties of aperiodic monoids lying beyond the classes of commutative and idempotent monoids. Many of them are of particular interest because of their relationships with well-discussed classes of regular languages: J-trivial monoids (associated with piecewise-testable languages [Simon]), R- and L-trivial monoids [Fich], and, further beyond, the class of all aperiodic monoids (related to star-free languages [Sch]). Given the results of chapters Three and Four, the Membership Problem in these pseudovarieties must be intractable, the question being, for which classes it remains *NP*-complete.

Walking up the lattice of these pseudovarieties, one encounters a class containing the aperiodic commutative monoids (denoted **AC** in Figure 5.1), the class **J** of J-trivial

monoids, for which a Class Test has been found by Stern [Ste1]. It is based on the following property of semiautomata with J-trivial transformation monoid.

Fact 5.1: [Simon] The transformation monoid of semiautomaton (X,A) is J-trivial **iff** (X,A) is a acyclic directed graph **and** for every $B \subseteq A$, every connected component of (X,B) is dominated by a unique state.

Stern's observation is that every (X,B) does not need to be computed; instead of working on every subalphabet, notice that there cannot be more SCCs than there are states in X, regardless of the alphabet, as a consequence of the acyclicity of (X,A).

Stern's sequential algorithm

1. Verify that (X,A) is acyclic;
2. for every $x \in X$, compute the set $\Xi(x) = \{ a \in A \mid xa = x \}$;
3. for every pair $x \neq y$ of states, do
   - 3.1. build $M = (X, \Xi(x) \cap \Xi(y))$;
   - 3.2. compute the transitive closure $\overline{M}$ of M;
   - 3.3. for every $z \in X$, verify that (z,x) and (z,y) are not both edges of $\overline{M}$.

Step 3.2 uses Warshall's algorithm [AHU2] to compute the transitive closure of a graph with m vertices and mn edges, which takes $O((mn)^3)$ time; this gives an overall complexity of $O(m^5n^3)$. This algorithm can be run in parallel, after some minor adjustments.

Parallel algorithm

1. Compute the SCCs of (X,A); verify that they are trivial;
2. for every $x \in X$ in parallel, compute the set $\Xi(x) = \{ a \in A \mid xa = x \}$;
3. for every pair $x \neq y$ of states in parallel, do
   - 3.1. build $M = (X, \Xi(x) \cap \Xi(y))$;
   - 3.2. for every $z \in X$, check that x and y are not both accessible from z in M.

Proposition 5.2: The Class Test in J belongs to NC^2.

Proof: Steps 1 and 3.2 are instances of problems in NC^2; the rest is in NC^1. □

The pseudovariety J is the intersection of R and L, the classes of R- and L-trivial monoids. This is made clear by these two facts, taken from [Fich].

Fact 5.3: A transformation monoid is R-trivial **iff** its semiautomaton is acyclic.

Fact 5.4: The transformation monoid of a semiautomaton (X,A) is L-trivial **iff** its

inverse semiautomaton $(2^X, A^{-1})$ is acyclic.

Corollary 5.5: The Class Test in **R** belongs to NC^2 / $O(mn)$. □

Whether the Slass Test for L-trivial monoids can be done in NC is not known.

Proposition 5.6: The Membership Problem in **R** is NP-complete.

Proof: By Corollary 3.13, all that is needed is a proof that the Problem in this case belongs to NP. For this, it suffices to show that every $f \in \langle A \rangle$ has an expression w of length at most $m(m-1)$, with $m = |X|$, a consequence of the acyclicity of the semiautomaton. To see this, notice that, upon applying $\phi(w)$ character-by-character on some state x, a cycle-free path from x to xf is defined. Now, if $w = uav$ for some $a \in A$, then either $x\phi(ua) = x\phi(u)$ for all $x \in X$, so that a is redundant, or a brings an x one step nearer to its image xf, with state $x\phi(u)$ never to be revisited again on the path from x to xf. With one path per state and a maximal length of $m-1$ for each path, this gives the desired upper bound on the length of w. □

The complexity of the Membership Problem in L-trivial monoids is unknown; if the Problem for this class were not NP-complete, this would be a first case where the computational complexity of the Membership Problem for a given class would not be matched by a similar complexity for its dual. The conservative attitude is therefore to conjecture that the Problem is NP-complete also for L-trivial monoids. Notice that, here too, it suffices to prove that the Problem is in NP, for example by establishing a polynomial upper bound on the length of the expression of the elements of the monoid.

Further above is the pseudovariety of all aperiodic monoids, denoted **A** in Figure 5.1. A look at the monoid used by Kozen to prove the *PSpace*-completeness of the Membership Problem in the general case [Koz] reveals that this monoid is aperiodic, so that the highest possible complexity can be reached without having to introduce groups. An interesting open problem is therefore to verify whether there is a unique largest class of aperiodic monoids in which the Membership Problem is in NP, a pattern similar to the one encountered in this thesis for the complexity classes FL and FP.

Notice that testing whether a transformation monoid is aperiodic is already coNP-hard, with the possibility still open that it be *PSpace*-complete [Stel].

5.2 Non-aperiodic pseudovarieties

Another direction of research is to look at how more complex the Membership Problem becomes when the monoid ceases to be group-free. This question has been settled for commutative monoids in chapter Three. Another non-aperiodic class worth being investigated is the pseudovariety of monoids of threshold one, denoted $\mathbf{Th}_1$ in Figure 5.1. They have several properties in common with idempotent monoids, so that the action of a transformation in the semiautomaton of such a monoid can be described quite well. Here are some facts about this class, taken from [GrR].

Fact 5.7: In the free monoid of threshold one on alphabet A, for every $w_1, w_2 \in A^*$, one has $\phi(w_1)\, H\, \phi(w_2)$ **iff** $\alpha(w_1) = \alpha(w_2)$ **and** there exist decompositions $w_1 = u_1 a x_1 b v_1$ and $w_2 = u_2 a x_2 b v_2$ with $a, b \in A$ (possibly the same), such that

$\alpha(u_1) = \alpha(u_2) = \alpha(w_1) - \{a\}$,
$\alpha(v_1) = \alpha(v_2) = \alpha(w_1) - \{b\}$,
$\phi(u_1) = \phi(u_2)$, and $\phi(v_1) = \phi(v_2)$.

Remark: The conditions for having $\phi(w_1) = \phi(w_2)$ are here more stringent than in idempotent monoids, where every H-class consisted of exactly one element, so that the above conditions sufficed to ensure that $\phi(w_1) = \phi(w_2)$.

Fact 5.8: Every H-class in a monoid of threshold one is a group. These groups are isomorphic within the same J-class (a J-class is a disjoint union of H-classes).

Fact 5.9: For any two elements f and g of a monoid of threshold one,

$$f\ J\ g \quad \text{iff} \quad \alpha(f) = \alpha(g).$$

This last Fact is analogous to Fact 4.7. Actually, results similar to those of section 4.3 exist for all monoids of threshold one.

Fact 5.10: Any element f in a monoid of threshold one can be decomposed into $f = uxv$, with $f\ J\ u\ J\ v$. If $f\ J\ g$ with $g = u_1 x_1 v_1$, $g\ J\ u_1\ J\ v_1$, then $fg\ H\ uv_1$ and $gf\ H\ u_1 v$.

Proposition 5.11: Let the transformation monoid of (X,A) be of threshold one. For every SCC K of (X,A), there is a subset $B = \alpha(K)$ such that, for every $g \in \langle A \rangle$:

a/ if $\alpha(g) \subseteq B$, then $Kg \subseteq K$;
b/ if $\alpha(g) \not\subseteq B$, then $Kg \cap K = \emptyset$;
c/ there is a $c \geq 1$ such that, for all g, $\alpha(g) = B$, one has $|Kg| = c$.

Proof: The demonstration for parts b/ and c/ is identical to the idempotent case. For part a/, let $a(g) \subseteq a(K)$; for any $x \in K$, there is a transformation f, $a(f) = a(K)$, such that $x \in Xf$. Observe that $f \; H \; f^j$ for all $j \geq 1$, and that there is an integer r for which $f^r = f$. Consider now the transformations $f = f \cdot f^{r-1}$ and $h = gf = gf \cdot f^{r-1}$, with $a(g) \subseteq a(K)$, and observe that $h \; J \; f$. Applying Fact 5.10 on the product fh, with $u = f$ and $v_1 = f^{r-1}$, one gets $fh = fgf \; H \; f \cdot f^{r-1} = f$, which implies $fgf \; R \; f$, and therefore there exists a transformation k such that $f = fgfk$. Hence, for any state $z \in X$ for which $zf \in K$, one has $zfg \in K$ as well. □

The questions relative to the Membership Problem in this class would therefore be:
1/ Whether it is possible to break the membership test into a test inside an idempotent monoid and a test inside a group of permutations. It is not excluded that there actually be more that one of either type of test: for example, if there definitely is a group-membership test to be done to see whether the action of f on Xf, which is that of a permutation, is compatible with membership in the monoid, there could also be other group membership tests involved.
2/ If this decomposition is feasible, whether it does not lead to an exponential or intractable complexity. In the commutative case (the algorithm of Section 3.1), both the aperiodic part and the stabilizer (the Abelian group) were generated by subsets of the initial generating set; this is not likely to remain the case: the decomposition might lead to instances with an exponential number of generators, or it might involve by itself some intractable computations.
3/ There remains also the problem of identifying a monoid of threshold one: the Class Test for this pseudovariety is still an open problem.

VI Transformation semigroups

The previous chapters have been concerned with transformation monoids, instead of semigroups. Although the difference seems negligible when it comes to testing membership in a given semigroup, it becomes significant when the discussion involves pseudovarieties.

This difference comes from the existence of classes which differ when defined in terms of semigroups, and not when defined in terms of monoids. This can be seen by observing that pseudovarieties are characterized with defining identities, which must be satisfied by every element of the semigroup, including the element 1 in the case of a monoid; introducing 1 in the identity may then lead to a condition on the other elements which is more stringent than desired. For instance, consider the idempotent pseudovariety of semigroups N_3, defined by the identities $g^2 = g$ and $fghk = fhgk$ [Fen,Ger]. If S is a monoid, then the second identity must be satisfied in particular when $f = k = 1$, which implies $gh = hg$ for all $g, h \in S$; in other words, this means that all monoids belonging to N_3 must be commutative, and therefore belong to the class J_1.

This discrepancy does not extend to all types of classes, however: all pseudovarieties of commutative semigroups and of monoids coincide, and the same applies to the classes discussed in chapter Five. There remain two cases where this makes a difference. First, the results of chapter Four have to be revisited, since the lattice of idempotent semigroup pseudovarieties is much richer than with monoids. Second, there exist classes of threshold-t semigroups which do not have to contain all the threshold-t commutative semigroups, which means that Corollary 3.13 does not apply to them, and that there may therefore exist classes of non-idempotent aperiodic semigroups where the Membership Problem is not *NP*-hard.

It is also worth noticing that an operation done at some points in this thesis, namely ignoring generators and working in the semiautomaton (X,B) instead of the original (X,A), with $B \subset A$, corresponds to applying a homomorphism such that $g \to g$ if $g \in \langle B \rangle$ and $g \to 1$ if $g \notin \langle B \rangle$. This implies that $\langle B \rangle$ must be a monoid.

Hence, the choice of writing this thesis in terms of monoids meant working with a simpler lattice of pseudovarieties, and allowed to reason in terms of ignoring characters and working in a subgraph of the original semiautomaton, an operation conceptually convenient, but which can be done without.

6.1 Idempotent semigroups

Idempotent pseudovarieties were originally studied in terms of semigroups [Ger,Fen]; the lattice of monoid classes used in chapter Four has been obtained by applying a reasoning similar to the one used above for the class $\mathbf{N}_3$. The full lattice for semigroups is represented in Figure 6.1, where the dotted lines are drawn between those pseudovarieties which collapse together upon passing from semigroups to monoids. Two new class names are introduced, $\mathbf{M}_2$ and $\mathbf{N}_3$, while the other labels indicate the pseudovarieties of semigroups whose definitions coincide with those used in chapter Four, and for which the results proved in that chapter remain valid.

The class $\mathbf{M}_2$ is defined by the identities $g^2 = g$ and $ghg = g$. Replacing g by 1 in the latter shows that all the monoids of $\mathbf{M}_2$ belong to $\mathbf{J}_1$. It can also be seen that a semigroup of this class coincides with the set $\{ ab \mid a,b \in A \}$, so that both the Class Test and the Membership Problem have an easy solution.
Already discussed in the introduction to this chapter is the class $\mathbf{N}_3$, defined by the congruence $\approx_3$.

Definition: For all $w_1, w_2 \in A^+$,

$$w_1 \approx_3 w_2 \text{ iff } \alpha(w_1) = \alpha(w_2) \text{ and } w_1 = au_1b \text{ and } w_2 = au_2b$$

for some $a,b \in A$ which do not have to be different. There are no further restrictions on u_1 and u_2, as long as they allow w_1 and w_2 to satisfy to the first condition. This is equivalent to the defining identity $fghk = fhgk$.

Proposition 6.1: The Class Test for $\mathbf{N}_3$ belongs to $NC^1/O(mn^3)$.

Proof: The demonstration consists in showing first that testing for the condition $fghk = fhgk$ on the generators suffices. Let $|g| = 1$, let $h = a_1 \cdots a_m$, and notice that f and k do not move, and play a role only through their last and first character, respectively, so that no assumption has to be made on them, other that they differ from the identity. Then

$$fghk = fga_1 \cdots a_m k = fga_1a_2 \cdots a_m k = fa_1ga_2 \cdots a_m k$$
$$\cdots = fa_1a_2 \cdots a_m g\ k = fhgk,$$

by operating a sequence of commutations on single characters. For $|g| > 1$, notice that g can be transferred to the right character by character. The first step of the algorithm consists therefore in testing whether $abcd = acbd$ for every $a,b,c,d \in A$, which is $NC^1 / O(mn^4)$.
The sequential upper bound can be brought down to $O(mn^3)$ by verifying instead the two conditions $abcb = acb$ and $abac = abc$. If they are satisfied by all the generators, then one has

$$abcd = abcbcd = acbcd = acbd$$

by first applying idempotency, then using each of the two identities.
The condition $abcd = acbd$ does not imply idempotency, however, so that this property has to be verified separately. It is claimed that testing $a^2 = a$, $abcb = acb$, and $abac = abc$ suffices. Indeed, if the generators satisfy this, then any element of the semigroup with expression awb, where $a, b \in A$, $w \in A^*$, is idempotent: one gets

$$awbawb = awabwb = aawbwb = aawwbb = awwb$$

by commuting w across a and b character by character. The inside of $awwb$ can then be simplified in a similar fashion. Testing the n generators for idempotency is $NC^1/O(mn)$. □

Theorem 6.2: The Membership Problem in N_3 semigroups is in $NC^2 / O(mn)$.

Proof: From the definition of $\approx_3$, it can be seen that the following algorithm tests for membership in an N_3 semigroup.

Algorithm

1. Compute the maximal alphabet $\alpha(f)$;
2. Find the first character, $a \in \alpha(f)$ such that $af = f$;
3. Find the last character, $b \in \alpha(f)$ such that $f = fb$;
4. Compute the product g of all generators of $\alpha(f)$, in any order;
5. $f \in \langle A \rangle$ iff $f = agb$.

Analysis: Step 1 is $NC^1 / O(mn)$ by Proposition 4.10; Steps 2,3, and 5 are trivially $NC^1 / O(mn)$; Step 4 is $NC^2 / O(mn)$ by Fact 2.6. □

There is therefore a unique largest class of idempotent semigroups for which the Membership Problem belongs to FL, a result identical to the one obtained for monoids, where J_1 played this role.

Further above, it can be seen that the class N_4 remains the largest one where the Membership Problem is not NP-hard, since the monoid constructed for Theorem 4.25 belongs to the pseudovariety of semigroups **XR**, one of the first two classes not included in N_4. It is important to notice, however, that N_4 is no longer the largest class of aperiodic semigroups in which the Problem is not NP-hard, since Corollary 3.13 holds only for monoids. Indeed, in the next section, pseudovarieties of non-idempotent semigroups are discussed, where membership can be tested in polynomial time.

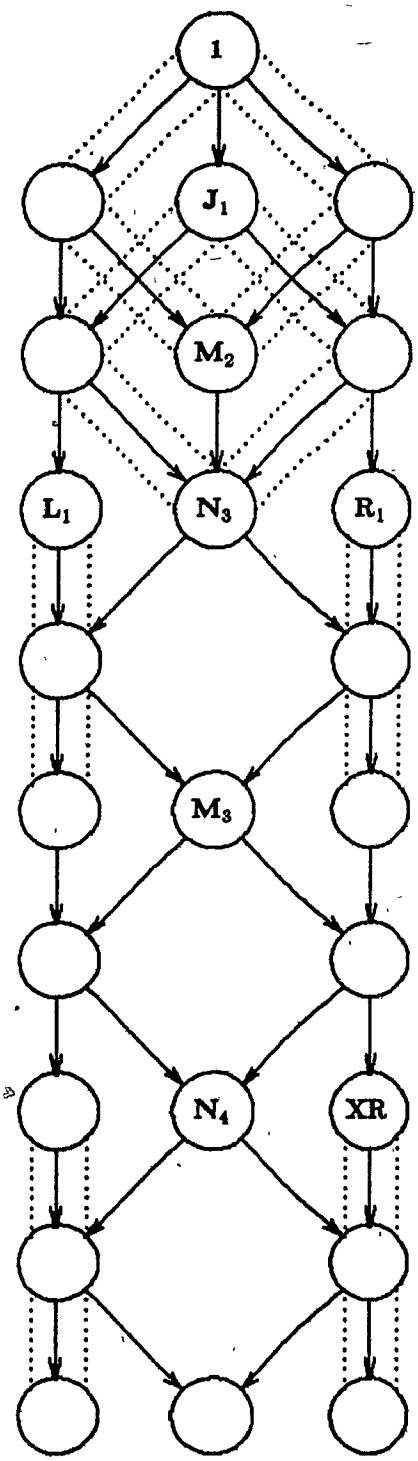

FIGURE 6.1: Pseudovarieties of idempotent semigroups

6.2 Non-idempotent aperiodic semigroups

This section discusses cases of pseudovarieties of aperiodic semigroups of threshold $t > 1$, which do not include the class of commutative aperiodic semigroups of threshold t, and in which the *NP*-hardness result of chapter Three therefore does not apply. The first example is the pseudovariety **Nil** of nilpotent semigroups, included in the class **J** of J-trivial semigroups [Eil].

Definition: The class $\mathbf{Nil}_k$ is defined by the identities $g_1 \cdots g_k = h_1 \cdots h_k$. This means that all the words of length k or more are expressions of the same semigroup element. The pseudovariety **Nil** is the union of all classes $\mathbf{Nil}_k$, $k \geq 1$.

In terms of transformations, this definition has the following equivalent.

Fact 6.3: The transformation semigroup of semiautomaton (X,A) belongs to the class $\mathbf{Nil}_k$ **iff** the following holds:

-the semiautomaton is acyclic;
-every connected component of (X,A) is dominated by a unique sink state;
-there are no trivial loops elsewhere than on the sinks;
-for any state, all the directed paths of length k starting from this state lead to a sink state.

Proof: The first two conditions express that the semigroup is J-trivial (Fact 5.1). For the next two, the *(if)* part is trivial, and the *(only if)* part is shown as follows. If $\langle A \rangle \in \mathbf{Nil}_k$, then for every state x and words $v, w \in A^+$ with $|w| = k$, one has $x\phi(wv) = x\phi(w)$. If $x\phi(w)$ is not a sink, which means a loop for $\phi(v)$ on this state, then the acyclicity of the graph implies the existence of a word $u \in A^+$ for which $x\phi(wu)$ is a sink. But then $\phi(w) \neq \phi(wu)$ with $|w| = k$, a contradiction with $\langle A \rangle \in \mathbf{Nil}_k$. □

Notice that the fourth condition is the only one to mention the parameter k, so that the first three suffice to determine membership in **Nil**. Observe also that this proposition implies an upper bound on k: if $|X| = m$ and $\langle A \rangle \in \mathbf{Nil}$, then $\langle A \rangle \in \mathbf{Nil}_{m-1}$. The Class Test in $\mathbf{Nil}_k$ is done as follows.

Algorithm

1. Test whether (X,A) is acyclic;
2. find the sinks: states y such that $ya = y$ for all $a \in A$;
3. verify that there are no trivial loops on non-sink states;
4. partition (X,A) into connected components and verify that each contains exactly one sink;

5. compute the longest path length between two states, and verify that it is at most k.

Proposition 6.4: The Class Test in $\mathbf{Nil}_k$ belongs to NC^2 / $O(mn)$.

Proof: The validity of the algorithm is immediate by Fact 6.3. Step 1 is $NC^2/O(mn)$, while Steps 2 and 3 are $NC^1/O(mn)$. In Step 4, once the partition into CCs has been done, it suffices to verify that there are as many connected components as there are sinks: by acyclicity of the graph, each CC must contain at least one sink. This step is therefore $NC^2/O(mn)$. Step 5 is done in sequential using a breadth-first search starting from the sinks and working against the direction of the edges: $O(mn)$. In parallel, the test works as follows. Let a CC K contain sink z, and assume without loss of generality that $k = 2^q$.

For every $x \in$ K in parallel, let $L(x) = \{ xa \mid a \in A \}$;
for $p = 0$ to q do
    for every $x \in$ K in parallel, let $M(x) = \bigcup_{y \in L(x)} L(y)$;
    for every $x \in$ K, let $L(x) = M(x)$;
if $L(x) = \{z\}$ for every x, then return *True*, else *False*.

Since each step is NC^1 and k is $O(m)$, this gives NC^2. □

Theorem 6.5: The Membership Problem in $\mathbf{Nil}_k$ is feasible in polynomial time whenever k is a constant; it is NP-complete in the general case.

Proof: Observe that every element of a semigroup belonging to $\mathbf{Nil}_k$ has an expression of length at most k; with $|A| = n$, this means for the semigroup a cardinality of $O(n^k)$, so that membership can be tested by a brute-force enumeration of all the elements: $NC^1/O(mn^k)$.

For the second part of the theorem, observe first that the Membership Problem is in NP, since Nil $\subseteq$ J. The proof is completed by a reduction from the 'minimum cover' problem [GJ]. An instance of this problem is a set T, a collection U of subsets of T, and an integer $m \leq |T|$; the question is to decide whether there is a collection U' of m or less elements of U such that $\bigcup_{u \in U'} u = T$.

First, reduce trivially to a test for the existence of a U' of size exactly m.

Next, construct the semiautomaton (X,A) as follows. For each $u \in U$, define a generator $a \in$ A. For each element t_i of T, build a connected component

$$K_i = \{ x_i, p_1, \cdots, p_m, q_1, \cdots, q_m \}$$

and partition A into two subsets $B_i = \{ a \in A \mid t_i \in u \}$ and $C_i = A - B_i$.

The generators act on K_i as follows:

-if $a \in B_i$, let $x_i a = q_1$; otherwise, let $x_i a = p_1$;

-for $1 \leq r \leq m-1$, let $p_r\, a = q_{r+1}$ if $a \in B_i$, and $p_r\, a = p_{r+1}$ if $a \in C_i$; meanwhile, let $q_r\, a = q_{r+1}$ for all $a \in A$;

-finally, for all $a \in A$, $p_m\, a = q_m\, a = p_m$.

Define the test-transformation f as: $x_i f = q_m$, and $p_r f = q_r f = p_m$ for all the other states p_r, q_r of the connected component K_i.

A connected component K_i is represented on Figure 6.2. In this picture, the arrows are labelled with the set of those generators whose action they correspond to.

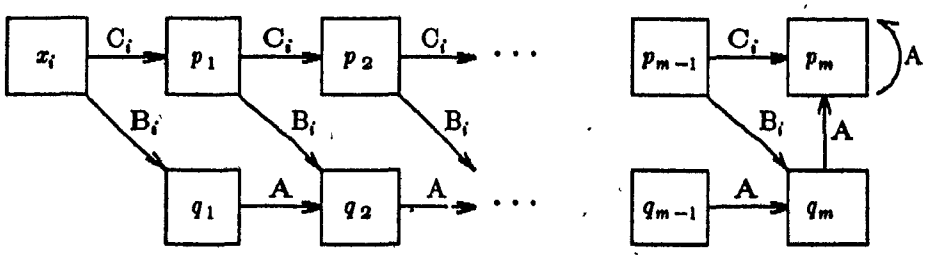

FIGURE 6.2: Connected component for the proof of Theorem 6.5

Proposition 6.6: The transformation semigroup of this semiautomaton belongs to the class $\mathbf{Nil}_k$, where $k = m+1$. □

Lemma 6.7: $f \in \langle A \rangle$ **iff** the instance of 'minimum cover' admits a solution.

Proof: *(if)* Let a solution of 'minimum cover' be a list $u_1, \cdots, u_l$ of elements of U, with $l \leq m$. For every $t_i \in T$, this implies that at least one of the subsets in the list, say u, contains t_i, which means that the corresponding generator a belongs to B_i. It can then be verified that $x_i\, a_1 \cdots a_l = q_l$. If $l < m$, then append enough generators to give an expression of length m: let $g = a_1 \cdots a_l\, b_{l+1} \cdots b_m$, with no conditions on the b_i's, and verify that g acts as f on every connected component of the semiautomaton.

(only if) Let $f \in \langle A \rangle$, with expression $a_1 \cdots a_l$. The condition $q_1 f = p_m$ implies $l = m$, while $x_i f = q_m$ means that at least one of $a_1, \cdots, a_l$ must belong to B_i, so that element t_i of T is covered. Therefore, $u_1, \cdots, u_m$ is a list of m elements of U which cover T. □

The discussion continues with two more pseudovarieties, which both contain the class **Nil**. These are **D**, associated with definite languages [Eil,PRS], and its dual $\mathbf{D}^R$.

Definition: The class $\mathbf{D}_k$ is defined by the identity $x_1 \cdots x_k = y\, x_1 \cdots x_k$. In other words, for all expressions $v, w \in A^+$ such that $|w| = k$, one has $\phi(vw) = \phi(w)$. The pseudovariety **D** is the union of all classes $\mathbf{D}_k$, $k \geq 1$; it is itself included in the class **L** of all L-trivial semigroups.

Considered first is the dual class $\mathbf{D}^R$, included in the class R of R-trivial semigroups. The characterization of this pseudovariety is analogous to the one of Nil, and is demonstrated in a similar fashion.

Fact 6.8: The transformation semigroup of semiautomaton (X,A) belongs to the class $\mathbf{D}_k^R$ **iff** the following holds:

-the semiautomaton is acyclic;

-there are no trivial loops elsewhere than on the sinks;

-for any state, all the directed paths of length k starting from this state lead to a sink state.

Fact 6.9: The Class Test in $\mathbf{D}_k^R$ belongs to NC^2 / $O(mn)$. □

Observe that the acyclicity condition implies that, if the transformation semigroup of a semiautomaton with m states belongs to $\mathbf{D}^R$, then it is in $\mathbf{D}_k^R$, for some $k < m$. A result analogous to Theorem 6.6 can therefore be demonstrated for $\mathbf{D}_k^R$.

Theorem 6.10: The Membership Problem in $\mathbf{D}_k^R$ is feasible in polynomial time whenever k is a constant; it is NP-complete in the general case. □

Similar results can be demonstrated for the class D. First, the equivalent of Theorem 6.10 is proved directly from the inclusion of Nil in D, and the following fact.

Fact 6.11: [PRS] If the transformation semigroup of a semiautomaton with m states belongs to D, then it is in $\mathbf{D}_k$, for some $k < m$.

The Class Test in D has been done in polynomial time [PRS]. A test in parallel can be done in NC^2, based on the following observation.

Fact 6.12: [PRS] The transformation semigroup of (X,A) belongs to $\mathbf{D}_k$ **iff** for every word $w \in A^*$ of length k, every connected component of the semiautomaton is mapped by $\phi(w)$ onto a unique state.

Proposition 6.13: The Class Test in D belongs to NC^2.

Proof: For every two states x, y, define the language

$$L(x,y) = \{ w \in A^* \mid x\phi(w) = y \text{ and } |w| = k \};$$

the condition that $L(x,y) = L(z,y)$ for every state z belonging to the same connected component as x is equivalent to having $\langle A \rangle \in \mathbf{D}_k$. Observe that, since $L(x,y) = \varnothing$ when y is not in the CC of x, the test can be restricted to states belonging to the same connected component.

The algorithm works as follows: for every connected component $\{ x_1, \cdots, x_j \}$, a set

$$Y = \{ x_{1,0}, \cdots, x_{1,k}, x_{2,0}, \cdots, x_{j,k} \}$$

is defined, on which the semiautomaton (Y,A) is built, as follows. For all $i \leq j$ and $l < k$, every $a \in A$, if $x_i \, a = x_k$ in the original semiautomaton, then let $x_{i,l} \cdot a = x_{k,(l+1)}$; next, let $x_{i,k} \, a = x_{i,k}$ for all $a \in A$.
For every $1 \leq h, i \leq j$, the deterministic finite automaton built on (Y,A) with $x_{i,0}$ as initial state and $x_{h,h}$ as final state accepts the language $L_{ih} = L(x_i, x_h) \cdot A^*$. For any other $g \neq i$, one has $L_{gh} = L_{ih}$ iff $L(x_i, x_h) = L(x_g, x_h)$. The characterization reduces therefore to testing, for every $1 \leq g, h \leq j$, whether $L_{1h} = L_{gh}$.

Parallel algorithm

1. Partition the semiautomaton into connected components; in each of them, distinguish a state x_1.
2. For every connected component in parallel, do
   2.1. construct the semiautomaton (Y,A);
   2.2. for $2 \leq g \leq j$ and $1 \leq h \leq j$, test by mutual inclusion whether $L_{1h} = L_{gh}$.

Analysis: Step 1 is NC^2. Step 2 is a NC^1 reduction to the test for inclusion of regular languages (Fact 2.7), which is also NC^2. □

This chapter is completed with a discussion of the class $\hat{\mathbf{D}}_k$ which is associated with generalized-definite languages [Eil].

Definition: The class $\hat{\mathbf{D}}_k$ is defined by the identity

$$x_1 \cdots x_k \, y \; z_1 \cdots z_k = x_1 \cdots x_k \, z_1 \cdots z_k .$$

In other words, for all expressions $u, v, w \in A^*$ such that $|u| = |w| = k$, one has $\phi(uvw) = \phi(uw)$. This class includes both $\mathbf{D}_k$ and $\mathbf{D}_k^R$.

In terms of transformation semigroups, this definition translates into the following.

Fact 6.14: The transformation semigroup of (X,A) belongs to $\hat{\mathbf{D}}$ iff
- the transformation semigroup of (Y,A) is in $\mathbf{D}^R$, where Y is the original set, in which each SCC has been replaced by a sink state, and
- the transformation semigroup of (Z,A) is in $\mathbf{D}$, where Z is the set X, minus the states from which more than one SCC is accessible.

Proof: (only if) Let <A> belong to $\hat{\mathbf{D}}_k$, for some $k \geq 1$, and consider the word uv, with $|u| = |v| = k$. Then, for any state x and for any $w \in A^*$, one has $x \, \phi(uvwv) = \phi(uv)$, which implies that $x \, \phi(uv)$ and $x \, \phi(uvw)$ belong to the same SCC By a reasoning analogous to the one used for Fact 6.3, this implies that all the

strongly connected components of the semiautomaton must be maximal for A. The first condition comes as a direct consequence of this. The second condition is proved by considering that a state $x \in Z$ will be mapped by all $\phi(uv)$, u and v of length k, into the same SCC of (Z,A). The condition $\phi(uwv) = \phi(uv)$ implies then that the image of the SCC by $\phi(v)$ consists of a single state. Therefore, the transformation semigroup of (Z,A) is in D_{2k}.

(if) The second condition implies the existence of a parameter l such that, for every word u of length l, every state is mapped by $\phi(u)$ onto a sink of (Y,A), that is, into an SCC of the original semiautomaton. Meanwhile, the third condition implies the existence of a j such that, for every word v of length j, $\phi(v)$ maps each SCC onto a single state. As a consequence, the semigroup belongs to $\hat{D}_k$, where k is the larger of j and l. □

Observe that both parameters j and l are at most m, where m is the cardinality of X. This upper bound allows to state a result identical to Theorem 6.5.

Theorem 6.15: The Membership Problem in $\hat{D}_k$ is feasible in polynomial time for k constant; it is *NP*-complete in the general case. □

VII Related Problems

7.1 Intersection of regular languages

It has been argued in chapter One that the complexity of the Membership Problem does not rise as fast as that of many common computational problems on transformation monoids. Here are three such problems; the Membership Problem reduces to each one of them.

Intersection of two regular languages:

given n deterministic finite automata, decide whether the intersection of the languages they accept is non-empty.

Reduction: Testing membership of f in (X,A) can be transformed into testing whether the languages L_x, $x \in X$, have a non-empty intersection, where L_x is the language accepted by the deterministic finite automaton built from (X,A) by marking states x as initial, and xf as final.

Order of a monoid:

given a transformation monoid, compute the number of its elements.

Reduction: Transformation f belongs to $\langle A \rangle$ **iff** monoids $\langle A \rangle$ and $\langle A \cup \{f\} \rangle$ have the same cardinality.

Monoid isomorphism:

given two semiautomata, decide whether their transformation monoids are isomorphic.

Reduction: Transformation f belongs to $\langle A \rangle$ **iff** monoids $\langle A \rangle$ and $\langle A \cup \{f\} \rangle$ are isomorphic.

The latter two problems have been shown to be in NC^3 for Abelian permutation groups [McC]; finding the order of an arbitrary permutation group is also in NC [BLS].

The first of these problems is now discussed in more detail, as it was of instrumental use in Kozen's proof of the *PSpace*-completeness of the Membership Problem [Koz]. In the general case, both problems have the same complexity. This is also the case everywhere where the Membership Problem is *NP*-complete, since an element of the intersection of the languages can be guessed in the same way as an expression for the test-transformation in the Membership Problem. Here are two cases, however, where the language intersection problem is strictly harder.

Fact 7.1: Testing for intersection of regular languages whose syntactic monoid is an Abelian permutation group is *NP*-complete.

Proof: Reduce from 'one-in-three 3SAT', with no negated variables involved [GJ]. For every clause (u_1,u_2,u_3), construct an eight-state automaton over alphabet $\{a_1,a_2,a_3\}$, as follows. Label a state (the initial state) as *FFF*, and $FFFa_1$ as *TFF*, $FFFa_2$ as *FTF*, $FFFa_1a_2$ as *TTF*, and so on, so that there is a state for every combination of truth values for the three variables, and a generator maps a state to the one where the value of the corresponding Boolean variable has been flipped, so that each generator is a permutation of period two. It can be verified that <A> is Abelian. Label states *TFF*, *FTF*, and *FFT* as final, and take the following interpretation: given an expression, start with all three Boolean variables at *False*, and flip the value of variable u_i each time character a_i is read. Variable u_i will therefore have value *True* iff generator a_i appears an odd number of times in the word being read. An automaton built according to this construction is represented in Figure 7.1. □

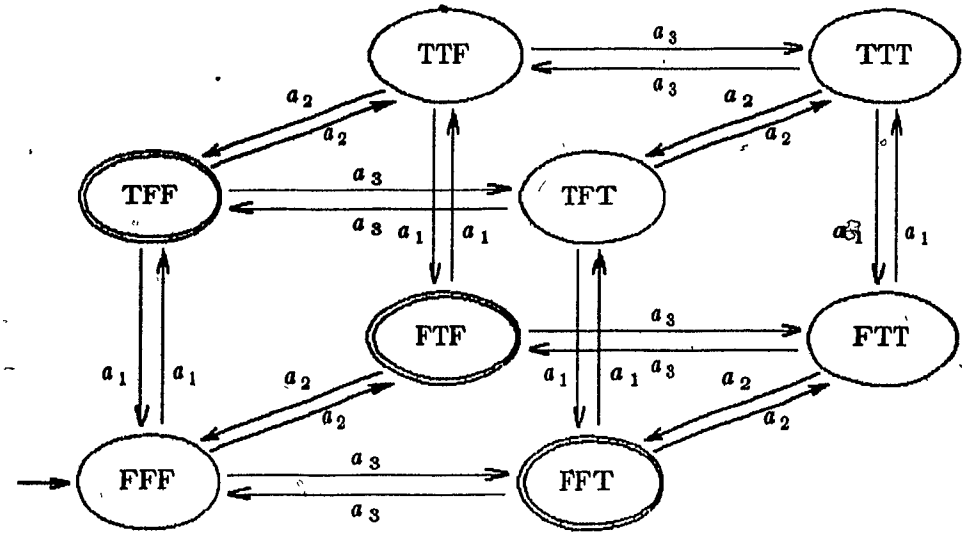

FIGURE 7.1: Automaton built for the proof of Fact 7.1

Fact 7.2: Testing for intersection of regular languages whose syntactic monoid is idempotent and commutative, that is, belongs to J_1, is *NP*-complete.

Proof: The reduction is from the same problem. For every variable u_i, define a generator a_i. For each clause, define an automaton with five states, as follows. The initial state is *FFF*. Define $FFFa_1$ as *TFF*, $FFFa_2$ as *FTF*, and $FFFa_3$ as *FFT*. These three states are final. For everyone of them, the generators which did not map *FFF* to it will map it to the fifth state, which is a sink. For instance: $TFFa_2 = TFFa_3 = SINK$. With the interpretation that all variables are initially set

to *False*, and that variable u_i is set to *True* as soon as generator a_i appears at least once, the rest of the proof is straightforward. An automaton built according to this construction is represented in Figure 7.2. Notice that trivial loops are not represented on this diagram: one has $TFFa_1 = TFF$, $FTFa_2 = FTF$, and $FFTa_3 = FFT$. □

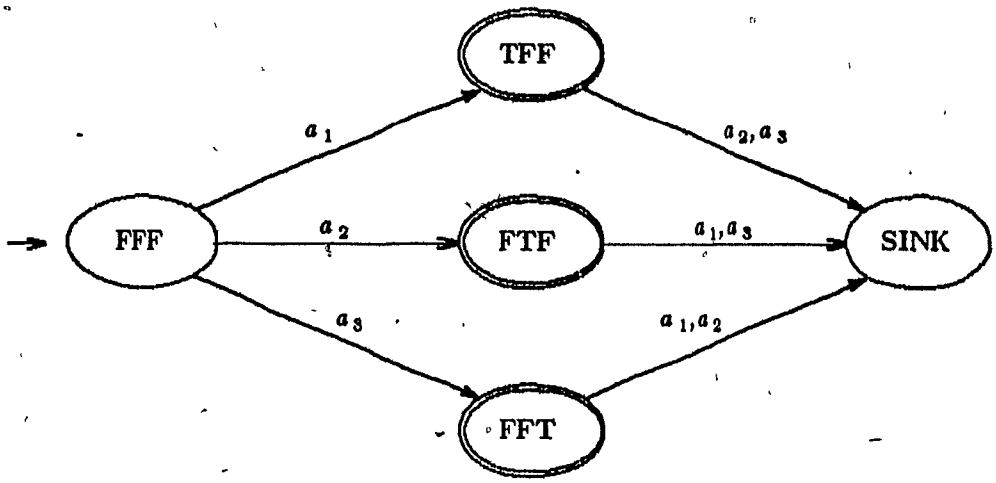

FIGURE 7.2: Automaton built for the proof of Fact 7.2

7.2 Reduction of finite automata

A computational problem related to the topic of this thesis consists, given a deterministic finite automaton M, in building another automaton which accepts the same language as M with a minimal number of states. This problem is already known to have a sequential solution [LP,HU]; this section presents an algorithm which does this in parallel.

The sequential algorithm is based on the idea of merging together states that could not be distinguished between one another. An equivalent definition for indistinguishability is introduced, which can be tested for in parallel. Define first a deterministic finite automaton as $M = (X, A, \delta, q_0, F)$.

Lemma 7.3: Let M be as above. Two states q_i and q_j are indistinguishable, in the sense defined in [LP,HU], **iff** the automata M and M′ accept exactly the same language, where $M' = (X, A, \delta', q_0, F)$, with, for $a \in A$, $\delta'(q_i, a) = \delta(q_j, a)$, $\delta'(q_j, a) = \delta(q_i, a)$, and $\delta'(q, a) = \delta(q, a)$ otherwise. In other words, indistinguishable states are interchangeable.

Proof: Let w be a word in A^*, and define

$$\Pi_M(q,w) = \begin{cases} 1 & \text{if } (q,w) \overset{*}{\vdash} (p,e) \text{ and } p \in F \\ 0 & \text{if } (q,w) \overset{*}{\vdash} (p,e) \text{ and } p \notin F \end{cases}$$

States q_i and q_j are defined to be indistinguishable **iff** for every word $w \in A^*$, reading w on M starting at q_i will lead to a final state **iff** reading w from q_j also leads to a final state, that is, in this notation: $\Pi_M(q_i,w) = \Pi_M(q_j,w)$. It is claimed that this is equivalent to having $\Pi_M(q_i,w) = \Pi_{M'}(q_i,w)$.

The proof is by induction on the number of occurences of q_i or q_j on the path travelled along from q_i (resp. q_j) while reading w. With no such occurence, one has $\Pi_{M'}(q_i,w) = \Pi_M(q_j,w)$, since $\delta'(q_i,a) = \delta(q_j,a)$, and the rest is unchanged. Similarly, one has $\Pi_{M'}(q_j,w) = \Pi_M(q_i,w)$, and therefore

$$\Pi_{M'}(q_j,w) = \Pi_M(q_i,w) = \Pi_{M'}(q_i,w) = \Pi_M(q_j,w),$$

whenever q_i and q_j cannot be distinguished by the word w.

With k occurences of q_i or q_j, let the path be decomposed into

$$(q_i,w) \overset{*}{\vdash} (q_l,u) \overset{*}{\vdash} (p,e),$$

where q_l is the first occurence of q_i or q_j encountered along the path, and u is the appropriate suffix of w. Then there are $k-1$ such occurences between q_l and p, and, by the induction hypothesis, this implies

$$\Pi_{M'}(q_i,w) = \Pi_{M'}(q_l,u) = \Pi_M(q_l,u) = \Pi_M(q_j,w);$$

Similarly, $\Pi_{M'}(q_j,w) = \Pi_M(q_i,w)$, when starting from q_j. Hence, $\Pi_{M'}(q_i,w) = \Pi_M(q_i,w)$ whenever $\Pi_M(q_j,w) = \Pi_M(q_i,w)$. □

Now let q_i and q_j be indistinguishable. If the path traversed while reading w doesn't pass through either q_i or q_j, then $w \in L(M)$ **iff** $w \in L(M')$; else

$$w \in L(M) \text{ iff } (q_0,w) \overset{*}{\vdash} (q_l,u) \overset{*}{\vdash} (p,e),$$

where $p \in F$ and q_l is the first occurence of q_i or q_j met along the path. This is equivalent to having

$$(q_0,w) \overset{*}{\vdash} (q_l,u) \text{ and } \Pi_M(q_l,u) = 1,$$

that is,

$$(q_0,w) \overset{*}{\vdash} (q_l,u) \text{ and } \Pi_{M'}(q_l,u) = 1,$$

which means $w \in L(M)$ **iff** $w \in L(M')$.

Parallel algorithm

1. For every pair of states $\{ q_i, q_j \}$ in parallel, construct the automaton M', as specified in the lemma.
2. For every such automaton M', test whether $L(M) = L(M')$.
3. Construct a graph with n nodes, each labelled with a state of M. Draw an edge

between nodes q_i and q_j whenever the corresponding states are indistinguishable. Each connected component of the resulting graph corresponds to a state of the minimal automaton.

Analysis: At Step 1, each of the $\frac{1}{2}n(n-1)$ automata M' is built in constant depth. Step 2 is done by testing mutual inclusion of the languages, using a variant of the algorithm presented in Fact 2.7: NC^2. Step 3 consists in identifying the CCs in a graph, a problem also in NC^2. The construction of the transition table for the minimal automaton is done in constant depth in a straightforward fashion.

Theorem 7.4: A deterministic finite automaton can be minimized in NC^2. □

Bibliography

[AHU1] A.Aho, J.Hopcroft, J.Ullman *The Design and Analysis of Computer Algorithms,* Addison-Wesley (1974).

[AHU2] A.Aho, J.Hopcroft, J.Ullman *Data structures and algorithms,* Addison-Wesley (1983).

[BLS] L.Babai, E.M.Luks, A.Seress Permutation groups in *NC*, *Proc. 19th ACM STOC Symposium* (1987) pp. 409-420.

[Be] M.Beaudry Testing Membership in Commutative Transformation Semigroups, in *Automata, Languages and Programming,* Lecture notes on Computer Science **267**, Springer-Verlag (1987) pp. 542-549.

[Bar] D.Barrington *Bounded-width Branching Programs,* Ph.D. Thesis, M.I.T. (1986).

[BT] D.Barrington, D.Thérien Finite monoids and the Fine Structure of NC^1, *Proc. 19th ACM STOC Symposium* (1987) pp. 101-109.

[BP] G.Bilardi, F.P.Preparata Size-time complexity of Boolean networks for prefix computation, *Proc. 19th ACM STOC Symposium* (1987) pp. 436-442.

[Bor] A.Borodin On relating time and space to size and depth, *SIAM J. Comp.* **6** (1977) pp. 733-744.

[CFL] A.K.Chandra, S.Fortune, R.Lipton Unbounded fan-in circuits and associative functions, *J. Comput. System Sci.* **30** (1985) pp. 222-234.

[CP] A.H.Clifford, G.B.Preston *The Algebraic Theory of Semigroups, Volume 1, 2nd. ed.* Amer. Math. Soc. Mathematical Surveys no.7 (1964).

[Cook] S.A.Cook The Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control* **64** (1985) pp. 2-22.

[CW] D.Coppersmith, S.Winograd Matrix Multiplication via Arithmetic Progressions, *Proc. 19th ACM STOC Symposium* (1987) pp. 1-6.

[Eil] S.Eilenberg *Automata, Languages and Machines, Vol. B,* Academic Press (1976).

[Fen] C.Fennemore All varieties of bands I,II, *Mathematische Nachrichten* **48** (1971) pp. 237-252 and 253-262.
C.Fennemore All varieties of bands, *Semigroup Forum* **1** (1970) pp. 172-177.

[Fich] F.Fich *Languages of R-Trivial and related Monoids,* MSc Thesis, University of Waterloo (1979).

[FHL] M.Furst, J.Hopcroft, E.M.Luks Polynomial-Time Algorithms for Permutation Groups, *Proc. 21st. IEEE FOCS Symposium* (1980) pp. 36-41.

[GJ] M.Garey, D.Johnson *Computers and Intractability, a Guide to the Theory of NP-Completeness,* Freeman (1979) .

[Ger] J.A.Gerhard The Lattice of Equational Classes of Idempotent Semigroups, *J.Algebra* **15** (1970) pp. 195-224.

[Gre] J.A.Green On the structure of Semigroups, *Ann. Math.* **54** (1951) pp. 163-172.

[GrR] J.A.Green, D.Rees Semigroups such that $x^r = x$, *Proc. Camb. Phil. Soc* **48** (1952) pp. 35-40.

[GBB] J.W.Grzymala-Busse, Z.Bavel Characterization of state-independent automata, *Theoretical Computer Science* **43** (1986) pp. 1-10.

[Hof] C.M.Hoffmann *Group-Theoretical Algorithms and Graph Isomorphism,* Lecture Notes in Computer Science **136** Springer-Verlag (1982).

[HU] J.Hopcroft, J.Ullmann *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley (1979).

[Jer] M.Jerrum A Compact Representation for Permutation Groups, *Proc. 23rd*

IEEE FOCS symposium (1982) pp. 126-133.

[Jon] N.D.Jones Space-bounded Reducibility among Combinatorial Problems, *J. Comput. System Sci.* **11** (1975) pp. 68-75.

[JL] N.D.Jones, W.T.Laaser Complete Problems for deterministic polynomial time, *Theoretical Computer Science* **3** (1977) pp. 105-117.

[Koz] D.Kozen Lower Bounds for natural proof systems, *Proc. 18th. IEEE FOCS symposium,* (1977) pp. 254-266.

[Lall] G.Lallement *Semigroups and combinatorial applications,* Addison-Wesley (1979).

[LP] H.Lewis, C.Papadimitriou *Elements of the Theory of Computation,* Prentice-Hall (1981).

[LMK] E.M.Luks, P.McKenzie Fast Parallel Computation with Permutation Groups, *Proc. 26th. IEEE FOCS symposium* (1985) pp. 505-517.

[McK] P.McKenzie *Parallel Complexity and Permutation Groups,* Ph.D. Thesis, University of Toronto (1984).

[McC] P.McKenzie, S.A.Cook The parallel complexity of the Abelian permutation group membership problem, *Proc. 24th. IEEE FOCS Symposium* (1983) pp.154-161.

[Mul] K.Mulmuley A Fast Parallel Algorithm To Compute The Rank Of A Matrix Over An Arbitrary Field, *Proc. 18th ACM STOC Symposium* (1986) pp. 338-339.

[Pan] V.Pan *How to Multiply Matrices Faster,* Lecture Notes on Computer Science **179**, Springer-Verlag (1984).

[PRS] M.Perles, M.O.Rabin, E.Shamir The theory of definite automata, *IEEE Trans. Elec. Comp.* **EC-12**, (1963) pp. 233-243.

[Pin] J.-E. Pin *Variétés de langages formels,* Masson (1984).

[Sch] M.P.Schützenberger On finite monoids having only trivial subgroups, *Information and Control* **8** (1965) pp. 190-194.

[Simon] I.Simon Piecewise testable events, in *Automata Theory and Formal Languages,* Lecture Notes in Computer Science **33**, Springer-Verlag (1975) pp. 214-322.

[Sims] C.Sims Computational methods in the study of permutation groups, in *Computational Problems in Abstract Algebra (John Leech, Ed.),* Pergamon Press (1970) pp. 169-183.

[Ste1] J.Stern Complexity of some Problems from the Theory of Automata, *Information and Control* **66** (1985) pp. 163-176.

[Ste2] J.Stern Characterization of some classes of regular events, *Theoretical Computer Science* **35** (1985) pp. 17-42.

[Wie] H.Wielandt *Finite Permutation Groups,* Academic Press (1964).

APPENDIX Abelian Group Membership in sequential

The sequential complexity of the Membership Problem in an Abelian group of permutations has not yet been studied; all the available results are valid for arbitrary groups [FHL,Sims]. In this Appendix, a translation into a sequential algorithm of the parallel algorithm of McKenzie and Cook [McC] is sketched. It is seen that the complexity thus obtained is dominated by the resolution of a system of linear congruences, a problem to which Abelian Group Membership is NC^1 equivalent [McC]. In what follows, it is assumed that the reader is familiar with McKenzie and Cook's parallel algorithm.

The algorithm consists first in computing a tentative expression f_i for f in every orbit (SCC) of (X,A). If $f \in \langle A \rangle$, then $f = f_i 1_i$, where 1_i is a permutation which acts as the identity on the orbit K_i. Finding f_i amounts to collecting the edge labels on any path from a state x of K_i to its image $xf = xf_i$ ($O(mn)$), and to identifying the conditions for a permutation to act on K_i as the identity. These conditions can be expressed as a system of linear congruences on the exponents of the generators, modulo the period of the group, which is the least common multiple of the periods of the generators.

Let B_i be the vector of exponents of the generators in a tentative expression f_i for f on the orbit K_i, and X_i a vector of exponents for an expression of 1_i. Let the matrix M^i define the conditions on X_i to represent an expression for the identity on K_i. Denote Y the vector of the exponents of the generators in the final expression of f. Then Y and X_i must satisfy to $Y \equiv B_i + M^i X_i \pmod{q}$, where q is the period of the group.

M^i can be given as a lower triangular matrix for which every column $(M^i)_j$ represents an expression of 1_i, with coefficients $M^i_{kj} = 0$ for all $k < j$. This is done as follows: for every generator a_j, find the smallest t such that a_j^t restricted on orbit K_i can be expressed in terms of generators $a_{j+1}, \cdots, a_n$. To do so, verify whether there is a path from some x to xa_j^t in the graph $(K_i, \{a_{j+1}, \cdots, a_n\})$, and keep the labels encountered on this path as an expression. Then one will have $M^i_{jj} = q_j - t$ where q_j is the period of a_j, and the entry M^i_{kj}, $k > j$ is the number of occurences of generator a_k in the expression thus computed. This implies an accessibility test for every generator and every possible value of t, $1 \leq t \leq r_i$, where r_i is the number of states in K_i. The accessibility test can actually be done

using a depth-first search, which gives $O(r_i^2 n^2)$ inside an orbit, $O(m^2 n^2)$ over the whole semiautomaton.

The sequential complexity will be dominated by the next step, which consists in deciding whether these systems have a solution. There are as many such $n \times n$ systems as there are orbits, the number of orbits being counted as $O(m)$, and they can be merged into a unique $O(mn) \times O(mn)$ system. The problem of solving a $d \times d$ system of linear congruences modulo q, whose complexity is denoted $SLC(d)$, has a complexity of at most $O(d^3)$, if arithmetic operations modulo q are counted for $O(1)$, and reduces in the case of q prime to the multiplication of matrices over a field [Pan], which is $O((d)^{2.376})$ [CW].