FAULT-TOLERANT COMPUTATION IN SEMIGROUPS AND SEMIRINGS

C. N. Hadjicostis

MIT Res. Lab. of Electronics
50 Vassar St
Cambridge, MA 02139
USA

G. C. Verghese

EECS Department, MIT
MIT Room 10-093
Cambridge, MA 02139
USA

Abstract

The traditional approach to fault-tolerant computation and signal processing has been via modular redundancy. Although universal and simple, modular redundancy is inherently expensive and inefficient. By exploiting particular structural features of a computation or an algorithm, arithmetic codes and recently developed Algorithm-Based Fault Tolerance techniques manage to offer more efficient fault coverage at the cost of narrower applicability and harder design. Previous work has shown that a variety of useful results and constructive procedures can be obtained when the computations take place in an abelian group. In this paper, we develop a systematic algebraic approach for computations occurring in semigroups or in higher semigroup-based algebraic structures, such as semirings. We thereby extend the previous framework to a more general setting that includes many non-linear signal processing applications.

1 Introduction

Systems designed with the ability to detect and, if possible, correct internal failures are called *fault-tolerant*. The design of such systems is motivated by applications that require high reliability, such as controllers of life-critical systems or applications that take place in a remote or hazardous environment. A necessary condition for a system to be fault-tolerant is that it exhibit *redundancy*, so that it is able to distinguish between valid and invalid states or, equivalently, between correct and incorrect results. Redundancy, however, is expensive and counter-intuitive to the traditional notion of system design. The success of a fault-tolerant system relies on making efficient use of hardware by adding redundancy in those parts of the system that are more liable to failures than others.

N-Modular redundancy, the traditional approach to fault tolerance, operates by replicating the computational system: N identical modules perform the exact same computation separately and in parallel. Their outputs are then compared and the final result is chosen based on what the majority of the modules decided. There exist a number of hybrid methods as well as a variety of approaches towards voting. Many examples of commercial applications of modular redundancy are referenced in [1].

While universally applicable and simple, modular redundancy is inherently expensive and inefficient. A more efficient, but harder to implement, approach towards fault tolerance in computation is the use of *arithmetic codes*. Arithmetic codes are a class of error correcting codes with properties that remain invariant under the operations of interest. They are typically used as shown in Figure 1. In order to provide fault tolerance to the computation of $g_1 \circ g_2 = r$, we first add redundancy to the representation of the data by using suitable and efficient encodings (mappings ϕ_1 and ϕ_2 in the figure[1]). Then, operation $\diamond$ is performed on the encoded data ($\diamond$ is not necessarily the same as the desired operation $\circ$ on the original data). The redundancy added to the data is subsequently used to perform error detection and correction. The result is finally decoded back to its original form through the use of the mapping σ.

This work has been supported in part by the USA Department of the Navy, Office of the Chief of Naval Research, contract number N00014-93-1-0686 as part of the Advanced Research Projects Agency's RASSP program.

[1]This situation can be generalized to p operants and, correspondingly, p encoding mappings ϕ_i for $1 \leq i \leq p$.

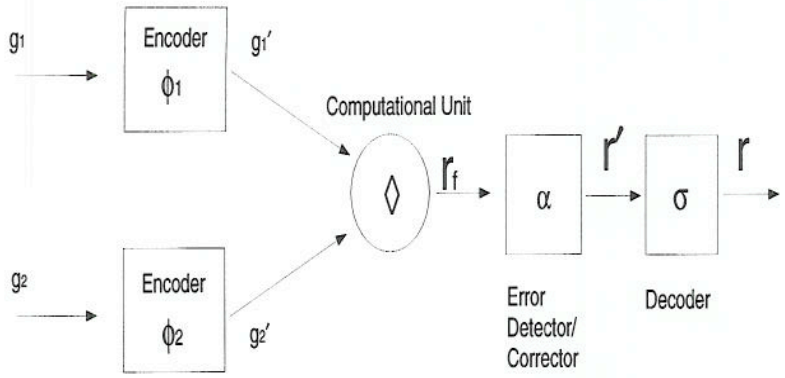

Figure 1: Fault tolerance through the use of arithmetic codes.

Highly involved arithmetic coding techniques are known as *Algorithm-Based Fault Tolerance* (ABFT) schemes. These schemes usually deal with real/complex arrays of data in multiprocessor concurrent systems. The term was introduced by J. Abraham and coworkers [3]-[7] in 1984. The classic example of ABFT is the protection of $N \times N$ matrix multiplication on a multiprocessor array, [3]. Since then, a variety of computationally intensive algorithms, such as other matrix arithmetic [3], [4], fault-tolerant FFT computational systems [6], A/D conversion and digital convolution [1] have been adapted to the requirements of ABFT.

As described in [5], there are three key steps involved in ABFT:

1. Encode the input data for the algorithm (just as in the general case of arithmetic coding).
2. Reformulate the algorithm so that it can operate on the encoded data and produce decodable results.
3. Distribute the computational tasks among different failure-prone parts of the system so that any errors occurring within these subsystems can be detected and, hopefully, corrected.

The most important step in both arithmetic coding and ABFT implementations is the detection of "exploitable structure" in an algorithm in a way that can provide efficient fault coverage. An important attempt to provide a systematic approach towards recognition and exploitation of such special structure was made in [1]. What is required in the framework of [1] is that the operation (or, more generally, the computational task) can be modeled as an *abelian group* operation. The framework extends naturally to other algebraic structures that have the underlying characteristics of an abelian group, such as rings, fields, modules and vector spaces. Therefore, a relatively extensive set of computational tasks (including the above mentioned ABFT schemes) can be modeled using this framework.

In this paper, we present the extension of the above results to a more general setting: the requirement that the computation occurs in an abelian group is relaxed to the less strict requirement of an underlying *abelian semigroup* structure. A framework for investigating such fault-tolerant systems in a mathematically rigorous and fruitful way is now available. Important results from group and semigroup theory can be directly applied in systems of interest that comply with our requirements. Moreover, the framework extends to higher algebraic systems with an underlying semigroup structure. We present one such extension, namely the extension to the semiring structure.

Detailed connection to actual hardware realizations and their failure modes is *not* addressed in this paper. There are many research issues that arise in making this hardware connection in a systematic and general way, but we leave these as topics for follow-on research.

2 Group Framework

Arithmetic codes and ABFT schemes have been constructed for particular algorithms, but a systematic and mathematically rigorous way of addressing the design of arithmetic codes for a general computational task has not yet been developed. A considerable step in this direction was taken in [1]. By concentrating on computational tasks that can be modeled as abelian group operations, one can impose sufficient structure

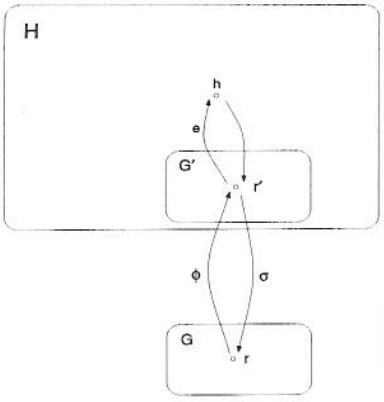

Figure 2: Fault tolerance in a computation using an abelian group homomorphism.

upon the computations to allow accurate characterization of the possible arithmetic codes and the form of redundancy that is needed.

In [1], the given group computation is protected through the use of an arithmetic coding scheme like the one shown in Figure 1. Redundancy is added to the operands before operating on them to allow for error detection and correction. In this setting, it can be shown that the encoding operations[2] ϕ_1 and ϕ_2 are forced to be the exact same *group homomorphism* that maps elements of the group G, in which the original computation is formulated, into a larger group H. We will denote this homomorphic mapping by ϕ (refer to Figure 2). The original operands, as well as the results of all valid computations, fall in a subgroup G' of H. Any result that falls outside of G' is invalid and is detected as erroneous. It may or may not be correctable, depending on the homomorphism that is being used. The error detection and correction properties are based on the structure of the cosets of G' in H. In the absence of errors, results lie in the zero coset (that is, in G' itself). Every detectable error forces the result of the computation into a nonzero coset, while every correctable error forces the result into a coset that is uniquely associated with that particular error.

The above results were extended in [1] to higher algebraic systems with an embedded group structure, such as rings, fields, and vector spaces. The framework is therefore extended enough to embrace a large variety of known arithmetic coding and ABFT schemes that were already developed in some other way.

One of the most important results in [1] is obtained for the special case of *separate* codes. These are codes in which redundancy is added through a separate "parity" channel. The group homomorphism for coding now maps G to $H = G \times T$, where T is the group of parity symbols. Therefore, finding a suitable homomorphism reduces to finding a homomorphism π such that $[g, \pi(g)]$ is the element of H corresponding to the operand g. Making the reasonable assumption that parity information is efficiently used, we are led to require that the homomorphism π be surjective (onto). The problem of finding all possible parity codings is thereby reduced to that of finding all epimorphisms (surjective homomorphisms) from G. By an important homomorphism theorem from group theory, these epimorphisms are isomorphic to the *canonical* epimorphisms, namely those that map G to its factor (or quotient) groups G/N, where N denotes a subgroup of G. Hence the problem of finding all possible parity codings reduces to that of finding all possible subgroups of G.

3 Extension to Semigroup-Based Computations

In this section, we extend the results for the group case to a less constrained setting in which we only require the computations to have an underlying *semigroup* structure. A semigroup $\mathcal{S} = [S, \circ]$ is just a set S that is closed under an associative binary operation $\circ$. Familiar examples of semigroups that are not groups are the set of positive integers under the operation of addition (or multiplication), the set of polynomials with

[2] The results are valid in the more general case, where we have p operants and p encoding mappings.

real coefficients under the operation of polynomial multiplication (or polynomial substitution), and the set of $N \times N$ matrices under matrix multiplication. More examples of semigroups and discussion on semigroup theory can be found in [8], [9].

In this paper, we only consider the case of *abelian* semigroups[3], although the results can be extended (in a non-trivial way, [2]) to the non-abelian case. We also assume that the semigroups under consideration have an identity element, i.e. that they are actually *monoids*; this involves no loss of generality, as an identity element can be trivially adjoined to a semigroup if it does not initially posses one. We will denote the identity element by $0_\circ$.

The model of a system that protects computation in an abelian monoid $\mathcal{S} = [S, \circ]$ is almost the same as the one for the group case (refer to Figure 1), the only difference being that the overall computation takes place in a semigroup (rather than in a group). Under some natural assumptions[4], it can be shown that all ϕ_i's must be the exact same semigroup homomorphism ϕ, and that $\sigma^{-1} = \phi$. It follows that redundancy is added to the original computation in $\mathcal{S} = [S, \circ]$ by mapping it to a larger abelian monoid $\mathcal{H} = [H, \diamond]$. This establishes an one-to-one correspondence between semigroup homomorphisms from S and arithmetic codes that can provide fault tolerance to computations in S.

The additional elements of H are used to detect errors and, if possible, correct erroneous results. The computation of $s_1 \circ s_2$ is carried out in H to produce the result

$$h' = \phi(s_1) \diamond \phi(s_2) \diamond e$$

where e denotes the error, modeled as an element of H. For purposes of completeness, we have assumed an additive error model, that is, we have assumed that the effect of a hardware fault on the final result can be modeled as an aggregate error e composed with the uncorrupted result $\phi(s_1) \diamond \phi(s_2)$ in the fashion shown above. In general, we should not expect to model the error in this way. Also note that, if the additive error is not invertible, we cannot expect anything more than detection of the error, not correction. Although the specifics of the error model are extremely important for the error detection and correction procedures, they do not affect the results we present here in any significant way.

If we focus on the special case of separate codes, we obtain even stronger results. The redundant semigroup H satisfies $H = S \times T$, where S is the original semigroup and T is the parity semigroup, [2]. Correspondingly,

$$\phi(s) = [s, \pi(s)] \quad \text{for all } s \in S$$

Using the fact that the mapping ϕ is a homomorphism, we can easily prove that the mapping π has to be a homomorphism as well. If, as in the case of groups, we restrict this parity mapping to be onto, it turns out that we can again obtain a complete characterization of all possible parity mappings and thus of all separate codes. However, the role that was played in the group framework by the subgroups N of the group G is now played by the so-called *congruence relations* in S, [8], [9].

An equivalence relation $\sim$ on the elements of S is called a congruence relation if $a \sim a', b \sim b' \Rightarrow a \circ b \sim a' \circ b'$. Let the set $S/\sim$ denote the set of equivalence classes of S under the congruence relation $\sim$. For congruence classes $[a], [b]$, we define the following binary operation:

$$[a] \otimes [b] = [a \circ b]$$

With this definition, $[S/\sim, \otimes]$ is a semigroup.

We are now in a position to apply a homomorphism theorem from semigroup theory (Theorems 3.29 and 3.30 in [9]) that is the natural generalization of the theorem used in the group case. Specifically, the surjective homomorphisms from S are isomorphic to the *canonical* surjective homomorphisms, namely those that map S to its factor (or quotient) semigroups $S/\sim$, where $\sim$ denotes a congruence relation in S. Hence the problem of finding all possible parity codings comes down to that of finding all possible congruence relations in S.

The above result on separate codes was used extensively in [2] in order to characterize all such codes for various semigroups of interest, such as the set of non-negative integers under the operation of addition (or

[3] *Abelian* semigroups are ones in which the binary operation is commutative.

[4] The assumptions are that the decoding mapping $\sigma : S' \longmapsto S$ (where $S' \subset H$ is the subset of valid results) is one-to-one and that $\phi_i(0_\circ) = 0_\diamond$ for $1 \leq i \leq p$.

multiplication), the set of integers under the MIN (or MAX) operation, and others. For example, the possible separate codes for the semigroup of positive integers under addition were shown to be of two forms:

1. For $n \in \{0, 1, 2, ...\}$, $\pi(n) = n \bmod M$, where M is any finite integer. In the special case when M is infinite, π is an isomorphism, whereas when $M = 0$ we get a trivial homomorphism onto the semigroup of a single identity element.
2. For finite positive integers k, M we have:
   $\pi(n) = n$ if $n < kM$
   $\pi(n) = n \bmod M$, otherwise
   Note that if $M = 0$ the mapping reduces to: $\pi(n) = n$ if $n < k$, $\pi(n) = 0_0$, otherwise.

The corresponding parity encodings for the group of integers under addition can only be of the first form mentioned above. Clearly, the relaxation from a group to a semigroup structure has opened more possibilities for parity encodings; their construction, however, became harder.

4 Semiring Extension

In this section we extend the previous results to a *semiring* framework. A semiring is a higher structure that comprises two operations associated by some distributive laws. More specifically, a semiring $\mathcal{R} = [R, +, \times]$ is a set of elements R together with two operations $+$ and $\times$ (called *addition* and *multiplication* respectively), and two distinguished elements 0_+ and $1_\times$ such that:

- R forms an abelian monoid under the operation of addition. The identity element of the additive monoid is 0_+. It also forms a monoid under the operation of multiplication. The identity element of the multiplicative monoid is $1_\times$.
- All $a, b, c \in R$ satisfy the *distributive laws*:
  $a \times (b + c) = (a \times b) + (a \times c)$, and
  $(b + c) \times a = (b \times a) + (c \times a)$.

Clearly, every ring is a semiring. The most natural example of a semiring that is not a ring is the set of non-negative integers under integer addition (additive operation) and multiplication (multiplicative operation). More examples of semirings and related theory can be found in [10].

Protection of a semiring computation is achieved by mapping the elements of $\mathcal{R} = [R, +, \times]$ to elements in a larger (redundant) semiring $\mathcal{H} = [H, \oplus, \otimes]$. Once again, the model of the computational system looks exactly like the one in Figure 1, except that instead of a single operation we are now dealing with two semiring operations. It can be shown that, under the same assumptions as in the semigroup case[5], all mappings ϕ_i have to be the exact same semiring homomorphism that we will denote by ϕ from now on, [2]. This result establishes an one-to-one correspondence between semiring homomorphisms of R and mappings that can be used to protect semiring computations in R.

By focusing on the special case of separate codes, we can obtain stronger results. In such a case, the redundant semiring is of the form $H = R \times T$, where T is a parity semiring. The homomorphism ϕ can be described by

$$\phi(r) = [r, \pi(r)] \quad \text{for all} \quad r \in R$$

where π can easily be shown to be a semiring homomorphism, [2]. If we restrict ourselves to π's that are onto, we obtain a complete characterization of all possible separate codes for protecting computations in R. Naturally, the situation is slightly more complicated than in the semigroup case. The role of semigroup congruence relations is now replaced by *semiring congruence relations*. An equivalence relation $\sim$ on the elements of a semiring $[R, +, \times]$ is called a semiring congruence relation if

$$\text{For } a, b, a', b' \in R, \quad a \sim a', b \sim b' \Rightarrow a+b \sim a'+b' \text{ and } a \times b \sim a' \times b'$$

[5] The assumptions are that the decoding mapping $\sigma : R' \longmapsto R$ (where $R' \subset H$ is the subset of valid results) is one-to-one, and the identities of R map to the identities of H, i.e. $\phi_i(0_+) = 0_\oplus$ and $\phi_i(1_\times) = 1_\otimes$ for $1 \leq i \leq p$.

Let us define binary operations $\oplus$ and $\otimes$ as follows: $[a] \oplus [b] = [a+b]$, and $[a] \otimes [b] = [a \times b]$. With this definition, the set $[R/\sim, \oplus, \otimes]$ (the set of semiring congruence classes of R under relation $\sim$ with operations $\oplus$ and $\otimes$) forms a semiring.

Just like in the semigroup case, a theorem establishes an one-to-one correspondence between onto homomorphisms of a semiring R and the set of its semiring congruence classes $\sim$, [2]: the surjective homomorphisms from R are isomorphic to homomorphisms that map R onto $R/\sim$ where $\sim$ is a semiring congruence relation. Hence, the problem of finding all possible parity codes for a semiring computation in R reduces to finding all possible semiring congruence relations. Some further analysis of such homomorphisms in terms of *semiring complexes* and *ideals* can be found in [2].

The above results were used in [2] to construct separate codes for computations in the semiring of positive integers under regular integer addition (additive operation) and multiplication (multiplicative operation). Specifically, it was shown that the parity checks for this semiring are exactly the same as the parity checks for the semigroup of positive integers under addition (see Section 3). This is a generalization of Peterson's result in [11], that is, all possible parity check codes for the ring of integers perform addition and multiplication modulo M (where M is some positive integer).

More examples of the use of the semiring framework in constructing arithmetic codes for semiring computations can be found in [2]. As pointed out there, the potential applications of this framework go beyond these simple examples: not only can the framework be used for developing arithmetic codes, but it can also assist in making efficient use of redundancy and in exploiting the error model more fully.

References

[1] P. E. Beckmann, *Fault-Tolerant Computation Using Algebraic Homomorphisms.* PhD Thesis, EECS, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[2] C. N. Hadjicostis, *Fault-Tolerant Computation in Semigroups and Semirings.* M.Eng. Thesis, EECS, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[3] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE Transactions on Computers,* vol. 33, pp. 518-528, June 1984.

[4] J.-Y. Jou and J. A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent parallel structures," *Proceedings of the IEEE,* vol. 74, pp. 732-741, May 1986.

[5] V. S. S. Nair and J. A. Abraham, "Real-number codes for fault-tolerant matrix operations on processor arrays," *IEEE Transactions on Computers,* vol. 39, pp. 426-435, April 1990.

[6] J.-Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks," *IEEE Transactions on Computers,* vol. 37, pp. 548-561, May 1988.

[7] J. A. Abraham, "Fault tolerance techniques for highly parallel signal processing architectures," *Proc. of SPIE,* vol. 614, pp. 49-65, 1986.

[8] E. S. Ljapin, *Semigroups.* Volume Three, Translations of Mathematical Monographs, American Mathematical Society, Providence, Rhode Island, 1974.

[9] R. Lidl and G. Pilz, *Applied Abstract Algebra.* Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1985.

[10] J. S. Golan, *The Theory of Semirings with Applications in Mathematics and Theoretical Computer Science.* Longman Scientific & Technical, Essex, England, 1992.

[11] W. W. Peterson and E. J. Weldon Jr, *Error-Correcting Codes.* The MIT Press, Cambridge, Massachusetts, 1972.