Extending CAS with Algebraic Reductions

Zongzhe Yuan
Christ's College

UNIVERSITY OF
CAMBRIDGE

A dissertation submitted to the University of Cambridge
in partial fulfilment of the requirements for the degree of
Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
UNITED KINGDOM

Email: zy272@cl.cam.ac.uk

April 14, 2018

Declaration

I Zongzhe Yuan of Christ's College, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 14,235

Signed:

Date:

Abstract

This is the abstract. Write a summary of the whole thing. Make sure it fits in one page.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to the Path Problem and Algebraic Solution

The path problem has always fascinated mathematicians and computer scientists. At the very beginning, programmer and scientists designed algorithms to solve each of the path problem. The most famous path problem is the shortest distance problem and there are several well-known algorithm that can solve such a problem: Dijkstra's algorithm, BellmanFord algorithm, A* search algorithm and FloydWarshall algorithm. People use different primitive metrics and various complicated algorithms to solve different path problems.

However, such an approach has its obvious shortcomings. Even at some point, designing a new algorithm can "steal" the ideas of the original algorithm, people must design a completely new independent algorithm in the face of each new problem (new metric), and this makes it difficult to have a generic (or framework) approach to solve this type of problem. Even if the path problem has minor changes to the problem, it is difficult for people to solve the new problem by slightly modifying existing algorithms.
Hence, lots of predecessors have found the algebraic approaches to work around this kind of problem. Using the knowledge of abstract algebra, people find that the routing problem (path problem) can be represent using a data structure called "semiring" $(S, \oplus, \otimes, \bar{0}, \bar{1})$[1, 2, 3, 4, 5]. For example, the popular "shortest path problem" can be represented as $(S, min, +, \infty, 0)$[4] and the "maximal capacity path problem" can be represented as $(S, max, min, 0, \infty)$.

For each path problem that represented as a semiring, we can construct a corresponding matrix semiring that represent the concrete problem (the edges and the distances for the shortest path problem for example). Then using matrix multiplication and stability of the closure (the semiring), we can solve the real problem of each concrete path problem. However, the simple matrix approach can only solve the ”trivial” path problem. Those complicated problem, for example the widest shortest path problem that constructed from the shortest path problem semiring and the widest path problem (maximal capacity path problem) semiring using lexicograhpic product, can't be solved by this ”traditional” theory approach. Some times even the method can find an optimal solution, there is no guarantee to find all optimal solutions using the classical method.

Therefore, people have found a non-classical theory of algebraic path finding method, so that algebras that violated the distribution law can be accepted. This non-classical theory can handle the problem the simple classical theory can't handle, such as the problem that can't be solved by Dijkstra or Bellman-Ford. This kind of method is dedicated to finding the local optimal solution at first, and then the local optimal solution is exactly the same as the global optimal solution by some verification or some addition restriction on the computation. For example, the famous protocol, the routing information protocol which is based on distributed Bellman-Ford algorithms is one of the non-traditional theory.

However, even so, RIP will also have a series of problems. For example, when a node does not have edges connected to it, the RIP matrix calculation (without pre-setting the maximum number of calculation steps) will continue infinitely. Even if the maximum number of calculation steps is set in advance, RIP still has some deficiencies in efficiency.

So while we use another protocol BGP (Border Gateway Protocol), we add an annihilator to the entire complex semiring. At the same time, we found that when we represent the path, we may have a loop path (a node in the path has been passed more than once). So we need our problem set to change from the original path to elementary path (A path p is elementary if no node is repeated). In this process, we performed two operations $S \to S$, and here we call it reduction in general, which comes to the main problem of our project.

1.2 Reduction

Algebraic reduction, introduced by Ahnont Wongseelashote in 1977[2] is an unary operator for a given set of problem, $reduce : S \longrightarrow S$. It has several properties, satisfying $reduce(\emptyset) = \emptyset$, $\forall A \in S, reduce(reduce(A)) = reduce(A)$ (which is called idempotent property) and $\oplus : S \times S \rightarrow S, \forall A, B \in S, reduce(reduce(A) \oplus B) = reduce(A \oplus B) = reduce(A \oplus reduce(B))$ (which is left and right invariant property) and this paper will discuss these properties in the later section.

It is hard to specify the reduced problem set in the most programming language. However, in the world of logic and those programming language that can be used to prove properties, programmer can represent the reduced problem set as $\{x|x \in S, reduce(x) = x\}$ which is also a subset of the original problem set. The idea to represent the reduced set explicitly is to form a pair $< x, Pr(x) >$ where $x \in S$ is the element in the set and $Pr(x) : r(x) = x$ is the proof that the element is in the subset (the element will not change after the reduced function, otherwise it will be reduced).

The first example of the reduction is id which maps all the stuff to itself. Another example is the min-set where $min_{\leq}(x) = \{x \in S|\forall y \in S, \neg(y < x)\}$. Regarding to $\mathcal{P}(S)$ it works well with $\cup$ that the min-set contain all the elements and remove the element that is non-trivial set. And this reduction is used in the construction of elementary path.

1.3 Motivation

Wongseelashote defined the reduction operator in his paper[2]. However the definition there is not constructive and it is the traditional reduction. Gurney and L11 claimed reductions could be used for "non-traditional" reductions such as elementary paths and combining elementary paths with lexicographic product [6]. However, Gurney and L11 never worked out the details, and the reduction there is still not constructive.

It is worth mentioning that, the algebraic approaches that using the matrix as the computation rely on the property of the operators a lot, for example, the left and right distributivity of the semiring. However, most of the cases mentioned above are aimed at some simple problems, or the ideal situations. In reality, we need to face the problem that, in the most time for the com-

plex path problem (especially for lexicographic product). The CAS system can derive most of the properties for the new semirings from the original "simple" simirings. However, sometimes the problem set is not the original problem set the provided to us, but the subset of it. As I mentioned before, as the problem that consist of the lexicographic product of the shortest path problem semiring and the maximum capacity semiring, there exists some path that have 0 capacity which shouldn't be concerned in the solution of the problem. This kind of reduction can be represent as a unary operation on the original problem set in our paper.

Another example, when we are doing path problems, for the most time the path that has negative value or have infinite value is not quite interesting. However, the operation we defined there is on the whole families of object. When we are defining some data structure, like semiring, we want to know that the properties (like commutative, selective and etc..) of the proper subset (the set of objects after reduction) will hold or not, or for some cases we can't do further calculation.

As the method mentioned previously, the algebraic approach to the path problem is depend on the properties of semirings. If one, or some of the properties don't hold for the semiring, the algebraic approach can't guarantee to find the optimal solution (depends on the semiring structure and the operations). Thus, after we applying those reductions on the original problem set, there is no guarantee that the original properties will still hold for the new subset of problem set, and it comes to our point. The existing CAS system doesn't have the functionality to derive and prove the properties for the reduced problem. Hence, I want to figure out the relationship between the reduction and those properties for those operators on the problem set.

Furthermore, in most programming languages, it is extremely difficult to express the reduction properly, on contrast, we can represent the reduction as a proof or proposition in our proof world. The second goal to the project is to figure out the friendly-extraction to those reduction.

Chapter 2

Background

2.1 Basic Definition

In the world of logic, we need to define several basic concepts before we are getting started.
In mathematics, A binary relation R in an arbitrary sets (or classes) S (here I restrict the relationship to be inside a single set, or say the element from the same type) is a collection of ordered pairs of elements of set (type) S, which is a subset of the Cartesian product $S \times S$. In order to link the mathematical concept with the proposition in logic, I provides each relation a representation (hold or not) as a boolean value, which is regarded as a property in Coq. $\Lambda S : Type.brel$(Binary relation) : $S \to S \to bool$.
Then, I define the basic operations for a given (but arbitrary) type: binary operation : $\Lambda S.S \to S \to S$ and unary operation : $\Lambda S.S \to S$.
And I extend the relation with a product type and a reduced version. I define the composition, identity and production for a unary operation, production and reduction for binary operation.

2.2 Semiring and its Properties

In abstract algebra, a semiring is a data structure $(S, \oplus, \otimes, \bar{0}, \bar{1})$ where S is a set and $\oplus, \otimes$ are two operators : $S \times S \to S$. $(S, \oplus)$ is a commutative semigroup (has associative property) and $(S, \otimes)$ is a semigroup: $\forall a, b, c \in S, a \oplus (b \oplus c) = (a \oplus b) \oplus c$, $a \oplus b = b \oplus a$ and $a \otimes (b \otimes c) = (a \otimes b) \otimes c$. $\oplus, \otimes$ are also left and right distributive on S : $\forall a, b, c \in S : a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

and $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$. $\bar{0}$ is the identity of $\oplus$ and $\bar{1}$ is the identity of $\otimes$: $\forall a \in S, a \oplus \bar{0} = a = \bar{0} \oplus a$ and $\forall a \in S, a \otimes \bar{1} = a = \bar{1} \otimes a$. Finally, $\bar{0}$ is the annihilator of $\otimes$: $\forall a \in S, a \otimes \bar{0} = \bar{0} = \bar{0} \otimes a$.

For Binary Relationship

Here we have three properties for a binary relationship in set S. As it is mentioned in the previous section, in order to define the proposition in the logic proof, the binary relationship will return a boolean value to indicate the relationship is holding or not.
Then for a given relationship $\backsim$: $S \to S \to bool$, we have reflexivity: $\forall a \in S, a \backsim a \equiv true$, symmetric : $\forall a, b \in S, a \backsim b \equiv b \backsim a$, and transitivity: $\forall a, b, c \in S, a \backsim b \equiv true \wedge b \backsim c \equiv true \to a \backsim c \equiv true$.

For Binary Operation

Then we have six properties for a binary operator in set S. For a given operator $\bullet : S \times S \to S$ and the equality (actually an arbitrary binary relationship) in S, we have:
congruence: $\forall s_1, s_2, t_1, t_2 \in S, s_1 \equiv t_1 \wedge s_2 \equiv t_2 \to s_1 \bullet s_2 \equiv t_1 \bullet t_2$
associativity: $\forall a, b, c \in S, a \bullet (b \bullet c) \equiv (a \bullet b) \bullet c$
commutativity: $\forall a, b \in S, a \bullet b = b \bullet a$
selectivity: $\forall a, b \in S, a \bullet b \in \{a, b\}$
hasId: $\exists \bar{0} \in S, \forall a \in S, a \bullet \bar{0} = a = \bar{0} \bullet a$
hasAnn: $\exists \bar{1} \in S, \forall a \in S, a \bullet \bar{1} = \bar{1} = \bar{1} \bullet a$
In this paper, we will mostly concentrate on those properties that are holding or not in the reduced set of problem.

2.3 Algebraic approach to the Path Problem

As we mentioned above, there are several properties which are really significant to the path problem by using the algebraic approaches. One of them is the distributivity properties for the two operators in a given semiring. Left distributivity: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ and right distributivity: $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$. The importance of these two properties is, when we are using matrix multiplication to compute the path problem represent as a semiring, we need there

two distributivity properties to guarantee the correctness of the computation (those equation will be shown later). Distributivity is part of the properties inside the semiring, and that becomes one of the reason that semiring has been choosen to represent a path problem.
As a result, we modified the original path problem algorithm and the original metric to our new modified metric and matrix equations which is called the generic algorithm.
Another important crucial point in our project is ”order”. As the example provided above, the min-set reduction can be applied in any problem set with some ”order”. (carnonically order and natural order)

2.4 Combinator for Algebraic System

Combinator for Algebraic system (CAS)[7] is introduced in $L11$. It is a language to design algebraic systems, in which many algebraic properties are automatically received and people can combine different operators to obtain a new semiring[7]. We can also generalize a more complex path problem (in other words, we can abstract a more complex path problem with this new semiring, such as the lexicographic products [5]). CAS can easily return the properties of those combined-operation semirings and it is already defined in Coq[8] (mentioned in $L11$ by Dr Timothy Griffin).

Chapter 3

Related Work

Maybe I should put this part into the background part

3.1 Algebraic Apporach to the Path Problem

3.2 CAS

3.3 Semirings and Path Spaces

Chapter 4

Design and Implementation

4.1 Product Theory and Product Semigroup

4.2 Reduction Properties and Reduction Theory

Here I define the reduction in a binary operation in three different version: reduce the result of the operation, only reduce the argument of the operation and reduce both result and argument of the operation. $bop - reduce : \forall S : Type.\forall r : unary - opS.\forall b : binary - opS.\lambda x, y : S.r(bxy)$, $bop - reduce - args : \forall S : Type.\forall r : unary - opS.\forall b : binary - opS.\lambda x, y : S.b(rx)(ry)$, $bop - full - reduce : \forall S : Type.\forall r : unary - opS.\forall b : binary - opS.\lambda x, y : S.r(b(rx)(ry))$.

4.3 Direct representation of Reduced Semigroup

(Well defined, but not extraction friendly)

4.3.1 Homomorphism

4.4 Another Representation of Reduced Semigroup – RSemigroup

(Given the properties of semigroup together with the reduction proof together.)

(For those two sections, I am just thinking how to write it properly)

4.5 One approach based on the Annihilator

Chapter 5

Evaluation

For any practical projects, you should almost certainly have some kind of evaluation, and it's often useful to separate this out into its own chapter.

Chapter 6

Summary and Conclusions

As you might imagine: summarizes the dissertation, and draws any conclusions. Depending on the length of your work, and how well you write, you may not need a summary here.

You will generally want to draw some conclusions, and point to potential future work.

Bibliography

[1] B. A. CARR. An Algebra for Network Routing Problems. *IMA Journal of Applied Mathematics*, 7(3):273–294, June 1971.

[2] Ahnont Wongseelashote. Semirings and path spaces. *Discrete Mathematics*, 26(1):55 – 78, 1979.

[3] Seweryn Dynerowicz and Timothy G. Griffin. On the forwarding paths produced by internet routing algorithms. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10. IEEE, 2013.

[4] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[5] Alexander JT Gurney and Timothy G. Griffin. Lexicographic products in metarouting. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 113–122. IEEE, 2007.

[6] Alexander James Telford Gurney. Construction and verification of routing algebras.

[7] Timothy G. Griffin and Joo Lus Sobrinho. Metarouting. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 1–12. ACM, 2005.

[8] The Coq proof assistant. https://coq.inria.fr.