Telling a Story

Zongzhe Yuan

May 31, 2018

1 Introduction To the Problem

The discussion of our problem originated from a concept that was mentioned in the L11 Algebraic Path Problem class but was not introduced in detail [1]. In L11 class, the concept of reduction was proposed, along with some examples, and was used to solve some problems that we could not solve before. However, in addition to the definition of reduction provided in the lecture, the property of reduction has not been thoroughly discussed. At the same time, although we have found the definition of reduction and practical examples in other papers and thesis before, unfortunately, as in the case of L11, none of them discussed the property of reduction itself in detail and provided a soundness reasoning process. This makes the detailed discussion of reduction a valuable research topic, and becomes to the topic to our project.

In our research, we will not only conduct detailed reasoning on the properties of reduction, but we will also accomplish the following three goals:

- The definition in L11 and the definition of reduction in the previous paper are all defined mathematically, which leads to the representation of reduction become implementation unfriendly. Hence we tried to find an implementation friendly representation equivalent to traditional reduction representation.
- The functionality of classical reduction is limited, and it can not represent some reduction (especially one of the reduction we used in the construction of our path problem). We need a more generalized way to define the reduction, the generalized reduction, and also do reasoning on it.
- Based on the application of reduction to the real problems, we define a class of reduction, predicate reduction on the basis of the previous section generalized reduction. Although predicate reduction is more concrete than generalized reduction (which means that there are some reductions that cannot be represented by predicate reduction), it is a fairly generalized reduction, and we can also use it to define a lot of concrete instance of reduction.

Finally, we will use the predicate reduction defined earlier, combined with the properties of reduction and some other mathematical structures, to define the path problem we encountered in L11 lecture.

2 Basic Definition

Before all the introductions begin, we first introduce a series of mathematical concepts that we will use in our project and will focus on. Let us begin with our problem set S, which is a collection of elements. Here in our project, we force our problem set S to be non-trivial, which means there at least one element inside the problem set.

2.1 Equality

In problem set S, the first thing we need to consider is the equality $=_S$. In the mathematical definition we can think of equality as a particular binary relationship, which is a set of ordered pairs $=_S \in S \times S$. And for equality, we have some properties that need to be discussed.

Congruence:

$$\forall a, b, c, d \in S, a =_{S_1} b \wedge c =_{S_1} d \rightarrow a =_{S_2} c = b =_{S_2} d \tag{1}$$

Reflexive:

$$\forall a \in S, a =_S a \tag{2}$$

Symmetric:

$$\forall a, b \in S, a =_S b \rightarrow b =_S a \tag{3}$$

Transitive:

$$\forall a, b, c \in S, a =_S b \wedge b =_S c \rightarrow a =_S c \tag{4}$$

2.2 Unary Operator

What we need to discuss next is the unary operator acting on S, and our subsequent reduction will be expressed as an unary operator. A unary operator r on S is a function on S, $r : S \rightarrow S$. For the unary operators, we mainly care about the following properties, and these properties are also mainly discussed later when we discuss the properties of reduction.

Congruence:

$$\forall a, b \in S, a =_S b \rightarrow r(a) =_S r(b) \tag{5}$$

Idempotent:

$$\forall a \in S, r(a) =_S r(r(a)) \tag{6}$$

Preserve Id: Given a binary operator $\oplus : S \times S \to S$

$$\exists i \in S, \forall a \in S, a \oplus i =_S a =_S i \oplus a \to r(i) =_S i \tag{7}$$

Preserve Annihilator: Given a binary operator $\oplus : S \times S \to S$

$$\exists a \in S, \forall x \in S, x \oplus a =_S a =_S a \oplus x \to r(a) =_S a \tag{8}$$

Left Invariant: Given a binary operator $\oplus : S \times S \to S$

$$\forall a, b \in S, r(a) \oplus b =_S a \oplus b \tag{9}$$

Right Invariant: Given a binary operator $\oplus : S \times S \to S$

$$\forall a, b \in S, a \oplus r(b) =_S a \oplus b \tag{10}$$

2.3 Binary Operator

The last thing we need to discuss, and also a major focus of this project is the binary operator under problem set S. A binary operator $\oplus$ under S is a function $\oplus : S \times S \to S$. And we will mainly discuss the following properties about the binary operator.

Congruence:

$$\forall a, b, c, d \in S, a =_S b \wedge c =_S d \to a \oplus c =_S b \oplus d \tag{11}$$

Associative:

$$\forall a, b, c \in S, a \oplus (b \oplus c) =_S (a \oplus b) \oplus c \tag{12}$$

Commutative:

$$\forall a, b \in S, a \oplus b =_S b \oplus a \tag{13}$$

Not Commutative:

$$\exists a, b \in S, a \oplus b \neq_S b \oplus a \tag{14}$$

Selective:

$$\forall a, b \in S, a \oplus b =_S a \bigvee a \oplus b =_S b \tag{15}$$

Left Distributive: Given another binary operator $\times : S \times S \to S$

$$\forall a, b, c \in S, a \otimes (b \oplus c) =_S (a \otimes b) \oplus (a \otimes c) \tag{16}$$

Not Left Distributive: Given another binary operator $\times : S \times S \to S$

$$\exists a, b, c \in S, a \otimes (b \oplus c) \neq_S (a \otimes b) \oplus (a \otimes c) \tag{17}$$

Right Distributive: Given another binary operator $\times : S \times S \to S$

$$\forall a, b, c \in S, (a \oplus b) \otimes c =_S (a \otimes c) \oplus (b \otimes c) \tag{18}$$

Not Right Distributive: Given another binary operator $\times : S \times S \to S$

$$\exists a, b, c \in S, (a \oplus b) \otimes c \neq_S (a \otimes c) \oplus (b \otimes c) \tag{19}$$

3 Semiring and Path Problem

Before we going into our real problem, let us introduce a basic definition, called semiring. This section will start with a basic mathematical structure called "semiring". Then we will mention why semiring will be applied to solve the path problem.

3.1 Semiring

In abstract algebra, a semiring is a data structure $(S, \oplus, \otimes, \bar{0}, \bar{1})$ where S is a set (Type) and $\oplus, \otimes$ are two binary operators : $S \times S \to S$.

$(S, \oplus)$ is a commutative semigroup (has associative property) and $(S, \otimes)$ is a semigroup:

$$\forall a, b, c \in S, a \oplus (b \oplus c) = (a \oplus b) \oplus c, a \oplus b = b \oplus a$$

$$\forall a, b, c \in S, a \otimes (b \otimes c) = (a \otimes b) \otimes c$$

$\oplus, \otimes$ are also left and right distributive on S :

$$\forall a, b, c \in S : a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

$$\forall a, b, c \in S : (a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$$

$\bar{0}$ is the identity of $\oplus$ and $\bar{1}$ is the identity of $\otimes$:

$$\forall a \in S, a \oplus \bar{0} = a = \bar{0} \oplus a$$

$$\forall a \in S, a \otimes \bar{1} = a = \bar{1} \otimes a$$

Finally, $\bar{0}$ is the annihilator of $\otimes$:

$$\forall a \in S, a \otimes \bar{0} = \bar{0} = \bar{0} \otimes a$$

Some definition will include that $\bar{1}$ is the annihilator of $\oplus$:

$$\forall a \in S, a \oplus \bar{1} = \bar{1} = \bar{1} \oplus a$$

3.2 Semiring Representation of Path Problem

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, Bellman–Ford algorithm, A* search algorithm and Floyd–Warshall 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" $(S, \oplus, \otimes, \bar{0}, \bar{1})$[2, 3, 4, 5, 6]. For example, the popular "shortest path problem" can be represented as $(S, min, +, \infty, 0)$[5] 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. Here provides two examples that if we following the basis of distributed Bellman-Ford algorithm Protocol (For example RIP protocol) that calculate the path from i to destination j by using the knowledge from its immediate, neighbourhood and applying it own path to the neighbour available to the process, we will get the result by using the matrix multiplication.

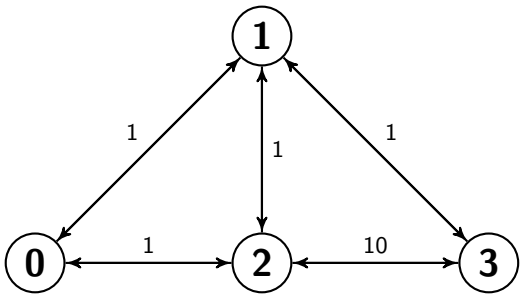

Figure 1: First Example Of RIP solution 1

We will get the following initial path problem adjacency matrix by using the RIP protocol.

$$\begin{bmatrix} \infty & 1 & 1 & \infty \\ 1 & \infty & 1 & 1 \\ 1 & 1 & \infty & 10 \\ \infty & 1 & 10 & \infty \end{bmatrix}$$

After using the RIP protocol for matrix calculations (multiplication), we obtained such an adjacency matrix as the result:

$$\begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

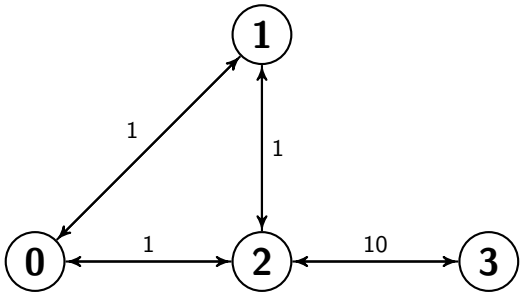

Figure 2: First Example Of RIP solution 2

We will get the following initial path problem adjacency matrix by using the RIP protocol.

$$\begin{bmatrix} \infty & 1 & 1 & \infty \\ 1 & \infty & 1 & \infty \\ 1 & 1 & \infty & 10 \\ \infty & \infty & 10 & \infty \end{bmatrix}$$

After using the RIP protocol for matrix calculations (multiplication), we ob-

tained such an adjacency matrix as the result:

$$\begin{bmatrix} 0 & 1 & 1 & 11 \\ 1 & 0 & 1 & 11 \\ 1 & 1 & 0 & 10 \\ 11 & 11 & 10 & 0 \end{bmatrix}$$

4 Previous Problem in L11

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. This leads to the problem that not all real-world problems can be solved directly with the simple matrix semiring. For example, we may meet the following situation (node 3 is not connected to all other nodes).

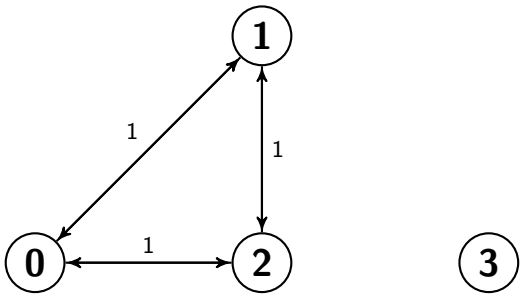

Figure 3: First Example Of RIP solution 3

We will get the following initial path problem matrix by using the RIP protocol.

$$\begin{bmatrix} 0 & 1 & 1 & \infty \\ 1 & 0 & 1 & \infty \\ 1 & 1 & 0 & \infty \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

After using the RIP protocol for n-step matrix calculations (multiplication), we obtained such a matrix:

$$\begin{bmatrix} 0 & 1 & 1 & n+1 \\ 1 & 0 & 1 & n+1 \\ 1 & 1 & 0 & n+1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

The result shows that if we do not limit the number of steps in the calculation, eventhough we may get the results of the path we need in the early step (eg path from a to b), but a better solution for the entire matrix may be found after "many: steps (or the calculation will counting to infinity, which is shown in this example).

4.1 Possible Solution

Therefore, in the course L11, instead of using RIP and simple matrix semiring approach, we used another protocol called BGP. We start from the simple $(\mathbb{N}, min, +)$ semiring that calculate the shortest distance, and we use lexicographic product (need reference to the Background/Definition) to construct a new semiring that contains the shortest-path metric and the set of its path.

Here we need to define some new operators/new rules for our semigroup. Assume $(S, \bullet)$ is a semigroup. Let

$$lift(S, \bullet) \equiv (fin(2^S), \hat{\bullet}) \tag{20}$$

where $X \hat{\bullet} Y = \{x \bullet y | x \in X, y \in Y\}$.

Then we can use our $lift$ to construct a bi-semigroup. Assume $(S, \bullet)$ is a semigroup. Let

$$union\_lift(S, \bullet) \equiv (\mathcal{P}(S), \cup, \hat{\bullet}) \tag{21}$$

where $X \hat{\bullet} Y = \{x \bullet y | x \in X, y \in Y\}$, and $X, Y \in \mathcal{P}(S)$, which is the set of finite subsets of S.

Then for a given graph $G = (V, E)$, we define

$$path(E) \equiv union\_lift(E^*, .) \tag{22}$$

where . is the concatenation function of sequence.

Finally we get our "shortest paths with paths" semiring from a given graph $G = (V, E)$:

$$spwp \equiv AddZero(\bar{0}, (\mathbb{N}, min, +) \overrightarrow{\times} path(E)) \tag{23}$$

4.2 Introduce Reduction into Our Problem

Here comes to the problem, when we are using *spwp* for doing calculations, because there may have loops in our $path(E)$, we need a lot of extra computation (though it will eventually yield correct results) to prove the paths have loops are not the shortest path. In order to eliminate these paths with loops, we introduced a new concept in the L11 course, the reduction.

If $(S, \oplus, \otimes)$ is a semiring and r is a function from S to S, then r is a reduction if $\forall a, b \in S$,

$$r(a) = r(r(a))$$

$$r(a \oplus b) = r(r(a) \oplus b) = r(a \oplus r(b))$$

and

$$r(a \otimes b) = r(r(a) \otimes b) = r(a \otimes r(b))$$

And, if $(S, \oplus, \otimes)$ is a semiring and r is a reduction, then $red_r(S) = (S_r, \oplus_r, \otimes_r)$, where

$$S_r = \{s \in S | r(s) = s\}$$

$$x \oplus_r y = r(x \oplus y)$$
$$x \otimes_r y = r(x \otimes y)$$

After that we went back to our path problem that for a given path p, we say p is elementary if there is no node inside p that is repeated. Then we can define our elementary path using the reduction

$$r(X) = \{p \in X | p \text{ is elementary } \} \tag{24}$$

and

$$epaths(E) = red_r(paths(E)) \tag{25}$$

Therefore, our path problem semiring became to

$$AddZero(\bar{0}, (\mathbb{N}, min, +) \overrightarrow{\times} epath(E)) \tag{26}$$

However, we still encounter the problem that there exists elements in our problem set that has a path distance value but does not have edges in its path. So we need to define the second reduction, which turns all elements that don't satisfy the condition into a single element (the *zero* we added into our semiring).

$$\begin{array}{rcl} r_2(inr(\infty)) & = & inr(\infty) \\ r_2(inl(s, \{\})) & = & inr(\infty) \\ r_2(inl(s, W)) & = & inl(s, W) \end{array} \tag{27}$$

Therefore, our path problem semiring becomes to

$$red_{r_2}(AddZero(\bar{0}, (\mathbb{N}, min, +) \overrightarrow{\times} epath(E))) \tag{28}$$

It is worth mentioning that we did not discuss the properties of reduction in detail in the L11 course. We also did not know whether the original semiring still satisfies the semiring property after the reduction. At the same time, whether some properties of the original operators (such as commutative rules) remains or not after reduction is also a mystery to us.

These are the topics that we need to focus on in our discussion. It is worth mentioning that the first reduction (reduce all pathes that are not elementary) that we defined above does not follow the property of the reduction on the *min* operation (doesn't have left/right invariant property). Imagine we have path $A = \{a, b, c, d\}$ that is longer but does not have loop and $B = \{a, b, b\}$ that is shorter but does have loop. Then $r(A \oplus B) = r(B) = \infty$ because B is shorter but has loop, but $r(A \oplus r(B)) = r(A \oplus \infty) = r(A) = A$ because B has loops but A doesn't.

Therefore, we not only discuss the properties of reduction, but also try to find another way to represent reduction to solve our path problem.

5 Classical Reduction, Example and Reasoning

5.1 Origin of the Concept of Reduction.

In order to better understand reduction and further analyze it in detail, we trace the source to find out if there is any other definition and study of reduction before us. In fact, the earliest definition we could find about reduction came from Ahnont Wongseelashote in 1977[3]. Wongseelashote also encountered the problem similar to ours when studying the path problem in the paper. In fact, our definition of reduction is very similar to Wongseelashote's definition of reduction (the real reason is that L11 refers to the Wongseelashote definition when the definition of reduction was given). However, unfortunately, Wongseelashoth only put forward the concept of reduction in the paper without detailed reasoning and structure proof.

Next we found that Alexander James Telford Gurney also mentioned the concept of reduction in his Ph.D thesis[7]. Gurney's discussion of reduction is also based on the definition by Wongseelashote, and it also does not focus on reduction for detailed structure proof.

5.2 Classical Reduction Definition

Therefore, here we refer to the definition of reduction in L11 (also the definition of Wongseelashote in paper) as the classical reduction, and we give the name of the way in which Wongseelashote represented reduction in paper as the traditional representation of reduction.

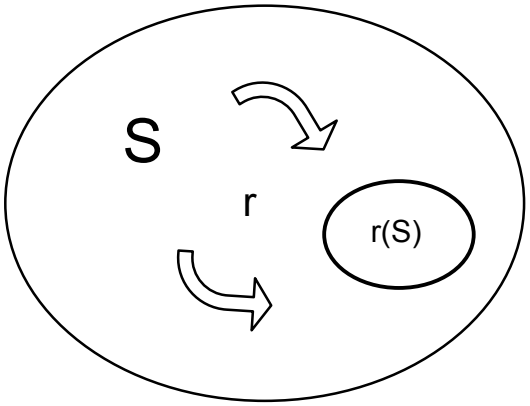

Figure 4: Illustration of Reduction

For a problem set S, the reduced problem set under reduction $r : S \to S$ is represented as

$$r(S) \equiv \{s \in S | r(s) =_S s\} \tag{29}$$

Then for the problem set S that represented as a semirgroup $(S, =_S, \oplus)$, the reduced problem set will be represented as $(r(S) \equiv \{s \in S | r(s) =_S s\}, =_S, \oplus_r)$ where

$$\forall a, b \in \{r(S) \equiv \{s \in S | r(s) =_S s\}, a \oplus_r b \equiv r(a \oplus b) \tag{30}$$

It is worth mentioning that although we use reduction to reduce the problem set from S to $\{r(S) \equiv \{s \in S | r(s) =_S s\}$, our equality $=_S$ is still the original equality $=_S$, . This can be seen from the figure (5.2), although we currently only focus on a subset of the original problem set S, the equality established on the problem set is still established on this subset. Here as the reduction r need to have the following properties.

Congruence:

$$\forall a, b \in S, a =_S b \to r(a) =_S r(b) \tag{31}$$

Idempotent:

$$r(a) = r(r(a)) \tag{32}$$

Left Invariant:

$$r(a \oplus b) = r(r(a) \oplus b) = r(a \oplus r(b)) \tag{33}$$

Right Invariant:

$$r(a \otimes b) = r(r(a) \otimes b) = r(a \otimes r(b)) \tag{34}$$

5.3 Example on L11 Lecture

To help us understand reduction and study its properties, we found another example of reduction in the course of L11.

The reduction is call $min_{\leq}$ (Martelli's semiring)[8] which means removing all the superset. For a given graph $G = (V, E)$, A cut set $C \in E$ for nodes i and j is a set of edges such there is no path from i to j in the graph $(V, E - C)$. C is minimal if no proper subset of C is a cut set. And Martelli's semiring is such that $A^{(*)}(i, j)$ is the set of all minimal cut sets for i and j. So Martelli's semiring is represented as

$$(S, \oplus, \otimes, \bar{0}, \bar{1})$$

where

$$S = min_{\leq}(2^{2^E})$$

$$X \oplus Y = min_{\leq}(\{U \cup V | U \in X, V \in Y\})$$

$$X \otimes Y = min_{\leq}(X \cup Y)$$

$$\bar{0} = \{\{\}\}$$

$$\bar{1} = \{\}$$

Here we can easily prove that $min_{\leq}$ satisfies our previous definition of reduction. So $min_{\leq}$ is a classical reduction.

5.4 Reasoning on Classical Reduction

Next we need to make a detailed reasoning the properties of the reduction. Because we are talking about classical reduction, we assume that our reduction r has properties of congruence (31), idempotent (32), left invariant (33) and right invariant (34).

First we discuss the properties of equality on reduction set. We assume the equality $=_S$ has the properties of reflexive (2), symmetric (3), transitive (4) and congruence (1) on the original problem set S. Then in the definition of our reduction representation we mentioned "although we use reduction to reduce the problem set from S to $\{r(S) \equiv \{s \in S | r(s) =_S s\}$, we equality $=_S$ is still the original equality $=_S$", so on reduced problem set $\{r(S) \equiv \{s \in S | r(s) =_S s\}$,the equality $=_S$ still has the property of reflexive, symmetric, transitive and congruence.

Next we need to discuss the properties of the binary operator under reduced problem set. Recalling to our definition, for a binary operator $\oplus : S \times S \to S$, the reduced binary operator under reduction r is defined as

$$\forall a, b \in \{r(S) \equiv \{s \in S | r(s) =_S s\}, a \oplus_r b \equiv r(a \oplus b)$$

So we first assume that the original binary operator $\oplus$ has certain properties on S, and then reasoning that these properties are remained after reduction ($\oplus_r$ still has the same property under $\{r(S) \equiv \{s \in S | r(s) =_S s\}$)

Commutative: we initially assume that $\oplus$ is commutative under S, which is $\forall a, b \in S, a \oplus b =_S b \oplus a$. Then we want to proof that $\forall a, b \in \{s \in S | r(s) =_S s\}, a \oplus_r b =_S b \oplus_r a$.

$$\begin{array}{lll} a \oplus_r b & =_S \; r(a \oplus b) & \text{by the definition of } \oplus_r \\ & =_S \; r(b \oplus a) & \text{by commutative of } \oplus \text{ and congruence of } r \\ & =_S \; b \oplus_r a & \text{by the definition of } \oplus_r \end{array} \tag{35}$$

So we can conclude that commutative of $\oplus$ plus congruence of r implies commutative of $\oplus_r$.

Congruence: we initially assume that $\oplus$ is congruence under S, which is $\forall a, b, c, d \in S, a =_S b \wedge c =_S d \to a \oplus c =_S b \oplus d$. Then we want to proof that $\forall a, b, c, d \in \{s \in S | r(s) =_S s\}, a =_S b \wedge c =_S d \to a \oplus_r c =_S b \oplus_r d$.

$$\begin{array}{lll} a \oplus_r c & =_S \; r(a \oplus c) & \text{by the definition of } \oplus_r \\ & =_S \; r(b \oplus c) & \text{by congruence of } \oplus \text{ and congruence of } r \\ & =_S \; r(b \oplus d) & \text{by congruence of } \oplus \text{ and congruence of } r \\ & =_S \; b \oplus_r d & \text{by the definition of } \oplus_r \end{array} \tag{36}$$

So we can conclude that congruence of $\oplus$ plus congruence of r implies congruence of $\oplus_r$.

Selective: we initially assume that $\oplus$ is selective under S, which is $\forall a, b \in S, a \oplus b =_S a \bigvee a \oplus b =_S b$. Then we want to proof that $\forall a, b \in \{s \in S | r(s) =_S$

$s\}, a \oplus_r b =_S a \bigvee a \oplus_r b =_S b$. By using the selective property, we can split the cases of $a \oplus b$.

case 1: $a \oplus b =_S a$

$$\begin{array}{lll} a \oplus_r b & =_S & r(a \oplus b) \quad \text{by the definition of } \oplus_r \\ & =_S & r(a) \quad \text{congruence of } r \\ & =_S & a \quad \text{because } r(a) = a \end{array}$$

case 2: $a \oplus b =_S b$

$$\begin{array}{lll} a \oplus_r b & =_S & r(a \oplus b) \quad \text{by the definition of } \oplus_r \\ & =_S & r(b) \quad \text{congruence of } r \\ & =_S & b \quad \text{because } r(b) = b \end{array}$$

So we can conclude that selective of $\oplus$ plus congruence of r implies selective of $\oplus_r$.

Associative: we initially assume that $\oplus$ is associative under S, which is $\forall a, b, c \in S, a \oplus (b \oplus c) =_S (a \oplus b) \oplus c$. Then we want to proof that $\forall a, b, c \in \{s \in S | r(s) =_S s\}, a \oplus_r (b \oplus_r c) =_S (a \oplus_r b) \oplus_r c$.

$$\begin{array}{lll} a \oplus_r (b \oplus_r c) & =_S & r(a \oplus r(b \oplus c)) \quad \text{by the definition of } \oplus_r \\ & =_S & r(a \oplus (b \oplus c)) \quad \text{by right invariant property of } r \text{ on } \oplus \\ & =_S & r((a \oplus b) \oplus c) \quad \text{by associative of } \oplus \text{ and congruence of } r \\ & =_S & r(r(a \oplus b) \oplus c) \quad \text{by left invariant property of } r \text{ on } \oplus \\ & =_S & (a \oplus_r b) \oplus_r c \quad \text{by the definition of } \oplus_r \end{array} \tag{37}$$

So we can conclude that associative of $\oplus$ plus congruence, left and right invariant of r on $\oplus$ implies associative of $\oplus_r$. Here we find that associative is not the same as the other three properties under binary operators. The proof that a binary operator satisfies the associative under reduction uses the left/right invariant properties of reduction. And in the later section we will focus on the properties of the associative and do reasoning on it.

Left Distributive: by adding another binary operator $\otimes : S \times S \to S$, the reduced binary operator $\otimes_r$ as $\forall a, b \in \{s \in S | r(s) =_S s\}, a \otimes_r b \equiv r(a \otimes b)$, we initially assume that $\oplus$ and $\otimes$ are left distributive under S, which is $\forall a, b, c \in S, a \otimes (b \oplus c) =_S (a \otimes b) \oplus (a \otimes c)$. Then we want to proof that $\forall a, b, c \in \{s \in S | r(s) =_S s\}, a \otimes_r (b \oplus_r c) =_S (a \otimes_r b) \oplus_r (a \otimes_r c)$.

$$\begin{array}{lll} a \otimes_r (b \oplus_r c) & =_S & r(a \otimes r(b \oplus c)) \quad \text{by the definition of } \oplus_r \text{ and } \otimes_r \\ & =_S & r(a \otimes (b \oplus c)) \quad \text{by right invariant property of } r \text{ on } \otimes \\ & =_S & r((a \otimes b) \oplus (a \otimes c)) \quad \text{by left distributive of } \oplus \text{ and } \otimes \\ & =_S & r(r(a \otimes b) \oplus r(a \otimes c)) \quad \text{by left and right invariant of } r \text{ on } \oplus \\ & =_S & r((a \otimes_r b) \oplus (a \otimes_r c)) \quad \text{by the definition of } \otimes_r \\ & =_S & (a \otimes_r b) \oplus_r (a \otimes_r c) \quad \text{by the definition of } \oplus_r \end{array} \tag{38}$$

So we can conclude that left distributive of $\oplus$ and $\otimes$ plus congruence, left and right invariant of r on $\oplus$, right invariant of r on $\otimes$ implies left distributive of $\oplus_r$ and $\otimes_r$.

Similarly we can conclude that right distributive of $\oplus$ and $\otimes$ plus congruence, left and right invariant of r on $\oplus$, left invariant of r on $\otimes$ implies right distributive of $\oplus_r$ and $\otimes_r$.

Thus, we get the conclusion that distributive of $\oplus$ and $\otimes$ plus congruence, left and right invariant of r on $\oplus$ and $\otimes$ implies distributive of $\oplus_r$ and $\otimes_r$. It is also worth mentioning that the property of the distributive is also one of the major features we will discuss in the later section.

6 New Reduction Representation

The previous section have mentioned that our definition of classical reduction traditional representation is implementation unfriendly.

This is because inside traditional reduction representation our actual problem set is $r(S) = \{s \in S | r(s) =_S s\}$ which is a type with record. Record is difficult to be accurately represented in most situations.

Imagine we have two different reductions r_1 and r_2 for S, then the problem set after applying reduction r_1 and r_2 will be $r_2(r_1(S)) = \{y \in \{x \in S | r_1(x) =_S x\} | r_2(y) =_S y\}$.

The nest record of reduction in our problem domain makes all the reasoning and calculation step into trouble. At the same time, since our real problem set has changed from the original S to the current $r(S) = \{s \in S | r(s) =_S s\}$ (suppose we only have one reduction r), the domain of our binary operator has changed. If we originally have a binary operator $\oplus : S \times S \to S$, then the binary operator after reduction will become the binary operator

$$\oplus_r : \{s \in S | r(s) =_S s\} \times \{s \in S | r(s) =_S s\} \to \{s \in S | r(s) = s\}$$

And if we apply two/more reductions simultaneously, the entangling of reductions with problem set and operator makes it almost impossible for us to define the combinator that exists on reduction and solve the problem at all.

Our goal is to redefine the equality and binary operators without changing the problem set S. Then the reduced problem set based on $(S, =_S, \oplus)$ by reduction r will become $(S, =_S^r, \oplus^r)$, which is still a problem set on S, but not $(\{x \in S | r(x) =_S x\}, =, \oplus_r)$. Then we will prove the isomorphism between these two representations.

6.1 Generalized Representation of Equality and Binary Operator

Let us look back and forth at the figure (5.2). In the figure, our reduction will reduce the problem set S to a subset of S. In our traditional representation, because reduction has already brought problem set from S to $\{x \in S | r(x) =_S x\}$,

which means when we are comparing two elements from the problem set, those two elements are already inside $\{x \in S | r(x) =_S x\}$, we can directly use the original equality $=_S$ as well.

However if we want to keep our problem set still as S, when we are doing equality comparison, we must guarantee that the elements we compared are already inside the reduced problem set. Thus we defined the new equality as

$$\forall a, b \in S, a =_S^r b \equiv r(a) =_S r(b) \tag{39}$$

The same reason for our binary operator $\oplus$, when we defined $\oplus_r$ earlier, because the arguments we need have been reduced to $\{x \in S | r(x) =_S x\}$, we only need to consider that the result of operation must also be in $\{x \in S | r(x) =_S x\}$, so we give the definition above (30).

Similarly to the equality, if we want to keep our problem set still as S, when we are doing binary operation, we not only need to consider that the result of the operation is in the reduced problem set, we also need to ensure that the two arguments of the binary operator are in the reduced problem set. Therefore we define our binary operator under reduction:

$$\forall a, b \in S, a \oplus^r b \equiv r(r(a) \oplus r(b)) \tag{40}$$

6.2 Isomorphism between Transitional and Generalized Representation

Next we need to prove that our new generalized representation is isomorphic to the previous traditional representation.

6.2.1 Isomorphic On Equality Properties

First we discuss the properties on equality between the traditional representation and the generalized representation.

Congruence: we need to proof that $=_S$ is congruence on $\{x \in S | r(x) =_S x\}$ is isomorphic to $=_S^r$ is congruence on S, which means

$$\forall a, b, c, d \in \{x \in S | r(x) =_S x\}, a =_S b \wedge c =_S d \rightarrow a =_S c = b =_S d$$

$$\longleftrightarrow$$

$$\forall a, b, c, d \in S, a =_S^r b \wedge c =_S^r d \rightarrow a =_S^r c = b =_S^r d$$

Reflexive: we need to proof that $=_S$ is reflexive on $\{x \in S | r(x) =_S x\}$ is isomorphic to $=_S^r$ is reflexive on S, which means

$$\forall a \in \{x \in S | r(x) =_S x\}, a =_S a$$

$$\longleftrightarrow$$

$$\forall a \in S, a =_S^r a$$

Symmetric: we need to proof that $=_S$ is symmetric on $\{x \in S | r(x) =_S x\}$ is isomorphic to $=_S^r$ is symmetric on S, which means

$$\forall a, b \in \{x \in S | r(x) =_S x\}, a =_S b \rightarrow b =_S a$$

$$\longleftrightarrow$$

$$\forall a, b \in S, a =_S^r b \rightarrow b =_S^r a$$

Transitive: we need to proof that $=_S$ is transitive on $\{x \in S | r(x) =_S x\}$ is isomorphic to $=_S^r$ is transitive on S, which means

$$\forall a, b, c \in \{x \in S | r(x) =_S x\}, a =_S b \wedge b =_S c \rightarrow a =_S c$$

$$\longleftrightarrow$$

$$\forall a, b, c \in S, a =_S^r b \wedge b =_S^r c \rightarrow a =_S^r c$$

These proof are really trivial and only need the property of idempotent of r, which is $\forall a \in S, r(a) = r(r(a))$.

6.2.2 Isomorphic On Binary Operator Properties

Next we discuss the properties on binary operator $\oplus$ (and $\otimes$ when we are talking about distributive) between the traditional representation and the generalized representation.

Commutative: we need to proof that $\oplus_r$ is commutative on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ is commutative on S, which means

$$\forall a, b \in \{x \in S | r(x) =_S x\}, a \oplus_r b =_S b \oplus_r a$$

$$\longleftrightarrow$$

$$\forall a, b \in S, a \oplus^r b =_S^r b \oplus^r a$$

Selective: we need to proof that $\oplus_r$ is selective on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ is selective on S, which means

$$\forall a, b \in \{x \in S | r(x) =_S x\}, a \oplus_r b =_S a \bigvee a \oplus_r b =_S b$$

$$\longleftrightarrow$$

$$\forall a, b \in S, a \oplus^r b =_S^r a \bigvee a \oplus^r b =_S^r b$$

Congruence: we need to proof that $\oplus_r$ is congruence on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ is congruence on S, which means

$$\forall a, b, c, d \in \{x \in S | r(x) =_S x\}, a =_S b \wedge c =_S d \rightarrow a \oplus_r c =_S b \oplus_r d$$

$$\longleftrightarrow$$

$$\forall a, b, c, d \in S, a =_S^r b \wedge c =_S^r d \rightarrow a \oplus^r c =_S^r b \oplus^r d$$

Associative: we need to proof that $\oplus_r$ is associative on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ is associative on S, which means

$$\forall a, b, c \in \{x \in S | r(x) =_S x\}, a \oplus_r (b \oplus_r c) =_S (a \oplus_r b) \oplus_r c$$

$$\longleftrightarrow$$

$$\forall a, b, c \in S, a \oplus^r (b \oplus^r c) =_S^r (a \oplus^r b) \oplus^r c$$

Left Distributive: we need to proof that $\oplus_r$ and $\otimes_r$ are left distributive on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ and $\oplus^r$ are left distributive on S, which means

$$\forall a, b, c \in \{x \in S | r(x) =_S x\}, a \otimes_r (b \oplus_r c) =_S (a \otimes_r b) \oplus_r (a \otimes_r c)$$

$$\longleftrightarrow$$

$$\forall a, b, c \in S, a \otimes^r (b \oplus^r c) =_S^r (a \otimes^r b) \oplus^r (a \otimes^r c)$$

Right Distributive: we need to proof that $\oplus_r$ and $\otimes_r$ are right distributive on $\{x \in S | r(x) =_S x\}$ is isomorphic to $\oplus^r$ and $\oplus^r$ are right distributive on S, which means

$$\forall a, b, c \in \{x \in S | r(x) =_S x\}, (a \oplus_r b) \otimes_r c =_S (a \otimes_r c) \oplus_r (b \otimes_r c)$$

$$\longleftrightarrow$$

$$\forall a, b, c \in S, (a \oplus^r b) \otimes^r c =_S^r (a \otimes^r c) \oplus^r (b \otimes^r c)$$

These proof will be shown in Coq file and we only need the property of congruence and idempotent of r, which is $\forall a \in S, r(a) = r(r(a))$ and $\forall a, b \in S, a =_S b \rightarrow r(a) =_S r(b)$.

This proves that our generalized representation of reduction is isomorphic to the traditional representation of reduction, which means all the properties we have proved in the classical reduction section could be used directly into our generalized representation.

7 Generalized Reduction

In our previously defined classical reduction (either in traditional representation or generalized representation), reduction must satisfy four properties that it is congruence, idempotent, left and right invariant. However, the reduction we encounter in reality does not all have the above four properties, especially the left/right invariant properties. In particular, the elementary reduction in the elementary path problem that we mentioned earlier, which will also be used

in the later section to construct our final path problem, does not satisfy the properties of the left/right invariant on the *min* operator.

Assuming $P_1, P_2 \in Path, P_1 \leq P_2, loop(P_1) = true \wedge loop(P_2) = false$. Then $r(min(P_1, P_2)) = r(P_1) = \infty$. However $r(min(r(P_1), P_2)) = r(min(\infty, P_2))r(P_2) = P_2$.

Therefore, in order to define our elementary path reduction and also to better represent other reductions, we generalize the definition reduction as: r is a reduction on S if r has the properties of congruence and idempotent, and get rid of the constraint of left/right invariant properties, and we name this kind of reduction as generalized reduction.

Do not confuse generalized representation of reduction with generalized reduction here. Generalized representation of reduction is a new representation of reduction, compared to and isomorphic to the traditional representation of reduction, that is implementation friendly. A reduction that represented by generalized representation could be classical reduction (that has left/right invariant properties) or generalized reduction (left/right invariant properties can be gotten rid of). Generalized reduction we defined here is a kind of reduction that is only asked for idempotent and congruence properties without forced to be left/right invariant. Generalized reduction could be represented by using traditional representation or generalized representation.

Because the generalized representation is implementation friendly, and we have proved the isomorphism between the generalized representation and traditional representation, we will use generalized representation to represent reduction in the following paragraph.

7.1 Properties of Generalized Reduction, Pseudo Associative and Pseudo Distributive

We have reasoned about the relationship between reduction and equality/binary operators in the previous section of classical reduction. Since in the last section we have already reached a grossized representation is isomorphic to the traditional representation, and the generalized reduction is just the classical reduction without left/right invariant properties, by assuming that the reduction r is congruence and idempotent, we can have the following conclusion:

If $=_S$ is congruence/reflexive/symmetric/transitive on the problem set S, then $=_S^r$ is congruence/reflexive/symmetric/transitive on the problem set S.

If $\oplus$ has the properties of commutative/selective/congruence on the problem set S, then $\oplus^r$ has the properties of commutative/selective/congruence on the problem set S.

There is another property that we may be interested in, if i/a is the identity/annihilator for $\oplus$ on S, and the reduction r has the properties of preserve id/ann (7,8), then i/a is the identity/annihilator for $\oplus^r$ on S.

The more troublesome is the properties of associative and distributive. Since the proof used the property of left/right invariant, and we don't have those properties inside generalized reduction by default, we cannot directly get the these two properties. So we conclude that generalized reduction with left/right properties will allow the operator(s) remaining associative and distributive properties.

However, the elementary path reduction we want to define afterwards is only a generalized reduction and does not have left/right invariant properties on the *min* operator. We still want to discuss the properties and relationship between the reduction and the operator and discuss whether *min* has associative property under reduction, and discuss whether the entire semiring has distributive properties.

Therefore, we found two sufficient condition, separately for associative properties and distributive properties respectively that we can prove the properties if we can prove the sufficient condition is holding.

Pseudo Associative:

$$\forall a, b, c \in S, r(r(r(a) \oplus r(b)) \oplus r(c)) = r(r(a) \oplus r(r(b) \oplus r(c))) \tag{41}$$

Pseudo Left Distributive:

$$\forall a, b, c \in S, r(r(a) \otimes r(r(b) \oplus r(c))) = r(r(r(a) \otimes r(b)) \oplus r(r(a) \otimes r(c))) \tag{42}$$

Pseudo Right Distributive:

$$\forall a, b, c \in S, r(r(r(a) \oplus r(b)) \otimes r(c)) = r(r(r(a) \otimes r(c)) \oplus r(r(b) \otimes r(c))) \tag{43}$$

It is easy to prove $\oplus^r$ is associative on S, and $\oplus^r$ and $\otimes^r$ are distributive on S by using pseudo associative and pseudo distributive properties.

Associative: by definition of $=_S^r$, $\oplus^r$

$$\forall a, b, c \in S, a \oplus^r (b \oplus^r c) =_S^r (a \oplus^r b) \oplus^r c$$

$$\longleftrightarrow$$

$$r(r(r(a) \oplus r(r(r(b) \oplus r(c))))) =_S r(r(r(r(r(a) \oplus r(b))) \oplus r(c)))$$

$r(r(r(a) \oplus r(r(r(b) \oplus r(c)))))$

$$\begin{array}{lll} =_S & r(r(a) \oplus r(r(r(b) \oplus r(c)))) & \text{by idempotent property of } r \\ =_S & r(r(a) \oplus r(r(b) \oplus r(c))) & \text{by idempotent property of } r \text{ and congruence of } \oplus \\ =_S & r(r(r(a) \oplus r(b)) \oplus r(c)) & \text{by pseudo associative} \\ =_S & r(r(r(r(a) \oplus r(b))) \oplus r(c)) & \text{by idempotent property of } r \text{ and congruence of } \oplus \\ =_S & r(r(r(r(a) \oplus r(b))) \oplus r(c))) & \text{by idempotent property of } r \end{array} \tag{44}$$

Left Distributive: by definition of $=_S^r$, $\oplus^r$ and $\otimes^r$

$$\forall a, b, c \in S, a \otimes^r (b \oplus^r c) =_S^r (a \otimes^r b) \oplus^r (a \otimes^r c)$$

$$\longleftrightarrow$$

$$r(r(r(a)) \otimes r(r(r(b) \oplus r(c)))) =_S r(r(r(r(a) \otimes r(b))) \oplus r(r(r(a) \otimes r(c))))$$

$r(r(r(a)) \otimes r(r(r(b) \oplus r(c))))$

$$\begin{array}{lll} =_S & r(r(a) \otimes r(r(b) \oplus r(c)))) & \text{by idempotent property of } r \text{ and congruence of } \otimes \\ =_S & r(r(r(a) \otimes r(b)) \oplus r(r(a) \otimes r(c))) & \text{by pseudo left distributive} \\ =_S & r(r(r(r(a) \otimes r(b))) \oplus r(r(r(a) \otimes r(c)))) & \text{by idempotent property of } r \text{ and congruence of } \oplus \\ =_S & r(r(r(r(r(a) \otimes r(b))) \oplus r(r(r(a) \otimes r(c))))) & \text{by idempotent property of } r \end{array} \tag{45}$$

Similar proof for right distributivity.

So we can conclude that for generalized reduction, although it may not have the property of left/right invariant, if we can directly prove the properties of pseudo associative/pseudo distributive, then we can also prove that binary operator(s) remains associative/distributive under reduction.

8 Predicate Reduction

In order to better implement the construction of the path problem we mentioned earlier, we are going to define a type of reduction and perform detailed reasoning on its properties in this section. If we deeply consider our reduction, whether it is to eliminate the path with a loop, or to turn an unqualified element tuple into a specific element – additive identity/multiplicative annihilator, we can easily find the situation that our reduction is depending on a predicate and reducing the elements that satisfy/not satisfy the condition. Therefore, we call this kind of reduction as predicate reduction.

We define a new type of reduction, predicate reduction based on generalized reduction. Predicate reduction is a kind of generalized reduction, but it is not as much generalized as the generalized reduction. Generalized reduction could define almost all kinds of reductions, but predicate reduction couldn't, and it is the reason why it is not as much generalized as the previous one. For example, when we are talking about the min set $min_{\leq} = \{x \in X | \forall y \in X, y \not\leq x\}$, which is introduced as Martelli's semiring in our previous section, we can't define $min_{\leq}$ using predicate reduce. The reason is, when we are defining a predicate, the predicate only know the information on its current element, but not the whole problem set. Predicate reduction is still a kind of generalized reduction but not classical reduction because it is not forcing the reduction to have left/right invariant properties. Have or not those properties depends on the predicate inside reduction. For example, we will use predicate reduction to define three different reductions in the later section, two of them (min plus with ceiling and reduction annihilator) has the properties of left/right invariant on both operator, but one of them (elementary path) doesn't on its *min* operator.

Here comes to our definition. Initially for our problem set S, we define a predicate P, which is a function from S to *bool* and we need to provide a specific element c (the predicate reduction will reduce all the element that satisfied the

predicate to $c \in S$). Then we can define the predicate reduction as

$$\forall a \in S, r_p(a) = \begin{cases} c & P(a) = true \\ a & otherwise \end{cases} \tag{46}$$

As we mentioned previously, predicate reduction is defined based on the generalized reduction, which means it has all the properties that generalized reduction has. Hence, we are not interested on the properties of equality, congruence, commutative and selective, has identity and has annihilator properties of binary operator any more. We mainly talk about the associative and distributive in the reasoning section (although we may use the properties of has identity and has annihilator during the process of proof).

8.1 Properties of Predicate

In predicate reduction, we hope to explore the properties of predicate reduction that are different from generalized reduction. But as the predicate reduction is a kind of generalized reduction, it must have the properties of the idempotent and congruence property, which are $\forall a \in S, r(a) = r(r(a))$ and $\forall a, b \in S, a = b \rightarrow r(a) = r(b)$. So we need to do reasoning on these two properties. For the property of the idempotent, we need to discuss the input element, split the case that the element satisfies the predicate or not. If the element doesn't satisfy the predicate, then $r(a) = a = r(r(a))$. If the element does satisfy the predicate, then $r(a) = c$ and $r(r(a)) = r(c)$, hence we need to prove/provide the property that c must satisfy the predicate

$$P(c) = true \tag{47}$$

For the property of the congruence, we can conclude from

$$\forall a, b \in S, r_p(a) = \begin{cases} c & P(a) = true \\ a & otherwise \end{cases} = r_p(b) = \begin{cases} c & P(b) = true \\ b & otherwise \end{cases}$$

that we need such a property

$$\forall a, b \in S, a = b \rightarrow P(a) = P(b) \tag{48}$$

which is the congruence of the predicate P.

Next we need to reduce the binary operator according to the generalize representation we defined earlier using the predicate reduction. For our problem set S and a given binary operator $\oplus$, we defined reduced binary operator as

$$\forall a, b \in S, a \oplus_r b = r_p(r_p(a) \oplus r_p(b)) \tag{49}$$

Since we completely use the generalize reduction representation defined in the previous section, the reduction we defined has all the above-mentioned properties, such as commutative, selective, congruence, and so on. What deserves

our attention is the associative property of a single binary operator and the distribution law of reduced semiring.
We also mentioned above that under our generalize reduction representation, the reduction we defined may not satisfy the properties of the left/right invariant (like the *min* operator under the elementary path problem), so we need to prove (or give certain preconditions) that our reduction to satisfy the pseudo-associative and pseudo-distributive law.

9 Path Problem Construction

References

[1] Timothy Griffin. Algebraic path problem, 2017.

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

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

[4] 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.

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

[6] 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.

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

[8] Alberto Martelli. A gaussian elimination algorithm for the enumeration of cut sets in a graph. 23(1):58–73.