Construction and verification of routing algebras

Alexander James Telford Gurney

30 April 2009

UNIVERSITY OF CAMBRIDGE

Darwin College
Computer Laboratory

This dissertation is submitted for the degree of Doctor of Philosophy

Declarations

- This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.
- It is not substantially the same as any dissertation that I have submitted or will be submitting for a degree or diploma or other qualification at any other university.
- It does not exceed 60 000 words (including tables and footnotes, but excluding appendices, bibliography, photographs and diagrams).

A. J. T. G.
2009–4–30

Summary

Standard algorithms are known for finding the best routes in a network, for some given notion of route preference. Their operation succeeds when the preferences satisfy certain criteria, which can be expressed algebraically. The Internet has provided a wide variety of route-finding problems for which these criteria can be hard to verify, and the ambitions of network operators and researchers are more diverse still.

One solution is to provide a formal means of describing route preferences in such a way that their correctness criteria can be automatically verified. This thesis makes the following contributions:

- It shows that analysis of generic route-finding algorithms can be separated from study of the specific algebraic structures that encode route preferences. This separation extends to more complex cases than are typically considered in this context.

- Lexicographic choice is analyzed in detail, covering deduction rules for correctness criteria as well as its uses. Design constraints applicable to interdomain routing protocols are derived from a study of lexicographic choice for hierarchical networks.

- Previous results on algorithmic correctness have been extended in two ways. An account has been given of the relationship between two proof strategies for finding the criteria associated with the existence of a Nash equilibrium. This leads to a new proof which is both shorter and more general.

- Multipath routing has been incorporated into the algebraic framework. This allows unified treatment of both single- and multipath algorithms: they are the same, but instantiated over different algebras. Another application of the mathematics allows a rigorous treatment of the handling of erroneous routes.

The examples and applications demonstrate the power of the algebraic approach in permitting analysis of systems that would otherwise be much more difficult to treat.

Contents

Chapter 1

Introduction

The problem of *finding the best path* has captivated the interest of mathematicians and engineers ever since it was discovered that even very large and complicated pathfinding problems could be solved on a digital computer.

This problem has multiple origins, and a true unification of the efforts of disparate researchers and practitioners did not occur until the 1960s. Prior to that date, independent groups with different backgrounds and experiences tackled the pathfinding problem each in their own way: operational researchers, electrical engineers, recreational mathematicians, power engineers, cyberneticists and telephony engineers all worked on various aspects of pathfinding, frequently without being aware of the contributions being made by others.

Major theoretical work has since been done which places all of these previous techniques into a single framework, using the language of graph theory and linear algebra. The mathematics that we use is shaped by the fact that it has been developed to solve particular problems. Historically, these have often been associated with new technology. Our efforts are also informed by the need to carry out practical computation, and to prove that what we have done is correct.

The same pattern can be found today. Pathfinding, or *routing*, in the context of the Internet is a demanding problem for many reasons: the network is large, diverse, under multiple ownership, and its continued operation is of immense importance. Consequently, protocols for finding 'optimal' paths down which data might flow have been developed and have accrued a great many extensions and complications; and this has outpaced traditional theory. Since we would like to design new protocols with confidence, we have a problem. Without an adequate theory, it is difficult to tell whether a proposed new technology will indeed compute what is desired, or do it efficiently; and it is difficult to alter a design that is known not to work correctly, without some guidance as to what precisely is going wrong.

We do have a substantial amount of practical experience with Internet techno-

logy, and there are theoretical results for some important aspects of routing, at least in specific cases. What is needed, however, is a way of going from the specific to the general, so that we can understand not only the technology that we have, but what we *might* have.

This thesis investigates several aspects of this goal:

1. We know several techniques for finding optimal paths when the definition of optimality is such that different network participants would nevertheless make the same choice: that is, if some node wishes to rank paths in a certain way, then it must be that all of its neighbours would want it to make the same choice. If preferences are not consistent in this way, then it still may be possible in some circumstances to find an assignment of paths to nodes that is in a Nash equilibrium. Several special cases are known, but there is no known rule for the most general case.

2. We need to understand the relationship between single-path and multipath routing in algebraic terms. In the multipath scenario, each node may select several equally-preferred paths at once. Operationally, this seems to make sense; but it is not obvious whether or how the convergence criteria for the single-path case carry across.

3. Routing preferences are often described as a combination of simpler preference schemes, but the corresponding correctness analysis has not been decomposed in the same way. We will treat the case of the *lexicographic* combination of routing algebras and derive compositional design rules.

4. It is common in networking, and ubiquitous in the Internet, to divide the network into several zones, and to have different path selection rules for intra-zone and inter-zone routes. How can we understand this algebraically, and what correctness conditions apply?

In practical terms, the aim is to find design rules that can guide the process of creating a new routing metric. Beyond this, investigation of the mathematics and its applications may yield new insight into the nature of the routing problem and our ability to analyze routing situations.

Chapter 2 introduces the fundamental mathematical definitions which will be needed throughout, as well as giving necessary background for Internet routing. It establishes several principles that are used throughout this thesis:

- Pathfinding algorithms intended for a specific setting can often be generalized to operate over some given algebraic object, which encodes information about path preference.
- Algebraic objects can often be defined compositionally in terms of simpler objects.
- Proofs about whether particular facts are true about a given object can be based on the same compositional structure.
- It is sometimes useful to remove an axiom from a definition, so long as it can be identified whether a given structure obeys that axiom or not.

This chapter also discusses the idea of a *metalanguage* that can be used to define algebras for use in routing.

Routing with multiple paths is covered in Chapter 3. This chapter shows how certain mathematical concepts, which have not previously been used in this context, are useful in understanding the problem and its solution.

An important class of algorithms are those which yield Nash equilibria: in fact, these are the standard pathfinding algorithms when used with algebras having a certain property. Some proofs are known for special cases of when this is possible. Chapter 4 presents a new proof that covers more cases and is significantly less complicated than previous efforts. It also relates previous proofs for the asynchronous message-passing setting to the standard setting that uses linear algebra.

Chapters 5 and 6 present comprehensive accounts of several important ways of building new algebras for routing. Lexicographic choice is covered in Chapter 5, including if-and-only-if deduction rules for the key algebraic properties. This is extended in Chapter 6 to an investigation of routing based on network areas; such schemes incorporate lexicographic choice but present new problems. In addition to deduction rules, this chapter looks at the Border Gateway Protocol: it shows that some operational issues observed with BGP are intrinsic to the problem of interdomain routing, and identifies certain modifications that are safe or unsafe to make to the protocol.

Some of the material in this thesis has been published elsewhere.

- Many of the definitions and theorems in Chapters 5 and 6, as well as one of the proofs in Appendix A, appear in the paper of Gurney and Griffin (2007).

- The paper of Griffin and Gurney (2008) includes material related to Chapter 4, but the main theorems of that chapter do not appear in the paper.

Both of these papers duplicate some of the definitions used in Chapter 2.

Chapter 2

Algebraic routing

Ever since routing protocols were first being designed, there has been a sense in the community that a more modular approach would serve us better than the monolithic architectures we have tended to produce. There are many ways in which this might be envisaged: modularity of design is not the same thing as modularity of implementation. The genuinely difficult task of creating router software and hardware that is 'pluggable' or 'programmable' has received much attention, and there are many gains to be made in this area even without making changes to the protocol suite. This thesis is concerned with an algebraic approach to routing, and the use of mathematical means to justify design decisions about the operation of routing protocols. That approach does not require any particular implementation to be built in a modular or programmable way, although this might still be a good idea for other reasons. Rather, it gives us a better conceptual grasp on how existing or future systems might work.

An ARPANET pioneer once wrote,

> Hopefully, a structured approach to understanding, analyzing and synthesizing routing algorithms will make the task of adapting to changing circumstances an easier one. (McQuillan 1974).

This approach is what *algebraic routing* is intended to be.

The fundamental idea is to separate the *decision process* from the *database management*. A typical routing protocol specification will contain many details about just how route information is communicated, how databases are maintained, and so on; but the 'brains' of each protocol is still the part which decides which of the given routes is to be preferred. We make a clear distinction between these concerns. Again, we are following long-established principles of protocol design. For example, McQuillan, Richer and Rosen (1980) described distributed routing as a combination of three components, for *measurement*, *dissemination*, and *calculation*. Although

dynamic response to measured network characteristics has often been found to be undesirable, distinguishing between these protocol components has clearly been a long-standing goal.

The metarouting approach is based upon a particular perception of routing. At the most basic level, we consider that the routing process and in particular the decision procedure is something that can be semantically rich. We are not limited to solving a simple numerical shortest paths problem: there is a wide choice of information that can be associated with a route, and many ways of using that information to make decisions, and those decisions can lead to more than one kind of optimal result. The fact that this complexity exists means that we need to come up with some means for managing it. This includes having a formal notation and semantics for routing problems. We will be able to use these semantics to draw conclusions about particular instances of routing problems, in a formal or even a mechanized way.

In the context of networking, the routing process exists in order to automate the creation of forwarding tables; these tables give rules for how each router is to deal with data which it receives. Forwarding does not intrinsically require routing, since tables could be partly or entirely determined manually (as was the case in early telephony). Whether manual or automatic, the routing problem is to find paths for data transit, according to some given constraints.

In the more general setting, the purpose of the computation might be different. It could be that we are seeking not simply a path, but some more complicated pattern of data transmission. Alternatively, we might just be trying to compute aggregate information about the network, not a path through it. The path-finding idea will however be taken as the main motivation.

Network management as an engineering process involves the attempt by operators to *realize* their intentions about how network traffic should behave. This is done by creating the physical network and configuring the protocol machinery. Now, operators' intentions may not be particularly precise and may not be formalized in all respects. Even if they are, it is not an easy task to translate these requirements into a network configuration: constraints of the technology may not allow a network to behave in the way an operator would like. Configuring a network can be seen as *translating* traffic engineering intentions into the 'language' of network protocols and equipment. Different systems will present different possibilities for how this might be done. In the crudest sense, if a network operator wants traffic to flow down one path rather than another, then he or she can alter the link weights so that the dynamic operation of the routing protocol will select the desired route. In practice,

especially for large networks, the intent of operators will be more nuanced and difficult to express; engineering expertise may be required to select appropriate trade-offs between conflicting high-level goals.

We will use the term *routing language* to refer to the ensemble of configuration possibilities admitted by real-world network equipment in relation to routing. This is inherently an informal characterization, because only some aspects of the technology are amenable to a formal treatment. The concept of routing language is further idealized since not all aspects of a particular network configuration can be expected to arise as the result of deliberate choices made by the operator. It nevertheless opens the possibility of comparing different routing languages with respect to their expressivity, treated separately from questions of operator intent.

We will use *routing algebras* to formalize certain aspects of routing language. A particular routing algebra will specify what data is associated with each path, and provide operations for manipulating that data (for the purpose of computing 'optimal' paths).

A *routing problem* will mean an instance of a pathfinding problem, using some routing algebra. It consists of a graph, with labelled edges over an algebra, together with a definition of 'optimal path assignment' given in terms of the symbols of the algebra. Using algebra for this is not new, although it is only comparatively recently that the scope of this approach has been broadened from the traditional semiring model; see for example Griffin, Shepherd and Wilfong (2002).

The main difference is that the algebras we will consider will themselves have been *constructed* according to a well-defined metalanguage. Each expression in this metalanguage will define some unique routing algebra. This idea is explored further in Section 2.4. The other parts of this chapter will explore the algebraic foundations of routing.

2.1 *Routing solutions and optimality criteria*

How are we to understand routing? There are several possible *points of view* that will help to explain certain aspects of the problem.

- Someone who is running a routing protocol, as a participant in the distributed route-finding process, wants to obtain the 'optimal' routes to every other participant, and to be be able to receive traffic in return. The definition of 'optimal' may vary, and different participants may have different ideas about

which routes are preferable. From this perspective, there is a lack of global knowledge or global control, (even in the case of link-state protocols). The network participants must individually trust the routing process to deliver the routing data which is desired.

- The designer of a routing protocol must lay down the rules that are to be followed by these individual participants. Great care must be taken in ensuring that the process is correct and can be correctly and efficiently implemented.
- A vendor of routing equipment is obliged to implement common protocols (or design their own), but they have considerable flexibility in implementation techniques and the provision of extensions. The correctness and efficiency of a routing protocol and its implementation are of major concern to vendors, so that their products will be attractive to customers.

To these we can add a *metarouting perspective*: we want to establish a common language and set of tools to allow discussion and solution of design problems in routing. This means that we will not always be taking the point of view of the protocol designer; rather, we are trying to support the designer in the task of creating a protocol.

A central question is: How do we know which routes are the best? Once we have established a mathematical model of route choice, we will be able to characterize what route optimality might mean, and prove that various design choices do or do not lead to optimality in all circumstances.

Our basic model of the network is a *graph*. The *nodes* of the graph represent routers, and the *arcs* are physical or logical connections between routers; see Figure 2.1. It is possible that this model will need to be altered or extended to deal with additional facets of Internet routing; for example, there is neither an inherent notion of *partitioning*, nor an architecture for *naming and addressing*. For now, we will assume that the graph model is sufficient. We will also assume that all graphs are finite—that is, that there are only finitely many nodes—and that there can be no arc from a node to itself.

In the attempt to generalize shortest-path algorithms to best-path algorithms, various mathematical structures have been proposed to replace the original use of natural numbers as path weights. However, there does not appear to be a single most-general structure which covers all known examples of best-path algorithms. For example, in Dijkstra's algorithm (discussed further in Section 2.3), there are two operations which are applied to path weights:

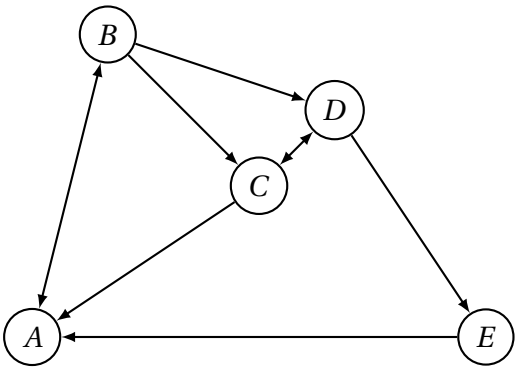


Figure 2.1 A directed graph

1. Extending the weight of a path by the weight of a new link, by adding the two numbers together.

2. Comparing the weights of two paths with (≤), to see whether a current best-path estimate should be replaced by an alternative route.

Alternatively, the comparison operation can be replaced by the use of a binary minimum operator, which returns the smaller of its two arguments. For natural number weights, as in the original formulation, these amount to the same thing: they are equivalent ways of finding the least of two numbers. If weights are not natural numbers, then this relationship may no longer hold. For example, it may not always be possible to decide which of two weights is better: in this case, some further effort is needed in the design of the algorithm, to deal with the new situation. Similarly, there may be a binary operator (such as set union) for which the result is different from either of its operands—it must be decided whether this is appropriate for the algorithm and for the problem at hand. So even for the comparatively simple algorithm of Dijkstra, there is already a difficulty in establishing a model that is intended to cover a great many potential choices for path preference.

In order to discuss these models in detail, some basic mathematical terminology will need to be introduced. The content of the definitions in the next section is standard, but the names attached to some of the definitions are not universally recognized, since they have not been used consistently by previous authors. This is indicated in each case.

Symbol	Interpretation
$\mathbb{N}$	Natural numbers (starting with zero)
$\mathbb{N}^{\infty}$	Natural numbers, plus infinity
$\mathbb{Z}$	Integers
$\mathbb{R}$	Real numbers
$\mathbb{R}_{\geq 0}$	Positive real numbers (including zero)
$\mathbb{R}^{\infty}_{\geq 0}$	Positive real numbers, plus infinity

Figure 2.2 Standard number sets

2.2 *Fundamental definitions*

We first introduce some notation.

- The constant function with value a will be denoted 'κ_a'; so $\kappa_a(b) = a$ for all b.
- Anonymous functions are written as lambda expressions, so for example '$\lambda x\,.\ x+1$' is the function that takes its argument x to $x+1$.
- If f is a function from a set A to a set B, and X is a subset of A, then we will write '$f(X)$' for the set $\{f(x) \mid x \in X\}$.
- If A is a set then '$\wp A$' denotes its power set (the set of all subsets of A).
- The symbols '$\sqcup$' and '$\sqcap$' will be used for the binary operations of numeric minimum and maximum (for example, $3 \sqcup 5 = 3 = 3 \sqcap 1$).

Figure 2.2 shows the notation that will be used for standard number sets. The asymptotic complexity symbol O has the following interpretation:

Definition 2.1. If f and g are functions from $\mathbb{N}$ to $\mathbb{R}_{\geq 0}$, then '$f \in O(g)$' or '$f(n) \in O(g(n))$' means that there exist constants $c \in \mathbb{R}_{\geq 0}$ and $k \in \mathbb{N}$ such that $f(n) \leq c \cdot g(n)$ whenever $n > k$.

2.2.1 *Semigroups*

Definition 2.2. A *semigroup* consists of a set and an associative binary operation; so, if $(S, \oplus)$ is a semigroup, then

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

for all a, b and c in S.

Definition 2.3. A semigroup $(S, \oplus)$ is *commutative* if

$$\forall a, b \in S : a \oplus b = b \oplus a.$$

Definition 2.4. If $(S, \oplus)$ is a semigroup and

$$\forall a \in S : a = a \oplus a,$$

then it is said to be *idempotent* or a *band*. A commutative idempotent semigroup is called a *semilattice*.

Definition 2.5. A semigroup $(S, \oplus)$ may have some special elements.

- If i is an element of S with $i \oplus s = s$ for all s, then i is said to be a *left identity*.
- Dually, if $s = s \oplus i$ for all s, then i is a *right identity*.
- An element that is both a left and a right identity is called simply an *identity*.
- If z is an element of S and $z = z \oplus s$ for all s, then z is called a *left zero* or *left annihilator*.
- Dually, if $s \oplus z = z$ for all s then z is a *right zero* or *right annihilator*.
- An element that is both a left and a right zero is called a *zero* or *annihilator*

If a two-sided identity exists then it is unique, and the semigroup is called a *monoid*.

Definition 2.6. A semigroup $(S, \oplus)$ is *left cancellative* if

$$\forall a, b, c \in S : c \oplus a = c \oplus b \implies a = b;$$

it is *right cancellative* if

$$\forall a, b, c \in S : a \oplus c = b \oplus c \implies a = b,$$

and *cancellative* if it is both left and right cancellative.

Definition 2.7. A semigroup $(S, \oplus)$ is *0-cancellative* if it has an annihilator 0 and

$$\forall a, b, c \in S : ((c \oplus a = c \oplus b \neq 0) \vee (a \oplus c = b \oplus c \neq 0)) \implies a = b$$

Definition 2.8. A semigroup $(S, \triangleleft)$ is a *left zero semigroup* if every element is a left zero; that is, if $a \triangleleft b = a$ for every a and b in S. Similarly, $(S, \triangleright)$ is a *right zero semigroup* if every element is a right zero.

In this thesis, the symbols '$\lhd$' and '$\rhd$' will always stand for the binary operators

$$a \lhd b \stackrel{\text{def}}{=} a \tag{2.1}$$

$$a \rhd b \stackrel{\text{def}}{=} b \tag{2.2}$$

respectively.

Definition 2.9. A semigroup $(S, \oplus)$ is *left condensed* if $a \oplus b = a \oplus c$ for all a, b and c in S. Similarly, it is *right condensed* if $b \oplus a = c \oplus a$ for all a, B and c in S.

Note that a left zero semigroup is always left condensed, but the converse is not necessarily true. Likewise, a right zero semigroup is always right condensed, but there are right condensed semigroups that are not right zero semigroups.

Definition 2.10. A *bisemigroup* is a set S together with two associative binary operations $\oplus$ and $\otimes$; and so $(S, \oplus)$ and $(S, \otimes)$ are both semigroups.

Definition 2.11. A *semiring* is a bisemigroup $(S, \oplus, \otimes)$ with the additional properties that:

1. $\oplus$ is commutative;
2. $\otimes$ distributes over $\oplus$: $c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$ for all a, b and c in S;
3. $\oplus$ has an identity, which is also an annihilator for $\otimes$;
4. $\otimes$ has an identity.

Several related structures have been studied, sharing the same fundamental idea of a set with two related operations, but differing in the axioms which are required. The terminology is not fully standard: the name 'semiring' has been applied not only to the definition above, but also to various other definitions with subtle differences. A comprehensive account of the terms used in the literature is given by Głazek (2002).

If $(S, \oplus, \otimes)$ is a semiring, then we can form the semiring $(M_n(S), \oplus, \cdot)$ of n by n matrices over S, where

$$(A \oplus B)_{ij} \stackrel{\text{def}}{=} A_{ij} \oplus B_{ij}$$

$$(A \cdot B)_{ij} \stackrel{\text{def}}{=} \bigoplus_{k \in N} (A_{ik} \otimes B_{kj}).$$

The semiring axioms are straightforward to verify. The identity for $(M_n(S), \oplus)$ is the matrix all of whose entries are ∞, where ∞ is the identity for $(S, \oplus)$. The identity for $(M_n(S), \cdot)$ is the matrix I given by

$$I_{ij} = \begin{cases} 1 & i = j \\ \infty & i \neq j, \end{cases}$$

where 1 is the identity for $(S, \otimes)$.

Definition 2.12. The *closure* of a matrix A is given by

$$A^* \stackrel{\text{def}}{=} I \oplus A \oplus A^2 \oplus A^3 \oplus \cdots = \bigoplus_{k \geq 0} A^k.$$

This solves the fixed-point equation $X = A \cdot X \oplus I$, just as in regular languages $a^* = \epsilon + a + aa + aaa + \cdots$ solves $x = ax + \epsilon$.

2.2.2 *Order theory*

Definition 2.13. A *relation* on a set S is a subset of $S \times S$. If R is a relation on S, we write '$a\ R\ b$' for '$(a, b) \in R$'.

Definition 2.14. A binary relation $\preceq$ on a set S may have some, all, or none of the following properties:

reflexivity	$a \preceq a$	(2.3)
symmetry	$a \preceq b \implies b \preceq a$	(2.4)
antisymmetry	$(a \preceq b \wedge b \preceq a) \implies a = b$	(2.5)
transitivity	$(a \preceq b \wedge b \preceq c) \implies a \preceq c$	(2.6)
totality	$a \preceq b \vee b \preceq a$	(2.7)

where each free variable is universally quantified.

Definition 2.15. Some relations with particular combinations of properties have special names:

preorder	reflexive, transitive
equivalence relation	reflexive, symmetric, transitive
partial order	reflexive, antisymmetric, transitive
preference relation	reflexive, transitive, total
linear order	reflexive, antisymmetric, transitive, total.

From a given preorder, there are several useful derived relations which can be constructed. If $(\preceq_S)$ is a preorder on S then $(\prec_S)$, $(\sim_S)$ and $(\#_S)$ are defined as follows:

$$a \prec_S b \overset{\text{def}}{\iff} a \preceq_S b \wedge \neg(b \preceq_S a) \tag{2.8}$$

$$a \sim_S b \overset{\text{def}}{\iff} a \preceq_S b \wedge b \preceq_S a \tag{2.9}$$

$$a \#_S b \overset{\text{def}}{\iff} \neg(a \preceq_S b) \wedge \neg(b \preceq_S a). \tag{2.10}$$

The $(\prec_S)$ relation is the strict version of $(\preceq_S)$, indicating when one element is preferred to another and they are *not* equivalent. The relation $(\sim_S)$ provides the equivalence: note that $(\preceq_S)$ arises as the union of $(\prec_S)$ and $(\sim_S)$. Finally, $(\#_S)$ indicates when two elements are incomparable, meaning that neither is preferred to the other. The distinction between equivalence and incomparability is subtle: the former typically means that elements are of equal quality, whereas the latter means that their relative merits cannot be assessed.

Definition 2.16. A *chain* in a set S, with a binary relation $(\preceq)$, is a subset A of S for which the restriction of $(\preceq)$ to A is a linear order.

Definition 2.17. An *antichain* in a set S, with a binary relation $(\preceq)$, is a subset A of S such that for all distinct a and b in A, $\neg(a \preceq b)$.

Definition 2.18. Let $(S, \preceq)$ be a preorder. S satisfies the *descending chain condition (DCC)* if there is no infinite chain $s_1 > s_2 > s_3 > s_4 > s_5 > \cdots$ of elements of S. If $(S, \preceq)$ satisfies the DCC then we say that $(\preceq)$ is *well-founded*.

The DCC guarantees that every subset of S has a minimal element—perhaps more than one. Here, $a \in A \subseteq S$ is minimal in A if and only if there is no b in A such that $b < a$. Conversely, any preorder in which every subset has a minimal element satisfies the DCC.

Theorem 2.1. *Let $(S, \oplus)$ be a semilattice. Then*

$$a \preceq^L b \overset{\text{def}}{\iff} a = a \oplus b$$

$$a \preceq^R b \overset{\text{def}}{\iff} b = b \oplus a$$

are partial orders, and they are dual to one another.

Proof. We first show that the relations are dual:

$$a \preceq^{\mathrm{L}} b \iff a = a \oplus b \iff b \preceq^{\mathrm{R}} a.$$

Now, if we can prove that $(\leq^{\mathrm{L}})$ is a partial order, then it will follow that its dual $(\leq^{\mathrm{R}})$ is also a partial order. We need to prove reflexivity, antisymmetry and transitivity.

- For all a, because S is idempotent, we have $a = a \oplus a$; so $(\leq^{\mathrm{L}})$ is reflexive.
- Suppose that $a \leq^{\mathrm{L}} b$ and $b \leq^{\mathrm{L}} a$. Then $a = a \oplus b$ and $b = b \oplus a$. Because S is commutative, $a \oplus b = b \oplus a$, so $a = b$ and $(\leq^{\mathrm{L}})$ is antisymmetric.
- Suppose that $a \leq^{\mathrm{L}} b$ and $b \leq^{\mathrm{L}} c$. Then $a = a \oplus b$ and $b = b \oplus c$. Now, $a \oplus c = (a \oplus b) \oplus c = a \oplus (b \oplus c) = a \oplus b = a$, so $a \leq^{\mathrm{L}} c$. This shows transitivity.

Therefore, both relations are partial orders. □

These two partial orders are called the *natural orders* derived from $\oplus$. We will normally use the left natural order, $\leq^{\mathrm{L}}$. There are alternative definitions of natural order which apply to other kinds of semigroup, but for the purposes of this thesis we will only need to use the definition for semilattices.

Definition 2.19. If x and y are elements of a preorder $(S, \preceq)$ and $z \preceq x$ and $z \preceq y$, then z is called a *lower bound* of x and y. If z is the maximal element of the set of lower bounds of x and y, then it is called the *greatest lower bound*. An order in which any two elements have a greatest lower bound is called *complete*.

Theorem 2.2. *Let $(S, \leq)$ be a complete partial order and define $\oplus$ so that $a \oplus b$ is the greatest lower bound of a and b. Then $(S, \oplus)$ is a semilattice, and the left natural order over it is $(\leq)$ itself.*

Proof. The operation $\oplus$ is certainly associative, commutative and idempotent, by definition of greatest lower bounds; and the fact that greatest lower bounds always exists ensures that it is well-defined. Now,

$$\begin{aligned} & a = a \oplus b \\ \iff & a \leq a \wedge a \leq b \wedge (c \leq a \wedge c \leq b \implies c \leq a) \\ \iff & a \leq b \end{aligned}$$

so the left natural order coincides with $(\leq)$. □

Definition 2.20. An equivalence relation $(\sim)$ on a semigroup $(S, \oplus)$ is a *congruence* if

$$\begin{aligned} a \sim b \implies \quad & (a \oplus c) \sim (b \oplus c) \\ & \wedge (c \oplus a) \sim (c \oplus b) \end{aligned}$$

for all a, b and c in S.

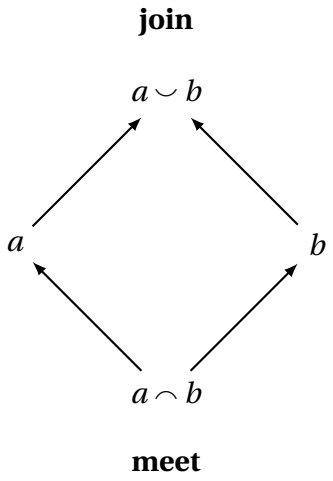


Figure 2.3 Meets and joins in a lattice.

Definition 2.21. If $(\sim)$ is a congruence on $(S, \oplus)$ then we can form the *quotient* $(S/\sim, \oplus')$, which is a semigroup. Its elements are the equivalence classes of S. If ρ is the function taking an element to its equivalence class, then

$$\rho(a) \oplus' \rho(b) \stackrel{\text{def}}{=} \rho(a \oplus b)$$

for all a and b in S: because $(\sim)$ is a congruence, this is sufficient to define $(\oplus')$ and ensure that $S/\sim$ is a semigroup.

Definition 2.22. A partial order is a *lattice* if any two elements have a greatest lower bound and a least upper bound. Alternatively: $(S, \frown, \smile)$ is a lattice if $(S, \frown)$ and $(S, \smile)$ are semilattices. Then $\frown$ is called the *meet* and $\smile$ the *join* operation—see Figure 2.3.

Definition 2.23. A partial order is *bounded* if it has a least element and a greatest element.

Definition 2.24. A lattice $(S, \frown, \smile)$ is *distributive* if

1. $c \frown (a \smile b) = (c \frown a) \smile (c \frown b)$ and
2. $c \smile (a \frown b) = (c \smile a) \frown (c \smile b)$

for all elements a, b and c of S.

Definition 2.25. An element c of a semilattice $(S, \oplus)$ is *prime* if and only if

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

and c is not the identity of S. Equally, an element of a lattice $(S, \frown, \smile)$ is *meet-prime* if it is prime in the semilattice $(S, \frown)$.

Definition 2.26. In a partial order $(S, \leq)$, an *upper set* is any subset A of S which is *upward closed*; that is, if x is in A and $x \leq y$, then y is also in A.

Note that the empty set is, vacuously, an upper set. The upper sets of an order are closed under the operations of union and intersection, and therefore form a distributive lattice.

Definition 2.27. In a partial order $(S, \leq)$, a *filter* is a subset A of S which is an upper set, and is closed under taking greatest lower bounds: so if x and y are in A, then $x \frown y$ exists and is in A.

Definition 2.28. For an element x of a partial order $(S, \leq)$, the set

$$\uparrow x \stackrel{\text{def}}{=} \{y \in S \mid x \leq y\}$$

is the *principal filter* generated by x. Since this is necessarily an upper set, we will still call $\uparrow x$ the *principal upper set* in cases when S does not have greatest lower bounds.

If A is a subset of $(S, \leq)$ then take $\uparrow A$ to be the union of all sets $\uparrow x$ for x in A. Figure 2.4 illustrates this definition.

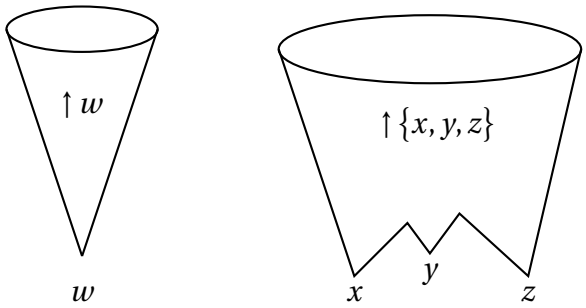


Figure 2.4 An element and its upper set; an upper set that is the union of three principal upper sets.

Definition 2.29. If $(S, \leq)$ is a partial order, and $(S, \leq_L)$ is a linear order such that

$$\forall x, y \in S : x \leq y \implies x \leq_L y$$

then $(\leq_L)$ is a *linear extension* or *linearization* of $(\leq)$.

Theorem 2.3. *Every partial order has a linearization. Furthermore, if $x \# y$ in a partial order $(S, \leq)$ then there is a linearization $(\leq_L)$ of $(\leq)$ in which $x <_L y$.*

Proof. This was proved by Szpilrajn (1930). □

2.2.3 *Graphs*

Definition 2.30. A *graph* consists of a set N of *nodes* together with a binary relation E on N. Elements of E are *arcs*.

If E is symmetric then the graph is *undirected*; otherwise it is *directed*. A graph is *simple* if E is irreflexive.

If i is a node, then write

$$iE \stackrel{\text{def}}{=} \left\{ j \in N \;\middle|\; (i,j) \in E \right\} \tag{2.11}$$

$$Ei \stackrel{\text{def}}{=} \left\{ j \in N \;\middle|\; (j,i) \in E \right\} \tag{2.12}$$

for the sets of nodes to which i is connected by outgoing and incoming arcs respectively.

Definition 2.31. If $G = (N, E)$ is a graph, then a sequence $[n_1, n_2, \ldots, n_k]$ is a *path* if $n_i \; E \; n_{i+1}$ for all i with $1 \leq i < k$. It is a *cycle* if $n_1 = n_k$. It is a *simple* path if all nodes are distinct.

Definition 2.32. A *weighted* graph is a graph equipped with a function $w : E \longrightarrow S$, where S is some set.

The set S which provides the arc weights will probably have defined over it either an order (indicating preference) or a binary operator (encoding preference-based choice). If $(S, \preceq)$ is a preorder, then it is always possible to adjoin a least-preferred element, if one is not already present: take $S' \stackrel{\text{def}}{=} S \cup \{\infty\}$ and $(\preceq') = (\preceq) \cup \left\{ (x, \infty) \;\middle|\; x \in S' \right\}$, where ∞ is not in S. Then ∞ is the topmost or maximal element of the new order.

Likewise, if $(S, \oplus)$ is a semigroup, then a new two-sided zero can be adjoined, which in the natural order derived from $(\oplus)$ will be the greatest element. If this is done, then we can take w to be defined on the whole of $N \times N$, with $w(i,j) = \infty$ if (i,j) is not in E.

Definition 2.33. If $G = (N, E)$ is weighted by $w : E \longrightarrow (S, \otimes)$, then the *weight* of a path $[n_1, n_2, \ldots, n_k]$ is given by

$$w(n_1, n_2) \otimes w(n_2, n_3) \otimes \cdots \otimes w(n_{k-1}, n_k).$$

We write $w(p)$ for this expression if p is a path.

Definition 2.34. If $G = (N, E)$ is weighted by $w : E \longrightarrow (S, \preceq)$, then a path p in G is a *shortest path* if there is no path q, having the same first node and the same last node as p, for which $w(q) \prec w(p)$.

Likewise, if w is a function into $(S, \oplus)$, then p is a shortest path if there is no path q, having the same first node and the same last node as p, for which $w(q) = w(q) \oplus w(p)$.

Definition 2.35. The *adjacency matrix* of a graph $G = (N, E)$ is the N by N matrix A with $A_{ij} = 1$ if iEj, and $A_{ij} = 0$ if $\neg(iEj)$, for all i and j in N. (In fact, this is just a concrete representation of the E relation.)

If G is weighted by $w : E \longrightarrow S$, and S has a two-sided zero ∞ then the adjacency matrix A of G is given by

$$A_{ij} = \begin{cases} w(i,j) & iEj \\ \infty & \neg(iEj) \end{cases}$$

for all i and j in N.

It is also possible to use functions as the arc labels of a graph. These are composed along each path, but they may also be applied to values originated at the source node. Algebras like this can be defined over semigroups or over orders; here, these are called *semigroup transforms* and *order transforms* respectively. Semigroup transforms include the *algebras of endomorphisms* of Gondran and Minoux (2001), but are more general since the functions here are not required to be endomorphisms of the given semigroup. The class of order transforms is similar to the algebras used by Sobrinho (2005) and Griffin and Sobrinho (2005).

Definition 2.36. A structure $(S, \oplus, F)$ is a semigroup transform if $(S, \oplus)$ is a semigroup and F is a set of functions from S to S.

Definition 2.37. A structure $(S, \preceq, F)$ is an order transform if $(S, \preceq)$ is a preorder and F is a set of functions from S to S.

Both of these structures are related to *actions* over semigroups or orders. An action of a semigroup $(A, \otimes)$ on a set X is an operation

$$\star : A \times X \longrightarrow X$$

for which $(a \otimes b) \star x = a \star (b \star x)$ for all a and b in A and x in X; there are similar conditions for actions of other structures. See Kilp, Knauer and Mikhalev (2000) for a thorough account of monoid actions.

The canonical example is that A might be a set of endofunctions on X, with $(\otimes)$ being composition and $(\star)$ being function application. This is essentially what the definition of a semigroup transform provides, though the action is on a semigroup $(S, \oplus)$ rather than on a set. Semigroup transforms also incorporate bisemigroups (and semirings) since another standard example is a semigroup acting on itself by multiplication: $(\star)$ and $(\otimes)$ are the same, as are A and X. From the function viewpoint, a semigroup transform is constructed from a bisemigroup by choosing F to be

$$\{\lambda y \,.\, x \otimes y \mid x \in S\}.$$

The relationship between order semigroups and order transforms is similar.

2.3 *Algorithms and properties*

It is difficult to identify the origin of pathfinding algorithms. The task is somewhat like trying to find the source of a river; there are many beginnings, and in many cases it is not obvious whether the earliest research should be counted as part of the field as it is currently recognized. It is safe to say that methods for finding optimal paths in weighted graphs have been a topic of serious investigation since at least the 1950s; related work was carried out during the previous several decades, but did not at that stage form part of a unified research agenda.

This research has produced algorithms in two families, which we here refer to as the *Dijkstra* and *Bellman-Ford* traditions. A third strand of research is associated with Stephen Kleene's work on regular languages and automata (Kleene 1956); the construction of the deterministic finite automaton for a given regular expression, has since been reinterpreted as a special case of the Floyd-Warshall algorithm (Rote 1985). It is now recognized that all of these techniques can be seen as solving the same 'shortest paths' problem, even though they originate from different domains of application. In recent years, work on the shortest path problem has concentrated on three main areas:

1. Techniques for the efficient implementation of shortest-path algorithms, especially for the special case in which path weights are natural numbers with a known upper bound. There are also special algorithms using a heuristic which are applicable for certain kinds of navigational problems.

2. Generalized or algebraic treatments of the problem, resulting in generic 'best path' algorithms, with finding the shortest path as one special case among

many other interpretations, dependent upon the choice of algebra.

3. Implementation of shortest-path or best-path algorithms in a distributed fashion, and associated problems in complexity and network dynamics.

There are other path problems which are related to finding the best path. These include:

- The *minimum spanning tree* problem: given a connected weighted graph, find a subgraph which connects all of the nodes (a *spanning tree*) and which has minimal weight among all spanning trees.
- The *Steiner tree* problem: given a weighted graph G and a subset M of its nodes, find a subgraph of G that spans M, and has minimal weight among all such subgraphs. The geometric Steiner problem is related: given several points in a metric space, find a minimal tree which connects them all.
- The *Hamiltonian path* problem: find a path which connects all nodes, does not pass through any node more than once, and which has minimal weight. A related problem is to find a cycle with the same properties.
- The *Eulerian cycle* problem: find a cycle in which each arc in the graph appears exactly once.

Although these can look quite similar to the best-path problem, this is deceptive: their complexity properties are different, as are the algorithmic techniques used to solve them.

We will now discuss the well-known Dijkstra and Bellman-Ford algorithms in detail. The algorithm of Dijkstra (1959), a refinement of the method of Moore (1959), is essentially combinatorial in nature. Dijkstra's insight was that it is possible to find shortest paths (at least when arc weights are natural numbers) by examining each arc once and only once, if they are taken in the correct order. This is only possible when arcs of negative weight cannot occur (Cormen, Leiserson and Rivest 1990). See Figure 2.5.

The efficiency of Dijkstra's algorithm depends on the method used to select the next node from the queue of unprocessed nodes. There is a known equivalence between priority queues and sorting, the most complete version of which is due to Thorup (2007). This equivalence states that if n keys can be sorted in $S(n)$ time per key, then a priority queue can be built for which extraction of the minimal element

```
d(0) := 0
for n in N \ {0}:
    d(n) := ∞

Q := N
while Q is not empty:
    choose i from Q so that d(i) ≤ d(j) for any j in Q
    Q := Q \ {i}
    for each j in iE:
        d(j) = min{d(j), d(i) + w(i, j)}
```

Figure 2.5 Pseudocode for Dijkstra's algorithm

takes $O(S(n))$ time, and vice versa. The resulting complexity for Dijkstra's algorithm will be $O(|E| + |N|\,S(|N|))$.

For example, since pure comparison-based sorting of n keys is in $O(n \log n)$, the most generic form of Dijkstra's algorithm has worst-case running time in $O(|E| + |N| \log |N|)$. This was achieved by Fredman and Tarjan (1987) using Fibonacci heaps. Note that if the graph is dense, so that $|E|$ is close to $|N|^2$, then a naive linked-list implementation with running time in $O(|N|^2)$ has comparable asymptotic performance to the Fredman-Tarjan version.

The running time can be reduced even further if arc weights are integers and an upper bound on path weight is known, as non-comparison-based methods can then be used; for example, the algorithm of Fredman and Willard (1994) has $S(n) = \log n / \log\log n$, and that of Ahuja et al. (1990) has $S(n) = \sqrt{\log C}$ where C is the maximum possible path weight. Which of these two methods is faster depends on the relationship between $|N|$ and C. Similarly, use of randomization, hashing, pointer arithmetic and other techniques can yield better complexity bounds, but at the cost of only being applicable to more specific versions of the shortest-path problem.

Dijkstra's algorithm can only be correctly applied when no arc weight is negative. If negative arcs are possible, but there are no negative-weight cycles, then the Bellman-Ford algorithm can be used instead. This was developed separately by Ford and Fulkerson (1956) and by Bellman (1958). See Figure 2.6 for the pseudocode.

These algorithms are only for finding *single-source* shortest paths, as opposed to shortest paths between *all pairs* of nodes. One could run an algorithm repeatedly, once for each node in the graph, in order to obtain shortest paths for each origin-destination pair. There are various other ways in which the all-pairs solution can be

```
d(0) := 0
for n in N \ {0}:
    d(n) := ∞

repeat |N| − 1 times:
    for each (i, j) in E:
        d(j) := min{d(i), d(i) + w(i, j)}
```

Figure 2.6 Pseudocode for the Bellman-Ford algorithm

```
for each (i, j) in N × N:
    if (i, j) is in E then:
        d(i, j) := w(i, j)
    else:
        d(i, j) := 0

for each k in N:
    for each i in N:
        for each j in N:
            d(i, j) := min{d(i, j), d(i, k) + d(k, j)}
```

Figure 2.7 Pseudocode for the Floyd-Warshall algorithm

computed more efficiently. One of these is the algorithm of Floyd (1962), Warshall (1962) and Roy (1959), shown in Figure 2.7. Johnson's algorithm is a combination of the Dijkstra and Bellman-Ford algorithms, using an initial Bellman-Ford run to compute a re-weighted graph that is suitable as input for Dijkstra's algorithm (Johnson 1977). All of these techniques seem superficially quite similar to one another, and indeed there is a way of perceiving all shortest-path algorithms as specializations of a single 'ur-algorithm' based on matrix multiplication. That is, each of the named algorithms we know about can be thought of as carrying out the same calculation, but in a way that is specialized for some particular class of shortest-path problems where certain optimizations apply that cannot be made in the general case.

The modern synthesis of shortest-path algorithms treats the underlying problem in terms of linear algebra. Recall that every graph can be represented by its adjacency matrix. The operation of finding the shortest path can be carried out by applying a

certain matrix iteration based on the adjacency matrix of the graph:

$$\sigma : X \mapsto I \sqcup (A \cdot X)$$

where A is the adjacency matrix and I is the identity matrix. It can be shown that $\sigma^k(I)_{ij}$ contains the length of the shortest path between i and j among all paths of length at most k. Because we are limited to simple paths, the matrix $\sigma^{|N|}(I)$ must contain the shortest paths among all paths of any length. We have

$$\sigma^k(I) = I \sqcup A \sqcup A^2 \sqcup \cdots \sqcup A^{k-1} \sqcup A^k$$

and

$$\sigma^{|N|} = A^* = \bigsqcup_{h \geq 0} A^h.$$

Here, the matrix addition and multiplication operations were defined in terms of underlying operations on natural numbers—addition, and binary minimization. We can replace these by other operations, and also change the type of matrix elements, to obtain a new algorithm that operates according to the same principle, but with a different notion of 'best path'. This will be discussed further in the next section.

2.3.1 *Genericized algorithms*

The replacement of $(\mathbb{N}, \min, +)$ by a semiring in the solution of fixed-point matrix equations was first done by Carré (1971). Previously-known algorithms for solving path problems could then be reinterpreted in terms of linear algebra: the Bellman algorithm is Jacobi elimination; the Ford-Fulkerson algorithm is Gauss-Seidel elimination; and the Floyd-Warshall algorithm is Jordan elimination. This connection suggested that better algorithms for the shortest path problem could be developed based on known techniques for solving matrix equations; for example, Rote (1985) discovered a systolic algorithm based on observations about the flow of computation in Gauss-Jordan elimination.

Whereas Carré considered only idempotent semirings, Lehmann (1977) was able to drop this requirement and therefore make available a wider variety of instances of the path problem. In particular, the problem of counting the number of paths between each origin-destination pair can be solved by using the semiring $(\mathbb{N}, +, \times)$, which is not idempotent, in the familiar matrix algorithm, if all arcs are given weight 1. Other counting problems admit solutions by means of related non-idempotent semirings.

In a similar way, the standard single-path algorithms can be extended in various ways to deliver multiple paths. These may or may not be of equivalent cost. An example of the latter is the problem of finding the list of the best k paths for each source-destination pair, for some fixed k. Extensions to the algebraic system to deal with multiple paths have been attempted by Wongseelashote (1979) and Mohri (2002), among others, although there is no general theory of these problems other than the standard semiring approach.

Accounts of these generic matrix-based algorithms have been given by Carré (1979), Rote (1990), and Gondran and Minoux (1984, 2001), among others. A detailed history of research on this topic appears in Chapter 8 of Zimmermann (1981), covering especially the period from 1950 to 1980.

The precise list of algebraic properties demanded from the semiring structure is not standard in the literature. Indeed, there is an alternative development of the algebraic theory that uses ordered semigroups instead: some care is needed to untangle exactly which axioms are needed for which algorithmic applications. Note that for the purpose of *running* an algorithm, such as the matrix iteration method, it is only necessary for operators of the appropriate types to be present; but in order to ensure that the computation will terminate with the desired result, the algebraic structure needs to satisfy certain additional properties.

This fact only becomes a problem once we start trying to construct algebraic systems that have more complicated behaviour, and whose properties are consequently more difficult to verify. It is therefore important to make certain of which properties are required and for what reason. If we can prove that some property is not needed after all, then we not only save ourselves the effort of verifying that condition, but we also make it possible to use a wider variety of concrete algebras with confidence. More subtly, we know that a structure that satisfies all of the semiring axioms *except* distributivity can still be used in the iteration algorithm (since we still have two binary operations over the same set), but computes a different kind of result: rather than obtaining the best paths between each pair of nodes, it finds a stable solution among path assignments.

This can be explained as a Nash equilibium: a state from which no player can improve by deviation. In this context, the players are the nodes, each of which has their own preferences among possible paths. They are only allowed to choose routes which are consistent with the choices made by their neighbours. An assignment of paths to nodes is stable when no node can choose a better path from the candidates made available as a result of the other nodes' choices.

A further wrinkle with the distributivity property is that it seems to be 'fragile', meaning that it may not be preserved by several of the semiring-based constructions we would like to carry out. Given the importance of this property in determining the kind of solution which the iterative algorithm produces, it is essential to provide a complete account of how this and other properties are related to algebraic constructions.

The property of *distributivity* for a semiring is related to several other properties for other structures. In a semiring $(S, \oplus, \otimes)$, distributivity is the requirement that

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

for all a, b and c. This is clearly similar to the criterion for a function f over $(S, \oplus)$ to be a homomorphism,

$$f(a \oplus b) = f(a) \oplus f(b),$$

for all a and b. Indeed, this homomorphism property is the exact analogue of distributivity when dealing with monoid endomorphisms (Gondran and Minoux 2001) or with semigroup transforms (Definition 2.36).

In order theory, there is the property of an order semigroup $(S, \preceq, \otimes)$ that

$$a \preceq b \implies c \otimes a \preceq c \otimes b$$

for all a, b and c. This property will be referred to as *monotonicity*; it has previously also been called *isotonicity* (Griffin and Sobrinho 2005; Sobrinho 2005). This is of interest because there are several constructions relating monotonic order semigroups to distributive semirings (see for example Theorem 3.6). It has also been related to algorithmic convergence in its own right, in the case of linear orders (Cormen, Leiserson and Rivest 1990).

The name 'monotonicity' has alternatively been applied to the property

$$\forall a, c : a \preceq c \otimes a$$

or its strict variant

$$\forall a, c : a \prec c \otimes a.$$

The latter property will be called *increasing* and the former *nondecreasing*. These are related to the existence of local optima, or Nash equilibria, even in the absence of distributivity. This connection is explored in Chapter 4.

For symmetry between the order and semigroup cases, the distributivity property will sometimes be referred to as monotonicity, to allow the same name to be used in proofs and arguments which relate to both structures.

2.3.2 *Influence of the Internet*

The development of computer networks provided a further impetus to the study and use of generic best-path algorithms. The design of the ARPANET routing algorithm was directly inspired by the work of Carré, and various semiring structures were considered for modelling the notion of 'best path' in this context (McQuillan 1974; McQuillan, Richer and Rosen 1980). It was not at all clear how the various attributes of a network path—such as end-to-end delay, bandwidth, reliability, and router processing speed—could be summarized so as to make it clear when one path is better than another, and consequently many possible designs had to be explored. This was effectively abandoned in the eventual protocol, which had a single numeric metric for 'cost'; it was up to the individual network operator to set this value appropriately based on whatever attributes he or she considered important. In addition, most work on shortest path algorithms had concentrated on methods that could be run on a single computer, and where the graph was unchanging during the program's run; in a routing context, however, there are compelling reasons to have the computation be distributed rather than centralized, and able to adapt dynamically to changing network conditions.

There are three general mechanisms by which routing protocols compute their routes.

1. **Link state**. Participants in the protocol exchange information about *all* arcs in the network and their attributes; consequently, each participant is able to build the same internal map of the network. Conventional shortest-path algorithms can be run against this map, and the paths obtained by each participant will be consistent since they are working with the same information.

2. **Distance vector**. In this model, participants send and receive information about destinations and their associated least-cost reachability estimates. In contrast to the link-state model, no information about network topology is (explicitly) propagated. Instead, each participant maintains a vector of the current estimated cost for reaching each destination. Updates from neighbours may result in this estimate being updated, and the new vector can then be transmitted onwards.

3. **Path vector**. The path vector model is essentially the same as the distance vector model, the difference being the information that is associated with each destination: in path vectoring, it is a route through the network, as opposed to

the *weight* of the route. Maintaining an explicit path is a way to avoid loops, because prospective extensions to the path can be checked to ensure that a loop is not created—this is not possible if only the cost of the path is available.

In practice, it may not be possible to assign a particular protocol's mode of operation to just one of these categories.

The growth of computer networks made issues of scalability more acute. Convergence time for distributed shortest path protocols can be exponential in the number of nodes in the network. The nascent Internet was not only larger, but more diverse than before: a single definition of 'best path' could no longer be enough to satisfy the conflicting demands of different network operators. The concept of a 'network of networks', made up of several participants each with their own systems and local administrative control, was a natural fit for a new routing technology, the split between exterior and interior routing. The original Exterior Gateway Protocol (EGP) operated by assuming that participant networks had their own distinctive routing protocols, and were joined together in a tree: EGP would manage the connections between these autonomous systems, while within each network a separate protocol would connect border routers and local systems, without any knowledge of how the outside world was structured (Rosen 1982).

EGP was only ever intended as a stopgap until a new and better protocol could be developed for connecting the Internet's autonomous systems. The first version of the Border Gateway Protocol (BGP) was adopted as a standard in 1990, following previous experimental deployment (Lougheed and Rekhter 1989, 1990); the version currently ubiquitous for the Internet is BGP-4 (Rekhter and Li 1995). There have been many other alterations and extensions to BGP and related routing technology since 1995; some of these have been included in an updated version of the BGP-4 standard (Rekhter, Li and Hares 2006).

BGP is notable for the degree of flexibility it permits in network configuration, even disregarding the existence of some extensions which have been standardized. Router vendors may also offer their own modified versions of the protocol.

BGP is an example of a path vector protocol, but unlike previous algorithms and protocols, it does *not* solve the shortest path problem. Instead, it finds solutions to the *stable paths problem*, a related but distinct combinatorial problem. This is discussed in detail in Chapter 4.

Aside from BGP, the surviving routing protocols in the present Internet are:

1. **Routing Information Protocol (RIP)**. This is a distance vector protocol using a simple numeric 'cost' metric. The first version of the protocol was standardized in RFC 1058 (Hedrick 1988), and a second version in RFC 2453 (Malkin 1998). There is also a 'next generation' RIP for IPv6 defined in RFC 2080 (Malkin and Minnear 1997).

2. **Open Shortest Path First (OSPF)**. This is a hybrid protocol: OSPF divides the network into *areas* connected by a *backbone*; within each area, pathfinding uses a link-state mechanism, but the exchange of routing information along the backbone uses a distance-vector method. The original protocol was described in RFC 1131 (Moy 1989), and by Coltun (1989). A second version was standardized as RFC 2328 (Moy 1998b), and a third version, for IPv6, as RFC 5340 (Lindem et al. 2008). The most comprehensive description of OSPF is in the two books by Moy (1998a, 2000).

3. **Intermediate System to Intermediate System (IS-IS)**. This is a link-state protocol. IS-IS was initially developed as an OSI standard, and was published as ISO/IEC 10589 in 1992; a draft of this document appeared as RFC 1142 (Oran 1990). This was later updated as RFC 1195 (Callon 1990) for use in TCP/IP networks; the ISO/IEC document has received several corrections and was most recently updated in 2002.

4. **Enhanced Interior Gateway Routing Protocol (EIGRP)**. EIGRP is the incompatible replacement for the (non-Enhanced) IGRP, both of which are Cisco-proprietary rather than open standards. They both use the same metric for route comparison, but otherwise operate differently. EIGRP uses the DUAL algorithm to establish best routes while avoiding transient loops (Garcia-Luna-Aceves 1993). It is described in Cisco Systems White Paper 16406.

There is a considerable engineering effort associated with ensuring that these distributed protocols operate properly, even aside from the details of the best-path computation being performed (Bertsekas and Gallager 1992). Another important consideration is that the route computation should be consistent with the network's forwarding regime. Internet forwarding is hop-by-hop: the sender of data does not control the entire route, but can only pass it on to some neighbour indicated by the routing computation. Consequently, the design of an Internet routing protocol

should make sure that the computed routes do indeed correspond to the path that would be taken by forwarded data (Feamster and Balakrishnan 2005).

This is related to the problems in routing theory of understanding what kinds of best-path computation are possible, what their performance characteristics are, and whether successful termination can be guaranteed. It is possible to reason about these in the abstract, just as is done with shortest-path problems, orthogonally to the distributed computation issue. For example, Griffin, Shepherd and Wilfong (2002) define both the stable paths problem as a mathematical object, and a simple path-vector protocol which can solve (certain) stable paths problems in a distributed or centralized fashion. Along the same lines, Sobrinho (2005) considers an asynchronous message-passing algorithm for solving a path problem given in algebraic terms.

2.4 *Metalanguages*

The previous section introduced the split between algebra and algorithm in the solution of path problems. Various kinds of path problems can be presented, with the nature of a solution being couched in terms of the algebra of path weights. Several methods are available for computing such solutions, but in order for these to apply it may be necessary to ensure that the algebra has some additional properties.

The promise of *metarouting* is to provide a rigorous means of defining the algebras which underlie path problems, so that their correctness properties can be automatically inferred (Griffin and Sobrinho 2005). This is associated with the goal of providing genericized routing algorithms, which can be instantiated with any algebra having the appropriate properties. Such a system would greatly simplify the engineering task of implementing a new routing protocol.

The defining characteristic of the metarouting approach, in contrast to previous work and in support of the implementation requirement, is the use of a *metalanguage* to define algebras. The metalanguage provides structure on which both theory and implementation can rely. Each expression in the metalanguage serves to define an algebra. There are some 'base' algebras built in, together with a collection of combinators for making new algebras from old. The term 'algebra' here refers to any of the mathematical objects used in the study of path problems (including semirings, ordered semigroups, and so on). Incorporation of several structures allows a wider range of constructions to be expressed than if only a single kind of algebra were present. The presence of multiple types of structure does not compromise the property deduction system, since required properties can be reformulated for each

case, but it does result in an increase in the number of deduction rules that need to be proved.

For the theory, use of a metalanguage gives us some structure to the algebras, and we can exploit this in our proofs. We need to be able to infer whether or not certain properties hold for a given algebra. This can be done in a compositional way, thanks to the structure provided by the metalanguage. Each base algebra and each combinator is associated with rules for property inference. The top-level properties that we seek can therefore be deduced from the properties of the base algebras involved and the rules for each combinator used. Ideally, this deduction process will be able to be easily automated; this will certainly be the case if we can find a rule-set that is complete.

There are at least two problems that might arise here. Firstly, finding the rules could be very difficult: there is no guarantee of how mathematically easy or hard this might be in a given case. We hope that the properties under discussion can be phrased mathematically in a way that will be susceptible to these 'compositional' proofs—if they are more or less 'constructive' then we probably have a good chance of being able to find rules. Secondly, we may find that the number of properties we need to track becomes very large, or perhaps even unbounded. There is no obvious reason why we should not see rules of the form

$$\mathrm{P}(S \boxtimes T) \iff (\mathrm{Q}_1(S) \wedge \mathrm{R}_1(T)) \vee (\mathrm{Q}_2(S) \wedge \mathrm{R}_2(T)) \vee \cdots$$

for some combinator $\boxtimes$, and infinite collections of properties $(\mathrm{Q}_i)_{i\in\mathbb{N}}$ and $(\mathrm{R}_j)_{j\in\mathbb{N}}$. If this does turn out to be the case, then an implementation of the property deduction system will have to do something more sophisticated than simply looking up rules in fixed tables.

The existence of a metalanguage has significant implications for practical implementation of a metarouting system. The hierarchical nature of metalanguage expressions means that we can apply well-known *code generation* principles to turn an expression into a corresponding piece of computer code. If the property checking process can be automated, then at the same time our implementation can say whether or not the generated code is indeed suitable for use in a particular context. There may be other advantages as well, including in particular the possibility of automatically carrying out certain optimizations, driven by the presence of appropriate algebraic properties. We will see some examples of this idea later.

Note that by use of a language, we are explicitly not attempting to be able to describe all possible algebras. That class would include any finite or infinite structure

with the appropriate operators and axioms; a wide class indeed. But the language, based on a comparatively small collection of base algebras and combinators, will probably not be able to express all of these. We hope at least to be able to cover examples that are of use in Internet routing. As a language design goal, this concept is somewhat imprecise; the next section will propose more concrete evaluation criteria.

It could be that we in fact can cover every single algebra with our language, if it is well-chosen. This is not necessarily a design goal, because many algebras seem to have little relevance for pathfinding. If we can generate anything, then we will certainly have satisfied our goal of generating everything that is useful—but it is more likely that we will have to demonstrate this in a more subjective way, by showing a range of useful examples that are covered.

Previously, Gouda and Schneider (2003) have considered methods for the compositional design of routing metrics. They consider two ways of combining algebras into a metric that supports the finding of shortest paths. In a similar way, Manger (2006, 2008) deals with lexicographic combinations of path algebras in order to solve the shortest path problem. This thesis represents an effort towards analysing such compositions in a more thorough way, by looking at a wider range of algebraic structures and a larger repertoire of properties. In addition, the property deduction rules which are sought are two-way; that is, they are 'if and only if' theorems which completely characterize when a given property holds for a composite algebra.

2.4.1 *Properties within the algebraic system*

A *property* of a class of algebras is a logical statement that may or may not hold for particular algebras in that class. We will speak of properties of semigroups, semirings, ordered sets, and the like. If a property P holds of an algebra S, we will write '$\mathrm{P}(S)$'. We will write properties using the standard symbols of first-order logic with equality:

$$\forall \quad \exists \quad \neg \quad \wedge \quad \vee \quad \Longrightarrow \quad \Longleftrightarrow \quad =$$

together with symbols appropriate for the algebra:

$$S \quad \oplus \quad \otimes \quad \preceq \quad F \qquad \ldots.$$

So if we say that

$$\mathrm{M} = \forall a, b, c \in S : c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$$

S	The underlying set of the algebra.
$\oplus$	The 'additive' binary operator.
$\otimes$	The 'multiplicative' binary operator.
$\preceq$	The preorder on S.
$\prec$	The strict version of ($\prec$): $x \prec y \overset{\text{def}}{\iff} x \preceq y \wedge \neg(y \preceq x)$.
$\sim$	The equivalence relation $x \sim y \overset{\text{def}}{\iff} x \preceq y \wedge y \preceq x$.
$\#$	The incomparability relation $x \# y \overset{\text{def}}{\iff} \neg(x \preceq y \vee y \preceq x)$.
F	The function set, a subset of $S \longrightarrow S$.

Figure 2.8 Symbols used in algebraic properties.

$\top$	The topmost element of a preorder.
$\bot$	The bottommost element of a preorder.
$\alpha_\oplus$	The identity for $\oplus$.
$\alpha_\otimes$	The identity for $\otimes$.
$\omega_\oplus$	The annihilator for $\oplus$.
$\omega_\otimes$	The annihilator for $\otimes$.

Figure 2.9 Symbols for special elements of algebras.

is a semiring property, we mean that it is satisfied by a semiring $(T, \boxplus, \boxtimes)$ if and only if

$$\forall a, b, c \in T : c \boxtimes (a \boxplus b) = (c \boxtimes a) \boxplus (c \boxtimes b)$$

is true; and then we would write '$\mathrm{M}((T, \boxplus, \boxtimes))$' or '$\mathrm{M}(T)$'.

The generic algebraic symbols have the interpretations listed in Figure 2.8. Not all of these will be present in every algebra: the properties are implicitly 'typed' and apply only when the symbols involved are appropriate for the algebra in question. There are some properties which apply in a more-or-less equivalent form for different algebras (for example, cancellativity can be formulated in terms of a binary operator $\otimes$ or a function set F, but is essentially the same property). In these cases, the same name will sometimes be used, but explicit definitions will always be given for each kind of algebra where the property applies.

The symbols for special elements listed in Figure 2.9 will sometimes be used in properties. These will only ever appear in subexpressions like '$\exists\top$' or '$x = \top$'; these will be taken to be false if there is no unique topmost element, so the property can still be well-defined. We can read these as shorthands for '$\exists t : \forall u : (u \prec t \vee u = t)$' and '$\forall u : (u \prec x \vee u = x)$' respectively, where t and u are variables not previously used, and likewise for subexpressions involving the other special elements.

Chapter 3

Minimal sets of paths

In considering route selection, it is often the case that in a particular path problem, there will be multiple paths of equivalent weight for some source-destination pair. Our algorithms tend to be presented in a way which assumes that only a single path is possible, or at least that if there are multiple paths, only a single one will be returned. This chapter is about the finding of multiple paths, using algebras of 'minimal sets'. Such constructions are based on operations like

$$\min(A) = \left\{x \in A \;\middle|\; \forall y \in A : \neg(y \prec x)\right\} \qquad \text{where } A \subseteq (S, \preceq) \tag{3.1}$$

which reduce a given set to a nub of minimal elements. In particular, if we are given an order transform $(S, \leq, F)$ then we can construct the minimal set algebra **minset**(S) as $(M(S), \oplus, F_M)$ where

1. $M(S)$ is $\{A \subseteq S \mid A = \min(A)\}$,
2. $A \oplus B$ is $\min(A \cup B)$, and
3. F_M is $\left\{f_m \;\middle|\; f \in F\right\}$ and $f_m(A) \stackrel{\text{def}}{=} \min\left\{f(a) \;\middle|\; a \in A\right\}$.

Properties of this algebra are investigated below, and in particular it will be demonstrated that $(\oplus)$ is associative.

Use of **minset** is preferable to any other way of resolving the multiple path issue. Other alternatives are to use a single-path algorithm with one of the following strategies:

1. Prefer older paths (so if a node is already using p and is presented with a path q of equivalent weight, then p will remain).
2. Prefer newer paths.
3. Linearize the order and run as normal.

The first two of these depend on the dynamics of the running algorithm or protocol. In the case of a synchronous algorithm, further tiebreaking (perhaps on node identifiers) must be introduced. The third option encompasses all means of extending a given order to be linear.

It will now be shown that none of these is viable for general use. This is because there are some algebras which have the correct properties for finding of global optima, but for which these strategies yield non-optimal results. It is not argued that these do not work in any circumstances, but only that there are some situations in which they fail.

Let U be the order semigroup

$$(\wp\{a,b,c\},\supseteq,\cup).$$

This is clearly monotonic: if A is a superset of B then $A\cup C$ is a superset of $B\cup C$. Consider a graph where node 1 has two paths to the origin, with weights $\{a\}$ and $\{b,c\}$, and where there is an arc from node 1 to node 2 labelled with $\{a,b\}$. Note that $\{a\}$ and $\{b,c\}$ are incomparable in U. Dynamically, node 1 might see either of its two possible paths first.

1. In strategy 1, if $\{b,c\}$ arrives first at 1 then it will be kept, and node 2 will in turn adopt $\{a,b,c\}$. But if $\{a\}$ arrives first then 2 can only get $\{a,b\}$, which is worse than $\{a,b,c\}$.

2. In strategy 2, if $\{a\}$ arrives first then 2 will get $\{a,b\}$, but this will soon be replaced by $\{a,b,c\}$ once 1 has received $\{b,c\}$. But if $\{b,c\}$ is the first to be received at 1, then 2 will get $\{a,b,c\}$ only to see it replaced by $\{a,b\}$.

So in both cases, it is possible for node 2 to end up with a non-optimal path, despite the algebraic properties of U.

Linearization of an order seems to offer a way out, since for U it is possible to find an extension of $(\supseteq)$ which is linear and monotonic. But in general, there are algebras which are monotonic but have no monotonic linearization. This means that strategy 3 does not justify the use of a single-path algorithm, because there may not be *any* way of forcing a linear order in a way that preserves monotonicity and hence existence of global optima.

Proof. We will give an example of a monotonic algebra that has no monotonic linearization. Let

$$S = (\mathbb{N}, \leq, \{s = \lambda x \,.\, x+1\}) \tag{3.2}$$

$$T = (\{0,1\}, =, \{i = \lambda x \,.\, x, n = \lambda x \,.\, (x+1) \bmod 2\}). \tag{3.3}$$

Consider the order transform

$$(S \times T, \preceq, \{(s,i),(s,n)\})$$

where

$$(w,x) \preceq (y,z) \iff w < y \vee (w = y \wedge x = z)$$

and

$$(s,i)(w,x) = (s(w), i(x)) = (w+1, x)$$
$$(s,n)(w,x) = (s(w), n(x)) = (w+1, (x+1) \bmod 2).$$

This is an example of a lexicographic product; such products will be discussed in Chapter 5.

It can be verified that this order transform is monotonic. But there is no linear order $(\leq_L)$ which maintains monotonicity. Consider a pair of elements $(k,0)$ and $(k,1)$ in $S \vec{\times} T$. If $(k,0) \leq_L (k,1)$ then by monotonicity we have

$$(s,i)(k,0) = (k+1,0) \leq_L (k+1,1) = (s,i)(k,1) \tag{3.4}$$

$$(s,n)(k,0) = (k+1,1) \leq_L (k+1,0) = (s,n)(k,1) \tag{3.5}$$

The same conclusion is reached if $(k,1) \leq_L (k,0)$. Hence $(\leq_L)$ is not a linear order. □

The failure of these strategies should make use of the **minset** operator more attractive, provided that it does have the right algebraic properties. So we now need to understand how to define algebras that make use of min, and how these behave in terms of the properties we need for correctness. The remainder of this section exhibits some definitions related to the min operation; the next section shows its algebraic properties.

The formulation in 3.1 assumes an underlying preorder, but it is also possible to make a similar definition over a commutative idempotent semigroup (a semilattice) in terms of its natural order. If $(S, \oplus)$ is a semilattice, define

$$\min(A) = \left\{ x \in A \;\middle|\; \forall y \in A : y = y \oplus x \implies x = x \oplus y \right\} \tag{3.6}$$

for $A \subseteq S$. This works because in the (left) natural order based on $\oplus$,

$$\begin{aligned}\neg(y < x) &\iff \neg(y = y \oplus x \wedge \neg(x = x \oplus y)) \\ &\iff \neg(y = y \oplus x) \vee (x = x \oplus y) \\ &\iff (y = y \oplus x \implies x = x \oplus y).\end{aligned}$$

It is clear that the two definitions of 'min' (Equations 3.1 and 3.6) are equivalent, in the sense that the preorder definition over the natural order of a semilattice is identical to the semilattice definition.

In the following discussion, take $(S, \preceq)$ to be a fixed preorder. It is clear that min is a function from $\wp S$ to $\wp S$. The following equations hold for all subsets A and B of S:

$$\min(A) = \min(\min(A)) \tag{3.7}$$

$$\min(A \cup B) = \min(\min(A) \cup B) \tag{3.8}$$

$$\min(A) \subseteq A. \tag{3.9}$$

Note also that $\min(\emptyset) = \emptyset$, and $\min(\{s\}) = \{s\}$ for any singleton subset $\{s\}$ of S. Within a set $\min(A)$, every pair of elements is either equivalent or incomparable.

If we are presented with two sets A and B of alternative routes, an obvious thing to do is to construct the set $\min(A \cup B)$, consisting of the best routes from either set. This gives us the $(\oplus)$ operator of **minset**(S) defined above. From Equation 3.8 we know that this $(\oplus)$ is associative, for

$$\begin{aligned}(A \oplus B) \oplus C &= \min(\min(A \cup B) \cup C) \\ &= \min(A \cup B \cup C) \\ &= \min(A \cup \min(B \cup C)) \\ &= A \oplus (B \oplus C)\end{aligned}$$

for all subsets A, B and C of a preorder. It is similarly easy to see that $(\oplus)$ is commutative.

It would be possible to define **minset**(S) so that the underlying set was $\wp S$ as opposed to $\{A \subseteq S \mid A = \min(A)\}$. If we use the whole of $\wp S$, then $(\oplus)$ is not necessarily idempotent: whenever A and $\min(A)$ are different, we have

$$A \oplus A = \min(A \cup A) = \min(A) \neq A.$$

However, $(\oplus)$ is idempotent for our definition of **minset**(S), since it is only ever applied to minimal sets. Equation 3.7 ensures that $(M(S), \oplus)$ is closed, because

$$\min(A \oplus B) = \min(\min(A \cup B)) = \min(A \cup B) = A \oplus B$$

These facts demonstrate that $(M(S), \oplus)$ is a semilattice.

A natural order can be derived for $(\oplus)$. This is given by

$$\begin{aligned} A \leq B &\iff A = A \oplus B = \min(A \cup B) \\ &\iff \min(A) = \min(A \cup B) \qquad \text{since } A = \min(A) \\ &\iff \forall b \in B : \exists a \in A : \{a\} = \min\{a, b\}. \end{aligned}$$

This relates to the interpretation of min as yielding the 'non-dominated' elements of the given set, in the language of economic utility.

If $(S, \preceq)$ is a preorder then define a new relation $(\preceq^{\text{nd}})$ on S by

$$x \preceq^{\text{nd}} y \overset{\text{def}}{\iff} \neg(y \prec x). \tag{3.10}$$

This says that x is 'not dominated by' y if and only if y is not preferred to x. Then by definition, $\min(A)$ is

$$\left\{ x \in A \;\middle|\; \forall y \in A : x \preceq^{\text{nd}} y \right\}.$$

In particular, an element x of S is in $\min\{x, y\}$ if and only if $x \preceq^{\text{nd}} y$. The natural order of **minset**(S) appears as

$$A = \min(A \cup B) \iff \forall x \in A, y \in A \cup B : x \preceq^{\text{nd}} y.$$

for all sets A and B. It is easy to see that $\preceq^{\text{nd}}$ is reflexive. It is also linear, since

$$\begin{aligned} &(x \preceq^{\text{nd}} y) \vee (y \preceq^{\text{nd}} x) \\ \iff &\neg(y \prec x) \vee \neg(x \prec y) \\ \iff &\neg((y \prec x) \wedge (x \prec y)). \end{aligned}$$

Similarly, if S is a linear order, then $(\preceq^{\text{nd}})$ is antisymmetric, and in this case $x \preceq^{\text{nd}} y \iff \neg(x \preceq y)$. But in general, $(\preceq^{\text{nd}})$ is not transitive: for example, if $z \prec x$, and y is incomparable to both x and z, then

$$x \preceq^{\text{nd}} y \text{ and } y \preceq^{\text{nd}} z \text{ but } \neg(x \preceq^{\text{nd}} z).$$

Because $(S, \preceq^{\text{nd}})$ is not a preorder, it is not appropriate for direct use in our algebraic framework. The encapsulation of $(\preceq^{\text{nd}})$ into the min operation allows this kind of preference to be treated equally with existing algebra and algorithms.

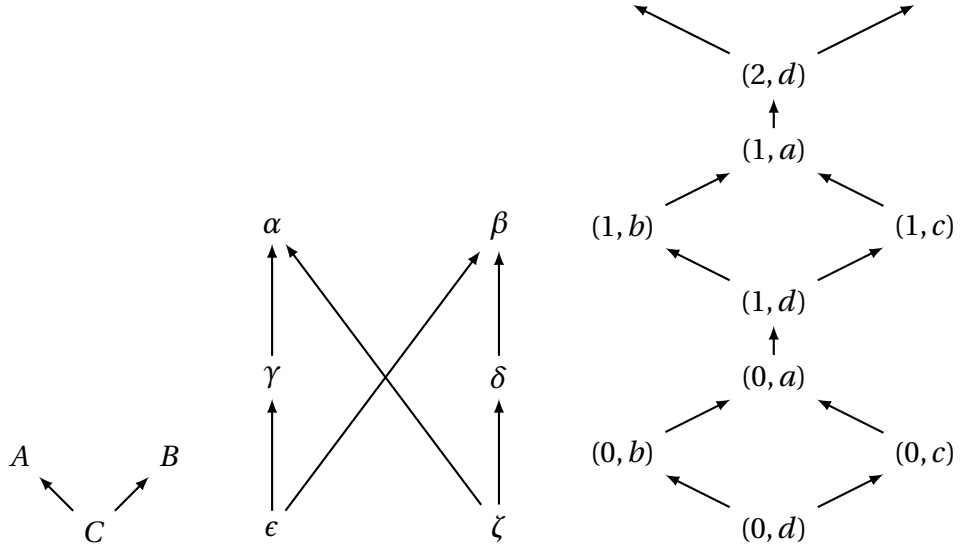


Figure 3.1 Three partial orders—from left to right, these are P, Q and R.

3.1 *The distributive lattice connection*

This section will show how the min operation is related to Birkhoff's representation theorem, and how this relationship manifests in terms of algebraic routing.

We will illustrate the various order-theoretic constructions by means of three running examples of partial orders. These are shown in Figure 3.1: P, Q and R. The order R is a lexicographic product of $\mathbb{N}$ with the four-element 'diamond' order.

The principal upper set generated by an element x of a partial order $(S, \leq)$ is

$$\uparrow x = \{y \in S \mid y \leq x\},$$

as in Definition 2.28. Suppose that S satisfies the descending chain condition (Definition 2.18). If this is so, then $\min(A)$ is nonempty for every subset A of S. Note that we do not require that this set be finite, as in a well-quasiordering, and so S may contain an infinite antichain (see Kruskal (1972) for the related theory of well-quasiorderings).

Lemma 3.1. *If S is an order satisfying the descending chain condition, then*

$$\uparrow \min(A) = \uparrow A$$
$$\min(\uparrow A) = \min(A)$$

for all subsets A of S.

Proof. For the first equality, it is clear that $\uparrow \min(A)$ is a subset of $\uparrow A$, by definition of the principal filter on a set. But if y is in $A \setminus \min(A)$ then there must be some element x of $\min(A)$ such that $y < x$. It follows that $\uparrow y$ is a subset of $\uparrow x$. Therefore $\uparrow (A \setminus \min(A))$ is a subset of $\uparrow \min(A)$, which completes the first part of the proof.

The second equality holds if

$$\min(A) = \min(A \cup (\uparrow A) \setminus A).$$

This is true since everything in $\uparrow A$ that is not in A must be strictly greater than at least one element of A. There is therefore no element of $(\uparrow A) \setminus A$ which is in $\min(A)$. □

Consequently,

$$\uparrow A = \uparrow B \iff \min(A) = \min(B). \tag{3.11}$$

since if $\uparrow A = \uparrow B$ then $\min(\uparrow A) = \min(\uparrow B)$ and so $A = B$, and symmetrically for the reverse direction.

This provides an isomorphism between upper sets and minimal sets; we can choose to represent an upper set A as the set $\min(A)$ of its minimal elements. This is not only more compact, but provides our useful min operation with a link to important areas of order theory. We can even do union and intersection operations using this form. We know that for all sets A and B,

$$\min(A \cup B) = \min(\min(A) \cup \min(B)).$$

It follows that when the upper sets A and B are being represented as $\min(A)$ and $\min(B)$, the representation of their union $A \cup B$ will be $\min(\min(A) \cup \min(B))$. In other words, the 'min-union' operation (which we have identified as being of operational importance in routing) corresponds to a standard operation on upper sets.

Recall the definition of meet-prime elements of a lattice (Definition 2.25). In the lattice of minimal sets, the meet-prime elements are those minimal sets C for which

$$C = \min(A \cup B) \implies C = A \vee C = B \tag{3.12}$$

for any minimal sets A and B. These are precisely those minimal sets which are singletons. If we perceive this lattice as the lattice of upper sets, the meet-prime elements are the principal upper sets (those generated by a single element of the original order).

Proof. If $C = \{c\}$ is a singleton and $\{c\} = \min(A \cup B)$, then every element of $A \cup B$ must be greater than or equal to c, and c itself must be in $A \cup B$. But A and B are themselves

minimal sets, so if $c \in A$ then $A = \{c\}$ and likewise for B. So at least one of A and B is equal to C. This shows that $\{c\}$ is meet-prime, for each c.

Now, suppose that C is meet-prime. Since C is a minimal set, all of the elements of C are equivalent or incomparable; and any subset of C is itself minimal. Let A and B be disjoint subsets of C. Then $C = A \cup B = \min(A \cup B)$, so by assumption, $C = A$ or $C = B$. This means that the only way to split C is as $C \cup \emptyset$, which means that C must be a singleton. □

The upper sets in our three partial order examples are as follows:

1. For P, they are $\emptyset$, $\{A\}$, $\{B\}$, $\{A, B\}$ and $\{A, B, C\}$.

2. For Q, there are fourteen of them:

   - first, we have the empty set;
   - next, the upper sets generated by each single element:

$$\begin{aligned} \alpha &\mapsto \{\alpha\} & \beta &\mapsto \{\beta\} \\ \gamma &\mapsto \{\alpha, \gamma\} & \delta &\mapsto \{\beta, \delta\} \\ \epsilon &\mapsto \{\alpha, \beta, \gamma, \epsilon\} & \zeta &\mapsto \{\alpha, \beta, \delta, \zeta\} \end{aligned}$$

   - finally, we can take unions of these principal upper sets to find seven more upper sets:

$$\{\alpha, \beta\}, \{\alpha, \beta, \delta\}, \{\alpha, \beta, \gamma\}, \{\alpha, \beta, \gamma, \delta\}, \{\alpha, \beta, \gamma, \delta, \epsilon\}, \{\alpha, \beta, \gamma, \delta, \zeta\}, Q$$

3. For R, we can follow much the same procedure. The principal upper set generated by each element is:

$$\begin{aligned} (k, a) &\mapsto \{(k, a)\} \cup U \\ (k, b) &\mapsto \{(k, a), (k, b)\} \cup U \\ (k, c) &\mapsto \{(k, a), (k, c)\} \cup U \\ (k, d) &\mapsto \{(k, a), (k, b), (k, c), (k, d)\} \cup U \end{aligned}$$

   where $U = \{(r, x) \mid k < r\}$. Taking unions, the only new upper sets we find are those generated by a pair $\{(k, b), (k, c)\}$. So we see that there is a correspondence between the upper sets of R and those of the diamond order (namely $\{a\}$, $\{a, b\}$, $\{a, c\}$, $\{a, b, c\}$ and $\{a, b, c, d\}$).

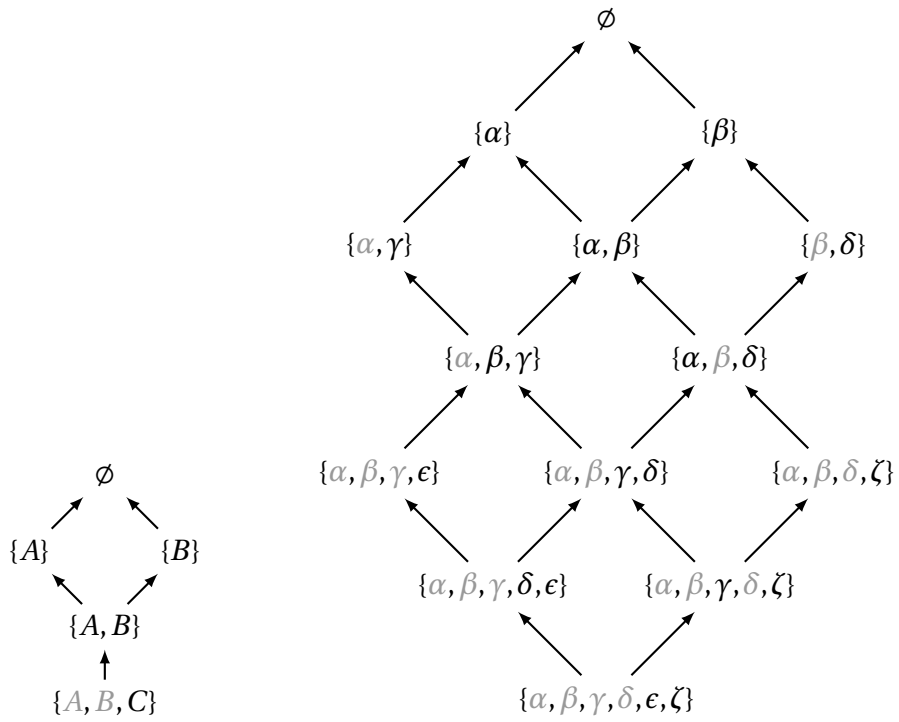


Figure 3.2 The lattices corresponding to the partial orders P and Q of Figure 3.1. Elements in grey are those which are not minimal in their sets.

These facts are consequences of the well-known representation theorem of Birkhoff (1937) which relates distributive lattices to partial orders:

Theorem 3.2. *A finite distributive lattice is isomorphic to the lattice of upper sets of the partial order of its meet-prime elements.*

This has been extended by to non-finite structures by means of Stone duality, but the correspondence between 'min' and upper sets only holds in the well-founded case. The more restrictive case here is equivalent to Theorem 10 in Chapter 9 of Birkhoff (1948).

The theorem reveals the **minset** operator as transforming a partial order into a distributive lattice. In particular, the fact that **minset**(S) is a distributive lattice, with the join operation being ($\oplus$), can be seen as generating greatest lower bounds where they did not exist before. Even if two elements of S do not have a greatest lower bound (such as α and β in Q) the corresponding singleton sets in **minset**(S) have a greatest lower bound (in Q, this is $\{\alpha, \beta\}$). Figure 3.2 shows the lattices corresponding to the partial orders P and Q.

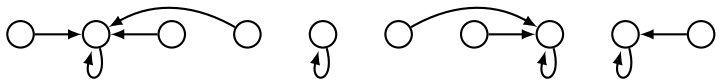


Figure 3.3 A reduction on a semigroup

This is a satisfying result for partial orders. In the case of a preorder, the minimal set algebra can still be constructed: the relationship with distributive lattices still holds, but is no longer an isomorphism.

Definition 3.1. For a preorder $(S, \preceq)$, define the *equivalence-free partial order* $(\preceq^{\text{po}})$ on S by

$$s \preceq^{\text{po}} t \overset{\text{def}}{\iff} s = t \vee (s \preceq t \wedge \neg(t \preceq s)).$$

This order is the same as the original, but with all equivalent elements now made incomparable. It is straightforward to prove that this is always a partial order; and if $(\preceq)$ was already a partial order, then $(\preceq) = (\preceq^{\text{po}})$.

Theorem 3.3. *If $(S, \preceq)$ is a preorder, then its upper sets are the same as those of $(S, \preceq^{po})$.*

Proof. A subset A of S is an upper set with respect to the partial order $(\preceq^{\text{po}})$ if and only if

$$(x \in A \wedge (x = y \vee x \prec y)) \implies y \in A;$$

that is,

$$((x \in A \wedge x = y) \implies y \in A) \wedge ((x \in A \wedge x \prec y) \implies y \in A).$$

The first implication is obviously true, and the second is the same as the condition for A to be an upper set with respect to the preorder $(\preceq)$. □

It follows that **minset**$(S, \preceq)$ is the same as **minset**$(S, \preceq^{\text{po}})$, and is therefore a distributive lattice. This gives a many-to-one relationship between preorders and distributive lattices: there are several preorders which yield the same lattice, and each lattice is the image of many preorders. However, there is still exactly one partial order which is associated with each distributive lattice.

3.2 *Reductions and congruences*

An approach to generalizing operations like min is the path algebra formulation of Wongseelashote (1976, 1979). Based on the equations (3.7) and (3.8), we can define

a *reduction* on any commutative semigroup $(X, \oplus)$ to be a function $r : X \longrightarrow X$ satisfying

$$r \circ r = r \tag{3.13}$$

$$r(a \oplus b) = r(r(a) \oplus b). \tag{3.14}$$

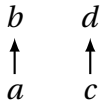
Note that for monoids, (3.13) is redundant since we can just set b to be the identity in (3.14). Figure 3.3 shows the graph of a reduction, demonstrating that each element is a fixed point or one application away from a fixed point.

So min is a reduction on $(\wp S, \cup)$. There are other examples for the k-best paths problem and for multisets. In general, reductions seem to be operators that put their argument into some canonical form.

The definition above bears some resemblance to the Kuratowski axioms for the closure operator of a topological space, or for its dual, the interior operator (see for example Kelley (1955)). These definitions are, however, not equivalent: the interior operator must satisfy

$$\text{int}(A \cup B) = \text{int}(A) \cup \text{int}(B)$$

so it is a reduction, but not every reduction obeys this axiom. In particular, min does not. In the order



we have

$$\min\{a, d\} = \{a, d\}$$

$$\min\{b, c\} = \{b, c\}$$

$$\text{but } \min(\{a, d\} \cup \{b, c\}) = \{a, c\} \neq \{a, b, c, d\}$$

and so min is not an interior operator.

Reductions do not seem to admit much of a structure theory in themselves, and it has proved difficult to characterize the space of all reductions. However, we can show that there is a relationship between reductions and *congruences* (see Definition 2.20). For each reduction there is a congruence, and for each congruence there is at least one reduction. This observation means that there is a reasonable algebraic way of characterizing these functions, which will allow us to discover more about them. Figure 3.4 shows how the reduction of Figure 3.3 yields a congruence.

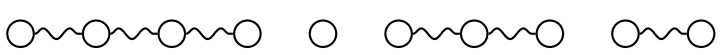


Figure 3.4 A congruence (derived from the reduction of Figure 3.3)

Lemma 3.4. *For any reduction r, define a relation $(\sim_r)$ on X by*

$$a \sim_r b \overset{\text{def}}{\iff} r(a) = r(b).$$

This $(\sim_r)$ is a congruence.

Proof. First we check that $(\sim_r)$ is an equivalence relation.

- Reflexivity: $r(a) = r(a)$ for all a, so $a \sim_r a$ always.
- Symmetry: $a \sim_r b \implies r(a) = r(b) \implies b \sim_r a$.
- Transitivity: If $a \sim_r b$ and $b \sim_r c$ then $r(a) = r(b)$ and $r(b) = r(c)$, so $r(a) = r(c)$ and $a \sim_r c$.

To prove that it is a congruence, suppose that $a \sim_r b$, so that $r(a) = r(b)$. Then

$$\begin{aligned} r(a \oplus c) &= r(r(a) \oplus c) && \text{by (3.14)} \\ &= r(r(b) \oplus c) && \text{since } r(a) = r(b) \\ &= r(b \oplus c) && \text{by (3.14).} \end{aligned}$$

Hence $(\sim_r)$ is indeed a congruence. □

We can also produce a reduction from a congruence. In fact, there will typically be many choices of reduction for a different congruence. The congruence $(\sim)$ splits X into equivalence classes, and this collection of classes is also a commutative monoid, referred to by the notation $X/\sim$. There is a homomorphism $\rho^\natural$ called the *natural map*, taking each element of X to its $\sim$-equivalence class. If we choose a function going in the other direction, taking each equivalence class to some representative element within the class, then the composition of these two functions will be a reduction. The choice of representatives means that there may be multiple reduction functions, although they all correspond to the same congruence and define the same equivalence classes. This is demonstrated in Figure 3.5, which shows a mapping from congruence classes to chosen representatives of each class. Note that these are different from the fixed points of the reduction of Figure 3.3.

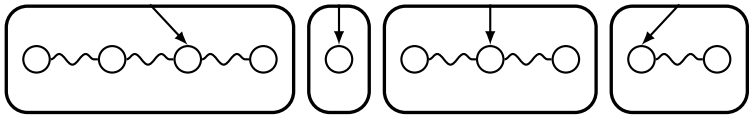


Figure 3.5 A possible choice for $\theta : X/\sim \longrightarrow X$

Lemma 3.5. *Let $(X, \oplus)$ be a semigroup, $\sim$ a congruence, and $\rho^\natural$ the natural map. If $\theta : X/\sim \longrightarrow X$ is such that $\rho^\natural \circ \theta = id$, then $\theta \circ \rho^\natural$ is a reduction; and $\sim$ is equal to $\sim_{\theta\circ\rho^\natural}$.*

Proof. Note that the condition $\rho^\natural \circ \theta = \mathrm{id}$ simply expresses that the representative for a class should be an element of that class. There is always at least one such θ, because there can be no empty classes. This condition also provides that θ must be one-to-one, for if $\theta(P)$ and $\theta(Q)$ then $(\rho^\natural \circ \theta)(P)$ and $(\rho^\natural \circ \theta)(Q)$ must also be equal; and then $P = Q$.

Now, $\theta \circ \rho^\natural$ satisfies the axioms for a reduction. For (3.13),

$$(\theta \circ \rho^\natural)^2 = \theta \circ (\rho^\natural \circ \theta) \circ \rho^\natural = \theta \circ \rho^\natural.$$

For (3.14),

$$\begin{aligned}
(\theta \circ \rho^\natural)(a \oplus b) &= \theta(\rho^\natural(a) \oplus \rho^\natural(b)) && \text{since } \rho^\natural \text{ is a homomorphism} \\
&= \theta(\rho^\natural(\theta(\rho^\natural(a))) \oplus \rho^\natural(b)) && \text{since } \rho^\natural \circ \theta = \mathrm{id} \\
&= (\theta \circ \rho^\natural)((\theta \circ \rho^\natural)(a) \oplus b) && \text{since } \rho^\natural \text{ is a homomorphism.}
\end{aligned}$$

Furthermore, the congruence derived from this reduction is $\sim$ again:

$$\begin{aligned}
a \sim_{\theta\circ\rho^\natural} b &\iff \theta(\rho^\natural(a)) = \theta(\rho^\natural(b)) \\
&\iff \rho^\natural(a) = \rho^\natural(b) && \text{since } \theta \text{ is one-to-one} \\
&\iff a \sim b && \text{by definition of the natural map.}
\end{aligned}$$

Hence for any congruence there is at least one equivalent reduction. □

We can therefore choose to represent any reduction r as a pair $(\sim, \theta)$, since this is enough to completely determine the function. In the case of min, the relevant congruence is the same as that for upper sets:

$$A \sim_{\min} B \iff \min(A) = \min(B) \iff \uparrow A = \uparrow B.$$

This implies that a $\sim_{\min}$-class consists of sets all of which have the same principal upper set. One choice for θ would be to take

$$\theta(P) = \bigcup_{A \in P} A$$

so that $\theta(P) = \uparrow(A)$ for any A in P. This choice has a mathematical advantage in that this θ is a homomorphism with respect to set union:

$$\begin{aligned}\theta(P \cup Q) &= \bigcup_{A \in P \cup Q} A \\ &= \left(\bigcup_{A \in P} A\right) \cup \left(\bigcup_{B \in Q} B\right) \\ &= \theta(P) \cup \theta(Q).\end{aligned}$$

However, this is not the only choice. We can recover min from $(\sim_{\min})$ by choosing

$$\theta'(P) = \bigcap_{A \in P} A = \min(\theta(P))$$

This demonstrates that although reductions are not homomorphisms, they are related, and that the idea of a congruence is the unifying concept.

Other examples treated by Wongseelashote (1979) can be handled similarly.

1. Consider the set Σ^* of words over an alphabet Σ. The function s on subsets of Σ^* given by

   $$s(A) \stackrel{\text{def}}{=} \left\{a \in A \,\middle|\, \text{no letter appears more than once in } a\right\}$$

   is a reduction on $(\wp\Sigma^*, \cup)$. This defines a congruence $(\sim_s)$ given by

   $$A \sim_s B \stackrel{\text{def}}{\iff} A \setminus \Sigma^R = B \setminus \Sigma^R$$

   where Σ^R is the set of words over Σ that contain a repeated letter. Note that unlike min, this s is a homomorphism. An equivalent definition of $(\sim_s)$ would be

   $$A \sim_s B \iff A \cup \Sigma^R = B \cup \Sigma^R.$$

   In either case,

   $$\theta(P) = \left(\bigcup_{A \in P} A\right) \setminus \Sigma^R$$

   is a function that yields the original s.

2. Let $(\preceq)$ be the order on Σ^* given by

   $$x \preceq y \stackrel{\text{def}}{\iff} \text{the letters of } x \text{ appear in order in } y.$$

   For example, ‘cod’ $\preceq$ ‘command’ and ‘rise’ $\preceq$ ‘armistice’. The ‘elem’ reduction of Wongseelashote is just the min function over this order.

3. Multisets over a set X are equivalent to functions from X to $\mathbb{N}^\infty$. Multiset union is given by

$$A \cup B \stackrel{\text{def}}{=} \lambda x \,.\, A(x) + B(x).$$

One reduction with respect to this operation is

$$r(A) \stackrel{\text{def}}{=} \lambda x \,.\, \begin{cases} 1 & \text{if } A(x) \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This effectively reduces a multiset to an ordinary set: each element appears at most once. The reduction can be perceived as the combination of a congruence and a function. The congruence makes two multisets equivalent if they are the same when multiplicity is disregarded:

$$A \sim B \iff (\forall x \in X : A(x) = 0 \iff B(x) = 0)$$

and the selection function chooses the appropriate element of the equivalence class:

$$\theta(P) = \lambda x \,.\, \min_{A \in P} A(x).$$

This θ exhibits the familiar pattern that the representative of an equivalence class is often some kind of 'summary' of the entire class, and is not an arbitrary choice.

The interpretation of reductions in terms of congruences is helpful because it clarifies the true role of a reduction as well as often being more algebraically useful. A reduction is not an arbitrary transformation that fulfils some unusual axioms, but instead arises as the combination of a congruence—to say which distinctions between elements are being ignored—and a choice of representative element from each equivalence class.

So far, this chapter has made little mention of the mapping of functions over minimal sets. The definition of **minset**(S) as an order transform includes a set of functions, each of which operates by

$$f_m(A) = \min \left\{ f(a) \;\middle|\; a \in A \right\}$$

where f is a function on the original S. The original definition of a reduction over a semiring $(S, \oplus, \otimes)$ includes the axiom

$$\forall A, B : r(A \otimes B) = r(r(A) \otimes B) = r(A \otimes r(B))$$

where

$$A \otimes B \stackrel{\text{def}}{=} \{a \otimes b \mid a \in A, b \in B\};$$

see Wongseelashote (1976, 1979). In terms of functions rather than an $(\otimes)$ operator, the axiom would become

$$\forall A, f : r(f(A)) = r(f(r(A))) \tag{3.15}$$

or $\forall f : r \circ f = r \circ f \circ r$ in point-free style. This property can be used to show that monotonicity of S implies distributivity of **minset**(S). In the case of min, it is guaranteed that $A = \min(A)$, and hence $(\min \circ f)(A)$ and $(\min \circ f)(\min(A))$ are equal as well.

Theorem 3.6. *Suppose that* $(S, \preceq, F)$ *is monotonic. Then* **minset**$(S) = (M(S), \oplus, F_M)$ *is distributive.*

Proof. Let f_m be a function from F_M, and let A and B be elements of $M(S)$. It must be shown that

$$f_m(A \oplus B) = f_m(A) \oplus f_m(B),$$

which by definition is equivalent to

$$(\min \circ f)(\min(A \cup B)) = \min\big(\min(f(A)) \cup \min(f(B))\big).$$

By (3.8), we have

$$\begin{aligned} \min\big(\min(f(A)) \cup \min(f(B))\big) &= \min\big(f(A) \cup f(B)\big) \\ &= \min(f(A \cup B)), \end{aligned}$$

so in order to prove distributivity, it is enough to show that

$$\forall f : \min \circ f \circ \min = \min \circ f.$$

This is (3.15), which has just been shown to hold for **minset**(S). □

The results of Chapter 4 will provide us with the information we need for the 'increasing' property in relation to minimal sets: if S is increasing, then algorithms over **minset**(S) converge to local optima.

Chapter 4

Convergence for non-distributive algebras

The property of *distributivity* or *monotonicity* for an algebra is sufficient to allow standard path-finding algorithms to reach a globally optimal path assignment. For algebras which fail to be distributive, certain results are known, but the full situation remains unclear. We know that for an algebra which is *selective* (a linear order) and *increasing*, the same standard algorithms will find a Nash equilibrium, or locally optimal path assignment. In the absence of selectivity, no theorem is presently available.

In this chapter, we will examine two convergence proofs for selective and increasing algebras, and explore the links between them. A new proof will then be presented which covers not only this case, but also a case related to multipath routing. This short proof also shows that the fixed point which is found by the standard algorithm is unique.

4.1 *Two convergence proofs*

The convergence of the standard matrix iteration $\sigma : X \mapsto AX \oplus I$ has been thoroughly explored in the case where the matrix elements are drawn from a semiring. But even if not all semiring axioms hold, convergence is still possible. An important case is that if a bisemigroup $(S, \oplus, \otimes)$ satisfies

1. $(\oplus)$ is commutative, idempotent and selective;
2. an identity for $(\oplus)$ exists and is an annihilator for $(\otimes)$;
3. for all x and y, $x = x \oplus (y \otimes x) \neq y \otimes x$

and is used to label a graph, then the σ iteration always converges to a fixed point. (It is also required that paths with loops be forbidden.) This third property is called the 'increasing' property, since it expresses the fact that any extension $y \otimes x$ of a path x must be strictly worse than x. Notably, the distributive property is not required.

For these 'increasing bisemigroups', the fixed point that is found is generally not a globally optimal path assignment. In the semiring case, it can be shown that the result of the iteration is a matrix whose (i, j) entry corresponds to the optimal path from node i to node j; that is, the sum $\bigoplus_{p \in P} p$ where P is the set of all paths from i to j. But in the absence of distributivity, the result of the iteration is instead a *local optimum* or *Nash equilibrium*. Let $\sigma^*(X)$ be the fixed point of σ starting from X as the initial matrix. Then

$$\sigma^*(X)_{ij} = \bigoplus_{k \in N} A_{ik} \otimes \sigma^*(X)_{kj} \oplus I_{ij}. \tag{4.1}$$

This equation expresses that fact that the (i, j) entry of $\sigma^*(X)$ is the best choice out of the set

$$\{I_{ij}\} \cup \{A_{ik} \otimes \sigma^*(X)_{kj} \mid k \in N\} \tag{4.2}$$

of all extensions of the paths chosen by neighbours of i in $\sigma^*(X)$. It is therefore a *stable solution* to the original path problem: a path assignment in which every node has the optimal path out of those paths which are consistent with the choices made by other nodes. In terms of game theory, no node can improve on the path it already has (given that it must choose an extension of some neighbour's path) and so this can be called a Nash equilibrium.

The *stable paths problem* is a combinatorial path problem formulated in order to model some aspects of the BGP route selection process (Griffin, Shepherd and Wilfong 1999, 2002; Griffin and Wilfong 2000). Though not using the same language of abstract algebra, the original publications show that if the path preferences of nodes follow the 'increasing' rule, then a stable solution exists and can be found through iteration. Later accounts of the stable paths problem have removed some of the elaborations present in the original model, such as the dynamics of message passing and queueing.

An instance of the stable paths problem is a graph G, with one node O identified as the origin, together with a list $\rho(i)$ of paths from i to O for each node i. This list is interpreted as providing the relative preferences of paths; not all paths from i to O will be on the list, and those that are not included are presumed to be forbidden. It is *not* specified how these rankings come about: they could be derived algebraically from a labelling of arcs, but this is not part of the model. A solution to the problem is

a path assignment with the property that no node is able to improve on the path it is assigned. For some instances of the stable paths problem, there are multiple stable solutions.

The strategy in the original paper is to prove the contrapositive: if there is no solution, then there must be a *dispute wheel* somewhere in the graph. This is an arrangement of nodes whose preferences collectively violate the rule. If these wheels cannot occur, then there must be a stable solution. In order to prove the absence of dispute wheels, another construction is used, the *dispute digraph*. This is a new graph based on the preference structure of the given stable paths problem. Cycles in the dispute digraph correspond to dispute wheels, so if the digraph is acyclic then there cannot be a dispute wheel. Path preferences derived from an algebra with the increasing rule always define acyclic digraphs; so in this case there are no dispute wheels in G, which means that at least one stable solution exists.

The dispute digraph is defined as follows. Its nodes are all of the simple paths in the graph. There are two kinds of arc: *transmission arcs* and *dispute arcs*. Whenever p is a path from j to k, and there is an arc in the original graph from i to j, then there is a transmission arc in the dispute digraph from p to $(i\,j)p$. The dispute arcs relate to preferences, and violations of the monotonicity rule. Suppose that there are two paths p and p' from j to k, and that p is preferred to p', but the extension $(i\,j)p'$ is preferred to $(i\,j)p$. Then a dispute arc exists in the digraph, between p and $(i\,j)p'$.

Once this graph has been shown to be acyclic, a proof based on 'histories' shows that the iterative algorithm must terminate. A *history* is a chain in the dispute digraph that is associated with each node. These histories change as the algorithm progresses, according to a fixed construction rule, and it can be shown that they never decrease in length. Since the dispute digraph is finite and acyclic, there must come a point when each history is fixed in size: from the construction rule, it can then be deduced that the current path assignment is stable, and the algorithm has terminated.

The rule for building histories is as follows. Whenever a node changes its current path, its associated history also changes. There are two alternatives: either the new path is better, or it is worse. Note that in the conventional shortest-path setting, this second case is impossible: it is the possibility for these kinds of dynamics which make the nondistributive termination proofs difficult. If the new path is better, this must be because it has been announced by some neighbour. The new history then consists of the neighbour's history, plus the new path; then the new link in the chain is a transmission arc. If the new path is worse, then it must be that a path which

was previously available has now disappeared, because a neighbour has withdrawn its prefix. In this case, the new history consists of the history of the withdrawing neighbour, followed by the path which was lost, so that the new link is a dispute arc. The intention behind this construction is that the history 'explains' how a value came to be adopted at a particular point (Griffin and Wilfong 2000).

An alternative proof style is used by Sobrinho (2003). This is based on a measurement of the network activity, at a given time, associated with each path. There is a similar construction to the dispute digraph, but based on a slightly different relation among paths. It is shown that each possible step in the asynchronous protocol results in a decrease in this activity measurement.

We will show how these two proofs are connected. First, a version of the Sobrinho proof is produced which is adapted for the the algebraic approach: it will be based on matrix iteration, rather than on an asynchronous message-passing model. This will allow a more direct comparison with the dispute digraph structure. It also shows that it is possible to characterize termination of this algorithm in terms of the familiar linear algebra structure.

Some generalizations and simplifications can be made at the same time; the original proof required all path weights to be distinct, whereas the new proof allows there to be two different paths with the same weight. An adaptation is also possible to the minimal-sets algebra, as a prelude to the ultrametric proof discussed in Section 4.2.

In the following, $(S, \oplus, F)$ will be taken to be an increasing bisemigroup with $(\oplus)$ being selective. Let $G = (N, E)$ be a graph weighted over S by w, and let P be the set of simple paths in G, together with the 'null path' ϕ. We require that $w(\phi) = \alpha_\oplus$, and if $w(p) = \alpha_\oplus$ then $p = \phi$.

The null path represents the absence of any route. It should not be confused with an empty path, which is a zero-length path from a node to itself. In our model, there will be many empty paths, but only one null path.

We will define an order on the set of simple paths. It is important that there should be no cycles: the foundation of our proof is that for any path with a certain property, there is another path which precedes it in the order and has a related property. If cycles were allowed, then although this statement could still be true, we could not use that relationship to prove convergence. The order is

$$p \preceq q \iff w(p) = w(p) \oplus w(q).$$

This is a preference relation.

Definition 4.1. A *path profile* is any function from P to $\mathbb{N}$.

This definition is different from that used by Sobrinho. He defined the *rank* of a path with respect to the $(\leq)$ order, and used functions from the set of ranks to $\mathbb{N}$. In fact, we do not lose anything if we omit the notion of rank. (The rank is elsewhere known as the *height* of an element in the order: the length of the longest path from that element to the bottom.)

The next step is to establish an order on path profiles. The matrix iteration will induce a decrease with respect to this order. This is the synchronous analogue of Sobrinho's theorem that network activity (the sending and receiving of messages and updating of local tables) is associated with such a decrease. In that formulation, each path is associated with a count: the number of nodes which are using the path, plus the number of messages in transit which mention it. The path profile order is based on ranks: it is permissible for a count to increase, so long as there is some other lower-ranked path for which the count decreases.

Definition 4.2. Let $(S, \leq_S)$ and $(T, \leq_T)$ be partially ordered sets, with $(S, \leq_S)$ well-founded and $(T, \leq_T)$ linear. Define a relation $(\sqsubseteq)$ on the set of functions from S to T as follows:

$$f \sqsubseteq g \iff \forall x \in S : f(x) >_T g(x) \implies (\exists y \in S : y <_S x \wedge f(y) <_T g(y)).$$

Then $(S \longrightarrow T, \sqsubseteq)$ is the *lexicographic power* of S and T, denoted $S \xrightarrow{\text{lex}} T$.

This ordered set is sometimes known as the *ordinal power* or *ordinal product* in the literature on ordinal arithmetic (Birkhoff 1948; Novák 1965). We use the term 'lexicographic power' to emphasize the nature of the order, rather than the role of the construction in transfinite mathematics.

Lemma 4.1. *The* $(\sqsubseteq)$ *relation is a partial order (provided that* $(S, \leq_S)$ *is well-founded and* $(T, \leq_T)$ *is linear).*

Proof. See Section A.1 in Appendix A. □

In the case when S is a linear order, $S \xrightarrow{\text{lex}} T$ reduces to the set of S-indexed sequences of elements of T, ordered lexicographically.

We use this order for the set $(P \longrightarrow \mathbb{N})$ of path profiles. The least element of $(P \longrightarrow \mathbb{N})$ is the function taking each input to 0. Sobrinho's ranks are natural numbers, plus infinity for the null path; as such, they form a linear order. We do not have this, but the definition here is enough for the following proofs. In the linear order case,

path profiles are tuples $(n_1, n_2, \ldots, n_{|P|}) \in \mathbb{N}^{|P|}$ with a lexicographic order. The critical step in the proof is that 'activity' of a certain kind at a higher rank is associated with another kind of 'activity' at a lower rank. Even though we do not have a linear order, this idea still makes sense: rather than speaking of ranks that are higher and lower in the linear order, we have paths which precede other paths in a partial order.

Definition 4.3. A subset R of $(P \longrightarrow \mathbb{N})$ is *component bounded* if there is a path profile m such that for every r in R, and for every p in P, $r(p) \leq m(p)$.

The above definition is for upper bounds: lower bounds always exist for each subset. This definition is distinct from the notion of boundedness with respect to the $(\sqsubseteq)$ order; the 'component bound' is that for each path p, the value $\max\{r(p) \mid r \in R\}$ is finite, whereas boundedness for $(\sqsubseteq)$ would mean that R had a least upper bound.

Any finite subset of $(P \longrightarrow \mathbb{N})$ is component bounded. If R_1 and R_2 are component bounded then $R_1 \cup R_2$ and

$$R_1 + R_2 \stackrel{\text{def}}{=} \{r_1 + r_2 \mid r_1 \in R_1, r_2 \in R_2\} \tag{4.3}$$

are also component bounded.

Lemma 4.2. *Suppose that P is finite. Let R be a subset of $(P \longrightarrow \mathbb{N})$. Then R satisfies the descending chain condition if and only if it is component bounded.*

Proof. Suppose that R is component bounded by the path profile m. Then $|R|$ is at most $\Pi_{p \in P}(m(p)+1)$, which is a finite number because P is finite. Because R is finite, it must satisfy the descending chain condition.

Now suppose that R is not component bounded. Then there is at least one path p for which $\{r(p) \mid r \in R\}$ is infinite. Let p_0 be a minimal such path; so if $p \prec p_0$, then $\{r(p) \mid r \in R\}$ is finite. There must then be a countable subset R' of R such that

1. $\{r'(p_0) \mid r' \in R'\}$ is infinite, and
2. if $p \prec p_0$, then for all r'_1 and r'_2 in R', $r'_1(p) = r'_2(p)$.

The set R' then forms an infinite descending chain. □

The fact that a descending chain cannot be of infinite length is an important step in the proof, because it means that a descending chain must eventually stabilize. We will show that this corresponds to convergence of the pathfinding process. So without this lemma, there would be the possibility that our algorithm gets closer and closer to its goal without ever actually reaching it.

Let A be the adjacency matrix of G, and define σ to be the function

$$\sigma(X) = I \oplus (A \cdot X). \tag{4.4}$$

We will define a function V to associate each matrix X with a path profile. To show that the sequence

$$X, \sigma(X), \sigma^2(X), \sigma^3(X), \ldots$$

converges for all X, we need to prove some facts about V:

1. the range of V contains no infinite descending chain;
2. $V(\sigma(X)) \sqsubseteq V(X)$ for all X; and
3. $V(X) = V(\sigma(X))$ implies $X = \sigma(X)$ for all X.

These facts together ensure that the sequence

$$V(X), V(\sigma(X)), V(\sigma^2(X)), V(\sigma^3(X)), \ldots$$

is monotonically descending, but is not of infinite length, and so we will eventually reach an index k for which $V(\sigma^k(X)) = V(\sigma^{k+1}(X))$, and then we obtain $\sigma^k(X) = \sigma^{k+1}(X)$.

Now, the domain of a path profile is the set of simple paths in a graph, whereas the traditional matrix iteration approach is for each matrix to contain the weights of paths rather than the paths themselves. For the purposes of this proof, we do need to distinguish different paths which have the same weight; this does not affect the termination properties of the iteration, but is required in particular in the proof of Lemma 4.4 below. There are several ways in which path information could be encoded or otherwise made available, accompanying the weight data in the matrix or alongside it. In the development below, we will assume that this has been done, either by augmentation of the algebra S or by using some companion data structure for each matrix. We will therefore write '$p = X_{ij}$' and mean that p is the (unique) path associated with the matrix entry X_{ij}.

Define the function V from $S^{N\times N}$ to $(P \longrightarrow \mathbb{N})$ by

$$V(X)(p) = \left|\left\{(i,j) \;\middle|\; p = \sigma(X)_{ij}\right\}\right| + \left|\left\{(i,j) \;\middle|\; p = \sigma(X)_{ij} \neq X_{ij}\right\}\right|. \tag{4.5}$$

Note that $V(X)(p)$ is either 0, 1 or 2 for all X and p. Therefore $V(X)$ is component bounded, by the function $m(p) \stackrel{\text{def}}{=} 2$. Note that the set P is finite, so Lemma 4.2 applies, and hence the range of V satisfies the descending chain condition.

Lemma 4.3. *Let δ be the function from $S^{N\times N}\times P$ to $\mathbb{Z}$ given by*

$$\delta(X)(p) = V(\sigma(X))(p) - V(X)(p).$$

Then for all X, and all nodes i and j:

1. $\delta(X)(\sigma(X)_{ij}) \leq 0$.
2. *If $\delta(X)(\sigma(X)_{ij}) = 0$ then $X_{ij} = \sigma(X)_{ij} = \sigma^2(X)_{ij}$.*
3. *If $\delta(X)(p) > 0$ for some path p from i to j, then $p = \sigma^2(X)_{ij} \neq \sigma(X)_{ij}$.*

Proof. It can be verified from the definition of δ that

$$\delta(X)(\sigma(X)_{ij}) = \begin{cases} 0 & \text{if } X_{ij} = \sigma(X)_{ij} = \sigma^2(X)_{ij} \\ -1 & \text{if } X_{ij} \neq \sigma(X)_{ij} \text{ or } \sigma(X)_{ij} \neq \sigma^2(X)_{ij} \\ -2 & \text{if } X_{ij} \neq \sigma(X)_{ij} \text{ and } \sigma(X)_{ij} \neq \sigma^2(X)_{ij} \end{cases}$$

$$\delta(X)(\sigma^2(X)_{ij}) = \begin{cases} 2 & \text{if } \sigma(X)_{ij} \neq \sigma^2(X)_{ij} \\ 0 & \text{if } \sigma(X)_{ij} = \sigma^2(X)_{ij} \text{ and } X_{ij} \neq \sigma(X)_{ij} \\ -1 & \text{if } \sigma(X)_{ij} = \sigma^2(X)_{ij} \text{ and } X_{ij} = \sigma(X)_{ij} \end{cases}$$

and $\delta(X)(p) = 0$ if p is a path from i to j that is equal to neither $\sigma(X)_{ij}$ nor $\sigma^2(X)_{ij}$. □

Lemma 4.4. *If $\delta(X)(p) > 0$ for some path p from i to j, then there is a node k and a path q from k to j such that $q \prec p$ and $\delta(X)(q) < 0$.*

Proof. If $\delta(X)(p) > 0$ then $p = \sigma^2(X)_{ij}$, and $\sigma(X)_{ij}$ is not equal to p, by Lemma 4.3(3).

Since $(\preceq)$ is a linear order, and $\sigma(X)_{ij}$ and $\sigma^2(X)_{ij}$ are different, they must be strictly ordered: either $\sigma(X)_{ij} \prec \sigma^2(X)_{ij}$ or $\sigma(X)_{ij} \succ \sigma^2(X)_{ij}$.

If $\sigma(X)_{ij} \prec \sigma^2(X)_{ij}$ then let k be i and let q be $\sigma(X)_{ij}$. By Lemma 4.3(1) it follows that $\delta(X)(q) \leq 0$. But it cannot be that $\delta(X)(q) = 0$, since then, by Lemma 4.3(2), we would have $\sigma(X)_{ij} = \sigma^2(X)_{ij}$. Therefore $\delta(X)(q) < 0$ as required.

Alternatively, $\sigma(X)_{ij} \succ \sigma^2(X)_{ij}$. The nodes i and j must be different, since otherwise $\sigma(X)_{ij}$ and $\sigma^2(X)_{ij}$ would both be equal to I_{ii}. Hence $\sigma^2(X)_{ij}$ is not the null path; there is some node k for which $\sigma^2(X)_{ij} = A_{ik} \otimes \sigma(X)_{kj}$. Let q be $\sigma(X)_{kj}$. Because $(\preceq)$ is increasing, we have $q \prec p$:

$$q = \sigma(X)_{kj} \prec A_{ik} \otimes \sigma(X)_{kj} = \sigma^2(X)_{ij} = p.$$

By Lemma 4.3(1), we have $\delta(X)(q) \leq 0$. If $\delta(X)(q) = 0$ then X_{kj} and $\sigma(X)_{kj}$ are equal, by Lemma 4.3(2), and

$$A_{ik} \otimes X_{kj} = A_{ik} \otimes \sigma(X)_{kj} = \sigma^2(X)_{ij} \prec \sigma(X)_{ij} \preceq A_{ik} \otimes X_{kj}.$$

This is a contradiction. Therefore $\delta(X)(q) < 0$ as required. □

Theorem 4.5. *For all X, $V(\sigma(X)) \sqsubseteq V(X)$.*

Proof. Suppose that for some p, $V(\sigma(X))(p) > V(X)(p)$. Then $\delta(X)(p) > 0$, and by Lemma 4.4, there is a $q \prec p$ such that $\delta(X)(q) < 0$; that is, $V(X)(q) < V(\sigma(X))(q)$. Hence $V(\sigma(X)) \sqsubseteq V(X)$. □

Finally, we show that when the descending chain of path profiles is stable (which must eventually happen), then the corresponding matrices must also be stable, giving us a fixed point of σ.

Theorem 4.6. *For all X, if $V(X) = V(\sigma(X))$ then $X = \sigma(X)$.*

Proof. If $V(X) = V(\sigma(X))$ then $V(X)(p) = V(\sigma(X))(p)$ for all p; so $\delta(X)(\sigma(X)_{ij}) = 0$ for all i and j. By Lemma 4.3, we then have $X_{ij} = \sigma(X)_{ij}$ for all i and j, so $X = \sigma(X)$. □

4.1.1 *Minimal sets*

Suppose that S is an algebra of minimal sets over S', where S' is increasing. Define V as:

$$V(X)(p) = \left|\left\{(i,j) \;\middle|\; p \in \sigma(X)_{ij}\right\}\right| + \left|\left\{(i,j) \;\middle|\; p \in \sigma(X)_{ij} \wedge p \notin X_{ij}\right\}\right|.$$

Then, as before, we have the following:

1. If $p \in \sigma(X)_{ij}$ then $\delta(X)(p) \leq 0$.
2. If $p \in \sigma(X)_{ij}$ and $\delta(X)(p) = 0$ then $p \in X_{ij}$ and $p \in \sigma^2(X)_{ij}$.
3. If $\delta(X)(p) > 0$ for a path p from i to j, then $p \in \sigma^2(X)_{ij}$ but $p \notin \sigma(X)_{ij}$.

These are just the same as in Lemma 4.3, but with '=' replaced by '$\in$'. We can now prove a new version of Lemma 4.4 for this definition.

Lemma 4.7. *If $\delta(X)(p) > 0$ for some path p from i to j, then there is a node k and a path q from k to j such that $q \prec p$ and $\delta(X)(q) < 0$.*

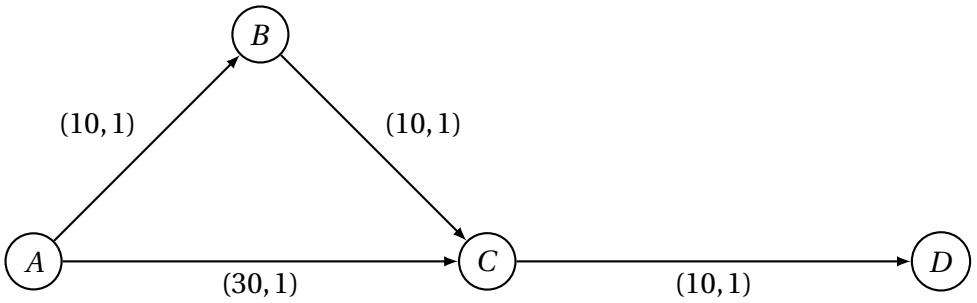


Figure 4.1 Labelled graph for Example 4.1.

Proof. If $\delta(X)(p) > 0$ then $p \in \sigma^2(X)_{ij}$, and $p \notin \sigma(X)_{ij}$.

We have $p = (i\,k)q$ for some q in $\sigma(X)_{kj}$; and since S' is increasing, we know that $q \prec p$. Certainly $\delta(X)(q) \leq 0$. We also know that if $\delta(X)(q) = 0$ then $q \in X_{kj}$; so if $q \notin X_{kj}$ then $\delta(X)(q) < 0$ as desired.

So suppose instead that $q \in X_{kj}$. Since $p = (i\,j)q \notin \sigma(X)_{ij}$, there must be some p' in $\sigma(X)_{ij}$ with $p' \prec p$. In this case, we will show that $\delta(X)(p') < 0$. As before, we have $\delta(X)(p') \leq 0$. If $\delta(X)(p') = 0$ then $p' \in \sigma^2(X)_{ij}$; but $p' \prec p$ and $p \in \sigma^2(X)_{ij}$. Therefore $\delta(X)(p') \neq 0$. □

Theorem 4.8. *For all X, if $V(X) = V(\sigma(X))$ then $\sigma(X) = \sigma^2(X)$.*

Proof. If $V(X) = V(\sigma(X))$ then $V(X)(p) = V(\sigma(X))(p)$ for all p, so $\delta(X)(p) = 0$ for all p. Therefore, for all p, we have either

1. $p \in X_{ij}$ and $p \in \sigma(X)_{ij}$ and $p \in \sigma^2(X)_{ij}$; or
2. $p \notin \sigma(X)_{ij}$ and $p \notin \sigma^2(X)_{ij}$.

Hence $\sigma(X)_{ij} = \sigma^2(X)_{ij}$ for each i and j; and $\sigma(X) = \sigma^2(X)$. □

4.1.2 *Examples*

Example 4.1. This example uses a lexicographic product; see Definition 5.1 for an explanation of the '$\vec{\times}$' symbol. It implements path choice based on the 'bandwidth-distance' pattern: each path has an associated minimum bandwidth and total distance, and when choosing paths the bandwidth attribute is the most significant, with distance being used as a tiebreaker betweek routes of equal bandwidth.

Let S be $(\mathbb{N} \cup \{\infty\}, \max, \min) \vec{\times} (\mathbb{N} \cup \{\infty\}, \min, +)$, and let G be the S-labelled graph in Figure 4.1. Then the path weights are pairs (b, d), where b is the minimum bandwidth along the path, and d is the total distance.

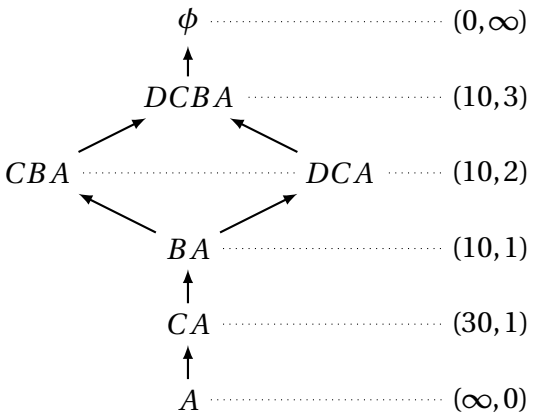


Figure 4.2 The path preference structure for Example 4.1, with associated S weights.

We will show the progress of the algorithm, and associated proof steps, for paths to the node A. The ordered set P of paths is shown in Figure 4.2. We begin with the matrix X_0 with all entries equal to the null path ϕ, whose weight is $(0,\infty)$. The paths computed from each node to A, at each stage of the computation, are shown in Figure 4.3(a); we have $\sigma^k(X_0) = \sigma^4(X_0)$ for every $k \geq 4$. Since this algebra is not monotonic, we have an example here of a node (D) whose path quality decreases: first it receives the path DCA, but it is then forced to use the less-preferred path $DCBA$ instead.

Now, in our convergence proof, we use the V function to map these matrices into a domain of functions, and show that the chain of functions is monotonically decreasing. The values assigned by V to each path in P, at each stage of the computation are shown in Figure 4.3(b). For a fixed p, the sequence of values seen for p follows a fixed pattern. First, the value is 0, indicating that this path is not being used. When the path is first seen, its value changes to 2; thereafter, it drops to 1 for as long as the path is still present. At the final stage, all paths have value either 0 or 1, since we have reached a stable state.

To show that the sequence $\left(V(\sigma^k(X))\right)_{k\geq 0}$ is decreasing, we use the δ function. The desired pattern is that whenever we see $\delta(X)(p)$ being positive, there should be a path q, which is preferred to p, for which $\delta(X)(q)$ is negative. Because of the lexicographic ordering of functions, this is enough to ensure that $V(\sigma(X))(p)$ is less than $V(X)(p)$. The values of δ are shown in Figure 4.3(c) In the proof of Lemma 4.4, there are two ways of finding a q for a given p: either q is a prefix of p which has just been made available by a neighbour, or q is a path from the same node as p which

	A	B	C	D
X_0	ϕ	ϕ	ϕ	ϕ
$\sigma(X_0)$	A	ϕ	ϕ	ϕ
$\sigma^2(X_0)$	A	BA	CA	ϕ
$\sigma^3(X_0)$	A	BA	CBA	DCA
$\sigma^4(X_0)$	A	BA	CBA	$DCBA$

(a) Paths obtained to A

	A	CA	BA	CBA	DCA	$DCBA$	ϕ
$V(X_0)$	2	0	0	0	0	0	3
$V(\sigma(X_0))$	1	2	2	0	0	0	1
$V(\sigma^2(X_0))$	1	0	1	2	2	0	0
$V(\sigma^3(X_0))$	1	0	1	1	0	2	0
$V(\sigma^4(X_0))$	1	0	1	1	0	1	0

(b) The V functions

	A	CA	BA	CBA	DCA	$DCBA$	ϕ
$\delta(X_0)$	-1	2	2	0	0	0	-2
$\delta(\sigma(X_0))$	0	-2	-1	2	2	0	-1
$\delta(\sigma^2(X_0))$	0	0	0	-1	-2	2	0
$\delta(\sigma^3(X_0))$	0	0	0	0	0	-1	0
$\delta(\sigma^4(X_0))$	0	0	0	0	0	0	0

(c) The δ functions

$\delta(X_0)(BA) > 0$	$\delta(X_0)(A) < 0$	path extension
$\delta(X_0)(CA) > 0$	$\delta(X_0)(A) < 0$	path extension
$\delta(\sigma(X_0))(CBA) > 0$	$\delta(\sigma(X_0))(BA) < 0$	path extension
$\delta(\sigma(X_0))(DCA) > 0$	$\delta(\sigma(X_0))(CA) < 0$	path extension
$\delta(\sigma^2(X_0))(DCBA) > 0$	$\delta(\sigma^2(X_0))(DCA) < 0$	lost a preferred path

(d) Proving that the V sequence decreases

Figure 4.3 Matrices and auxiliary functions for Example 4.1.

has just been lost. Both relationships are present in this example; see Figure 4.3(d). In the final case, it would also have sufficed to observe that $\delta(\sigma^2(X_0))(CBA) < 0$. But in general this would not be enough: the decrease at CBA occurs because this was a new path at the previous step, a fact which is unrelated to its being adopted as an alternative to DCA.

The sequences of paths, and associated changes in V, are associated with the convergence proof that is based on 'histories'. Applications of Lemma 4.4 correspond to successive applications of the history construction rule. In the case of this example, the applications of the lemma are shown in Figure 4.3(d), with accompanying reasons for why the path has changed in each instance. The corresponding histories are, for $\sigma^4(X_0)$:

$$\begin{aligned} A&: \quad A \\ B&: \quad BA \longrightarrow A \\ C&: \quad CBA \longrightarrow BA \longrightarrow A \\ D&: \quad DCBA \longrightarrow_d DCA \longrightarrow CA \longrightarrow A \end{aligned}$$

where all arrows indicate transmission arcs, except for the arrow labelled d, which is a dispute arc. According to the history construction, node D has $DCBA$ at the start of its history after losing the path DCA, and these are connected with a dispute arc. The path DCA came about because it was transmitted from node C which had CA, and this happened because C received the direct path to A. The histories for the other nodes arise in a similar way, and for each step in each history, the lemma provides a net decrease in the δ values.

In general, if $p_k \cdots p_2 p_1 p_0$ is a history, then

$$\begin{aligned} \delta(\sigma^{i_k}(X_0))(p_k) > 0 \qquad & \delta(\sigma^{i_k}(X_0))(p_{k-1}) < 0 \\ \delta(\sigma^{i_{k-1}}(X_0))(p_{k-1}) > 0 \qquad & \delta(\sigma^{i_{k-1}}(X_0))(p_{k-2}) < 0 \\ \vdots \qquad & \vdots \\ \delta(\sigma^{i_1}(X_0))(p_1) > 0 \qquad & \delta(\sigma^{i_1}(X_0))(p_0) < 0 \end{aligned}$$

for some increasing sequence $\{i_1, i_2, \ldots, i_k\}$. This provides the required link between this proof and that of Griffin and Wilfong (2000).

Example 4.2. Let $(S, \leq, \otimes)$ be as follows:

- S consists of pairs (d, o) of a *length* d in $\mathbb{N}$ and an *orientation* o in $\{h, v\}$, together with the special elements $\top$ and $\bot$.

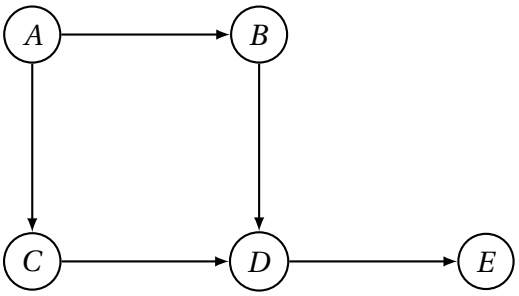


Figure 4.4 Labelled graph for Example 4.2. The horizontal arcs have weight $(1, h)$ and the vertical arcs have weight $(1, v)$.

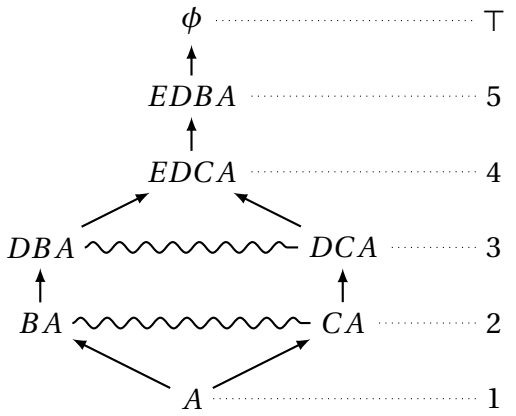


Figure 4.5 The path preference structure for Example 4.2, with associated costs (orientations omitted).

- $(d_1, o_1) \leq (d_2, o_2)$ if and only if $d_1 \leq d_2$; also, $\top$ is greater than all other elements and $\bot$ is less than all other elements.
- $(d_1, o_1) \otimes (d_2, o_2)$ is $(d_1 + d_2 + p, o_1)$, where p is 1 if o_1 and o_2 are different, and 0 if they are the same.

See Lengauer and Theune (1991) for a discussion of a similar algebra. The application is to define the cost of a path on a grid to be its length plus a penalty based on the number of corners. This has uses in the layout of integrated circuits. We will work with the algebra of minimal sets over S. Note that S is not monotonic, and is not a linear order, but it is increasing.

A labelled graph G is shown in Figure 4.4, and the corresponding path preference structure in Figure 4.5. As before, we consider paths from the single source A. The paths obtained are shown in Figure 4.6(a), and the V and δ functions in Figures 4.6(b) and 4.6(c) respectively.

	A	B	C	D	E
X_0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\sigma(X_0)$	$\{A\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\sigma^2(X_0)$	$\{A\}$	$\{BA\}$	$\{CA\}$	$\emptyset$	$\emptyset$
$\sigma^3(X_0)$	$\{A\}$	$\{BA\}$	$\{CA\}$	$\{DBA, DCA\}$	$\emptyset$
$\sigma^4(X_0)$	$\{A\}$	$\{BA\}$	$\{CA\}$	$\{DBA, DCA\}$	$\{EDCA\}$

(a) Paths obtained to A

	A	BA	CA	DBA	DCA	$EDCA$	$EDBA$	ϕ
$V(X_0)$	2	0	0	0	0	0	0	4
$V(\sigma(X_0))$	1	2	2	0	0	0	0	2
$V(\sigma^2(X_0))$	1	1	1	2	2	0	0	1
$V(\sigma^3(X_0))$	1	1	1	1	1	2	0	0
$V(\sigma^4(X_0))$	1	1	1	1	1	1	0	0

(b) The V functions

	A	BA	CA	DBA	DCA	$EDCA$	$EDBA$	ϕ
$\delta(X_0)$	-1	2	2	0	0	0	0	-2
$\delta(\sigma(X_0))$	0	-1	-1	2	2	0	0	-1
$\delta(\sigma^2(X_0))$	0	0	0	-1	-1	2	0	-1
$\delta(\sigma^3(X_0))$	0	0	0	0	0	-1	0	0

(c) The δ functions

Figure 4.6 Matrices and auxiliary functions for Example 4.2.

4.2 *Ultrametrics and a new proof*

We now present a new proof of convergence for nondistributive and increasing algebras. It is inspired by the proof of Sobrinho discussed above, though the mechanics are different. This proof is shorter than the previous two, and incorporates a proof of uniqueness of the fixed point. It covers the minimal-set case as well as the selective case.

The idea is to define a measurement of the distance between two matrices, and to show that application of σ to two matrices reduces the distance between them. This is analogous to the fixed point theorem of Banach (1922), but based on a different kind of metric space. Furthermore, the finiteness of our problem means that much of the apparatus of the fixed point theorem is unnecessary.

First, recall that the *height* of an element x of a partial order $(S, \leq)$ is the length of the longest chain in the lower set of x. In particular, the height of a minimal element is 1. It is easy to see that if $x < y$ then $h(x) < h(y)$, since all elements that are below x are also below y.

Now, not all of our orders $(S, \leq)$ will be finite, and some may not even be countable. It is nonetheless possible to define height on a finite subset of S defined by the weights of every simple path in a graph. Let W be the set $\{w(p) \mid p \in \mathscr{P}_s(G)\}$; then W is a finite subset of S. A height function can be defined on W alone, and it will then be a partial function on S; we still have $h(x) < h(y)$ if x and y are elements of W with $x < y$. In the following, h can be assumed to be a partial function that by construction will only be applied to values in its domain, namely the weights of paths in the graph over which A and σ are defined.

Let H be the extremal value

$$H = \max\{h(w) + 1 \mid w \in W\}. \tag{4.6}$$

Then the quantity $H - h(w)$ ranges between 1 and $H - 1$, for w in W. Clearly, if $x < y$ then $H - h(x) > H - h(y)$.

Assume that in the matrix iteration, each matrix entry is a set of paths, and that within each set, all path weights are equivalent or incomparable. We are therefore working with an algebra of minimal sets over some other algebra. This situation includes the selective case, since then each minimal set is a singleton (or empty). Let S be the underlying algebra; this will be assumed to be increasing, though it is not necessarily selective or monotonic.

We now define

$$d(X, Y) \stackrel{\text{def}}{=} \max\left\{H - h(z) \;\middle|\; \exists i, j : (z \in X_{ij} \wedge z \notin Y_{ij}) \vee (z \notin X_{ij} \wedge z \in Y_{ij})\right\}. \tag{4.7}$$

By convention, if X and Y are equal, we take $\max(\emptyset) = 0$. The following propositions are easily seen to be true for all X, Y and Z:

1. $d(X, Y) \geq 0$;
2. $d(X, Y) = 0$ if and only if $X = Y$;
3. $d(X, Y) = d(Y, X)$;
4. $d(X, Z) \leq \max(d(X, Y), d(Y, Z))$.

These facts make d into an *ultrametric*. Such structures have been extensively studied; see for example Kirk (2001). Other names for the same concept include 'isosceles space' and 'non-archimedean metric'. Ultrametric spaces are similar to metric spaces, but whereas a metric space must satisfy the triangle inequality

$$d(X,Z) \leq d(X,Y) + d(Y,Z),$$

ultrametric spaces must satisfy the fourth axiom above.

Intuitively, d measures the extent to which two matrices X and Y disagree. The set of paths which is in one matrix but not the other is

$$D(X,Y) \stackrel{\text{def}}{=} \bigcup_{i,j} \left(X_{ij} \setminus Y_{ij} \cup Y_{ij} \setminus X_{ij} \right).$$

These paths are ranked based on their S weights; the process is similar to finding the paths in $D(X,Y)$ whose S weights are minimal. But instead of taking

$$d(X,Y) = \min\left(D(X,Y)\right),$$

we define

$$d(X,Y) = \max\{H - h(z) \mid z \in D(X,Y)\}.$$

This is because we want d to take values in a linearly ordered set. Even if S is not linearly ordered, the set of all heights of elements of S must be a linear order. The reason for taking the maximal value of $H - h(z)$ rather than the minimal value of $h(z)$ is to reverse the sense of the order, because we want d to yield larger values for matrices which disagree about low-cost paths, smaller values for matrices which agree about low-cost paths but disagree about high-cost paths, and zero for matrices which agree on all paths.

The *ball* of radius r about a point X in an ultrametric space is the set of all points within a distance r of X: $\{Y \mid d(X,Y) < r\}$. Note that there is no distinction between open and closed balls in the context of ultrametric spaces. An ultrametric space is said to be *spherically complete* if every chain of balls has nonempty intersection. This is true for our example, since the space is finite. There is a general theorem that any strictly contractive mapping on a spherically complete ultrametric space has a unique fixed point (Prieß-Crampe 1990). Lemma 4.9 shows that our σ is strictly contractive, and so Theorem 4.10 is a special case of the Prieß-Crampe theorem, applicable when the range of d is finite. The proof is given here in full to illustrate how the unique fixed point result works in this case.

Lemma 4.9. *For all matrices X and Y, if $\sigma(X) \neq \sigma(Y)$ then $d(\sigma(X), \sigma(Y)) < d(X, Y)$.*

Proof. If $\sigma(X) \neq \sigma(Y)$ then $d(\sigma(X), \sigma(Y))$ cannot be 0. So there is a pair (i, j) on which $\sigma(X)$ and $\sigma(Y)$ disagree, and a path p that is in one matrix but not the other, having minimal height. Without loss of generality, assume that p is in $\sigma(X)_{ij}$ but not $\sigma(Y)_{ij}$.

There can be no p' in $\sigma(Y)_{ij}$ for which $w(p') \prec w(p)$. Suppose that we could find such a p'. Then that p' could not be in $\sigma(X)_{ij}$, because it is a set of equivalent or incomparable elements and already contains p. Hence both p and p' are in the disagreement set $D(X, Y)$. It is also the case that $H - h(p') > H - h(p)$. This contradicts the choice of p as an element of minimal height in $D(X, Y)$. Consequently no such p' can exist.

By the definition of σ, there must be a neighbour k of i and a path q in X_{kj} such that $w(p) = A_{ik} \otimes w(q)$. Because S is increasing, we then have $w(q) < w(p)$, so $h(q) < h(p)$ and $H - h(p) < H - h(q)$.

Is q also in Y_{kj}? This cannot be the case, since $\sigma(Y)_{ij}$ does not contain p nor any path that is preferred to p. If the path q had been in Y_{kj}, then $\sigma(Y)_{ij}$ would have to include its extension p; but this did not happen. Consequently the path q does not appear in Y_{kj}.

Hence X and Y disagree at (k, j), because X_{kj} contains the path q and Y_{kj} does not. Therefore the value of $d(X, Y)$ must be at least $H - h(q)$. We have

$$d(\sigma(X), \sigma(Y)) = H - h(p) < H - h(q) \leq d(X, Y)$$

as required. □

Theorem 4.10. *If S is increasing, then σ has a unique fixed point.*

Proof. We first show that if there is a fixed point, then it is unique. By the preceding lemma, we know that $d(X, Y) > d(\sigma(X), \sigma(Y))$ unless $\sigma(X) = \sigma(Y)$. So if $X = \sigma(X)$ and $Y = \sigma(Y)$ then $X = Y$, since otherwise we would have $d(X, Y) > d(X, Y)$.

It is also the case that there is at least one fixed point of σ, which we can obtain by repeated iteration. Take X to be any matrix, and consider the sequence $\{X, \sigma(X), \sigma^2(X), \ldots\}$. By Lemma 4.9,

$$d(X, \sigma(X)) > d(\sigma(X), \sigma^2(X)) > d(\sigma^2(X), \sigma^3(X)) > \cdots$$

is a strictly descending chain in $\mathbb{N}$, and it must therefore be of finite length. Therefore we reach a k for which $\sigma^k(X) = \sigma^{k+1}(X)$, and so this $\sigma^k(X)$ is the required fixed point.

□

This theorem proves that use of **minset**(S) always leads to a unique fixed point, provided that S is increasing. In particular, S need not be selective, and in this case **minset**(S) will also not be selective.

Another interpretation of the theorem is that is shows that even if an increasing nondistributive algebra S is not known to converge, then **minset**(S) will definitely converge. This is true, but it is important to realize that S and **minset**(S) may behave quite differently, and it is not necessarily possible to use one as a substitute for the other.

Suppose that x and y are incomparable elements of $(S, \oplus, F)$. Then their sum $x \oplus y$ will be some element of S that is below both x and y in the natural order; furthermore, given this element it is not necessarily possible to recover x and y. But in **minset**(S) the sum of $\{x\}$ and $\{y\}$ will be $\{x, y\}$. While this is also ordered below both $\{x\}$ and $\{y\}$, in this case the elements x and y can be recovered (though this is not to say that sets can be uniquely factorized). Now, it might be thought that these sets can be 'flattened' with $(\oplus)$, as

$$\bigoplus_{z \in \{x,y\}} z = x \oplus y$$

to recover the same results as if S had been used all along instead of **minset**(S). This is *not* the case, because of the nondistributivity of F over $(\oplus)$. In **minset**(S), a node might receive $\{x\}$ and $\{y\}$ from its two neighbours, compute $\{x, y\}$, and pass on $\{f(x), f(y)\}$ to another neighbour over an arc labelled with f. The analogous situation in S is for the node to receive x and y, compute $x \oplus y$, and pass on $f(x \oplus y)$. But unless f distributes over $(\oplus)$, the flattening $f(x) \oplus f(y)$ of $\{f(x), f(y)\}$ will not be equal to $f(x \oplus y)$.

Not every nondistributive algebra arises as **minset**(S) for some S. The results of Chapter 3 show that $A =$ **minset**(S) if and only if A is a distributive lattice. While there are many distributive lattices, there are even more partial orders and preorders, and so there remain many algebras for which we have not yet proved a convergence criterion.

Chapter 5

Lexicographic choice

One of our primary means of constructing new algebras will be the use of *lexicographic products*, which implement *lexicographic choice*.

Lexicographic choice is a method of making preference decisions based on multiple criteria. The available criteria are ranked in importance, and evaluated in this order; the evaluation ends when a criterion has yielded a decisive result. That is, the less important criteria are used to break ties arising from the more important end of the ranking. The term 'lexicographic' derives from the sorting order of dictionary head-words: the first letter is the most significant, then the second, and so on; so 'sausage' is sorted after 'sauce' and before 'saveloy'. In the routing context, the criteria will usually be different attributes of a route, such as the end-to-end delay or the bandwidth. Suppose that routes p, q and r have the attributes shown in Figure 5.1.

There are then six ways in which these attributes might be considered in a lexicographic order. We use the symbol '$\vec{\times}$' to denote such combinations, so '$B \vec{\times} D$' refers to the ordering where bandwidth is the primary criterion, with the delay attribute being used to break ties if the bandwidth was not decisive. The six combinations give rise to the orderings for p, q and r shown in Figure 5.2.

Evidently, decisions about the significance to be assigned to different attributes can completely change the resulting preference order on routes. The question of which of these six lexicographic products is 'the best' cannot be answered definitively, since any of them might be justified in a given scenario; the choice is an engin-

	Bandwidth (B)	Delay (D)	Reliability (R)
p	high	long	high
q	low	short	high
r	high	short	low

Figure 5.1 Three routes, each with three attributes

Lexicographic product	Preference
$B \vec{\times} D \vec{\times} R$	$r < p < q$
$B \vec{\times} R \vec{\times} D$	$p < r < q$
$D \vec{\times} B \vec{\times} R$	$r < q < p$
$D \vec{\times} R \vec{\times} B$	$q < r < p$
$R \vec{\times} B \vec{\times} D$	$p < q < r$
$R \vec{\times} D \vec{\times} B$	$q < p < r$

Figure 5.2 Six lexicographic products and the resulting route preferences

eering decision. From the metarouting point of view, we *can* provide the mathematical tools that will help the protocol designer to choose: and so we need to develop the theory of lexicographic products.

Lexicographic choice is important to us for two reasons. Most obviously, it is a useful way of combining algebras that is already familiar in Internet routing practice. But it is also an important test case for the metarouting approach: the operator is comparatively simple, while not being trivial. This chapter will be a case study in developing the correct algebraic definition of the product, and its accompanying property inference rules. This will demonstrate the metalanguage-based approach, and the rules themselves will also be of interest because of their implications for routing protocol design.

In general, if multiple criteria are to be taken into account when choosing routes, then lexicographic choice may be useful. The Border Gateway Protocol uses lexicographic choice for selecting routes, and there are many attributes that are used in the selection process. The main alternatives are to use only a single criterion, as in RIP, where a path weight is simply a number, or to combine multiple criteria according to some formula. This latter choice is the case for EIGRP, which combines delay and bandwidth values by a function

$$\lambda d, b \,.\, d + \frac{k}{b}$$

where k is a fixed constant (optionally, other criteria can be incorporated into the formula as well). The resulting weights are used to select routes, effectively defining a preference relation on delay-bandwidth pairs. In fact, the EIGRP metric also incorporates lexicographic choice: before this formula is applied, a check is made to ensure that individual attribute values are within certain bounds.

An advantage of the lexicographic method is that it is straightforward to use with non-numeric metrics, including those which are not linear orders. Decision-making

can also be more transparent, since it is obvious which component metrics have been involved in selecting a particular route. Consequently, it is clearer how weights can be manipulated to achieve a different outcome.

Lexicographic decision-making has been studied in the economics literature for many decades. Its most important early appearance was in the work of Debreu (1954), who showed that induced choice is not sufficient to represent all linear orders: his counterexample was that there is no injective order-preserving function from $\mathbb{R}_{\geq 0} \vec{\times} \mathbb{R}_{\geq 0}$ to $\mathbb{R}_{\geq 0}$. This demonstrates that the use of a formula does not subsume lexicographic choice even for linear orders, so lexicographic choice is a worthwhile method to investigate.

An objection to this choice method is that it does not model how human beings assess their options in life. We do not usually think of our decision-making criteria as being part of a fixed hierarchy; some other model of choice may be a better psychological fit. Is this a problem for the use of lexicographic choice in protocols? Routing metrics are designed so as to yield results that operators believe to be in some sense 'reasonable' for the way that traffic should flow. Lexicographic choice may not be the only method that is reasonable here, but the evidence from existing protocols suggests that it is an important part of the picture. In particular, we will see in Chapter 6 that lexicographic choice is essential for modelling hierarchical network partitions. The ranking of criteria makes sense here because each criterion is set at a different scope, and local configuration changes should have only a local effect.

5.1 *Lexicographic product in orders and monoids*

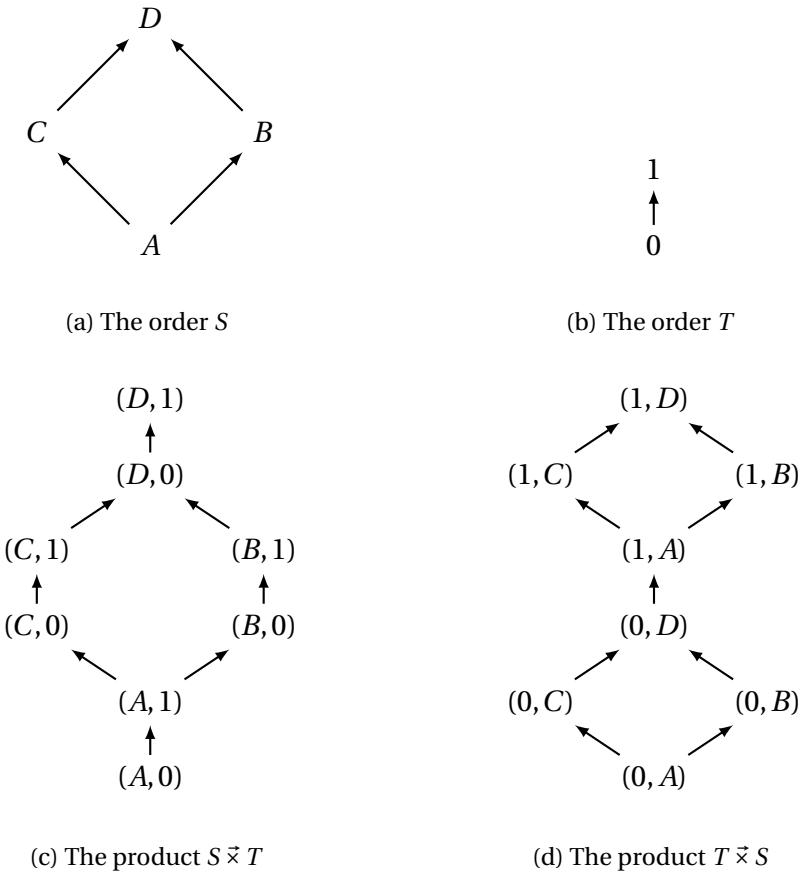
In the case of preorders, the lexicographic product is easy to define. Suppose that we are given two preorders $(S, \leq_S)$ and $(T, \leq_T)$. The order which implements lexicographic choice is their *lexicographic product* $(S \times T, \leq_{S \vec{\times} T})$, where

$$(s_1, t_1) \leq_{S \vec{\times} T} (s_2, t_2) \iff (s_1 <_S s_2) \vee (s_1 \sim_S s_2 \wedge t_1 \leq_T t_2). \tag{5.1}$$

Here, $s_1 \sim_S s_2$ if and only if both $s_1 \leq_S s_2$ and $s_2 \leq_S s_1$; and $s_1 <_S s_2$ if and only if $s_1 \leq_S s_2$ but $\neg(s_2 \leq_S s_1)$. So the T component is only used to break ties for S, and this tiebreaking does not occur if the S elements are incomparable.

This definition can be expressed in terms of relational algebra as

$$(\leq_{S \vec{\times} T}) = (<_S \times *_T) \cup (\sim_S \times \leq_T). \tag{5.2}$$



(a) The order S

(b) The order T

(c) The product $S \vec{\times} T$

(d) The product $T \vec{\times} S$

Figure 5.3 Lexicographic product of two partial orders

We can then derive relational characterizations of the strict version of the order, of equivalence, and of incomparability:

$$(<_{S \vec{\times} T}) = (<_S \times *_T) \cup (\sim_S \times <_T) \tag{5.3}$$

$$(\sim_{S \vec{\times} T}) = (\sim_S \times \sim_T) \tag{5.4}$$

$$(\#_{S \vec{\times} T}) = (\#_S) \cup (\sim_S \times \#_T). \tag{5.5}$$

Here, '$*_T$' denotes the complete relation on T, so that $t_1 *_T t_2$ for all t_1 and t_2 in T.

Figure 5.3 demonstrates the lexicographic product when applied to two partial orders S and T. The resulting *set* is the same as for the direct product, but the order on that set is not the direct product order. It can also be seen that the order of the operands is important: if they were reversed, then a different result would emerge from the lexicographic product: $S \vec{\times} T$ is not the same as $T \vec{\times} S$.

At the beginning of this chapter, we formed triple lexicographic products like '$B \vec{\times} R \vec{\times} D$'. This notation is ambiguous unless the $(\vec{\times})$ operator is associative. In fact, this is the case.

Theorem 5.1. *The lexicographic product operator on preorders is associative.*

Proof. Using the relational characterization, we have, for all preorders $(S, \leq_S)$, $(T, \leq_T)$ and $(U, \leq_U)$:

$$\begin{aligned}
(\leq_{S \vec{\times} (T \vec{\times} U)}) &= (<_S \times *_{T \vec{\times} U}) \cup (\sim_S \times \leq_{T \vec{\times} U}) \\
&= (<_S \times *_T \times *_U) \cup (\sim_S \times ((<_T \times *_U) \cup (\sim_T \times \leq_U))) \\
&= (<_S \times *_T \times *_U) \cup (\sim_S \times <_T \times *_U) \cup (\sim_S \times \sim_T \times \leq_U) \\
&= (((<_S \times *_T) \cup (\sim_S \times <_T)) \times *_U) \cup (\sim_{S \vec{\times} T} \times \leq_U) \\
&= (<_{S \vec{\times} T} \times *_U) \cup (\sim_{S \vec{\times} T} \times \leq_U) \\
&= (\leq_{(S \vec{\times} T) \vec{\times} U}).
\end{aligned}$$

□

We would like to define a similar operator for monoids. As a matter of convenience and mathematical cleanliness, such a definition would permit lexicographic choice to be expressed purely in terms of monoids, rather than in terms of an intermediate constructed partial order. More importantly, making a monoid definition in addition to the order definition will make clear some of the relationships between these two structures.

What does it mean to make a 'similar definition'? We already know how to turn a commutative and idempotent monoid into a partial order, and how to turn an order possessing least upper bounds or greatest lower bounds into a monoid. Our construction of a monoidal lexicographic product will thus be informed by these operations.

The monoidal lexicographic product we want is one which gives the same result as the order lexicographic product when applied to the same arguments. Of course, the arguments and results here are of different types—orders and monoids—so we need to make some kind of translation. The natural order seems to be the most obvious. But note that this only applies to certain monoids: those which are commutative and idempotent. So for this special case, we need

$$\mathbf{natord}\,((S, \oplus_S) \mathbin{\vec{\times}} (T, \oplus_T)) = \mathbf{natord}(S, \oplus_S) \mathbin{\vec{\times}} \mathbf{natord}(T, \oplus_T) \tag{5.6}$$

for all monoids $(S, \oplus_S)$ and $(T, \oplus_T)$. Here, **natord** returns the partial order corresponding to the given monoid. The first '$\vec{\times}$' is the desired monoidal version, and the second is the known preorder version.

Example 5.1. Let S be the semigroup $(\mathbb{N}, \sqcup)$ and let T be the semigroup $(2^{\mathbb{N}}, \cup)$. Their natural orders $(\leq_S)$ and $(\leq_T)$ are given by

$$m \leq_S n \overset{\text{def}}{\iff} m = m \sqcup n \iff m \leq n$$
$$A \leq_T B \overset{\text{def}}{\iff} A = A \cup B \iff A \supseteq B$$

and their lexicographic product $(\leq)$ is

$$\begin{aligned}(m, A) \leq (n, B) &\overset{\text{def}}{\iff} m <_S n \vee (m = n \wedge A \leq_T B) \\ &\iff m < n \vee (m = n \wedge A \supseteq B).\end{aligned}$$

We want to find a binary operator $(\oplus)$ for which $(m, A) \leq (n, B)$ if and only if $(m, A) = (m, A) \oplus (n, B)$. This will be the monoidal lexicographic product of S and T.

In this example, the operator can be verified to be

$$(m, A) \oplus (n, B) \overset{\text{def}}{=} \begin{cases} (m, A) & \text{if } m < n \\ (n, B) & \text{if } n < m \\ (m, A \cup B) & \text{if } m = n. \end{cases}$$

This is the only binary operator which stands in the required relationship with the natural order.

Let us try to define $(S, \oplus_S) \vec{\times} (T, \oplus_T)$ for general semigroups. The underlying set of the semigroup will certainly be $S \times T$; we need to define a semigroup operation that will match our understanding of how the lexicographic product 'ought' to behave.

We will first assume that S is selective and commutative: so for all s_1 and s_2 in S, $s_1 \oplus_S s_2$ is equal to s_1 or s_2 (or both), and $s_1 \oplus_S s_2 = s_2 \oplus_S s_1$. It follows that S has to be idempotent.

Suppose that s_1 and s_2 are distinct, and $s_1 \oplus_S s_2 = s_1$: we interpret the semigroup operation as choice, meaning that s_1 is strictly preferable to s_2. Since the S component of the lexicographic product must dominate the T component, we define $(s_1, t_1) \oplus (s_2, t_2) = (s_1, t_1)$ in this case. Because the S choice was decisive, the T values should not be considered. Symmetrically, if $s_2 = s_1 \oplus_S s_2 \neq s_1$, then the result of $(s_1, t_1) \oplus (s_2, t_2)$ should be (s_2, t_2).

The final case is that s_1 and s_2 could coincide. We should then consider the T component in determining the final choice: $(s_1, t_1) \oplus (s_1, t_2) = (s_1, t_1 \oplus_T t_2)$. This completes the definition.

Definition 5.1. Let $(S, \oplus_S)$ be a selective, commutative semigroup and T be a semigroup. Then $S \vec{\times} T = (S \times T, \oplus)$ is the *lexicographic product* of S and T, where

$$(s_1, t_1) \oplus (s_2, t_2) = \begin{cases} (s_1, t_1) & \text{when } s_1 = s_1 \oplus_S s_2 \neq s_2 \\ (s_2, t_2) & \text{when } s_2 = s_1 \oplus_S s_2 \neq s_1 \\ (s_1, t_1 \oplus_T t_2) & \text{when } s_1 = s_2. \end{cases}$$

We can prove that this does indeed define a semigroup, because the $\oplus$ operation is associative. Note that it does not have to be selective or commutative, since this is not required from T.

Theorem 5.2. *Let $(S, \oplus_S)$ be a selective, commutative semigroup and $(T, \oplus_T)$ be a semigroup. Then $S \vec{\times} T$ is associative.*

Proof. Let (s_1, t_1), (s_2, t_2) and (s_3, t_3) be elements of $(S \times T, \oplus)$, and let

$$(s_L, t_L) = \big((s_1, t_1) \oplus (s_2, t_2)\big) \oplus (s_3, t_3)$$
$$(s_R, t_R) = (s_1, t_1) \oplus \big((s_2, t_2) \oplus (s_3, t_3)\big).$$

Note that s_L and s_R will always be equal, since S is associative: $s_L = s_R = s_1 \oplus_S s_2 \oplus_S s_3$.

If S is selective then $s_L = s_R \in \{s_1, s_2, s_3\}$; and some of s_1, s_2 and s_3 may coincide. In each case, we find that t_L and t_R are the same:

$s_1 \oplus_S s_2 \oplus_S s_3$	t_L	t_R
$s_1 = s_2 = s_3$	$(t_1 \oplus_T t_2) \oplus_T t_3$	$t_1 \oplus_T (t_2 \oplus_T t_3)$
$s_1 = s_2$	$t_1 \oplus_T t_2$	$t_1 \oplus_T t_2$
$s_1 = s_3$	$t_1 \oplus_T t_3$	$t_1 \oplus_T t_3$
$s_2 = s_3$	$t_2 \oplus_T t_3$	$t_2 \oplus_T t_3$
s_1	t_1	t_1
s_2	t_2	t_2
s_3	t_3	t_3

Therefore $\oplus$ is associative. □

This agrees with Example 5.1. Let us consider an example where the second component is *not* idempotent.

Example 5.2. Let S be as in Example 5.1, and let T be $(\mathbb{R}_{\geq 0}, +)$. Their lexicographic product is $(S \times T, \oplus)$, where

$$(m,x) \oplus (n,y) \stackrel{\text{def}}{=} \begin{cases} (m,x) & \text{if } m < n \\ (n,y) & \text{if } n < m \\ (m,x+y) & \text{if } m = n. \end{cases}$$

This is associative and commutative, but is not idempotent; nor is it selective. Because $S \vec{\times} T$ is not idempotent, we cannot derive a natural order.

However, some elements are idempotent: since $(m,x) \oplus (m,x)$ is equal to $(m,2x)$, an element (m,x) is idempotent precisely when x is zero, and the set of idempotents $E(S \vec{\times} T)$ is $\{(m,0) \mid m \in \mathbb{N}\}$. This subsemigroup can have a natural order defined over it, which is isomorphic to the order on S:

$$(m,0) = (m,0) \oplus (n,0) \iff m \leq n.$$

We have seen that a lexicographic product operation can be defined over certain semigroups. If these are semilattices, this product corresponds to the definition of lexicographic product in terms of order relations.

So far, we have seen a definition of lexicographic products for the case where the first operand is a selective semilattice, and the second is any semigroup. The result of this operation will be a semigroup, but it will not necessarily be selective or idempotent. This restricts our ability to form multiple products. It would also be desirable to be able to form products where the first operand is not selective: in order-theoretic terms, when it is not necessarily a linear order, but could be a partial order or preorder.

If S is not selective, then Definition 5.1 is not enough: we need a case for when $s_1 \oplus_S s_2$ is neither s_1 nor s_2. Which element of T should be chosen in this case? It is tempting to say that it should be $t_1 \oplus_T t_2$; however, this would lead to associativity being violated. Suppose that s_1 and s_2 are elements of S with $s_1 \oplus_S s_2 \notin \{s_1, s_2\}$, and choose elements t and t' from T. Then, using the proposed rule, we have

$$\begin{aligned} &(s_1,t) \oplus \big((s_2,t) \oplus (s_1 \oplus_S s_2, t')\big) \\ =&(s_1,t) \oplus (s_1 \oplus_S s_2, t') \\ =&(s_1 \oplus_S s_2, t') \end{aligned}$$

whereas

$$\begin{aligned}&\big((s_1, t) \oplus (s_2, t)\big) \oplus (s_1 \oplus_S s_2, t')\\ =&(s_1 \oplus_S s_2, t \oplus_T t) \oplus (s_1, t')\\ =&(s_1 \oplus_S s_2, t \oplus_T t \oplus_T t').\end{aligned}$$

Unless T is trivial, $t \oplus_T t \oplus_T t'$ and t' will not be the same for every t and t' in T.

If T has an identity α_T, then we can define

$$(s_1, t_1) \oplus (s_2, t_2) = (s_1 \oplus_S s_2, \alpha_T) \qquad \text{when } s_1 \oplus_S s_2 \notin \{s_1, s_2\}$$

in addition to the previous cases, and associativity is regained.

Definition 5.2. Let $(S, \oplus_S)$ be a commutative idempotent semigroup and T be a monoid. Then $S \vec{\times} T = (S \times T, \oplus)$ is the *lexicographic product* of S and T, where

$$(s_1, t_1) \oplus (s_2, t_2) = \begin{cases} (s_1 \oplus_S s_2, t_1 \oplus_T t_2) & s_1 = s_1 \oplus_S s_2 = s_2 \\ (s_1 \oplus_S s_2, t_1) & s_1 = s_1 \oplus_S s_2 \neq s_2 \\ (s_1 \oplus_S s_2, t_2) & s_1 \neq s_1 \oplus_S s_2 = s_2 \\ (s_1 \oplus_S s_2, \alpha_T) & \text{otherwise.} \end{cases}$$

The two definitions 5.1 and 5.2 coincide when S is a selective commutative semigroup and T is a monoid, and in both cases, the $(\vec{\times})$ operator is associative.

There is an overlap between the classes of semigroups and orders: a semilattice can be perceived as either an ordered set in which any two elements have a unique greatest lower bound, or as a monoid whose operation is commutative and idempotent. The lexicographic product of two semilattices is the same, whether the order-theoretic or semigroup-theoretic product is used.

5.2 *Inference rules*

For use in the context of generic pathfinding algorithms, the key algebraic properties are that a structure be

- distributive or monotonic, for finding global optima; or
- increasing, for finding Nash equilibria.

We will also need to find inference rules for more basic properties: commutativity, idempotence, and selectivity. These rules are:

$$\text{COMM}(S \mathbin{\vec{\times}} T) \iff \text{COMM}(S) \wedge \text{COMM}(T) \tag{5.7}$$

$$\text{IDEM}(S \mathbin{\vec{\times}} T) \iff \text{IDEM}(S) \wedge \text{IDEM}(T) \tag{5.8}$$

$$\text{SEL}(S \mathbin{\vec{\times}} T) \iff \text{SEL}(S) \wedge \text{SEL}(T) \tag{5.9}$$

These apply when S and T are semigroups; they are the semigroup counterparts of the statements that the lexicographic product of orders preserves the properties of reflexivity, antisymmetry, and being a linear order. See Section A.2 in Appendix A for the proofs of these statements.

5.2.1 *Rules for global optima*

Our main results for the monotonicity property are based on a theorem by Saitô (1970). His theorem is for order semigroups, in the special case when the order is linear. (Recall that an order semigroup is a structure $(S, \preceq, \otimes)$ consisting of a preordered set and a semigroup operation over that set.) We will extend this to a more general case, and to a wider variety of structures. In each case, the theorem has substantially the same structure, but some of the properties needed are different.

The monotonicity property, MONO, is defined for order semigroups as

$$\text{MONO} = \forall a, b, c \in S : a \preceq b \implies (c \otimes a) \preceq (c \otimes b). \tag{5.10}$$

We will also need to track whether an order semigroup is left-cancellative (Definition 2.6) or left-condensed (Definition 2.9); these properties are

$$\text{CANC} = \forall a, b, c \in S : c \otimes a = c \otimes b \implies a = b \tag{5.11}$$

$$\text{COND} = \forall a, b, c \in S : c \otimes a = c \otimes b. \tag{5.12}$$

Note that these two properties are only for the semigroup part, not the order part, of the order semigroup.

The cancellative property is true for some important examples of semigroups, such as $(\mathbb{N}, +)$, but it fails to hold for some other equally important examples, including $(\mathbb{N}, \sqcup)$ and $(\wp A, \cup)$ for a set A.

The property of being condensed is stricter still. It is true for any left zero semigroup (where $c \otimes a = c$ for all a and c) and for any null semigroup (where there is a fixed element z such that $c \otimes a = z$ for all a and c), but these are not the only examples. If f is a function on S for which $f = f \circ f$, then let $a \oplus b \stackrel{\text{def}}{=} f(a)$; then $(S, \oplus)$ is a

Family	Property	Definition
Bisemigroups	M	$\forall a, b, c : c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b)$
	C	$\forall a, b, c : c \otimes a = c \otimes b \implies a = b$
	K	$\forall a, b, c : c \otimes a = c \otimes b$
Order semigroups	M	$\forall a, b, c : a \preceq b \implies (c \otimes a) \preceq (c \otimes b)$
	C	$\forall a, b, c : c \otimes a \sim c \otimes b \implies a \sim b \vee a \# b$
	K	$\forall a, b, c : c \otimes a \sim c \otimes b$
Semigroup transforms	M	$\forall a, b, f : f(a \oplus b) = f(a) \oplus f(b)$
	C	$\forall a, b, f : f(a) = f(b) \implies a = b$
	K	$\forall a, b, f : f(a) = f(b)$
Order transforms	M	$\forall a, b, f : a \preceq b \implies f(a) \preceq f(b)$
	C	$\forall a, b, f : f(a) \sim f(b) \implies a \sim b \vee a \# b$
	K	$\forall a, b, f : f(a) \sim f(b)$

Figure 5.4 Properties required for global optima, for several algebraic structures

left-condensed semigroup. Equally, for any left-condensed semigroup, such a function f can be found: let $f(a) \stackrel{\text{def}}{=} a \otimes a$. Consequently, the number of left-condensed semigroups of order n, up to isomorphism, is the same as the number of partitions of n. This series has generating function

$$\prod_{k \geq 1} \frac{1}{1 - x^k}$$

and begins

$$1, 2, 3, 5, 7, 11, 15, 22, 30, 42, \ldots$$

There are therefore more left-condensed semigroups than might be immediately apparent from considering the trivial examples, though most of these cannot be expected to be useful in routing.

Theorem 5.3 (Saitô)**.** *Let S and T be order semigroups whose orders are linear. Then*

$$\text{MONO}(S \vec{\times} T) \iff \text{MONO}(S) \wedge \text{MONO}(T) \wedge (\text{CANC}(S) \vee \text{COND}(T)) .$$

Note that we have one property applying to the first operand, and as an alternative, another property applying to the second. This allows binary lexicographic products to be monotonic for a wide choice of operands: while both must be monotonic, only one needs to have the further constraint of being cancellative or condensed.

This theorem can be extended to order semigroups where the order is not necessarily linear. Analogues of the theorem can also be proved for bisemigroups, for

order transforms, and for semigroup transforms. The theorem statement is the same, once the appropriate definitions of the properties have been adapted. See Figure 5.4 for how the properties appear for these different structures. What was MONO is now M, and similarly CANC and COND have become C and K. The new names reflect the generality of the properties: K applies to more than semigroups, so the name 'COND' is no longer appropriate.

Theorem 5.4. *For each family of structures in Figure 5.4, if S and T are both from that family, then*

$$\mathrm{M}(S \mathbin{\vec{\times}} T) \iff \mathrm{M}(S) \wedge \mathrm{M}(T) \wedge (\mathrm{C}(S) \vee \mathrm{K}(T))$$

where the properties M, C *and* K *are the appropriate versions.*

Proof. See section A.3. □

The combination of M and C yields 'strict monotonicity', the requirement that if a is strictly preferred to b then $c \otimes a$ is strictly preferred to $c \otimes b$. We define

$$\mathrm{SM} = \mathrm{M} \wedge \mathrm{C} = \forall a, b, c : a \prec b \implies c \otimes a \prec c \otimes b. \tag{5.13}$$

It has previously been shown (Gouda and Schneider 2003) that if $\mathrm{SM}(S)$ and $\mathrm{M}(T)$, then $\mathrm{M}(S \mathbin{\vec{\times}} T)$. In contrast, our theorem gives necessary and sufficient conditions, and in so doing reveals a second case: if T has the K property, and both S and T have M, then $\mathrm{M}(S \mathbin{\vec{\times}} T)$.

In extending Saitô's result to preorders rather than linear orders, the properties CANC and COND have changed to C and K. This is because of the new possibilities for the ordering of elements which may be encountered in a preorder: they may be equivalent, although not equal, or they may be incomparable. The new properties take account of these new cases.

The new K condition requires only that $c \otimes a$ always be equivalent to $c \otimes b$, rather than that they should be equal, as required by the original COND. Similarly, the C property has changed to allow cancelling when $c \otimes a$ is equivalent to $c \otimes b$: then a must be equivalent to b, *or* they must be incomparable.

The effect of this is to rule out the possibility that $c \otimes a \sim c \otimes b$ while a and b are strictly ordered, a state of affairs which would result in monotonicity being violated for the lexicographic product. Suppose that $s_1 \prec_S s_2$ and $s_3 \otimes_S s_1 \sim_S s_3 \otimes_S s_2$; then $(s_1, t_1) \prec (s_2, t_2)$ lexicographically. For monotonicity, we need

$$(s_3 \otimes_S s_1, t_3 \otimes_T t_1) \preceq (s_3 \otimes_S s_2, t_3 \otimes_T t_2)$$

for any s_3 and t_3, and this is only true when

$$t_3 \otimes_T t_1 \preceq t_3 \otimes_T t_2.$$

Unless K(T) holds, this will not be true in all cases, causing monotonicity to fail. The lack of C for S allows an apparent *reversal of preferences*, due to the previously-ignored T component being revealed.

We will now see several examples of the lexicographic product as it appears for the various algebraic families. These examples are intended to illustrate the diversity of structures which can be handled by the algebraic framework, and the ease of inference of the desired properties.

Example 5.3. Let S be $(\mathbb{N}, \leq, F)$, where F is the set of functions

$$\{\lambda x \,.\, ax + b \mid a \in \mathbb{N}, b \in \mathbb{N}\}.$$

This has the M property but not C, since if $x \leq y$ then $ax + b \leq ay + b$ for all natural numbers a and b, but if a is zero then we can have $ax + b = ay + b = b$ even though x and y are different.

Let T be the set $(\mathbb{N}, \leq, \{\kappa_n \mid n \in \mathbb{N}\})$; this has both the M and the K properties.

Then $S \vec{\times} T$ is monotonic. This product resembles the order lexicographic product $(\mathbb{N}, \leq) \vec{\times} (\mathbb{N}, \leq)$, but with a more complicated multiplicative part. As in the more familiar lexicographic product, there are two numeric components to the metric which are considered in order. But in this example, these two items of data are forced to arise as follows:

1. The first value arises from a composition of linear functions, the input of the composite function being determined by the source of the path. Suppose that a given path has arcs labelled with $\lambda x \,.\, a_i x + b_i$, for i in $\{1,2,3,4\}$. Then the weight of the whole path is given by

   $$a_4 a_3 a_2 a_1 x + a_4 a_3 a_2 b_1 + a_4 a_3 b_2 + a_4 b_3 + b_4$$

   where x is the value originated at the source node.

2. The second value is entirely determined by weight of the final arc on the path.

This demonstrates that property deduction does not require very much additional effort for more complicated examples, compared to the simple examples usually seen.

Example 5.4. Let S be the bisemigroup $(\mathbb{N}, \sqcup, +) \times (\mathbb{N}, \sqcup, +)$. This direct product has the M and C properties, because so does each of its components. If a graph is labelled by S, then the results of the iterative matrix algorithm may not correspond to any single path in the graph. For example, if there are two paths between node i and node j, one of weight $(3,5)$ and one of weight $(7,4)$, then their sum in S is $(3,4)$; this information is not associated with any one path, but with a combination.

Similar situations occur with other algebras which admit incomparability. Note that in this case, the algorithm is equivalent to performing two separate best-path computations, one for each component of the product: so the weight $(3,4)$ represents a path of weight 3 with respect to the first weighting and a separate path of length 4 with respect to the second weighting. However, there are other algebras for which results are not associated with a single path, and which cannot be decomposed in that way, such as the power set algebra $(\wp X, \cup, \cap)$ for a fixed set X.

Now, let T be the bisemigroup $(\{\top, \bot\}, \wedge, \vee)$, and form the lexicographic product $S \vec{\times} T$. Suppose that a graph is weighted by this product, and that every T-weight is $\bot$. Then every path also has a T-weight of $\bot$, because $\bot = \bot \vee \bot \vee \cdots \vee \bot$. When two paths are combined with the lexicographic $(\oplus)$ operation, we find that $\top$ only emerges in two situations:

1. If the S-weights of the paths are incomparable, then the output T-weight will be $\top$, the identity for $(\wedge)$.

2. If the S-weights are comparable, and the better path has a T-weight of $\top$, then the output T-weight will also be $\top$.

Consequently, the T component of the algebra acts as a flag indicating whether or not an S-weight is associated with a single path $(\bot)$ or a combination $(\top)$.

Because we have $\mathrm{M}(S)$, $\mathrm{C}(S)$ and $\mathrm{M}(T)$, the product $S \vec{\times} T$ is monotonic, and can be used for finding globally optimal paths. Each result $((a,b),t)$ consists of path weights a and b, which are the weights of optimal paths with respect to the two weightings, together with a flag value t showing whether or not the two weights correspond to the same path.

Example 5.5. It may not be obvious how the C property for preorders contributes to monotonicity. This example illustrates the need for the incomparability case. Let S be the order semigroup $(X, =, \lhd)$ for a fixed set X. Then S has the C property, because the order is discrete. For T, take $(\mathbb{N}, \sqcup, +)$. Both S and T are monotonic.

Our theorem tells us that $S \vec{\times} T$ should be monotonic, as indeed it is. The order here is

$$(s_1, t_1) \preceq (s_2, t_2) \iff (s_1 = s_2 \wedge t_1 \leq t_2)$$

and the multiplication is given by

$$(s_1, t_1) \otimes (s_2, t_2) = (s_1, t_1 + t_2).$$

Now, $((s,t) \otimes (s_1, t_1)) \preceq ((s,t) \otimes (s_2, t_2))$ if and only if $t + t_1 \leq t + t_2$ (and $s = s$). This is certainly implied by $t_1 \leq t_2$, so we have monotonicity. See Figure 5.5 for an illustration of the case $X = \{0, 1\}$.

One possible use of this example can be seen when $X = E$, and each arc in the graph has itself as a label—for example, an arc from node 17 to node 3 of weight 4 would be labelled $((17,3),4)$. When doing multipath routing in this scenario, each origin-destination pair will end up with several paths; because of the lexicographic order, there will be at least one path associated with each incoming adjacency of the destination node. We will therefore have found not just the shortest overall path, but a collection of shortest paths based on each possible next-hop. These paths will not in general be of equal length.

Note that the erasure effect of the $(\triangleleft)$ operator in the definition of S means that these longer alternative paths will not be propagated in the computation. So if a tuple $((5,90),47)$ appears in the output, this is to be interpreted as the length of the path that a packet would follow if it were forwarded from node 5 to its neighbour node 90, which might not be the overall best choice, and subsequent nodes forwarded it along the optimal path.

Example 5.6. In Example 5.5, the preorder semigroup $S \vec{\times} T$ itself has the C property, so we can choose any preorder semigroup U which is monotonic, and be sure that $S \vec{\times} T \vec{\times} U$ will be monotonic.

For example, let U be the 'reliability' algebra $([0,1], \geq, \times)$. This is monotonic because if $p \geq q$ then $rp \geq rq$ for any p, q and r in the interval $[0,1]$. This algebra is not cancellative (because of the presence of 0), but it does not need to be. The product algebra $S \vec{\times} T \vec{\times} U$ is monotonic, and can be used just as $S \vec{\times} T$ was in Example 5.5.

This demonstrates the possibilities of language-based design of routing algebra for incremental development. We can reuse the previously-established result for $S \vec{\times} T$ in assessing the properties of the new algebra, rather than having to prove everything all over again from scratch.

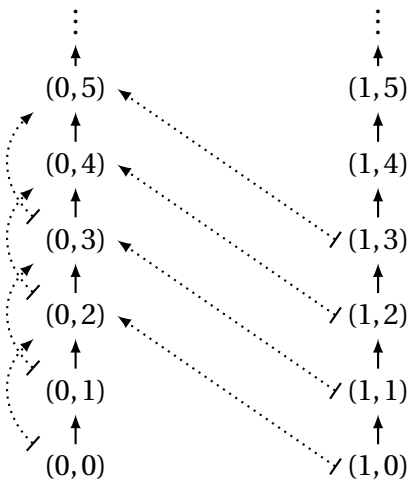


Figure 5.5 The order of Example 5.5, with dotted lines indicating multiplication by the element $(0,2)$.

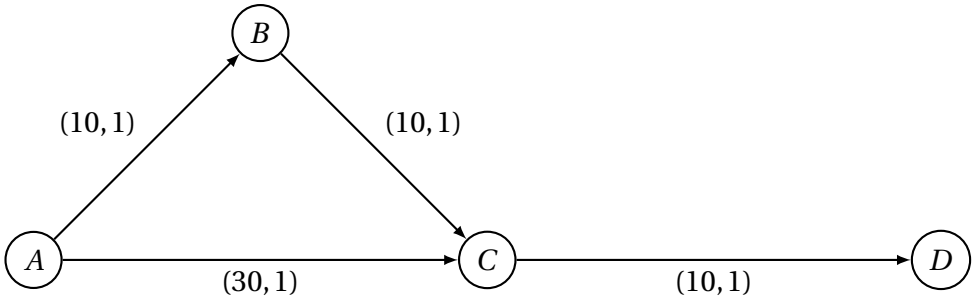


Figure 5.6 Graph for Example 5.7, showing that bandwidth-distance is not monotonic.

Example 5.7. Two of the metrics that are most frequently considered for network path selection are distance (or delay) and bandwidth. It is natural to model these as $S = (\mathbb{N}, \sqcup, +)$ and $T = (\mathbb{N}, \sqcap, \sqcup)$ respectively. The product $S \vec{\times} T$ is monotonic, by Theorem 5.4, because S has M and C, and T has M. But $T \vec{\times} S$ is not monotonic. Figure 5.6 shows a labelled graph which illustrates the problem. The intermediate node C prefers the wider and longer path ABC, but because the CD link is narrow, the paths $ABCD$ and ACD have the same bandwidth—and since ACD is shorter, it is preferred at D.

For the construction of not just binary but n-ary lexicographic products, we will need inference rules for the C and K properties. This will allow deduction of when an n-ary lexicographic product is monotonic, and hence suitable for algorithms that find global optima.

Theorem 5.5. *For each family of structures in Figure 5.4, if S and T are both from that family, then*

$$\begin{aligned} \mathrm{C}(S \mathbin{\vec{\times}} T) &\iff \mathrm{C}(S) \wedge \mathrm{C}(T) \\ \mathrm{K}(S \mathbin{\vec{\times}} T) &\iff \mathrm{K}(S) \wedge \mathrm{K}(T) \end{aligned}$$

where C *and* K *are the appropriate versions of the properties.*

Proof. See Lemma A.9 through Lemma A.12 in Appendix A. □

We are now able to derive an extension of Theorem 5.4 for n-ary lexicographic products.

Theorem 5.6. *Let $S_1, S_2, \ldots, S_k$ be structures from one of the standard families. Then*

$$\begin{aligned} \mathrm{M}(S_1 \mathbin{\vec{\times}} S_2 \mathbin{\vec{\times}} \cdots \mathbin{\vec{\times}} S_k) \iff\ & (\forall i \in \{1,\ldots,k\} : \mathrm{M}(S_i)) \\ & \wedge (\exists i \in \{1,\ldots,k\} : \forall j : j < i \implies \mathrm{C}(S_j) \wedge j > i \implies \mathrm{K}(S_j)) \end{aligned}$$

Proof. We proceed by induction on k. Note that the case $k = 2$ has already been proved as Theorem 5.4. Suppose that the theorem statement holds for some particular $k \geq 2$. Let S stand for the k-ary lexicographic product $S_1 \mathbin{\vec{\times}} S_2 \mathbin{\vec{\times}} \cdots \mathbin{\vec{\times}} S_k$, and let S_{k+1} be any algebra (of the appropriate kind). From Theorem 5.4, we have

$$\mathrm{M}(S \mathbin{\vec{\times}} S_{k+1}) \iff \mathrm{M}(S) \wedge \mathrm{M}(S_{k+1}) \wedge (\mathrm{C}(S) \vee \mathrm{K}(S_{k+1})).$$

Now, by Theorem 5.5 we know that

$$\mathrm{C}(S) \iff \forall i \in \{1,\ldots,k\} : \mathrm{C}(S_i).$$

The theorem statement for $\mathrm{M}(S)$ can then be expanded to yield

$$\begin{aligned} \mathrm{M}(S \mathbin{\vec{\times}} S_{k+1}) \iff\ & (\forall i \in \{1,\ldots,k+1\} : \mathrm{M}(S_i)) \\ & \wedge (\exists i \in \{1,\ldots,k\} : \forall j : j < i \implies \mathrm{C}(S_j) \wedge j > i \implies \mathrm{K}(S_j)) \\ & \wedge (\forall j : (j < k+1 \implies \mathrm{C}(S_j)) \vee (\mathrm{K}(S_{k+1}))) \\ \iff\ & (\forall i \in \{1,\ldots,k+1\} : \mathrm{M}(S_i)) \\ & \wedge (\exists i \in \{1,\ldots,k+1\} : \forall j : j < i \implies \mathrm{C}(S_j) \wedge j > i \implies \mathrm{K}(S_j)). \end{aligned}$$

which is the theorem statement for $k+1$. □

In general, n-ary lexicographic products that are monotonic can be divided into three parts:

1. an initial part which has the C property;
2. a single algebra which need not have the C or K properties; and
3. a final part which has the K property,

where each individual algebra has M. This gives insight into the possible designs of routing algebras based on the lexicographic product. In particular, we only get one chance to include an algebra that has neither the C nor the K property (unless we are willing to give up monotonicity). Another conclusion is that any tiebreakers which follow this component must have K, which is a very restrictive property: there are few examples of algebras that have it and are non-trivial.

Example 5.8. In Examples 5.5 and 5.6, we extended an algebra we had previously designed with a new component. Suppose that we wanted to do something similar with the distance-bandwidth algebra from Example 5.7. Let U stand for the new component to be added.

From Theorem 5.6, we know that $\mathrm{M}(S \vec{\times} T \vec{\times} U)$ requires that all three bisemigroups should be distributive, and that one of the following holds:

1. both S and T have C;
2. S has C and U has K; or
3. both T and U have K.

Because the bandwidth algebra does *not* have C, the only way to achieve distributivity overall is for U to have the K property. That is, the multiplicative component of U has to be a left zero semigroup. This is a very restrictive condition; there are few useful choices of U which could be inserted here. From a point of view of a protocol designer, the theorem has given us an important result: we should avoid putting an arbitrary semigroup in the U position, unless we are willing to forfeit the possibility of finding global optima. It would be extremely undesirable if a practical system was designed using such a metric without knowledge of this consequence. Furthermore, this design guidance was found as a straightforward application of the theorem, and did not require its own proof or any experimentation.

Family	Property	Definition
Bisemigroups	ND	$\forall a, c : a = a \oplus (c \otimes a)$
	I	$\forall a, c : a = a \oplus (c \otimes a) \neq (c \otimes a)$
Order semigroups	ND	$\forall a, c : a \preceq c \otimes a$
	I	$\forall a, c : a \prec c \otimes a$
Semigroup transforms	ND	$\forall a, f : a = a \oplus f(a)$
	I	$\forall a, f : a = a \oplus f(a) \neq f(a)$
Order transforms	ND	$\forall a, f : a \preceq f(a)$
	I	$\forall a, f : a \prec f(a)$

Figure 5.7 Properties relating to local optima.

Another possible design might be to insert a new component at the front, so that we would have $R \vec{\times} S \vec{\times} T$ for some bisemigroup R. Again, the theorem for n-ary lexicographic products tells us that any such R has to have both the M and the C properties (it should be a cancellative semiring). The theorem has provided the precise constraint to be satisfied.

5.2.2 *Local optima*

For finding local optima, the properties of interest are 'increasing' and 'nondecreasing'. As with monotonicity, these look slightly different for each of the four structures, while still expressing the same general idea: that extensions of paths are (strictly) less preferred than the originals. Note that we always have $\text{I}(S) \implies \text{ND}(S)$.

In the case of the lexicographic product, our general theorems are as follows.

Theorem 5.7. *For each family of structures in Figure 5.7, if S and T come from that family, then*

$$\text{ND}(S \vec{\times} T) \iff \text{I}(S) \vee (\text{ND}(S) \wedge \text{ND}(T))$$
$$\text{I}(S \vec{\times} T) \iff \text{I}(S) \vee (\text{ND}(S) \wedge \text{I}(T))$$

where the properties ND *and* I *are as appropriate for the family.*

Proof. See Section A.4 in Appendix A. □

It is then straightforward to prove when an n-ary lexicographic product is increasing.

Corollary 5.8. *Let S_1 through S_n be from one of the four families. Then* $\mathrm{I}(S_1 \vec{\times} \cdots \vec{\times} S_n)$ *if and only if there is some k with $1 \leq k \leq n$ such that* $\mathrm{I}(S_k)$ *and for $j < k$,* $\mathrm{ND}(S_j)$.

So a lexicographic product that is increasing has three parts. First, there are zero or more components that are merely nondecreasing. Then, a component that is increasing; and finally, the remaining components need not have any special properties at all. This means that in our lexicographic products, we can use increasing algebras to 'guard' any lower-priority algebras at all, and still be able to compute local optima using these metrics.

We are primarily interested in increasing algebras that fail to be monotonic, although of course there are those which have both properties. This is because the I property is associated with a different and more subtle kind of optimality than the M property. If an algebra is monotonic, then standard algorithms can be used to derive global optima for the best-path problem. If the algebra happens to be increasing as well, this fact does not affect the presence of that optimum or the difficulty of finding it. Put another way, the algebras that are increasing but not monotonic represent a part of the mathematical territory that is less familiar and less explored; they are interesting because they can be used in standard algorithms *despite* their lack of monotonicity.

The next chapter will present many examples of how the increasing property manifests in lexicographic products, in the context of interdomain routing.

5.3 *Errors and infinities*

There is a problem with the rules presented so far, connected with certain special elements of routing algebras. It is not unusual for an order to have a greatest element, or a semigroup to have a two-sided annihilator. For example, we might want to work with the order $(\mathbb{N}^\infty, \leq)$ which has ∞ as its greatest element. According to the definition of the increasing property, there can be no increasing function on $(\mathbb{N}^\infty, \leq)$, since ∞ cannot be mapped to anything strictly greater than itself.

In the case of the distributive property, the problem is with finite structures. The algebra $(\mathbb{N}^\infty, \sqcup, +)$ used for finding shortest paths will be realized in practice by some finite algebra, such as $(\{0, \ldots, U\}, \sqcup, \oplus)$, where $x \oplus y \stackrel{\text{def}}{=} (x+y) \sqcup U$ for some fixed upper bound U. This $(\oplus)$ is not cancellative, even though the idealized $(+)$ is cancellative. The failure noted in Figure 5.6 applies equally to this example: a lexicographic product using the finite algebra as its first component may fail to support the finding

of global optima, due to a loss of the distributivity property.

The underlying problem is that there is an overloading of the semantics of 'infinite' or 'error' elements. There are at least three separate concepts which are conflated:

1. A greatest element of an order—one which simply happens to be greater than any other, but is otherwise normal and does not represent any kind of failure.

2. An erroneous route, which should never be used for sending data. These may be encoded as maximal elements, because then they will be less preferred than any valid route.

3. A blank: in the operation of an algorithm, a greatest element might be used as the initial shortest path estimate, so that it can be overwritten once better information is found.

If infinite elements are regarded as erroneous, so that they represent the absence of a route (or of a valid route), then there is something wrong with the definition of lexicographic product. All definitions so far have permitted the appearance of pairs where one component is erroneous. This is not necessarily the right choice, if we are seeking to have the algebra incorporate the idea that all errors are mapped to a greatest element which can then be 'ignored'.

Although these elements are referred to as erroneous, this is not intended to suggest that they are the result of misconfiguration, or represent faults in the network. Rather, they are meant to be elements for which there is no associated route. This will typically be because there is a route which exists but is for some reason unacceptable in terms of preferences. It is natural to model such a situation in terms of the preference structure of an algebra, since that is the purpose of using routing algebra in the first place. This model may also be seen as modelling certain 'filtering' operations used in Internet routing, which permit routes to be excluded from the selection process according to the desires of operators, in addition to the normal routing procedure.

Instead, we should define the lexicographic product so that it is impossible to have such 'mixed' pairs, *if* that is the behaviour that is desired. If greatest elements are not intended to represent erroneous behaviour, but are simply elements which happen to be large, then the usual definition is the appropriate one.

Definition 5.3. Let $(S, \preceq_S, F)$ and $(T, \preceq_T, G)$ be order transforms, where S and T have greatest elements ∞_S and ∞_T respectively. Then $S \overrightarrow{\times}_\infty T$ is the following order transform:

- The underlying set is $(S \setminus \{\infty_S\}) \times (T \setminus \{\infty_T\}) \cup \{\infty\}$, where ∞ is a new element.
- The order is lexicographic with greatest element ∞, so that

$$(s_1, t_1) \preceq (s_2, t_2) \iff (s_1 \prec_S s_2) \vee (s_1 \sim_S s_2 \wedge t_1 \preceq_T t_2)$$

  and $(s, t) \prec \infty$.
- The function set consists of pairs (f, g) from $F \times G$, where

$$(f, g)(s, t) = \begin{cases} (f(s), g(t)) & f(s) \neq \infty_S \wedge g(t) \neq \infty_T \\ \infty & f(s) = \infty_S \vee g(t) = \infty_T \end{cases}$$

  and $(f, g)(\infty) = \infty$.

Call this the *absorbing lexicographic product* of S and T.

A similar definition can be made for semigroup transforms, and for the other structures. The important factor here is that an 'error' in either component is raised to the top level. This alternative definition is therefore appropriate for use when both components are capable of experiencing erroneous conditions, encoded by means of the greatest element. It seems that there should be a combinatorial explosion of variations on these products: what if we want to raise errors in the first component, but treat the greatest element of the second component as simply a large value? The problem becomes worse when moving to n-ary products. Furthermore, it seems that property deduction rules will have to be proved for each variation on the lexicographic product. This may lead to a further explosion in the number of properties which need to be tracked.

There is another unusual feature of the absorbing product in relation to semigroups. Recall that in $S \vec{\times} T$, if a and b are incomparable, then $(a, x) \oplus (b, y)$ will be $(a \oplus_S b, \infty_T)$. Is it truly correct to map this to a global ∞? In that case, all attempts to combine incomparable elements are deemed erroneous.

There is evidently considerable difficulty even in defining the correct operator, even before any attempt to prove its properties. We will now present one possible way of dealing with the problem of errors, using the reduction functions of Section 3.2. This approach may scale better than the definition of an entirely new operator for each possible notion of error.

The general principle is to define a reduction which will eliminate erroneous elements of an algebra, by mapping them onto a greatest element. If $(X, \oplus, F)$ is a semigroup transform, with $(\oplus)$ commutative and having identity α, and E is a subset of X, then define a function r_E on X by

$$r_E(x) = \begin{cases} x & x \notin E \\ \alpha & x \in E. \end{cases} \tag{5.14}$$

For this to be a reduction, it is required that E satisfies the property

$$\forall e \in E, x \in X : (x \in E \wedge e \oplus x \in E) \vee (x = e \oplus x). \tag{5.15}$$

It is then possible to define a new structure based on r_E.

Definition 5.4. Let $\mathbf{err}(X, E)$ be the semigroup transform $(X_E, \oplus_E, F_E)$, where

- X_E consists of those elements of X for which $r_E(x) = x$,
- $x \oplus_E y$ is $r_E(x \oplus y)$, and
- F_E consists of functions f_E for each f in F, and $f_E(x) = r_E(f(x))$.

This $(\oplus_E)$ can be verified to be associative, since r_E is a reduction. The other properties of $\mathbf{err}(X, E)$ will depend on the choice of X and E.

The congruence associated with such a reduction is related to the notion of a Rees congruence on a semigroup. A subset E of $(X, \oplus)$ is an *ideal* if

$$\forall x \in X, e \in E : (x \oplus e \in E) \wedge (e \oplus x \in E). \tag{5.16}$$

For a given ideal E, the relation

$$x \sim_E y \overset{\text{def}}{\iff} x = y \vee (x \in E \wedge y \in E) \tag{5.17}$$

is a congruence, called the *Rees congruence* with respect to E (Grillet 1995). In the case of our r_E, the congruence may not be a Rees congruence because E may not satisfy (5.16). This is in line with our general principle of not enforcing conditions which can be inferred: the definition of $\mathbf{err}(X, E)$ makes sense even when E is not an ideal, though it may not have desirable properties.

If X is a lexicographic product $S \mathbin{\vec{\times}} T$ where S has an identity α_S, we can choose

$$E_1 = \{\alpha_S\} \times T. \tag{5.18}$$

This is an appropriate choice since the possible loss of monotonicity comes when there are elements of $S \vec{\times} T$ for which

$$s_1 = s_1 \oplus_S s_2 \neq s_2$$
$$f(s_1) = \alpha_S = f(s_2)$$
$$t_2 = t_1 \oplus_T t_2 \neq t_1$$

as then we have

$$(s_1, t_1) = (s_1, t_1) \oplus (s_2, t_2) \neq (s_2, t_2)$$

but

$$(f(s_1), g(t_1)) \neq (f(s_1), g(t_1)) \oplus (f(s_2), g(t_2)) = (f(s_2), g(t_2)),$$

violating monotonicity due to the absence of cancellativity in S. In this example, we do not attempt to handle errors emanating from T. It can be verified that this E_1 satisifies (5.15), and so $\mathbf{err}(S \vec{\times} T, E_1)$ is a semigroup transform.

Theorem 5.9. *The algebra* $\mathbf{err}(S \vec{\times} T, E_1)$ *is distributive if*

$$(f, g)(s_1, t_1) \oplus (f, g)(s_2, t_2) = (f, g)((s_1, t_1) \oplus (s_2, t_2))$$

for all f *in* F, g *in* G, s_1 *and* s_2 *in* $S \setminus \{\alpha_S\}$, *and* t_1 *and* t_2 *in* T.

Proof. See Section A.5 in Appendix A. □

In a similar way, we can handle errors from the second component of a lexicographic product. There is a particularly important use for this capability in the case of finding a local optimum. For an algebra that is increasing but not necessarily monotonic, convergence is not assured if paths are permitted to have loops. If, however, it is only possible for simple paths to occur, then termination is guaranteed for an increasing algebra.

There are several ways in which this could be achieved. One is for the host algorithm to detect loops and intervene—so an otherwise good path would not be taken if it contained a loop. Another possibility is for this to happen 'automatically' by putting path information into the algebra, and forcing paths with loops to be considered as erroneous. Each of these ideas has merits. The main argument against including path information as a component of a routing algebra is that it is unnecessary and overcomplicated, compared to modifying an algorithm. It is easy to underestimate the difficulty of such changes: consider the 'minimal set' algebras. Avoiding loops for those algebras would mean that the algorithm would need to track

the contents of each set against a separately-maintained set of paths, duplicating the logic. If the path is part of the algebraic structure, then the desired behaviour is still obtained, without any need to rewrite the stock algorithm. In addition, as a matter of design philosophy, decisions about which paths ought to be taken naturally belong as part of the algebra, which exists to encode path selection rules, as opposed to being part of the algorithm, whose purpose is to solve the generic optimization problem.

We will show an example of how to ensure, in a multipath setting, that only simple paths emerge from the algorithm. The standard algebra of paths is to order them by length: we have a preference relation rather than a semilattice. A variation on the **minset** construction will convert such an algebra into one which can be used in the context of matrix operations.

Let P be the algebra of paths $(N^*, \preceq, C)$, where $p \preceq q$ if and only if p is shorter than q, and C consists of functions c_n for all n in N, which concatenate the node n onto the given path. It would also be possible for $(\preceq)$ to be the complete order on P. Let $(S, \leq, F)$ be an order transform, and construct the lexicographic product $S \vec{\times} P$.

Now, let E_2 be the subset of $S \vec{\times} P$ consisting of those pairs which contain a non-simple path:

$$\left\{(s,p) \in S \times P \;\middle|\; p \text{ is not simple}\right\}. \tag{5.19}$$

The **err** construction cannot be used directly, since E_2 does not satisfy the required property (5.15). However, there is a reduction which can be used over subsets of $S \vec{\times} P$. Let r be the function

$$r(A) \stackrel{\text{def}}{=} \min(A \setminus E_2); \tag{5.20}$$

this satisfies the reduction axioms (3.7) and (3.8). Consquently, a semigroup transform can be constructed where

1. the elements are subsets of $S \vec{\times} P$ which are fixed points of r;
2. the operation $(\oplus)$ is given by $A \oplus B \stackrel{\text{def}}{=} r(A \cup B)$; and
3. the functions are pairs (f, c_n) for f in F, where

$$(f, c_n)(A) \stackrel{\text{def}}{=} r(\{(f(s), c_n(p) \mid (s,p) \in A\}).$$

It can be seen that this algebra implements the simple paths criterion in the case of multipath routing: if during the course of computation a non-simple path is computed, it and its associated S-value will be removed from the candidate set.

It has been demonstrated in Chapter 4 that algorithmic convergence is guaranteed for a minimal set algebra if only simple paths are permitted, and the underlying algebra is increasing. So if $S \vec{\times} P$ is increasing, then this new algebra which forces **minset**$(S \vec{\times} P)$ to be restricted to simple paths will satisfy the requirements. In particular, if S is increasing then $S \vec{\times} P$ is increasing, by Theorem 5.7.

These examples serve as further evidence that the split between algebra and algorithm is useful, and that many capabilities can be put in the algebra as opposed to forming part of the algorithm. In the next chapter, this idea will be further tested by the modelling of network partitions: we will see that this too can be handled in the algebra, without modifications to the standard algorithms.

Chapter 6

Modelling network partitions

This chapter will illustrate the use of the metalanguage approach in the modelling of *network partitions*. In particular, the lexicographic product of Chapter 5 will be an important construction, because it can be used to model the way in which local policy can be overridden by constraints at a global level. Property deduction rules will be derived, illustrating the compositional approach to design that is enabled by the metarouting world-view.

The division of networks into areas is an important mechanism for the construction and maintenance of large networks. This especially applies to networks like the Internet, which are composed of heterogeneous systems and managed by separate, possibly competing, entities. There are many possible ways in which this division might work, some of which are represented in current practice.

In particular, this chapter seeks to extend our knowledge of the design space surrounding the Border Gateway Protocol. BGP is the unique example of an interdomain routing protocol, and much is known about its behaviour. However, less is known about whether these features are quirks of BGP's design, or whether they are intrinsic to the interdomain routing problem. In this chapter, we will see that *lack of monotonicity* is inevitable for any protocol that seeks to capture the nature of interdomain relationships, and provide route optimization at the same time; if only one of these tasks is attempted, then the protocol can be monotonic. Other features of BGP, such as the use of centrally-assigned autonomous system numbers or the lack of end-to-end attributes, are not essential. This points the way to a potential future alternative design that differs only slightly in implementation from the current BGP, while permitting far more expressivity to network operators.

In trying to understand possible area-based routing designs, it is natural to seek standard terminology among the various schemes that we know about. This is a necessary precursor to the building of a model of interdomain routing. A model should reflect conditions in the real world, so that conclusions drawn from its analysis can

later be implemented in reality. But it is not appropriate to simulate every nuance of current practice; rather, the goal is to identify just those features of routing and partitioning which are relevant.

6.1 *Network areas*

We will now look in detail at examples of partitioning in the Internet, in order to identify some common ground.

The Internet is the premier example of a 'network of networks', being a composite system not only made up of heterogeneous equipment, but also under the combined administrative control of a multiplicity of economic agents. The network is almost invariably understood to be a decentralized, distributed system; there are comparatively few areas in which participating networks can be said to be hierarchically organized. These include:

1. The social function by which new technology is standardized. The Internet Engineering Task Force and its associated bodies are 'central' in the sense that they possess a unique authority, but this is more a matter of coordination than of command.

2. IANA is responsible for coordinating the assignment of various names and numbers, such as IP address blocks, protocol numbers, and autonomous system numbers.

3. The Domain Name System is administratively and structurally hierarchical.

4. Sometimes, it makes sense to distinguish between 'tiers' of service providers, corresponding to the degree to which they are operationally independent.

The last of these points is the most relevant for a model of routing. It is important to emphasize that the grading of organizations into tiers is not an inherent property of the technology (one does not configure a router with a flag that indicates "I am a Tier 2 provider"), but one which can be observed, imprecisely, by examining that organization's relationships with others. This pattern frequently recurs in our attempts to understand the Internet at the level of independent participating networks.

For example, the coarsest partitioning of the Internet is into *autonomous systems* (ASes). An AS is commonly understood to be a network under a single administrative control, consisting of all and only those systems under that authority.

However, there are several ways in which this definition does not match reality. This stems from the fact that the term 'AS' only has meaning within the context of BGP, where it denotes an area that has its own external routing policy. This does not necessarily overlap with other notions of organizational independence.

- A single organization may use multiple AS numbers, if different parts of the organization require separate routing policies. This may also occur as a result of a corporate merger or acquisition.
- Multiple organizations may share an AS number if they have the same policy. For example, two companies which are multi-homed to the same two providers could use the same AS number.
- Organizations which lack an externally visible routing policy, such as most single-homed networks, need not have an AS number at all.
- AS numbers in the private range (64512–65534) can be reused at will for subareas of an autonomous system, as long as such information does not 'leak' outside the boundary of the AS.
- Assigned or unassigned numbers are occasionally hijacked for nefarious purposes, or used accidentally.
- AS 23456 has a special meaning, for the transition from 16-bit to 32-bit AS numbers.

It is therefore important to be clear about what is meant by the name 'autonomous system', and how this idea relates to other notions of network partition. Because of the BGP precedent, this chapter will use the term to mean a collection of systems having the same external routing policy, without any particular implication for how these systems might be owned or administered. That policy will be derived in part from the nature of the relationships between the owners or controllers of an AS, and the owners or controllers of its neighbours.

Regarding the relationships between ASes, it has similarly been observed that interactions can often be classified into types. In the influential model of Gao and Rexford (2000), these include *customer-provider* and *peer-peer* relationships. The distinction is that customers pay providers for their connectivity, whereas peers exchange routes and traffic on a mutual basis. These categories are not exhaustive, and do not in themselves cover the wide variety of legal or contractual relationships that

might exist with respect to two ASes. For example, peering agreements would generally cover only certain categories of traffic, perhaps providing financial sanctions for violations; this would be reflected in the routing policy, but can still be understood to be in the nature of a peering relationship. Even more so than with the AS definition, these terms are the result of observation of existing systems: they are not built in to the technology, but are characteristic of the way in which the technology is typically used. In practice, it can be difficult to label relationships in this way, particularly for organizations with a global presence; two ASes might have a peering relationship in one part of the world, and a customer-provider relationship elsewhere, for example.

The way in which routes are propagated and selected among customers, providers and peers may or may not be safe. Gao and Rexford (2000) identify constraints on routing policy which enable a stable routing solution to be found; these have been encoded algebraically by Griffin and Sobrinho (2005). In our algebraic formalism, this appears as the order transform $(L, \leq, F)$ where

- L is the set $\{c, r, p, \infty\}$ of identifiers representing customer routes, peer routes, provider routes and forbidden routes respectively.
- $(\leq)$ is given by $c < r < p < \infty$.
- F is the set $\{C, R, P\}$ of functions to be attached to arcs; the values of these functions on L are given as follows:

	c	r	p	∞
C	c	∞	∞	∞
R	r	∞	∞	∞
P	p	p	p	∞

This table expresses, for example, that all routes received from a provider should be considered as provider routes, whether they come from that provider's customers, peers, or its own upstream providers. If a route is passed on by a customer from one of their providers, we should not accept it: $C(p) = \infty$. As written, these rules disallow 'transitive peering', where a system passes on routes learnt from one peer to another of its peers. This can be permitted by instead defining $R(r) = r$. The existence of this algebra shows that path algorithms can operate not only based on a conventional notion of cost, but also on *policy*, which might seem like a more slippery and ill-behaved concept. In fact, certain aspects of the economic and commercial relationships among autonomous systems can be encoded in algebra; they therefore

become usable by generic algorithms, and susceptible to property analysis. For example, $(L, \leq, F)$ is both monotonic and increasing.

In BGP, autonomous system numbers appear in the AS_PATH route attribute. This attribute simultaneously serves many functions:

- It prevents AS-level loops. If the same AS number appears more than once in a path, then the route will not be used.
- It allows some route optimization. A shorter AS_PATH is better than a longer one.
- It permits filtering based on the downstream path.
- It permits monitoring and measurement of the AS path.

In addition, route preferences can be affected by *AS padding* and *AS poisoning*. Padding artificially lowers the preference of a route by repeating an AS number in the path, so that it will appear longer. This works because the uniqueness criterion is only applied in relation to each *new* entry in the path. Poisoning can be used to stop a route from being taken by some distant AS (presumably in favour of another route that would otherwise be less-preferred) by adding that AS's own number to the AS_PATH of the route to be suppressed.

Although this attribute is typically thought of as being a list of numbers, it in fact has a more complex structure. The path is a sequence of structures, each of which is either a list or a set of AS numbers. The admission of sets allows route aggregation while still maintaining some notion of the order of ASes along a path in the absence of aggregation. This feature is intended to improve BGP performance by merging route data from several prefixes together, and therefore reducing the size of routing tables and the number of routing messages. The AS_PATH attribute also incorporates lists and sets of confederations, which are regions below the AS level.

There are other contexts in which AS numbers may appear, mainly to support monitoring and analysis. For example, if routes are aggregated, the responsible AS puts its own number into the AGGREGATOR attribute. This is intended to aid debugging rather than as an input to route selection. BGP community numbers typically include the number of the responsible AS, which allows these values to be defined on a per-AS basis without global coordination. Finally, AS numbers are used between communicating routers in order to determine whether or not they belong to the same autonomous system, and therefore whether the adjacency should be op-

erated as an external or internal connection. See Huston (2006) for a discussion of autonomous system numbers and their uses.

6.2 *The road to BGP*

It is well known that the metric used by BGP is not monotonic, and does not lead to the guaranteed finding of globally optimal routing solutions. In this section we will see how this non-monotonicity arises, and in particular how it seems to be unavoidable for designs that are similar to BGP. These will be those metrics which attempt to do some route optimization, but subject to asymmetric policy constraints after the fashion of the customer-provider-peer system.

The hallmark of BGP's exterior metric is the combination of local policy and optimization. The LOCAL_PREF attribute allows each AS to independently decide on the preference assigned to incoming routes; and it is more significant in route choice than the AS_PATH. The route length optimization carried out through the AS_PATH attribute is always subject to the demands of policy, as expressed in local preference (among other means). The customer-provider-peer model is one attempt at identifying ways in which this attribute can be used so as to avoid protocol divergence.

We can model just these two attributes using a lexicographic product. Let $(L, \leq, F)$ be the customer-provider-peer algebra from the previous section, and let $(P, \leq, G)$ be as follows:

- P is the set of all finite lists of AS numbers, plus the special value $\top$.
- The choice relation $\leq$ is defined so as to prefer shorter paths, regardless of their content, and so that $\top$ is the unique maximal element.
- The operations are the functions c_n, for each AS number n, where $c_n(\ell)$ returns n prepended to ℓ if n is not already present in ℓ, and $\top$ otherwise.

This represents AS paths. Note that we do not attempt to handle path length overflow. There is no explicitly defined limit on the length of a path in the BGP standard, but there are limits on the byte size of an attribute and on the size of a message which imply that path length is bounded in practice. Existing implementations may have even smaller limits on the path lengths they are able to handle.

Now consider the product $L \vec{\times} P$. This will be monotonic if L and P are both monotonic, and if either L is cancellative or P is constant. We observe that L is clearly not cancellative, and P is clearly not constant. Therefore $L \vec{\times} P$ cannot be monotonic.

Furthermore, it is apparent that even simplified variations on $L \vec{\times} P$ will also fail to be monotonic. For example, the carrier set of L could be reduced in size, or P could be changed to a simple hop count rather than a list: but their lexicographic product would remain non-monotonic. This demonstrates that it is not necessary to go very far down the road to BGP before monotonicity is lost.

The presence of these two attributes, in that order, is of critical importance not only for BGP, but for any routing scheme applicable to the present Internet. In particular, the local preference attribute allows the encoding of economic relationships, as with the customer-provider-peer model. If this is combined with some form of route optimization, with the proviso that route quality is always subject to local preference, then we already have $L \vec{\times} P$ even before other possible attributes are considered. The optimization component could be omitted, so that interdomain routes were determined solely by local preference, and the resulting algebra would be monotonic. However, it seems doubtful that a BGP-like protocol would be effective without some form of route optimization.

This is evidence in favour of the proposition that not only BGP, but also designs similar to BGP, do not generally admit globally optimal solutions. Instead, we should only expect to find a local optimum, or Nash equilibrium; this corresponds to the algebra having the 'increasing' property, as is indeed the case with our $L \vec{\times} P$.

6.3 *The scoped product*

The lexicographic product is all very well as a construction if we are seeking only to model the fact that attributes are considered in order of significance. But when it comes to the use of areas, the simple lexicographic product is not enough to encode all of the desired behaviour. This section will explain and explore the *scoped product*, an operator that is similar to the lexicographic product but also allows partitioning of a network into domains.

Abstractly, the main characteristic of interdomain route choice is that there are *exterior attributes* and *interior attributes*, as illustrated in Figure 6.1. There may also be *end-to-end attributes* whose computation ignores area boundaries. The exterior attributes are those relating to the parts of a route that go between different areas; for example, the AS_PATH of BGP records the sequence of areas (ASes) through which a route passes, but it does not include information relating to the interior of an area (aside from the confederation information previously discussed). In contrast, the interior attributes relate to the route as it passes through a single area: these are not

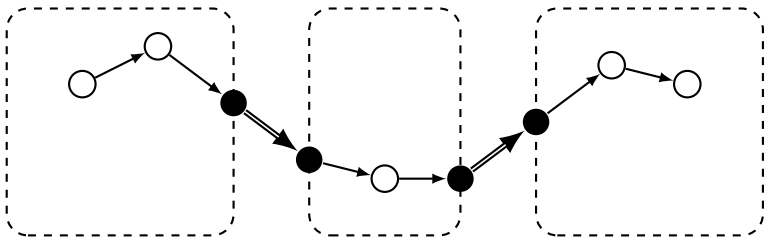


Figure 6.1 Route calculation can be the result of a combination of internal and external metrics. The double arrows are external arcs, which connect different areas. Within each area, shown with a dashed line, are the internal arcs. The black nodes are border routers and the white nodes are interior routers.

carried over between one area and the next. An example in BGP is the 'IGP distance to next hop', which is the distance, according to some interior protocol, to the appropriate border router in the current AS. Note that since the value is reset on passing to a new AS, it is possible for different areas to use different IGPs; in general, the set of interior attributes need not be the same between one area and the next, as long as the appropriate algebraic properties are satisfied at a global level.

A scoped product can be defined over semigroup transforms or over order transforms.

Definition 6.1. Let $(S, \oplus, F)$ and $(T, \boxplus, G)$ be semigroup transforms. Then $S \ominus T$ is a semigroup transform, the *scoped product* of S and T. Its carrier set is $S \times T$ and its operation is the lexicographic product of the $\oplus$ and $\boxplus$ operations. Its set of functions is

$$\{e(f,t) \mid f \in F, t \in T\} \uplus \{i(g) \mid g \in G\}$$

where $e(f,t)(s',t') = (f(s'),t)$ and $i(g)(s',t') = (s',g(t'))$.

Definition 6.2. Similarly, let $(S, \leq_S, F)$ and $(T, \leq_T, G)$ be order transforms. Then the scoped product $S \ominus T$ is defined to be $(S \times T, \leq, H)$, where $(\leq)$ is the lexicographic product of $(\leq_S)$ and $(\leq_T)$, and H is the same set of functions as in the previous definition.

The purpose of these definitions is to amend the notion of lexicographic choice with the idea that different attributes are associated with different network scopes. So the route choice is still lexicographic, but there is a new twist as to how path weights are derived in the first place. In the scoped product setting, there are two kinds of arcs—internal and external—to go along with the two components of the lexicographic product. An internal arc only affects the value of the internal metric.

By contrast, the external arcs have two roles: firstly, they alter the external metric value, and secondly, they provide a new internal value as a basis for subsequent route calculations. This means that the internal metric is indeed isolated to each domain, since passing from one domain to another (via an external arc) causes the value to be replaced.

Example 6.1. As a basic example of the scoped product, let

$$S = T = (\mathbb{N}, \min, \{\lambda y \,.\, x + y \mid x \in \mathbb{N}\}).$$

Then $S \ominus T$ allows us to maintain two different distance measurements: one external, and one internal. Unpacking the scoped product definition, we find that the two kinds of functions which can be attached to arcs operate as follows:

$$\begin{aligned} i(t)(n,m) &= (n, m+t) && \text{(internal arc)} \\ e(s,t)(n,m) &= (n+s, t) && \text{(external arc).} \end{aligned}$$

As desired, the internal distance is reset upon entering a new area, and within an area the external distance is held constant.

Figure 6.2 shows an example network configured with this algebra. The node Z originates the pair $(0,0)$.

Subsequently, two paths will become visible at B: the lower path has weight $(5,6)$ and the upper path has weight $(5,10)$, because

$$\begin{aligned} (5,6) &= (i(4) \circ e(5,2) \circ i(1))\,(0,0) \\ (5,10) &= (i(4) \circ e(5,6) \circ i(1))\,(0,0). \end{aligned}$$

The external weight in each case is 5, because the two external arcs both had this as their external weight. But the internal weights are different, and this happens even though both paths contain an internal arc of weight 4 and another of weight 1. The reason is that different values (2 and 6) were used to initialize the internal T weight on entry into the final region. The path that is chosen is $(5,6)$.

The node A will see three paths. There is a path going through B, whose weight is $i(3)(5,6) = (5,9)$. The weights of the other two are

$$\begin{aligned} (4,2) &= (i(2) \circ e(1,0) \circ i(2) \circ i(18) \circ i(1) \circ e(3,0) \circ i(1))\,(0,0) \\ (4,6) &= (i(6) \circ e(1,0) \circ i(3) \circ i(1) \circ e(3,0) \circ i(1))\,(0,0). \end{aligned}$$

The path $(4,2)$ is chosen; $(5,9)$ is rejected because its external weight of 5 is larger than 4, and $(4,6)$ is rejected because its internal weight of 6 is larger than 2. Note that

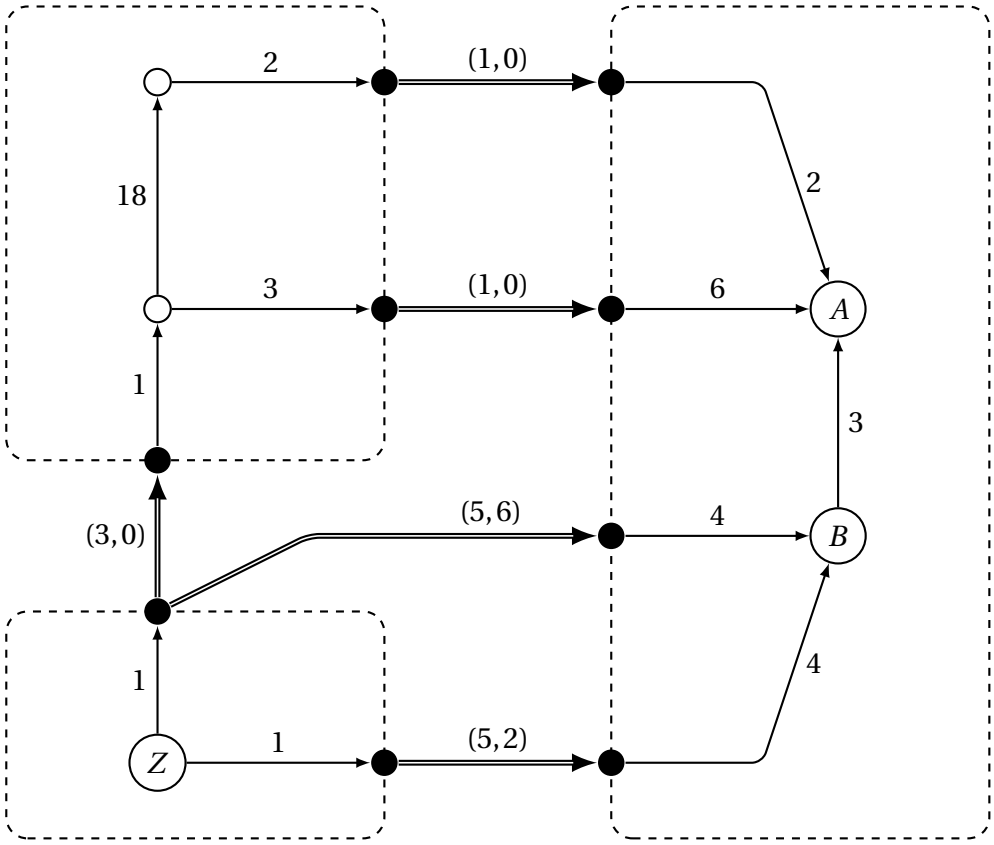


Figure 6.2 A scoped product that combines an external distance with an internal distance. The numbers on the internal arcs are the increments to be added to the internal distance. The pairs (a, b) on the external arcs represent the weight a to be added to the external distance, and the value b with which the internal distance will be initialized on entering the target region.

the internal weight calculation depends only on the final region: it is *not* equal to the total weight of all internal arcs along the path.

For analysis of this product from the point of view of its algebraic properties, and to permit exploration of alternative, related definitions, it is useful to have a definition that is in terms of more primitive operations.

Definition 6.3. Let **copy** be the unary operator given by

$$\mathbf{copy}(S, \oplus, F) \stackrel{\text{def}}{=} (S, \oplus, \{\text{id}\}) \tag{6.1}$$

$$\mathbf{copy}(S, \leq, F) \stackrel{\text{def}}{=} (S, \leq, \{\text{id}\}) \tag{6.2}$$

where id is the identity function on S.

Definition 6.4. Let **replace** be the unary operator given by

$$\mathbf{replace}(S, \oplus, F) \stackrel{\text{def}}{=} (S, \oplus, K(S)) \tag{6.3}$$

$$\mathbf{replace}(S, \leq, F) \stackrel{\text{def}}{=} (S, \leq, K(S)) \tag{6.4}$$

where $K(S)$ is the set $\{\kappa_s \mid s \in S\}$.

Definition 6.5. Let $\uplus$ be the binary operator given by

$$(S, \oplus, F) \uplus (T, \boxplus, G) = \begin{cases} (S, \oplus, F \uplus G) & \text{if } (S, \oplus) = (T, \boxplus) \\ \text{undefined} & \text{otherwise} \end{cases} \tag{6.5}$$

$$(S, \leq_S, F) \uplus (T, \leq_T, G) = \begin{cases} (S, \leq_S, F \uplus G) & \text{if } (S, \leq_S) = (T, \leq_T) \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{6.6}$$

We can now define

$$S \ominus T \stackrel{\text{def}}{=} (S \vec{\times} \mathbf{replace}(T)) \uplus (\mathbf{copy}(S) \vec{\times} T) \tag{6.7}$$

where S and T are either both semigroup transforms, or both order transforms.

It is now possible to provide a straightforward proof of the property deduction rules for the scoped product, since the rules for the lexicographic product are known, and those for **copy**, **replace** and $(\uplus)$ are simple to work out. For the $(\uplus)$ operator, when $S \uplus T$ is defined, we have

$$\text{M}(S \uplus T) \iff \text{M}(S) \wedge \text{M}(T) \tag{6.8}$$

$$\text{C}(S \uplus T) \iff \text{C}(S) \wedge \text{C}(T) \tag{6.9}$$

$$\text{K}(S \uplus T) \iff \text{K}(S) \wedge \text{K}(T). \tag{6.10}$$

The situation for **copy** and **replace** is even simpler. From the definitions of M and C, it is clear that $\mathbf{copy}(S)$ has both of these properties no matter which algebra is chosen for S. The K property is more difficult to satisfy. In the case of order transforms we have

$$\text{K}(\mathbf{copy}(S)) \iff \forall x, y \in S : x \sim y \tag{6.11}$$

whereas for semigroup transforms the rule is

$$\text{K}(\mathbf{copy}(S)) \iff \forall x, y \in S : x = y. \tag{6.12}$$

In the same way, **replace**(S) always has the M and K properties, but there are stricter conditions in the case of C. For order transforms,

$$\mathrm{C}(\mathbf{replace}(S)) \iff \forall x, y \in S : x \sim y \vee x \# y \tag{6.13}$$

and for semigroup transforms,

$$\mathrm{C}(\mathbf{replace}(S)) \iff \forall x, y \in S : x = y. \tag{6.14}$$

All of these rules follow directly from the relevant definitions.

It is now possible, given the definition of the scoped product in terms of other operators for which the deduction rules are known, to find the counterpart of Theorem 5.4: a rule describing precisely when a scoped product has the M property.

Theorem 6.1. *If S and T are both order transforms or both semigroup transforms, then*

$$\mathrm{M}(S \ominus T) \iff \mathrm{M}(S) \wedge \mathrm{M}(T).$$

Proof.

$$\begin{aligned}
&\mathrm{M}(S \ominus T) \\
\iff &\mathrm{M}(S \mathbin{\vec{\times}} \mathbf{replace}(T)) \wedge \mathrm{M}(\mathbf{copy}(S) \mathbin{\vec{\times}} T) \\
\iff &\mathrm{M}(S) \wedge \mathrm{M}(\mathbf{replace}(T)) \wedge (\mathrm{C}(S) \vee \mathrm{K}(\mathbf{replace}(T))) \\
&\quad \wedge \mathrm{M}(\mathbf{copy}(S)) \wedge \mathrm{M}(T) \wedge (\mathrm{C}(\mathbf{copy}(S)) \vee \mathrm{K}(T)) \\
\iff &\mathrm{M}(S) \wedge \mathrm{M}(T).
\end{aligned}$$

□

Compared to the rule for lexicographic product, it seems that scoped products are more forgiving. All that is needed for monotonicity is that the two inputs be monotonic; further properties, such as being cancellative or condensed, are not required.

A similar process can be used to find a deduction rule for the 'increasing' property. We first need to identify the conditions for I and ND in the context of $(\uplus)$, **copy** and **replace**. As before, these follow immediately from the definitions of the operators and properties. We have

$$\mathrm{I}(S \uplus T) \iff \mathrm{I}(S) \wedge \mathrm{I}(T) \tag{6.15}$$

for $(\uplus)$, for both semigroup transforms and order transforms. But I does *not* hold for either **copy**(S) or **replace**(S), since in both cases we would be able to deduce

$\forall s \in S : s \neq s$. For the ND property, we have ND(**copy**(S)) for all S, but ND(**replace**(S)) if and only if

$$\forall x, y \in S : x \sim y$$

in the order transform case, and

$$\forall x, y \in S : x = y$$

in the semigroup transform case.

The deduction rule for the increasing rule in scoped products is now easy to prove.

Theorem 6.2. *If S and T are both order transforms or both semigroup transforms, then*

$$\mathrm{I}(S \ominus T) \iff \mathrm{I}(S) \wedge \mathrm{I}(T).$$

Proof.

$$\begin{aligned} & \mathrm{I}(S \ominus T) \\ \iff & \mathrm{I}(S \vec{\times} \mathbf{replace}(T)) \wedge \mathrm{I}(\mathbf{copy}(S) \vec{\times} T) \\ \iff & \mathrm{I}(S) \wedge \mathrm{I}(T). \end{aligned}$$

□

This is *less* forgiving than the lexicographic product, because I is now required for both components. This is because new arcs are either interior or exterior arcs, so for a route to be increasing, both kinds of arcs need to increase the cost of the route; in other words, the interior and exterior algebras must both be strictly increasing.

Example 6.2. Recall Example 5.7, the lexicographic product of distance (S) and bandwidth (T). It was demonstrated that the product $S \vec{\times} T$ is monotonic, but $T \vec{\times} S$ is not. What about $S \ominus T$ and $T \ominus S$: are these monotonic, and what does the algebra look like in each case?

Unlike as with the lexicographic product, *both* of these two algebras are monotonic. This means that in both cases, global optima exist and can be found by standard algorithms. The verification is straightforward. Note that in order to use the **copy** and **replace** operators, we use semigroup transforms rather than bisemigroups in the construction of the product. We have

$$S = (\mathbb{N}, \sqcup, \{\lambda y \,.\, x + y \mid x \in \mathbb{N}\})$$
$$T = (\mathbb{N}, \sqcap, \{\lambda y \,.\, x \sqcup y \mid x \in \mathbb{N}\})$$

and it is obvious that monotonicity holds for both, since

$$\begin{aligned}(\lambda y\,.\,x+y)(a \sqcup b) &= x + (a \sqcup b)\\ &= (x+a) \sqcup (x+b)\\ &= (\lambda y\,.\,x+y)(a) \sqcup (\lambda y\,.\,x+y)(b)\end{aligned}$$

and

$$\begin{aligned}(\lambda y\,.\,x \sqcup y)(a \sqcap b) &= x \sqcup (a \sqcap b)\\ &= (x \sqcup a) \sqcap (x \sqcup b)\\ &= (\lambda y\,.\,x \sqcup y)(a) \sqcap (\lambda y\,.\,x \sqcup y)(b).\end{aligned}$$

Therefore, by Theorem 6.1, the products $S \ominus T$ and $T \ominus S$ are monotonic.

The semigroup transform $S \ominus T$ can be described explicitly as follows.

- The underlying set is $\mathbb{N} \times \mathbb{N}$, so each element is a pair (d, b) of a distance d and a bandwidth b.
- The operation $(\oplus)$ implements lexicographic choice:

$$(d_1, b_1) \oplus (d_2, b_2) = \begin{cases} (d_1, b_1 \sqcap b_2) & \text{if } d_1 = d_2 \\ (d_1, b_1) & \text{if } d_1 < d_2 \\ (d_2, b_2) & \text{if } d_2 < d_1. \end{cases}$$

  So to choose among a set of paths, we first pick the paths whose lengths are smallest, and from these pick those whose bandwidths are greatest.

- The set of functions is

$$\{e(s,t) \mid s \in S, t \in T\} \uplus \{i(t) \mid t \in B\}$$

  where $e(s,t)(d,b) = (s+d, t)$ and $i(t)(d,b) = (d, b \sqcup t)$. The 'external' functions act by adding on a distance value to the previous distance, and replacing the previous bandwidth value with an entirely new one. The 'internal' functions maintain the previous distance value, and add a new potential bottleneck to the previous bandwidth value (so it may be that the bandwidth is unchanged if the new bandwidth t is larger than the previous bandwidth b).

We can see that this algebra represents an area-based routing scheme wherein the inter-area path is chosen on the basis of its length, and the intra-area paths on the

basis of bandwidth. Note that internal paths make no contribution to the length, and bandwidth information does not leak from one area to the next.

Monotonicity can be verified directly. For external arcs we have

$$\begin{aligned} e(s,t)\,((d_1,b_1)\oplus(d_2,b_2)) &= (s+(d_1\sqcup d_2),t)\\ e(s,t)(d_1,b_1)\oplus e(s,t)(d_2,b_2) &= (s+d_1,t)\oplus(s+d_2,t)\\ &= ((s+d_1)\sqcup(s+d_2),t) \end{aligned}$$

and in the case of internal arcs,

$$i(t)\,((d_1,b_1)\oplus(d_2,b_2)) = \begin{cases} (d_1,(b_1\sqcap b_2)\sqcup t) & \text{if } d_1=d_2\\ (d_1,b_1\sqcup t) & \text{if } d_1<d_2\\ (d_2,b_2\sqcup t) & \text{if } d_2<d_1 \end{cases}$$

$$\begin{aligned} i(t)(d_1,b_1)\oplus i(t)(d_2,b_2) &= (d_1,b_1\sqcup t)\oplus(d_2,b_2\sqcup t)\\ &= \begin{cases} (d_1,(b_1\sqcup t)\sqcap(b_2\sqcup t)) & \text{if } d_1=d_2\\ (d_1,b_1\sqcup t) & \text{if } d_1<d_2\\ (d_2,b_2\sqcup t) & \text{if } d_2<d_1. \end{cases} \end{aligned}$$

Therefore $\mathrm{M}(S\ominus T)$ is assured.

Like the $(\vec{\times})$ operator, the $(\ominus)$ operator is associative. This fact allows the implementation of areas at more than one level of nesting in a straightforward way. In the current Internet, there is some capability for supporting nested areas. BGP allows subregions of an internal-BGP area to be defined, known as *confederations*, which operate to some extent as autonomous systems within the larger autonomous system. Although in principle this can be extended to arbitrary depth, current technology only allows such configurations in a restricted context. For example, the simultaneous use of confederations and *route reflectors* has no defined semantics, and is therefore only possible if users commit to a particular vendor whose equipment offers the behaviour they desire.

Example 6.3. Consider a triple scoped product $S\ominus T\ominus U$. The associativity of $(\ominus)$ means that there is no ambiguity in writing a triple scoped product in this way.

The choice is lexicographic, as before, but there are now *three* different kinds of function that may be attached to arcs. These correspond to the three levels in the hierarchy. The labels on arcs follow three different patterns depending on their position in the network topology.

The different kinds of arcs are:

1. arcs (f, t', u') at the S level, between one T area and another;
2. arcs (g, u') within a T area, between one U area and another;
3. arcs h within a U area.

Here, f, g and h are functions on S, T and U respectively, and t' and u' are elements of the sets T and U. These functions operate on triples (s, t, u) as follows:

$$(f, t', u')(s, t, u) = (f(s), t', u') \tag{6.16}$$

$$(g, u')(s, t, u) = (s, g(t), u') \tag{6.17}$$

$$h(s, t, u) = (s, t, h(u)). \tag{6.18}$$

The analogy with the binary scoped product is clear. At the innermost level, functions like h simply alter the U value, leaving the others alone. At the intermediate level, the T value is altered, the S value is maintained and the U value is replaced, because these arcs correspond to passage from one U area to another, but remain within a single T region. At the outermost level, the S value is altered and both the T and the U values are replaced, because these arcs represent passage to a new T area, and therefore to a new U area as well.

The associativity of $(\ominus)$ means that these relationships can be understood in several ways. The bracketing $S \ominus (T \ominus U)$ corresponds to the perception that $(T \ominus U)$ is a single algebra, which happens to be implemented as a scoped product. Equation 6.16 shows that the outer S value is altered while the inner $(T \ominus U)$ value is replaced; in both Equation 6.17 and 6.18 the S value is maintained while the $(T \ominus U)$ value is altered.

Alternatively, the bracketing $(S \ominus T) \ominus U$ groups the outer two levels together as a single algebra. In Equation 6.16 and 6.17 this outer value is altered while the inner U value is replaced, whereas in Equation 6.18 the outer $(S \ominus T)$ value is preserved while the U value is altered.

Figure 6.3 shows how the three different kinds of function are attached to arcs.

So far, our use of areas has involved the segregation of attributes into categories of internal and external. This contrasts with the algebras seen previously, for example in Chapter 5, in which the attributes were *end-to-end*; that is, each arc along the path contributed to the value. In area-based routing schemes, it is arguable that some kind of end-to-end measurement could be helpful in determining

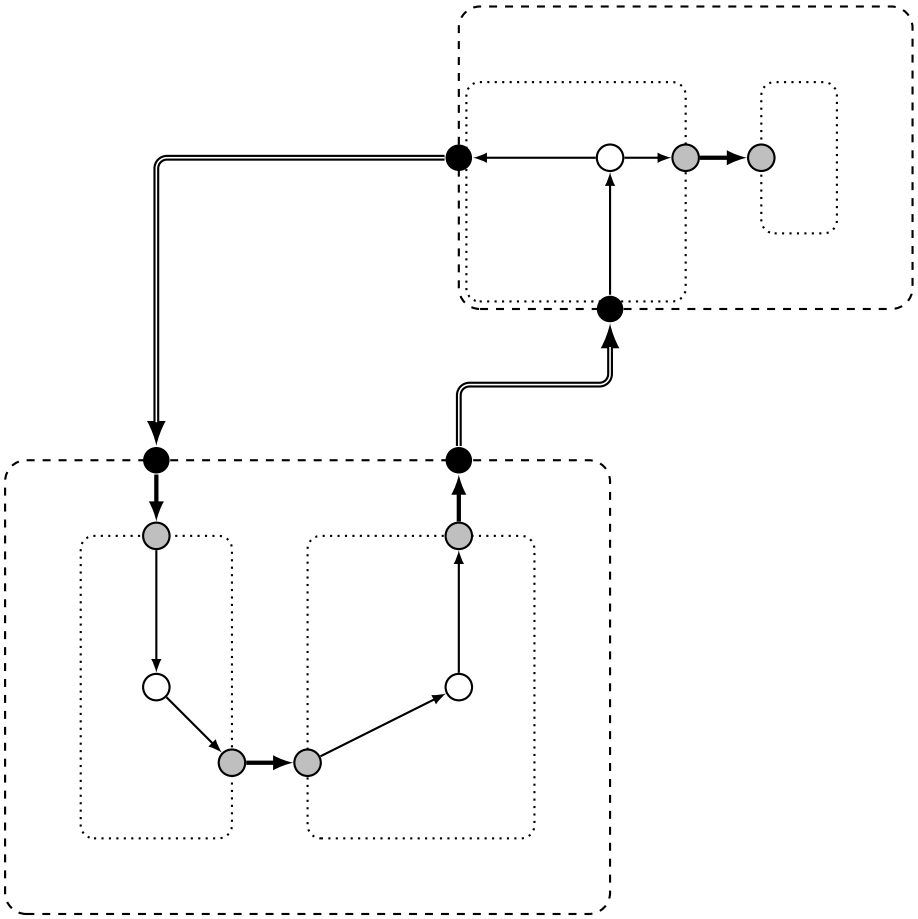


Figure 6.3 Nested areas with a triple scoped product. Double arrows are S arcs, thick arrows are T arcs, and thin arrows are U arcs. The dashed lines indicate the boundaries of a $(T \ominus U)$ region, and the dotted lines show each U region within a T region. The nodes are shaded according to whether they have arcs at the S level (black), at the T level (grey) or at the U level (white). This extends the previous division between internal nodes in white and external or border nodes in black.

the best routes. Several design choices are possible, and as the following examples demonstrate, each of these has implications for the correctness conditions of the whole algebra. Broadly speaking, if we start with $S \ominus T$ so that there is already a lexicographic choice, then a new end-to-end attribute U could be inserted in three positions: either before S, between S and T, or after T.

We already have notation for two of these cases: $U \vec{\times} (S \ominus T)$ and $(S \ominus T) \vec{\times} U$. Let us see what the property requirements are for these products.

Firstly, for $U \vec{\times} (S \ominus T)$ to be monotonic we need U and $S \ominus T$ to be monotonic, and either C(U) or K($S \ominus T$). But we can only have K($S \ominus T$) if **copy**(S) has the K property. From 6.11 and 6.12 it is clear that S has to be trivial in order to have K(**copy**(S)): if

it is a semigroup transform then S can only have one element, and if it is an order transform then it can consist of only one equivalence class. In both cases, there is no possibility of S influencing route choice. We can therefore discard $\mathrm{K}(S \ominus T)$ as a design goal: the only interesting way to achieve monotonicity for $U \vec{\times} (S \ominus T)$ is to have $\mathrm{C}(U)$. For this case, we need U to have both the M and C properties, and at the same time each of S and T must have the M property. See Example 6.4 for a discussion of an algebra following this pattern.

Alternatively, consider the product $(S \ominus T) \vec{\times} U$. If we want $S \ominus T$ to have the C property, then **replace**(T) must also have C. But this is only the case if T is trivial, as in the previous argument. So as before, we only have one interesting case: S, T and U are all monotonic, and U has the K property. Note that this is a significant restriction on any algebra which occurs in the final end-to-end position. While it might be tempting to add some kind of end-to-end distance as a final tiebreaker, the property analysis shows that this will lead to a loss of monotonicity, and hence to an inability of standard algorithms to find global optima.

For the increasing property, we have

$$\mathrm{I}(U \vec{\times} (S \ominus T)) \iff \mathrm{I}(U) \vee (\mathrm{ND}(U) \wedge \mathrm{I}(S) \wedge \mathrm{I}(T)).$$

An important consequence of this rule is that if U is increasing, then $S \ominus T$ need not be increasing nor even nondecreasing. It is therefore possible to use algebras employing area-based metrics in comparative freedom, if they are preceded in a lexicographic order by an increasing U. This fact is already implied by Theorem 5.7, but it is worth emphasizing in order to show the variety of algebraic combinations that are possible. In contrast, the only way for $(S \ominus T) \vec{\times} U$ to be increasing is when both S and T are increasing. This also corresponds to a case in Theorem 5.7.

Example 6.4. As shown above, for $U \vec{\times} (S \ominus T)$ to be monotonic we need each of S, T and U to be monotonic, and for U to have the C property. So let these algebras be as follows:

1. U is $(\mathbb{N}, \sqcup, \{\lambda y \,.\, x + y \mid x \in \mathbb{N}\})$.
2. S is $(\wp R, \cap, \{\lambda A \,.\, A \cup \{n\} \mid n \in N\})$, where R is a set of 'region identifiers'.
3. T is $([0,1], \sqcap \{\lambda p \,.\, pq \mid q \in [0,\ 1]\})$.

Then $\mathrm{M}(U \vec{\times} (S \ominus T))$. This algebra expresses an unusual model of route choice, but one which nonetheless admits global optima.

According to this scheme, a route p is better than another route q if

1. the end-to-end distance of p is less than that of q; or
2. the distances are the same, but p passes through a subset of the regions of q; or
3. they have the same length and pass through the same set of regions, but within the final region p is more reliable than q.

If p and q have the same end-to-end distance but their region sets are incomparable, then no decision can be made about which is better.

The design of this algebra exposes an engineering issue for end-to-end attributes. If such an attribute is to occur first in the order, then the region structure may be of no use at all in determining the best route. It may be that with this algebra, on a particular graph, the U distances alone are sufficient for determination of the best route. Such a design would be inappropriate for the Internet, since there are no circumstances in which it makes sense to violate each autonomous system's local policy on the basis of end-to-end considerations.

In other circumstances a similar design could be more useful. Note that the U algebra allows arc weights to be zero. This fact allows the treatment of U as an override for the $(S \ominus T)$ path selection process. Suppose that in some graph labelled with this algebra, all of the U weights are zero, and that the path selection algorithm, working solely on the basis of $(S \ominus T)$ produces results that for some reason are not those desired by the operator. Those paths can have their preference altered, by changing their U weights away from zero. This will permit other paths to be chosen in their place.

Figure 6.4 shows an example of this kind of manipulation. All U weights are assumed to be zero. The numbers on the internal arcs are the T-weights (probabilities); the pairs on the external arcs are the region identifier to be added, together with the new probability to be used as a basis for calculation in the destination region. The path to A chosen by B will be the direct path from region α to region γ, with weight $(0, \{\alpha\}, 0.99)$. This is preferred to the longer path through β, with weight $(0, \{\alpha, \beta\}, 0.99)$, because it passes through fewer regions. The low reliability of 0.15 on an arc of the shorter path did not affect the final reliability value for the overall path, due to the erasure effect of passing from one region to another. So this path might be of much lower quality than the weight calculation suggests, and the operator of region γ might want the longer path to be taken instead. This can be done by

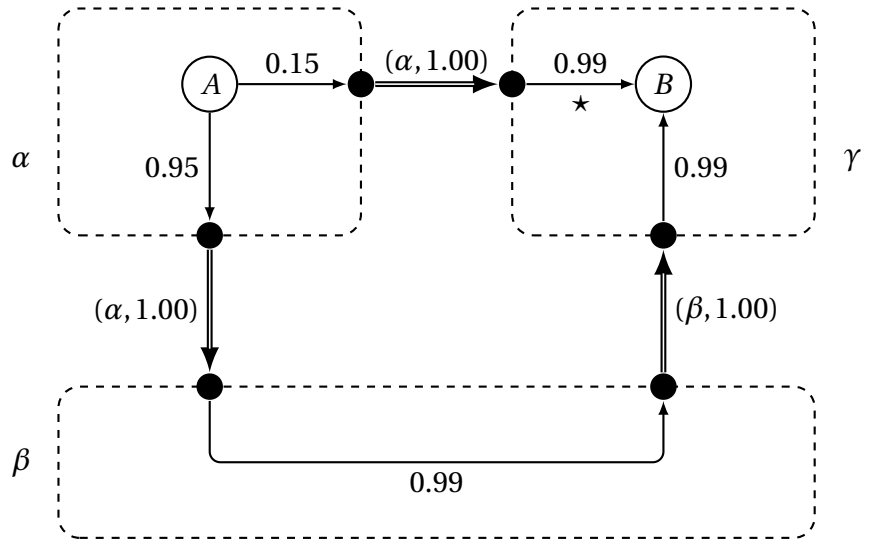


Figure 6.4 Overriding the area-based route choice with an end-to-end attribute.

altering the U weight on the arc marked with $\star$, say to a value of 10. This will result in the longer path being preferred at B, because its U weight is still zero. Note that the making of this change did not require any alteration in the α region, and that the existence of global optima is unaffected by the ability to change U weights in this way.

Of course, there is no guarantee that every possible alternative route choice will be able to be reached by suitable U manipulation. The algebra nevertheless demonstrates that some traffic engineering capabilities can be understood by incorporating suitable additional components (U) into a pre-existing algebra ($S \ominus T$), and that this does not impair our ability to determine correctness properties.

So far, we have seen several ways in which end-to-end attributes can be incorporated into an area-based routing scheme, while still being assured that a globally optimal path assignment can be found.

- An attribute that is used as a final tiebreaker, after the external and internal metrics have failed to produce a definitive route selection, can only be an algebra with the K property. In the case of a semigroup transform, this means that every function must be a constant function κ_x for some x. Tiebreaking can therefore only be performed on the basis of the final arc in the path: no other data can contribute to this choice. For order transforms, the situation is similar: the image of every function must be a set of elements that are all equivalent to one another, so again it is only the final arc which determines the

ultimate preference of the path. (If both the external and internal attributes are increasing, then any choice of attribute for the final tiebreaker is sufficient for local optima to be found.)

- An attribute in the initial position, considered before any external or internal attributes, must have the C property. This is the case for a wide variety of algebras, as demonstrated elsewhere in this thesis. This kind of design is alien to the current model of interdomain routing, but may have applications in other areas.
- Other schemes are possible only if either the internal or external attributes have no effect on route selection. These cases are degenerate, since they reduce to simple lexicographic products for which the relevant deduction rules are already known.

Another possibility is for end-to-end attributes to be considered after the external attributes, but before the internal attributes. This may be a closer fit for future changes to the Internet's interdomain routing scheme, since end-to-end data cannot override autonomous system policy (part of the external attributes) but still has the potential to have non-trivial effects on route choice (as is not the case for end-to-end attributes in the final tiebreaker position). The next two examples demonstrate this kind of scheme.

An algebraic description of an area-based system with intermediate end-to-end attributes is

$$\left(S \mathbin{\vec{\times}} T \mathbin{\vec{\times}} \mathbf{replace}(U)\right) \uplus \left(\mathbf{copy}(S) \mathbin{\vec{\times}} T \mathbin{\vec{\times}} U\right)$$

where S is the external (inter-area) algebra, U is the internal (intra-area) algebra, and T is the end-to-end algebra. A similar analysis to that of Theorem 6.1 shows that this product is monotonic if and only if all three of S, T and U are monotonic, and in addition one of the following conditions holds:

1. $\mathrm{C}(S) \wedge \mathrm{C}(T)$
2. $\mathrm{C}(S) \wedge \mathrm{K}(U)$
3. $\mathrm{K}(T) \wedge \mathrm{C}(T)$
4. $\mathrm{K}(T) \wedge \mathrm{K}(U)$.

The third of these is equivalent to T being trivial, and therefore can be ignored. We are left with precisely the same requirements as for the triple lexicographic product $S \mathbin{\vec{\times}} T \mathbin{\vec{\times}} U$ (see Theorem 5.6 and Example 5.8).

Example 6.5. The most interesting case is when $\mathrm{C}(S)$ and $\mathrm{C}(T)$, because the K property is so restrictive. A concrete example might be the following, for order transforms.

- Let S consist of lists of AS numbers (say), ordered by length. The set of functions F consists of functions of the form $\lambda \ell \,.\, n : \ell$, where n is an AS number and $(:)$ adds a new element to the beginning of a list.
- Let T be the end-to-end delay algebra $(\mathbb{N}, \leq, \{\lambda x \,.\, x + y \mid y \in \mathbb{N}\})$.
- The U algebra can now be any monotonic algebra at all. Let it be

  $$(\mathbb{R}^{\infty}_{\geq 0}, \geq, \{\lambda x \,.\, x \sqcup y \mid y \in \mathbb{R}_{\geq 0}\}) \times ([0,1], \geq, \{\lambda p \,.\, pq \mid q \in [0,1]\})$$

  the direct product of bandwidth and reliability.

This can be verified to be monotonic and therefore a globally optimal path assignment can always be found.

Suppose that we only want the algebra

$$\left(S \mathbin{\vec{\times}} T \mathbin{\vec{\times}} \mathbf{replace}(U)\right) \uplus \left(\mathbf{copy}(S) \mathbin{\vec{\times}} T \mathbin{\vec{\times}} U\right)$$

to be increasing. As before, the property rules for $(\uplus)$, $(\vec{\times})$, **copy** and **replace** can be expanded to yield only two cases in which the component algebras are nontrivial. These are:

1. $\mathrm{I}(S) \wedge \mathrm{ND}(T) \wedge \mathrm{I}(U)$
2. $\mathrm{ND}(S) \wedge \mathrm{I}(T)$.

Compare this to the rule for scoped product in Theorem 6.2. There, both the interior and exterior attributes were required to be increasing in order for the entire product to be increasing. We now see two ways of incorporating intermediate end-to-end attributes into such a scheme. Firstly, any nondecreasing algebra can be inserted between the exterior and interior parts of the algebra, and no other changes need to be made. Secondly, if the inserted algebra is not just nondecreasing but increasing, then it is sufficient for S to be merely nondecreasing, and for U to be any algebra at all. Local optima can still be found in this case, even though neither the exterior nor the interior algebra is increasing.

There is a common pattern between this rule and that of Example 6.5. In both cases, the insertion of an intermediate end-to-end algebra into a scoped product

resulted in a change in the property deduction rules. In both cases, the new rule reflects the rule for lexicographic products: recall that $S \vec{\times} T \vec{\times} U$ is increasing if S is nondecreasing and T is increasing. In the previous example, this was problematic since the lexicographic product monotonicity rule is more restrictive than the scoped product monotonicity rule. But in this example, the rule for the lexicographic product is less restrictive, and the new case that is revealed allows the design of a new family of algebras with components whose properties are less strong than in the standard scoped product case.

Example 6.6. For concreteness, suppose that T is the increasing algebra

$$(\mathbb{N}, \leq, \{\lambda x \,.\, x + y \mid y \in \mathbb{N} \setminus \{0\}\}).$$

Then the exterior algebra S could be anything that is nondecreasing, such as

$$(\wp R, \preceq_S, \{\lambda A \,.\, A \cup \{r\} \mid r \in R\})$$

where R is a set of region identifiers and $A \preceq_S B$ if and only if $|A| \leq |B|$. The interior algebra U could be any algebra at all, such as

$$(\wp\mathbb{N}, \preceq_U, \{\lambda A \,.\, f(A) \mid f : \mathbb{N} \longrightarrow \mathbb{N}\})$$

where $A \preceq_U B$ if and only if $\min(A) \leq \min(B)$. This U algebra would ordinarily not be usable for finding optima at all, since it is neither monotonic nor increasing.

- If f is $\lambda x \,.\, (x+3) \bmod 5$, A is $\{1\}$ and B is $\{2,3\}$ then we have $A \preceq_U B$ since $\min(A) < \min(B)$. But $f(A)$ is $\{4\}$ and $f(B)$ is $\{0,1\}$, so $f(B) \prec_U f(A)$. Consequently U is not monotonic.
- If f is $\lambda x \,.\, (10 - x) \sqcap 0$ and A is $\{7\}$ then $f(A) \prec_U A$, so U is not increasing (nor even nondecreasing).

The full product will nevertheless be increasing, allowing the computation of local optima by standard methods. This example demonstrates that even very unusual algebras can be used in route finding, so long as they appear in a suitable context. Here, the U algebra which allows any function to $\mathbb{N}$ to be an arc label is used as a region-internal metric, and is used subject to the shortest-distance algebra T. At the same time, use of S as an external algebra would not normally be acceptable since it is not increasing; but the presence of T allows that constraint to be relaxed.

6.4 *The road away from BGP*

We have now seen several designs that explore different aspects of how an interdomain routing protocol might arrive at its choice of best routes. In Section 6.2 it was argued that in today's Internet, and any system resembling it, we should not expect to be able to find globally optimal path assignments. Instead, we should concentrate our design efforts towards finding useful routing schemes that admit the possibility of finding local optima, or Nash equilibria. In terms of metarouting, this means the search for algebras having the increasing property, and whose design reflects the engineering desires of network operators. This second design goal is, as has been noted in Chapter 2, inherently not well-defined and probably can never be formally defined. Nonetheless, in this section several suggestions will be put forward about the design of any future routing protocol that is similar in spirit to BGP, but offers more capabilities and is demonstrably correct.

The following changes may be considered 'safe' in that they are slightly different from what is done today, but have the correct algebraic properties for the entire routing scheme to possess local optima. They can therefore be seen as the low-hanging fruit for adaptations of BGP.

The autonomous system path could be enriched by accompanying each AS number with a numeric weight. The cost of the path would then be the sum of these weights. This would subsume the current use of padding, while allowing more nuance in the path selection process. Poisoning could be implemented by adding an AS number with zero weight; this would prevent the route from being taken at the remote AS, but would not otherwise affect the weight. The AS numbers themselves remain present for transparency and loop-avoidance purposes, but otherwise are not needed. This change adds more capability to the AS_PATH attribute, is neutral with respect to convergence properties, and tidies up some uses of the attribute (like padding and poisoning) which make its semantics unclear. It comes at a cost of increased processing time and greater storage requirements, though this may be mitigated by choosing an alternative representation or implementing loop-avoidance differently. For example, it would be possible to store only the total weight in addition to the set of ASes on a path, rather than storing a list of pairs of an AS number and a weight.

Confederations and sub-ASes should be arbitrarily nestable. One precondition for this would be to disentangle the confederation path from the inter-AS path; currently, these are both present within the AS_PATH attribute. (This is an implement-

ation accident resulting from the desire to reuse AS_PATH behaviour for a then-new BGP feature, confederations.) Confederation information, which is conceptually separate and part of the internal detail of how a particular AS operates, should not be externally visible. This change would also allow a variety of algebras to be involved with the selection of the inter-confederation path; this could use the same mechanism as proposed in the previous point, but if the paths are separate there is no compulsion to use the same algebra. Associating all confederation-related data together permits arbitrary levels of nesting, without having to alter the AS_PATH description any further. It also allows easier interoperation with route reflectors. Furthermore, the data can be isolated from the outside world, with benefits for stability and security.

An end-to-end attribute or set of attributes could be introduced at an appropriate point in the path selection process. The previous section identified several possible points with associated design tradeoffs for the correctness properties which would be required.

BGP communities provide a general mechanism for various aspects of routing protocol control not otherwise represented in the standard (Traina, Chandrasekeran and Li 1996). Communities are numeric tags which can be attached to route data, and which *may* cause recipients to take some action—whether to do something, or what to do, is entirely up to the recipient. In terms of BGP semantics, communities represent a source of mysterious and potentially dangerous behaviour, because their effect is so unconstrained. They therefore do not fit in well to the algebraic viewpoint. It is possible to identify some ways in which the community model might be made more susceptible to algebraic analysis, but a great deal of further effort would be required to complete these proposals and carry out the necessary theory.

Chapter 7

Conclusion

This thesis has presented a comprehensive account of several new results in algebraic routing. The theorems that have been proved give complete criteria for the presence or absence of important properties in algebraic constructions, and also extend the scope within which locally optimal solutions are known to be found. The wider project of this work has been to test the extent to which the algebraic routing model is feasible in understanding the routing problem. To that end, many examples have been explored which go beyond those typically considered in the field: not just to exercise corner cases of the algebra, but also to approach routing problems which are known to be difficult, such as the analysis of network areas. Connections have also been made to pre-existing mathematical theory, such as with the distributive lattices of Section 3.1.

The separation between *algebra* and *algorithm* has been an important philosophical driver behind the methods used here. The relationship between pathfinding and linear algebra is fundamental, and preserving this relationship while tackling new problems is worthwhile. The linear algebra expresses the essence of what happens in the finding of a routing solution, even in the case of non-distributive algebra. Being clear about this fact not only makes a large amount of theory available, but also aids our perception of the problem. It is not obvious how the algorithm design process can take place without an understanding of the problem that an algorithm is attempting to solve.

As such, in this thesis the algorithms have been kept pure and generic, and all details of a particular route-finding problem are part of the algebra. This includes the avoidance of loops which is necessary for convergence in some cases. It is true that for any particular instantiation of an algorithm with an algebra, there may be many ways of achieving the same result without maintaining the separation. This possibility only reinforces the importance of the separation, because that kind of alternative design should be considered in precisely this way: an optimization specific to the cir-

cumstances, whose correctness depends upon the known correctness of the generic solution.

The same argument can in principle be applied to concerns relating to distributed algorithms, where the connection with matrix multiplication is not so obvious. This thesis has not attempted to provide a formal translation between a centralized and a distributed algorithm. Nevertheless, given what has been done, it is plausible that the right way to think about distributed pathfinding is by treating the generic algebra-algorithm combination as the standard of measurement, and proving correctness in relation to that. More research is needed before this conclusion can be truly supported.

Several other areas for further work have already been identified. The analysis of properties and operators is clearly not complete: the work that has been done in this thesis does not represent, nor is it intended to represent, a definitive account of an algebraic metalanguage. There is considerable scope not only for further mathematical work on the underlying theory, but also for development of a metalanguage as an object in itself. This would include the writing of a formal syntax and semantics for the language, and accompanying analysis of its expressive power and applications. It may also involve some research on human factors, in the case of a metalanguage intended to be employed by network operators as part of a routing system.

In order for that to be done, we must thoroughly investigate the notion of expressive power of a language. The language design in the present thesis has been motivated by its ability to cover examples that we recognize as being important, while retaining the possibility of full property inference in all cases. We do not yet have a theory of how different metalanguages—with different base cases, operators and properties—might allow different algebraic structures to be defined and analyzed.

The work that has been done on the algebraic theory still suffers from several weaknesses. The theory that has been used throughout this thesis has not been described in a way that would be expected for someone familiar with standard abstract algebra. In particular, we lack a comprehensive structure theory that is explained in terms of familiar algebraic concepts (homomorphisms, ideals, varieties, and the like). It may be that such a treatment would reveal patterns for property deduction which so far have remained elusive. Similar remarks apply to the treatment of properties as logical propositions, which in this thesis has been informal but could benefit from a more complete study.

The lexicographic product has been given a thorough treatment, but one area that is lacking is a fully abstract definition of what the product really is. Such a

description would probably involve the use of category theory, expressing a lexicographic product as some kind of universal object in a category with particular characteristics. This kind of definition would allow a deeper unification of the order-theoretic and semigroup-theoretic products, as well as possibly suggesting other instances of the same pattern.

In the analysis of deduction rules for the various properties and operators, some repeated patterns can be seen in the rule structure. For example, the monotonicity of the lexicographic product depends on each component being monotonic, and there is an additional constraint on either the first or the second operand. In the same way, the existence of the order-theoretic lexicographic product depends on either the first operand being a linear order, or the second operand having a greatest element. In both cases, there is a pattern of the desired result being achievable as a result of a property on one or the other component.

This suggests that it should be possible to develop higher-level theorems about property deduction rules in some cases. It could be that such theorems would reveal the logical structure behind these rules, and their relationship to the definition of the product. These higher-level proof strategies could enable us to find new rules more easily: perhaps, given the definition of the lexicographic product and of the monotonicity property, there is a theorem which says that there must be properties P and Q such that

$$\mathrm{M}(S \mathbin{\vec{\times}} T) \iff \mathrm{M}(S) \wedge \mathrm{M}(T) \wedge (\mathrm{P}(S) \vee \mathrm{Q}(T))$$

and even some information about what they might be.

Regarding convergence proofs, the work in Chapter 4 has extended our knowledge of which properties are required in different scenarios, but does not encompass the most difficult case. This is the question of what is required for convergence in the case of an algebra which is not monotonic (or distributive), is not selective, and is not an algebra of minimal sets over another algebra which is selective. Examples of such algebras include some which are related to path counting, as well as those related to the k-shortest paths problem. Because of the importance of these problems, it would certainly be desirable to know what the required properties for convergence are. It is likely that a proof for these cases would itself be of interest for the structure of the underlying combinatorial problem, just as the existing proofs for the selective nondistributive case relate to the selective version of the Stable Paths Problem. The ultrametric-based approach of Section 4.2 may be promising for this more general case.

In addition, all current convergence proofs assume that paths with loops are invalid, but it may be possible to still achieve convergence in the presence of a weaker guarantee, such as an upper bound on the permitted number of loops. It should be noted in this connection that we do not have any applications for which this is the case, though some may be found which are unrelated to network routing.

Alongside the correctness issue are the related problems of algorithmic complexity and performance. It seems plausible that there should be a direct relationship between the properties of an algebra and the asymptotic complexity of finding a path assignment for a graph parametrized over that algebra. In particular, we might expect that monotonic algebras have more favourable performance characteristics than those which are not monotonic. There are some algebras which are 'almost monotonic' in the sense that some but not all triples of elements exhibit monotonicity; perhaps it is possible to quantify the degree of near-monotonicity of an algebra and relate this to algorithmic performance. At the same time, we know of several optimized forms of Dijkstra's algorithm, for example, which are applicable to monotonic algebras which have several additional algebraic properties (such as the possibility of subtraction, possessing an upper bound, and so on). It should be possible not only to derive deduction rules for these properties, but also relate them to the algorithmic improvements which they permit.

In terms of applications, there are many aspects of routing protocol behaviour which are not well covered by the present algebraic model. These include unusual features of current protocols, such as BGP communities, but mainly involve aspects of routing unconnected with the path selection process: naming, addressing, forwarding, and asynchronicity. Some of these may be more susceptible than others to algebraic treatment, and some have already been considered in this way by others. The extensions to 'traditional' route-finding algebra which have been presented in this thesis already demonstrate that the scope of the theory is wider than might have been supposed. It is plausible that this is also true in relation to these other problems.

Generic pathfinding is not only applicable to the Internet. Other areas in which it has been used include circuit layout; road and rail navigation; planning, scheduling and resource allocation; dataflow analysis of computer programs; and algorithmic problems related to automata, including string matching. It would be interesting to see to what extent the metarouting approach can contribute to these areas of study, and vice versa.

Appendix A

Extended proofs

A.1 *Convergence*

Proof of Lemma 4.1. The relation is obviously reflexive. For the remaining properties of a partial order, we will show that if T is linear, but $(\sqsubseteq)$ fails to be transitive or antisymmetric, then S must contain an infinite descending chain.

For the antisymmetry axiom, suppose that $f \sqsubseteq g$ and $g \sqsubseteq f$. If $f(s) = g(s)$ for all s then $f = g$. So suppose, without loss of generality, that $f(s_0) <_T g(s_0)$ for some s_0 in S. Because $g \sqsubseteq f$, there exists some $s_1 <_S s_0$ with $g(s_1) <_T f(s_1)$. But $f \sqsubseteq g$ as well, so then there must also be $s_2 <_S s_1$ with $f(s_2) <_T g(s_2)$. This construction can be repeated: we find an infinite descending chain

$$s_0 >_S s_1 >_S s_2 >_S s_3 >_S \cdots$$

which cannot be the case if S is well-founded. Hence $(\sqsubseteq)$ must be antisymmetric.

The argument for transitivity is similar. Suppose that $f \sqsubseteq g$, $g \sqsubseteq h$, and $h(s_0) <_T f(s_0)$ for some s_0 in S. Towards a contradiction, assume that $h(s) \leq_T f(s)$ whenever $s <_S s_0$. By linearity of T, either $g(s_0) <_T f(s_0)$ or $f(s_0) \leq_T g(s_0)$.

- If $g(s_0) <_T f(s_0)$ then there exists $s_1 <_S s_0$ with $f(s_1) <_T g(s_1)$, since $f \sqsubseteq g$. Therefore $h(s_1) <_T g(s_1)$, by assumption. But $g \sqsubseteq h$, so there exists $s_2 <_S s_1$ with $g(s_2) <_T h(s_2)$; and $g(s_2) <_T f(s_2)$. We again obtain an infinite descending chain by reapplying the construction.

- If $f(s_0) \leq_T g(s_0)$ then $h(s_0) <_T g(s_0)$. Since $g \sqsubseteq h$, there exists an $s_1 <_S s_0$ with $g(s_1) <_T h(s_1)$. By assumption, $g(s_1) <_T f(s_1)$; and since $f \sqsubseteq g$, there exists $s_2 <_S s_1$ with $f(s_2) <_T g(s_2)$. The assumption again yields $h(s_2) <_T g(s_2)$. Once more we find an infinite descending chain.

Therefore the assumption cannot hold: there must be some $s <_S s_0$ with $f(s) <_T h(s)$. This means that $f \sqsubseteq h$. □

A.2 *Basic properties for the lexicographic product*

In this section, we first prove deduction rules for the fundamental properties of commutativity, idempotence, and selectivity. The bulk of this section is however devoted to detailed proofs for the monotonicity and increasing properties, for both the order-theoretic lexicographic product and the monoidal version.

Proof of 5.7 and 5.8. Let $(S, \oplus_S)$ and $(T, \oplus_T)$ be semigroups, where either S is selective or T has an identity. These conditions guarantee that their lexicographic product $(S \vec{\times} T, \oplus)$ exists.

We will prove that S and T are semilattices if and only if $(S \vec{\times} T, \oplus)$ is a semilattice.

First, suppose that S and T are both commutative and idempotent. Consider the two expressions

$$c_1 = (s_1, t_1) \oplus (s_2, t_2)$$
$$c_2 = (s_2, t_2) \oplus (s_1, t_1)$$

in $S \vec{\times} T$. Their first components are equal, since S is commutative. So $s_1 = s_1 \oplus_S s_2$ if and only if $s_1 = s_2 \oplus_S s_1$, and likewise for s_2. It follows that the second components of c_1 and c_2 must be equal, by definition of the lexicographic product. Also, if $s_1 = s_2$ and $t_1 = t_2$ then we obtain

$$(s_1, t_1) \oplus (s_1, t_1) = (s_1 \oplus_S s_1, t_1 \oplus_T t_1) = (s_1, t_1). \tag{A.1}$$

Thus $S \vec{\times} T$ is commutative and idempotent.

Now, suppose that $S \vec{\times} T$ is a semilattice. Then S and T must both be idempotent, by A.1. Because $c_1 = c_2$ for all choices of elements from S and T, it must be that S is commutative (since their first components have to be equal). If $s_1 = s_2$ then the second components of c_1 and c_2 are $t_1 \oplus_T t_2$ and $t_2 \oplus_T t_1$ respectively; since these are known to be equal, T has to be commutative. □

Proof of 5.9. Let S and T be as in the previous proof.

If both are selective then $S \vec{\times} T$ is selective. By examination of Definition 5.1, it is enough to prove that if $s_1 = s_1 \oplus_S s_2 = s_2$, then

$$(s_1 \oplus_S s_2, t_1 \oplus_T t_2) \in \{(s_1, t_1), (s_2, t_2)\}.$$

This follows immediately from selectivity of T.

If $S \vec{\times} T$ is selective then S and T are selective. Let t_1 and t_2 be two elements of T, neither of which is an identity for $(\oplus_T)$. (If no two such elements can be found, then T is trivial and therefore selective in any case.)

For any s_1 and s_2, if $s_1 \oplus_S s_2$ is equal to neither s_1 nor s_2 then

$$(s_1, t_1) \oplus (s_2, t_2) = (s_1 \oplus_S s_2, \alpha_T).$$

But if neither t_1 nor t_2 is α_T, then $S \mathbin{\vec{\times}} T$ cannot be selective. So $s_1 \oplus_S s_2$ must be equal to at least one of s_1 and s_2; in other words, S is selective.

If S is selective then it is idempotent. So

$$(s, t_1) \oplus (s, t_2) = (s \oplus_S s, t_1 \oplus_T t_2) = (s, t_1 \oplus_T t_2).$$

If $S \mathbin{\vec{\times}} T$ is to be selective then $t_1 \oplus_T t_2$ must be equal to t_1 or t_2, or both. Therefore T is also selective. □

A.3 *Properties for global optima*

The results of this section establish Theorem 5.4 for all four algebraic structures. We first prove the theorem for order semigroups and order transforms, and then for bisemigroups and semigroup transforms.

Proof of Theorem 5.4 for order semigroups and order transforms. For all order transforms S and T, we have

$$\begin{aligned}
\mathrm{M}(S \mathbin{\vec{\times}} T) &\implies \mathrm{M}(S) \wedge \mathrm{M}(T) \\
\mathrm{M}(S \mathbin{\vec{\times}} T) \wedge \neg \mathrm{C}(S) &\implies \mathrm{K}(T) \\
\mathrm{M}(S \mathbin{\vec{\times}} T) \wedge \neg \mathrm{K}(T) &\implies \mathrm{C}(S) \\
\mathrm{M}(S) \wedge \mathrm{M}(T) \wedge (\mathrm{C}(S) \vee \mathrm{K}(T)) &\implies \mathrm{M}(S \mathbin{\vec{\times}} T)
\end{aligned}$$

by the lemmas (A.1, A.2, A.3, and A.4). Consequently,

$$\mathrm{M}(S \mathbin{\vec{\times}} T) \iff \mathrm{M}(S) \wedge \mathrm{M}(T) \wedge (\mathrm{C}(S) \vee \mathrm{K}(T)).$$

Since this statement holds for all order transforms, the corresponding statement is true for order semigroups, because we can construct an order transform $(A, \leq_A, F_A)$ from any order semigroup $(A, \leq_A, \otimes_A)$, where

$$F_A = \left\{ \lambda x \,.\, y \otimes_A x \;\middle|\; y \in A \right\},$$

and the properties carry across. □

Proof of Theorem 5.4 for bisemigroups and semigroup transforms. The proof has the same structure as the previous proof, but relies instead on the lemmas A.5, A.6, A.7 and A.8. The same technique carries the semigroup transform proof over into the bisemigroup proof. □

In the following lemmas, let $(S, \preceq_S, F)$ and $(T, \preceq_T, G)$ be order transforms, with their lexicographic product being $(S \times T, \preceq, F \times G)$.

Lemma A.1. *If* $\mathrm{M}(S \vec{\times} T)$ *then* $\mathrm{M}(S)$ *and* $\mathrm{M}(T)$.

Proof. Suppose that $\mathrm{M}(S \vec{\times} T)$. Then for all (s_1, t_1) and (s_2, t_2) in $S \times T$ and all (f, g) in $F \times G$ we have

$$(s_1, t_1) \preceq (s_2, t_2) \implies (f(s_1), g(t_1)) \preceq (f(s_2), g(t_2)).$$

Choose $t_1 = t_2$, and note that then $g(t_1) = g(t_2)$ for any g. From the definition of the lexicographic product,

$$s_1 \preceq_S s_2 \iff (s_1, t_1) \preceq (s_2, t_1)$$

and

$$f(s_1) \preceq_S f(s_2) \iff (f(s_1), g(t_1)) \preceq (f(s_2), g(t_2))$$

and so $\mathrm{M}(S)$ is true.

Likewise, set $s_1 = s_2$, and choose any f to deduce $\mathrm{M}(T)$. □

Lemma A.2. *If* $\mathrm{M}(S \vec{\times} T)$ *and* $\neg\mathrm{C}(S)$ *then* $\mathrm{K}(T)$.

Proof. Suppose that C does not hold for S, and that $S \vec{\times} T$ is monotonic. Then there exist s_1 and s_2 in S, and f in F, so that $s_1 \prec_S s_2$ and $f(s_1) \sim_S f(s_2)$. Since $s_1 \prec_S s_2$, we have

$$(s_1, t_1) \prec (s_2, t_2)$$

for all t_1 and t_2 in T. The product $S \vec{\times} T$ is monotonic, so

$$(f(s_1), g(t_1)) \preceq (f(s_2), g(t_2))$$

for any g in T; and because $f(s_1) \sim_S f(s_2)$, it must be the case that $g(t_1) \preceq_T g(t_2)$. This is true for any t_1 and t_2, so we have $\mathrm{K}(T)$: for all t_1, t_2 and g, $g(t_1) \sim_T g(t_2)$. □

Lemma A.3. *If* $\mathrm{M}(S \vec{\times} T)$ *and* $\neg\mathrm{K}(T)$ *then* $\mathrm{C}(S)$.

Proof. Suppose that $\neg\mathrm{K}(T)$ and $\mathrm{M}(S \vec{\times} T)$. Then there exist t_1 and t_2 in T, and g in G, such that $g(t_1)$ and $g(t_2)$ are strictly ordered or incomparable. Therefore, either

$\neg(g(t_1) \preceq_T g(t_2))$ or $\neg(g(t_2) \preceq_T g(t_1))$. Without loss of generality assume it is the former.

If $s_1 \prec_S s_2$, then $(s_1, t_1) \preceq (s_2, t_2)$, by definition of the lexicographic product. Since $S \vec{\times} T$ is monotonic, for all f in F and g in G,

$$f(s_1) \prec_S f(s_2)$$
$$\vee (f(s_1) \sim_S f(s_2) \wedge g(t_1) \preceq g(t_2)).$$

But $g(t_1) \preceq g(t_2)$ is false. Hence $s_1 \prec_S s_2$ implies $f(s_1) \prec_S f(s_2)$, and similarly $s_2 \prec_S s_1$ implies $f(s_2) \prec_S f(s_1)$.

Therefore, if $f(s_1)$ and $f(s_2)$ are equivalent, then it cannot be that $s_1 \prec_S s_2$ or $s_2 \prec_S s_1$. Since s_1 and s_2 cannot be strictly ordered in this case, they must be equivalent or incomparable. This proves $\mathrm{C}(S)$. □

Lemma A.4. *If* $\mathrm{M}(S)$, $\mathrm{M}(T)$, *and either* $\mathrm{C}(S)$ *or* $\mathrm{K}(T)$, *then* $\mathrm{M}(S \vec{\times} T)$.

Proof. Suppose that $(s_1, t_1) \preceq (s_2, t_2)$. Then either $s_1 \prec_S s_2$, or $s_1 \sim_S s_2$ and $t_1 \preceq_T t_2$. Hence $s_1 \preceq_S s_2$, and by monotonicity of S we have $f(s_1) \preceq_S f(s_2)$ for any f in F. If $f(s_1) \prec_S f(s_2)$ then there is nothing more to prove, so consider the case when $f(s_2) \sim_S f(s_1)$.

If $\mathrm{K}(T)$, then $g(t_1)$ and $g(t_2)$ are equivalent for any g in G. Consequently,

$$(f(s_1), g(t_1)) \preceq (f(s_2), g(t_2))$$

which makes $S \vec{\times} T$ monotonic.

However, if $\mathrm{C}(S)$ is true, then $s_1 \sim_S s_2$ (or $s_1 \#_S s_2$, but this cannot be the case because we already know that $s_1 \preceq_S s_2$). Hence $t_1 \preceq_T t_2$, and since T is monotonoc we have $g(t_1) \preceq_T g(t_2)$ for any g in G. By definition of the lexicographic product, $(f(s_1), g(t_1)) \preceq (f(s_2), g(t_2))$. So in this case, $S \vec{\times} T$ is also monotonic. □

In the following lemmas, let $(S, \oplus_S, F)$ and $(T, \oplus_T, G)$ be semigroup transforms, with their lexicographic product being $(S \times T, \oplus, F \times G)$.

Lemma A.5. *If* $\mathrm{M}(S \vec{\times} T)$ *then* $\mathrm{M}(S)$ *and* $\mathrm{M}(T)$.

Proof. Suppose that for all s_1 and s_2 in S, t_1 and t_2 in T, f in F and g in G,

$$(f, g)((s_1, t_1) \oplus (s_2, t_2)) = (f, g)(s_1, t_1) \oplus (f, g)(s_2, t_2).$$

The first component of the left expression is $f(s_1 \oplus_S s_2)$ and the first component of the right expression is $f(s_1) \oplus_S f(s_2)$, and these are equal for all elements of S and F. Therefore S is monotonic.

Let $s_1 = s_2$; note that $f(s_1)$, $f(s_2)$, $f(s_1) \oplus_S f(s_2)$ and $f(s_1 \oplus_S s_2)$ are all equal. Then the second component of the left expression above is $g(t_1 \oplus_T t_2)$, and the second component of the right expression is $g(t_1) \oplus_T g(t_2)$. Therefore T is also monotonic. □

Lemma A.6. *If* $\mathrm{M}(S \vec{\times} T)$ *and* $\neg\mathrm{C}(S)$ *then* $\mathrm{K}(T)$.

Proof. Suppose that $S \vec{\times} T$ is monotonic but S does not have C. Then there exist s_1 and s_2 in S, and f in F, such that $f(s_1) = f(s_2)$ but $s_1 \neq s_2$. By Lemma A.5, $f(s_1 \oplus_S s_2) = f(s_1) \oplus_S f(s_2)$, and this must be equal to $f(s_1)$ by idempotence of S. Hence

$$(f, g)(s_1, t_1) \oplus (f, g)(s_2, t_2) = (f(s_1), g(t_1 \oplus_T t_2))$$

for all t_1 and t_2 in T and g in G, and by monotonicity of $S \vec{\times} T$ this is equal to

$$(f, g)\left((s_1, t_1) \oplus (s_2, t_2)\right).$$

The T component of this expression depends on the values of s_1 and s_2: by definition of $(\oplus)$, it will be either $g(t_1)$, $g(t_2)$, or $g(\alpha_T)$. (It cannot be $g(t_1 \oplus_T t_2)$, since s_1 and s_2 are different.) We therefore have one of the following:

1. $\forall t_1, t_2 : g(t_1 \oplus_T t_2) = g(t_1)$,
2. $\forall t_1, t_2 : g(t_1 \oplus_T t_2) = g(t_2)$, or
3. $\forall t_1, t_2 : g(t_1 \oplus_T t_2) = g(\alpha_T)$.

If the first case holds, set $t_1 = \alpha_T$ to obtain

$$\forall t_2 : g(t_2) = g(\alpha_T).$$

Similarly, if the second or third case holds, set $t_2 = \alpha_T$ to obtain

$$\forall t_1 : g(t_1) = g(\alpha_T).$$

From either of these, it follows that $g(t_1) = g(t_2)$ for all t_1 and t_2. This is the statement of $\mathrm{K}(T)$. □

Lemma A.7. *If* $\mathrm{M}(S \vec{\times} T)$ *and* $\neg\mathrm{K}(T)$ *then* $\mathrm{C}(S)$.

Proof. Suppose that $S \vec{\times} T$ is monotonic but $\mathrm{K}(T)$ is false. Then there are t_1 and t_2 in T, and g in G, for which $g(t_1)$ and $g(t_2)$ are different (and so t_1 and t_2 are also different). Therefore $g(t_1) \oplus_T g(t_2)$ must be different from at least one of $g(t_1)$ and

$g(t_2)$. Without loss of generality, assume that $g(t_1) \neq g(t_1) \oplus_T g(t_2)$. Note that it must also be the case that $g(\infty_T) \neq g(t_1) \oplus_T g(t_2)$, since otherwise $g(t_1)$ and $g(t_2)$ would both have to be equal to $g(\infty_T)$.

Suppose that $f(s_1) = f(s_2)$ for some s_1 and s_2 in S and f in F; it must be shown that $s_1 = s_2$. So assume towards a contradiction that s_1 and s_2 are not equal. Note that $f(s_1 \oplus_S s_2)$ and $f(s_1) \oplus_S f(s_2)$ are also equal to $f(s_1)$. Then

$$(f,g)(s_1,t_1) \oplus (f,g)(s_2,t_2) = (f(s_1), g(t_1) \oplus_T g(t_2))$$

and by monotonicity of $S \vec{\times} T$, this is equal to

$$(f,g)\left((s_1,t_1) \oplus (s_2,t_2)\right) = (f(s_1), g(t))$$

where

$$t = \begin{cases} t_1 & s_1 = s_1 \oplus_S s_2 \neq s_2 \\ t_2 & s_1 \neq s_1 \oplus_S s_2 = s_2 \\ \infty_T & s_1 \neq s_1 \oplus_S s_2 \neq s_2. \end{cases}$$

Therefore $g(t_1) \oplus_T g(t_2)$ must be equal to $g(t)$. We have already shown that $g(t_1) \oplus_T g(t_2)$ is not equal to either $g(t_1)$ or $g(\infty_T)$. Hence $g(t)$ must be equal to $g(t_2)$.

It follows that $s_1 \neq s_1 \oplus_S s_2 = s_2$, in order for this case to emerge. But s_1 and s_2 could be exchanged in this argument to yield $s_2 \neq s_2 \oplus_S s_1 = s_1$; this gives a contradiction. So $s_1 = s_2$ after all, which proves that S has the C property. □

Lemma A.8. *If* M(S), M(T), *and either* C(S) *or* K(T), *then* M($S \vec{\times} T$).

Proof. Suppose that M(S), M(T) and C(S). Write

$$L = (f,g)\left((s_1,t_1) \oplus (s_2,t_2)\right)$$
$$R = (f,g)(s_1,t_1) \oplus (f,g)(s_2,t_2).$$

For s_1 and s_2 in S, there are four cases:

1. $s_1 = s_2$. Then L is equal to $(f(s_1 \oplus_S s_2), g(t_1 \oplus_T t_2))$. Now, if $s_1 = s_2$ then $f(s_1) = f(s_2)$, and by idempotence this is equal to $f(s_1 \oplus_S s_2)$ as well. Then R is equal to $(f(s_1) \oplus_S f(s_2), g(t_1) \oplus_T g(t_2))$, and by monotonicity of S and T this is equal to L. Therefore $S \vec{\times} T$ is monotonic

2. $s_1 = s_1 \oplus_S s_2 \neq s_2$. Then L is equal to $(f(s_1), g(t_1))$. Since $s_1 \neq s_2$ and C(S), we know that $f(s_1) \neq f(s_2)$; and by monotonicity of S, $f(s_1) = f(s_1 \oplus_S s_2) = f(s_1) \oplus_S f(s_2)$. Therefore $R = (f(s_1), g(t_1)) = L$, and $S \vec{\times} T$ is monotonic

3. $s_1 \neq s_1 \oplus s_2 = s_2$. This is symmetrical with the previous case.

4. $s_1 \neq s_1 \oplus s_2 \neq s_2$. By the C property of S, we have $f(s_1) \neq f(s_1 \oplus_S s_2) \neq f(s_2)$ also. Then $L = (f(s_1 \oplus s_2), g(\infty_T))$ and $R = (f(s_1) \oplus f(s_2), \infty_T)$, and these are equal. (Recall that $g(\infty_T) = \infty_T$ is an axiom for semigroup transforms.) Hence $S \vec{\times} T$ is monotonic

Now suppose that instead of $\mathrm{C}(S)$ we have $\mathrm{K}(T)$. Then $L = (f(s_1 \oplus_S s_2), \infty_T)$ since g applied to anything yields the same as g applied to ∞_T, namely ∞_T. Also, $R = (f(s_1) \oplus_S f(s_2), \infty_T \oplus_T \infty_T)$. Clearly $L = R$ by monotonicity of S, and so $S \vec{\times} T$ is monotonic as well. □

We will now prove property deduction rules for C and K. Theorem 5.5 asserts that

$$\mathrm{C}(S \vec{\times} T) \iff \mathrm{C}(S) \wedge \mathrm{C}(T)$$
$$\mathrm{K}(S \vec{\times} T) \iff \mathrm{K}(S) \wedge \mathrm{K}(T)$$

for S and T being bisemigroups, order semigroups, semigroup transforms or order transforms. As before, we will prove each of these statements for the semigroup transforms and for order transforms; from these, the proofs for bisemigroups and order semigroups follow immediately. This is done in the next four lemmas.

Lemma A.9. *If $(S, \oplus_S, F)$ and $(T, \oplus_T, G)$ are semigroup transforms for which the lexicographic product exists, then $\mathrm{C}(S \vec{\times} T)$ if and only if $\mathrm{C}(S) \wedge \mathrm{C}(T)$.*

Proof. The product $S \vec{\times} T$ is cancellative if and only if

$$(f(s_1), g(t_1)) = (f(s_2), g(t_2)) \implies (s_1, t_1) = (s_2, t_2)$$

for all s_1 and s_2 in S, t_1 and t_2 in T, f in F and g in G. It is obvious that this holds if S and T are individually cancellative. For the other direction, set $t_1 = t_2$ in the implication above to see that S must be cancellative if $S \vec{\times} T$ is cancellative; likewise, set $s_1 = s_2$ to obtain the statement of cancellativity of T. □

Lemma A.10. *If $(S, \oplus_S, F)$ and $(T, \oplus_T, G)$ are semigroup transforms for which the lexicographic product exists, then $\mathrm{K}(S \vec{\times} T)$ if and only if $\mathrm{K}(S) \wedge \mathrm{K}(T)$.*

Proof. We have $\mathrm{K}(S \vec{\times} T)$ if and only if

$$(f(s_1), g(t_1)) = (f(s_2), g(t_2))$$

for all s_1, s_2, t_1, t_2, f and g. Again, if S and T both have K, then K clearly holds for their lexicographic product. And if the above statement holds, then setting $s_1 = s_2$ or $t_1 = t_2$ will yield $\mathrm{K}(T)$ or $\mathrm{K}(S)$ respectively. □

Lemma A.11. *If $(S, \preceq_S, F)$ and $(T, \preceq_T, G)$ are order transforms for which the lexicographic product exists, then* $\mathrm{C}(S \vec{\times} T)$ *if and only if* $\mathrm{C}(S) \wedge \mathrm{C}(T)$.

Proof. The C property holds for $S \vec{\times} T$ if and only if

$$(f(s_1), g(t_1)) \sim (f(s_2), g(t_2)) \implies (s_1, t_1) \sim (s_2, t_2) \vee (s_1, t_1) \# (s_2, t_2)$$

for all s_1, s_2, t_1, t_2, f and g. If $\mathrm{C}(S)$ and $\mathrm{C}(T)$ both hold, then

$$\begin{aligned}(f(s_1), g(t_1)) \sim (f(s_2), g(t_2)) &\implies f(s_1) \sim_S f(s_2) \wedge g(t_1) \wedge g(t_2) \\ &\implies (s_1 \sim_S s_2 \vee s_1 \#_S s_2) \wedge (t_1 \sim_T t_2 \vee t_1 \#_T t_2)\end{aligned}$$

from which it follows that (s_1, t_1) and (s_2, t_2) are either equivalent or incomparable. This proves $\mathrm{C}(S \vec{\times} T)$.

For the other direction, setting $t_1 = t_2$ in the statement of $\mathrm{C}(S \vec{\times} T)$ yields $\mathrm{C}(S)$ and setting $s_1 = s_2$ yields $\mathrm{C}(T)$. □

Lemma A.12. *If $(S, \preceq_S, F)$ and $(T, \preceq_T, G)$ are semigroup transforms for which the lexicographic product exists, then* $\mathrm{K}(S \vec{\times} T)$ *if and only if* $\mathrm{K}(S) \wedge \mathrm{K}(T)$.

Proof. The product $S \vec{\times} T$ has the K property if and only if

$$(f(s_1), g(t_1)) \sim (f(s_2), g(t_2))$$

which by definition of $(\sim)$ for the lexicographic product is equivalent to

$$(f(s_1) \sim_S f(s_2)) \wedge (g(t_1) \sim_T g(t_2)).$$

Hence $\mathrm{K}(S \vec{\times} T)$ if and only if $\mathrm{K}(S)$ and $\mathrm{K}(T)$. □

A.4 *Properties for local optima*

Proof of Theorem 5.7. The statements to be proved are

$$\mathrm{ND}(S \vec{\times} T) \iff \mathrm{I}(S) \vee (\mathrm{ND}(S) \wedge \mathrm{ND}(T)) \tag{A.2}$$

$$\mathrm{I}(S \vec{\times} T) \iff \mathrm{I}(S) \vee (\mathrm{ND}(S) \wedge \mathrm{I}(T)) \tag{A.3}$$

for each of the four standard structures. The following lemmas prove these for semigroup transforms and order transforms; the proofs for bisemigroups and order semigroups follow from these.

Note that $\mathrm{I}(S)$ implies $\mathrm{ND}(S)$ in all cases. □

Lemma A.13. *Equation A.2 holds for semigroup transforms.*

Proof. By definition, $\text{ND}(S \mathbin{\vec{\times}} T)$ if and only if

$$\begin{aligned}(s,t) &= (s,t) \oplus (f(s), g(t)) \\ &= (s \oplus_S f(s), t')\end{aligned}$$

for all s, t, f and g, where

$$t' = \begin{cases} t \oplus_T g(t) & \text{if } s = s \oplus_S f(s) = f(s) \\ t & \text{if } s = s \oplus_S f(s) \neq f(s) \\ g(t) & \text{if } s \neq s \oplus_S f(s) = f(s) \\ \alpha_T & \text{if } s \neq s \oplus_S f(s) \neq f(s). \end{cases}$$

So for this to hold, it must be that $s = s \oplus_S f(s)$; and if $s = f(s)$ then it is additionally required that $t = t \oplus_T g(t)$, since otherwise t' would not be equal to t. Equally, if this does hold, then $s = s \oplus_S f(s)$ and either $s \neq f(s)$ or $t = t \oplus_T g(t)$. □

Lemma A.14. *Equation A.3 holds for semigroup transforms.*

Proof. Following from the previous lemma, the only additional relationships that needs to be proved are

$$\begin{aligned}\text{ND}(S) \wedge \text{I}(T) &\implies \text{I}(S \mathbin{\vec{\times}} T) \\ \text{I}(S \mathbin{\vec{\times}} T) \wedge \neg\text{I}(S) &\implies \text{I}(T).\end{aligned}$$

If $\text{ND}(S)$ and $\text{I}(T)$ then we already have $\text{ND}(S \mathbin{\vec{\times}} T)$, and it only remains to show that, using the same notation as the previous proof, (s,t) must always be different from $(f(s), g(t))$. This is the case if T is increasing, as then t and $g(t)$ must be different.

Suppose that $S \mathbin{\vec{\times}} T$ is increasing but S alone is not. Then there exist f and s with $s = f(s)$, and yet

$$(s,t) = (s,t) \oplus (f(s), g(t)) \neq (f(s), g(t)).$$

Hence it must be that t and $g(t)$ are always different: we have $\text{I}(T)$ as required, since $\text{ND}(T)$ follows from the previous lemma. □

Lemma A.15. *Equation A.2 holds for order transforms.*

Proof. The product $S \mathbin{\vec{\times}} T$ is nondecreasing if

$$(s,t) \preceq (f(s), g(t))$$

always; that is, if

$$(s \prec_S f(s)) \vee (s \sim_S f(s) \wedge t \preceq_T g(t)).$$

This can be rewritten as

$$(s \preceq_S f(s)) \wedge (\neg(f(s) \preceq_S s) \vee t \preceq_T g(t))$$

This certainly is the case if S is increasing ($s \prec_S f(s)$ for all s and f), or if both S and T are nondecreasing. For the reverse direction, if $S \vec{\times} T$ is nondecreasing then S is nondecreasing; and if S is not strictly increasing, then it must be that $t \preceq_T g(t)$, so that T is nondecreasing. □

Lemma A.16. *Equation A.3 holds for order transforms.*

Proof. It remains to be shown that if S is nondecreasing and T is increasing, then $S \vec{\times} T$ is increasing; and if $S \vec{\times} T$ is increasing but S is not, then T must be increasing.

For the first of these, if $s \preceq_S f(s)$ and $t \prec_T g(t)$ always, then

$$(s \prec_S f(s)) \vee (s \sim_S f(s) \wedge t \prec_T g(t))$$

which is the statement of $S \vec{\times} T$ being increasing.

For the second, S and T have already been shown to be nondecreasing; so if $(s,t) \prec (f(s), g(t))$ but $s \sim_S f(s)$ it must be the case that $t \prec_T g(t)$. □

We will prove Theorem 5.7 in the case of semigroup transforms. The bisemigroup proof follows directly from this. Likewise, the proof for order transforms subsumes the proof for order semigroups. The proofs for the ND and I properties are done separately.

In the following, let $(S, \oplus_S, F)$ and $(T, \oplus_T, G)$ be semigroup transforms with their lexicographic product being $(S \times T, \oplus, F \times G)$.

Proof of Theorem 5.7(1) for semigroup transforms. The product $S \vec{\times} T$ is nondecreasing if and only if

$$(s,t) = (s,t) \oplus (f(s), g(t))$$

for all s in S, t in T, f in F and g in G. By definition of $(\oplus)$, this is equivalent to the statement that

$$(s,t) = \begin{cases} (s \oplus_S f(s), t \oplus_T g(t)) & s = s \oplus_S f(s) = f(s) \\ (s \oplus_S f(s), t) & s = s \oplus_S f(s) \neq f(s) \\ (s \oplus_S f(s), g(t)) & s \neq s \oplus_S f(s) = f(s) \\ (s \oplus_S f(s), \infty_T) & s \neq s \oplus_S f(s) \neq f(s). \end{cases}$$

Now, it is clear that this holds if and only if $s = s \oplus_S f(s)$ for all s and f, since otherwise the S components of the expressions will not be equal. (Note that this condition is $\text{ND}(S)$.) So the statement can be simplified further to

$$(s,t) = \begin{cases} (s \oplus_S f(s), t \oplus_T g(t)) & s = f(s) \\ (s \oplus_S f(s), t) & s \neq f(s). \end{cases}$$

This is true in all cases if and only if $t = t \oplus_T g(t)$ for all t and g, *or* $s \neq f(s)$ for all s and f. Since $\text{ND}(S)$ is already known to hold, the conjunction of that with the condition that $s \neq f(s)$ yields $\text{I}(S)$.

Therefore $\text{ND}(S \vec{\times} T)$ if and only if

1. $\text{I}(S)$, or
2. $\text{ND}(S)$ and $\text{ND}(T)$,

as required. □

Proof of Theorem 5.7(2) for semigroup transforms. If $\text{I}(S \vec{\times} T)$ then $\text{ND}(S \vec{\times} T)$, and by the previous proof we already then have either $\text{I}(S)$ or $\text{ND}(S) \wedge \text{ND}(T)$. It remains to be shown that

1. $\text{I}(S \vec{\times} T) \wedge \neg \text{I}(S) \implies \text{I}(T)$
2. $\text{I}(S) \implies \text{I}(S \vec{\times} T)$
3. $\text{ND}(S) \wedge \text{I}(S) \implies \text{I}(S \vec{\times} T)$.

These statements will now be proved.

1. Suppose that $\text{I}(S \vec{\times} T)$ and $\neg \text{I}(S)$. Then there exist s in S and f in F such that $s = s \oplus_S f(s) = f(s)$. (Recall that $\text{ND}(S)$ is guaranteed.) If $\text{I}(S \vec{\times} T)$ then for all t in T and g in G we have

   $$(s,t) = (s,t) \oplus (f(s), g(t)) \neq (f(s), g(t))$$

   If $s = f(s)$ but $(s,t) \neq (f(s), g(t))$ then $t \neq g(t)$. Since it is already known that T is nondecreasing, this demonstrates that it must be increasing as well.

2. Suppose that $\text{I}(S)$. Then $\text{ND}(S \vec{\times} T)$ by the proof for ND. It remains to be shown that $(s,t) \neq (f(s), g(t))$ for any s, t, f and g. But if S is increasing then we cannot have $s = f(s)$, so (s,t) can never be equal to $(f(s), g(t))$.

3. Similarly, if ND(S) and I(T) then ND($S \vec{\times} T$), and the fact that T is increasing means that t and $g(t)$ will never be equal. Therefore, (s,t) and $(f(s),g(t))$ cannot be equal, and $S \vec{\times} T$ must be increasing.

This completes the proof for the I property. □

We will now prove the same theorem for order transforms, and consequently for order semigroups. Let $(S, \preceq_S, F)$ and $(T, \preceq_T, G)$ be order transforms, with their lexicographic product being $(S \times T, \preceq, F \times G)$.

Proof of Theorem 5.7(1) for order transforms. The product $S \vec{\times} T$ is nondecreasing if $(s,t) \preceq (f(s),g(t))$ for all s, t, f and g. By definition of the lexicographic product, this is equivalent to

$$s \prec_S f(s) \vee \big(s \sim_S f(s) \wedge t \preceq_T g(t)\big).$$

Clearly, if $s \prec_S f(s)$ is always true then ND($S \vec{\times} T$). Equally, if $s \preceq_S f(s)$ and $t \preceq_T g(t)$ are always true, then ND($S \vec{\times} T$) as well.

For the other direction, if ND($S \vec{\times} T$) then $s \prec_S f(s)$ or $s \sim_S f(s)$ always; therefore, $s \preceq_S f(s)$ for all s and f. So S is nondecreasing. If S happens to be increasing, then we are done, but if not, it must be that $t \preceq_T g(t)$ for all t and g. Therefore ND($S \vec{\times} T$) implies that either I(S), or ND(S) and ND(T). □

Proof of Theorem 5.7(2) for order transforms. As before, it remains to be shown that

1. I($S \vec{\times} T$) $\wedge$ $\neg$I(S) $\implies$ I(T)
2. I(S) $\implies$ I($S \vec{\times} T$)
3. ND(S) $\wedge$ I(S) $\implies$ I($S \vec{\times} T$).

Now, I($S \vec{\times} T$) if and only if $(s,t) \prec (f(s),g(t))$ for all s, t, f and g, which by definition is equivalent to

$$s \prec f(s) \vee \big(s \sim_S f(s) \wedge t \prec_T g(t)\big).$$

If $S \vec{\times} T$ is increasing but S is not increasing, then at least S is nondecreasing (by the previous proof). There must therefore be some s and f with $s \sim_S f(s)$. So for any t and g, we must have $t \prec_T g(t)$ in order for $S \vec{\times} T$ to be increasing.

The other two statements are obvious consequences of the definition of I: they correspond to the two clauses of the disjunction. □

A.5 *Reductions*

Proof of Theorem 5.9. It must be shown that if $S \vec{\times} T$ is 'distributive except for α_S', then $\mathbf{err}(S \vec{\times} T, E_1)$ is distributive. This condition means that

$$(f,g)(s_1,t_1) \oplus (f,g)(s_2,t_2) = (f,g)\left((s_1,t_1) \oplus (s_2,t_2)\right)$$

for all f in F, g in G, s_1 and s_2 in $S \setminus \{\alpha_S\}$, and t_1 and t_2 in T.

An important property of the chosen set E_1 is that if (s,t) is in E_1, then so is $(f,g)(s,t)$ for any (f,g), because $f(\alpha_S) = \alpha_S$ always.

Firstly, we can show that

$$r_{E_1} \circ (f,g) \circ r_{E_1} = r_{E_1} \circ (f,g) \tag{A.4}$$

for all (f,g) in $F \times G$. Clearly, these two compositions are equal when applied to a pair (s,t) with $s \neq \alpha_S$, since then we have $(s,t) = r_{E_1}(s,t)$. Now consider applying both to a pair (α_S,t). We have

$$r_{E_1}(f(\alpha_S), g(t)) = r_{E_1}(\alpha_S, g(t)) = (\alpha_S, \alpha_T)$$

and

$$\left(r_{E_1} \circ (f,g)\right)(r_{E_1}(\alpha_S,t)) = \left(r_{E_1} \circ (f,g)\right)(\alpha_S,\alpha_T) = (\alpha_S,\alpha_T)$$

so the equation holds in this case as well.

Distributivity requires that

$$r_{E_1}\left(\left(r_{E_1} \circ (f,g)\right)(x) \oplus \left(r_{E_1} \circ (f,g)\right)(y)\right) = \left(r_{E_1} \circ (f,g) \circ r_{E_1}\right)\left(x \oplus y\right)$$

for all (f,g) in $F \times G$, and x and y in $S \times T$. This can be rearranged, using A.4 and other reduction properties, to yield the equivalent form

$$(f,g)(x) \oplus (f,g)(y) \sim (f,g)(x \oplus y)$$

where $(\sim)$ denotes the congruence induced by r_{E_1}. Hence the two sides must either be equal, or they must both yield elements of E_1. This is as should be expected, since the purpose of defining E_1 was to avoid the situation where we could find (α_S,t_1) and (α_S,t_2) emerging from two computations that ought to have been equivalent: now, these will be mapped on to the same element (α_S,α_T) in order to restore distributivity.

Suppose that x is in E_1 but y is not. Then it must be that $y = x \oplus y \neq x$, from the definition of E_1. We have

$$(f,g)(x) \oplus (f,g)(y) = (\alpha_S,t) \oplus (f,g)(y) = (f,g)(y)$$

for some t in T, and

$$(f,g)(x \oplus y) = (f,g)(y)$$

so distributivity holds for this case, and by symmetry for the case when y is in E_1 but x is not.

If both x and y are in E_1, then $(f,g)(x) \oplus (f,g)(y)$ and $(f,g)(x \oplus y)$ must also be in E_1, and therefore are congruent under $(\sim)$.

Finally, suppose that neither x nor y is in E_1. Then neither can have α_S in their S component, by definition of E_1. The distributive law from the theorem statement then implies that

$$(f,g)(x) \oplus (f,g)(y) = (f,g)(x \oplus y),$$

which proves the remaining case.

Therefore, $\mathbf{err}(S \vec{\times} T, E_1)$ is distributive. □

Acknowledgements

Above all, my supervisor Tim Griffin deserves my thanks for his tireless support and encouragement throughout the previous three years.

I would also like to thank M. Abdul Alim, John Billings, Ken Calvert, Jon Crowcroft, Marcelo Fiore, Jonathan Hayman, Chung-Kil Hur, Franck Le, Andrew Moore, Vilius Naudžiūnas, Andrew Pitts, Balraj Singh, João Sobrinho, Barney Stratford, Philip Taylor, David Turner, Jamie Vicary, and Gordon Wilfong for their valuable advice and support during the writing of this thesis.

I am grateful for the support of Cambridge libraries and librarians. These include in particular, Nicholas Cutler of the Computer Laboratory library, and the staff of the Rare Books Room and Manuscript Room in the University Library.

Bibliography

The official repository at http://www.rfc-editor.org/ has copies of all RFC documents referred to below.

Ahuja, Ravindra K., Kurt Mehlhorn, James Orlin and Robert Endre Tarjan. 1990. Faster algorithms for the shortest path problem. *Journal of the Association for Computing Machinery* 37(2): 213–223.

Banach, Stefan. 1922. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae* 3:133–181.

Bellman, Richard. 1958. On a routing problem. *Quarterly of Applied Mathematics* 16(1): 87–90.

Bertsekas, Dimitri P. and Robert Gallager. 1992. *Data networks*. 2nd edition. Prentice-Hall.

Birkhoff, Garrett. 1937. Rings of sets. *Duke Mathematical Journal* 3(3): 443–454.

Birkhoff, Garrett. 1948. *Lattice Theory*. 2nd edition. Colloquium Publications 25. New York: American Mathematical Society.

Callon, Ross W. 1990. *RFC 1195: Use of OSI IS-IS for routing in TCP/IP and dual environments*.

Carré, Bernard A. 1971. An algebra for network routing problems. *Journal of the Institute of Mathematics and its Applications* 7(3): 273–294.

Carré, Bernard A. 1979. *Graphs and networks*. Oxford University Press.

Coltun, Rob. 1989. OSPF: An Internet routing protocol. *ConneXions: The Interoperability Report* 3(8): 19–25.

Cormen, Thomas H., Charles E. Leiserson and Ronald L. Rivest. 1990. *Introduction to algorithms*. 1st edition. MIT Press/McGraw-Hill.

Debreu, Gerard. 1954. Representation of a preference ordering by a numerical function. In *Decision Processes*, edited by R. M. Thrall, C. H. Coombs and R. L. Davis. Wiley. Chapter 11, 159–166.

Dijkstra, Edsger W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

Feamster, Nick and Hari Balakrishnan. 2005. Correctness properties for Internet routing. *Proceedings of the 43rd Allerton Conference on Communication, Control and Computing*.

Floyd, Robert W. 1962. Algorithm 97: Shortest path. *Communications of the Association for Computing Machinery* 5(6): 345.

Ford Jr., Lester Randolph and Delbert Ray Fulkerson. 1956. Maximal flow through a network. *Canadian Journal of Mathematics* 8:399–404.

Fredman, Michael L. and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association for Computing Machinery* 34(3): 596–615.

Fredman, Michael L. and Dan E. Willard. 1994. Trans-dichotomous algorithms for minimum spanning tree and shortest path. *Journal of Computer and System Sciences* 48(3): 533–551.

Gao, Lixin and Jennifer Rexford. 2000. Stable internet routing without global coordination. *Proceedings of ACM SIGMETRICS*. 307–317.

Garcia-Luna-Aceves, J. J. 1993. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking* 1(1): 130–141.

Głazek, Kazimierz. 2002. *A guide to the literature on semirings and their applications in mathematics and information sciences: with complete bibliography*. Kluwer Academic Publishers.

Gondran, Michel and Michel Minoux. 1984. *Graphs and algorithms*. Wiley.

Gondran, Michel and Michel Minoux. 2001. *Graphes, dioïdes et semi-anneaux: Nouveaux modèles et algorithmes*. Tec & Doc.

Gouda, Mohamed G. and Marco Schneider. 2003. Maximizable routing metrics. *IEEE/ACM Transactions on Networking* 11(4): 663–675.

Griffin, Timothy G. and Alexander J. T. Gurney. 2008. Increasing bisemigroups and algebraic routing. In *Relations and Kleene algebra in computer science*, Lecture Notes in Computer Science 4988. Proceedings of the 10th International Confer-

ence on Relational Methods in Computer Science (RelMiCS 10). Springer-Verlag. 123–137.

Griffin, Timothy G., F. Bruce Shepherd and Gordon Wilfong. 1999. Policy disputes in path vector protocols. *Proceedings of the 7th IEEE International Conference on Network Protocols (ICNP 99)*.

Griffin, Timothy G., F. Bruce Shepherd and Gordon Wilfong. 2002. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking* 10(2): 232–243.

Griffin, Timothy G. and João Luís Sobrinho. 2005. Metarouting. *Proceedings of ACM SIGCOMM*. 1–12.

Griffin, Timothy G. and Gordon Wilfong. 2000. A safe path vector protocol. *Proceedings of the 19th Annual IEEE Conference on Computer Communications (INFOCOM 2000)*.

Grillet, Pierre Antoine. 1995. *Semigroups: An introduction to the structure theory*. Monographs and textbooks in pure and applied mathematics 193. Marcel Dekker.

Gurney, Alexander J. T. and Timothy G. Griffin. 2007. Lexicographic products in metarouting. *Proceedings of the 15th IEEE International Conference on Network Protocols (ICNP 07)*. 113–122.

Hedrick, Charles L. 1988. *RFC 1058: Routing Information Protocol*.

Huston, Geoff. 2006. Exploring autonomous system numbers. *The Internet Protocol Journal* 9(1): 2–24.

Johnson, Donald B. 1977. Efficient algorithms for shortest paths in sparse networks. *Journal of the Association for Computing Machinery* 24(1): 1–13.

Kelley, John L. 1955. *General topology*. Graduate Texts in Mathematics 27. Springer.

Kilp, Mati, Ulrich Knauer and Alexander V. Mikhalev. 2000. *Monoids, acts and categories: with applications to wreath products and graphs*. Expositions in Mathematics 29. Walther de Gruyter.

Kirk, William A. 2001. Contraction mappings and extensions. In *Handbook of metric fixed point theory*. Edited by William A. Kirk and Brailey Sims. Springer.

Kleene, Stephen C. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*, edited by Claude Elwood Shannon and John McCarthy. Princeton University Press. 3–42.

Kruskal, Joseph B. 1972. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory*, A, 13(3): 297–305.

Lehmann, Daniel J. 1977. Algebraic structures for transitive closure. *Theoretical Computer Science* 4(1): 59–76.

Lengauer, T. and D. Theune. 1991. Efficient algorithms for path problems with general cost criteria. In *Automata, Languages and Programming: 18th International Colloquium, Madrid, Spain; July 18–22, 1991*. Lecture Notes in Computer Science 510. Springer-Verlag. 314–326.

Lindem, Acee, Rob Coltun, Dennis Ferguson and John Moy. 2008. *RFC 5340: OSPF for IPv6*.

Lougheed, Kirk and Yakov Rekhter. 1989. *RFC 1105: A Border Gateway Protocol (BGP)*. Experimental.

Lougheed, Kirk and Yakov Rekhter. 1990. *RFC 1163: A Border Gateway Protocol (BGP)*.

Malkin, Gary Scott. 1998. *RFC 2453: RIP version 2*.

Malkin, Gary Scott and Robert E. Minnear. 1997. *RFC 2080: RIPng for IPv6*.

Manger, Robert. 2006. Composite path algebras for solving path problems in graphs. *Ars Combinatoria* 78(1): 137–150.

Manger, Robert. 2008. A catalogue of useful composite semirings for solving path problems in graphs. *Proceedings of the 11th International Conference on Operational Research (KOI 2006)*.

McQuillan, John Macrae. 1974. *Adaptive routing algorithms for distributed computer networks*. PhD thesis, Harvard University. Also appeared as BBN Technical Report 2831.

McQuillan, John Macrae, Isaac Richer and Eric C. Rosen. 1980. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications* 28(5): 711–719.

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

Moore, Edward F. 1959. The shortest path through a maze. *Proceedings of an International Symposium on the Theory of Switching: Cambridge, Massachusetts, USA; 2–5 April 1957*. Annals of the Computation Laboratory of Harvard University 30. Harvard University Press.

Moy, John. 1989. *RFC 1131: The OSPF specification*.

Moy, John. 1998a. *OSPF: Anatomy of an Internet routing protocol*. Addison-Wesley.

Moy, John. 1998b. *RFC 2328: OSPF version 2*.

Moy, John. 2000. *OSPF: Complete implementation*. Addison-Wesley.

Novák, Vítězslav. 1965. On the lexicographic product of ordered sets. *Czechoslovak Mathematical Journal*, new series, 15(2): 270–282.

Oran, David. 1990. *RFC 1142: OSI IS-IS intra-domain routing protocol*.

Prieß-Crampe, Sibylla. 1990. Der Banachsche Fixpunktsatz für ultrametrische Räume. *Results in Mathematics* 18(1–2): 178–186.

Rekhter, Yakov and Tony Li. 1995. *RFC 1771: A Border Gateway Protocol 4 (BGP-4)*.

Rekhter, Yakov, Tony Li and Susan Hares. 2006. *RFC 4271: A Border Gateway Protocol 4 (BGP-4)*. Replaces Rekhter and Li (1995).

Rosen, Eric C. 1982. *RFC 827: Exterior Gateway Protocol (EGP)*.

Rote, Günter. 1985. A systolic array algorithm for the algebraic path problem. *Computing* 34(3): 191–219.

Rote, Günter. 1990. Path problems in graphs. In *Computational graph theory*, edited by Gottfried Tinhofer et al. Computing Supplementa 7. Springer-Verlag. 155–189.

Roy, Bernard. 1959. Transitivité et connexité. *Comptes rendus hebdomadaires des séances de l'Académie des Sciences* 249:216–8.

Saitô, Tôru. 1970. Note on the lexicographic product of ordered semigroups. *Proceedings of the Japan Academy* 46(5): 413–416.

Sobrinho, João Luís. 2003. Network routing with path vector protocols: Theory and applications. *Proceedings of ACM SIGCOMM*. 49–60.

Sobrinho, João Luís. 2005. An algebraic theory of dynamic network routing. *IEEE/ACM Transactions on Networking* 13(5): 1160–1173.

Szpilrajn, Edward. 1930. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae* 16:386–389.

Thorup, Mikkel. 2007. Equivalence between priority queues and sorting. *Journal of the Association for Computing Machinery* 54(6): Article 28. Expanded version of an article that appeared in the 43rd IEEE Symposium on Foundations of Computer Science (FOCS 2002).

Traina, Paul, Ravishanker Chandrasekeran and Tony Li. 1996. *RFC 1997: BGP communities attribute*.

Warshall, Stephen. 1962. A theorem on Boolean matrices. *Journal of the Association for Computing Machinery* 9(1): 11–12.

Wongseelashote, Ahnont. 1976. An algebra for determining all path-values in a network with application to k-shortest-paths problems. *Networks* 6(4): 307–334.

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

Zimmermann, Uwe. 1981. *Linear and combinatorial optimization in ordered algebraic structures*. Annals of discrete mathematics 10. Elsevier North-Holland.