A Model of Internet Routing Using Semi-modules

John N. Billings and Timothy G. Griffin

Computer Laboratory, University of Cambridge
{John.Billings,Timothy.Griffin}@cl.cam.ac.uk

Abstract. Current Internet routing protocols exhibit several types of anomalies that can reduce network reliability. In order to design more robust protocols we need better formal models to capture the complexities of Internet routing. In this paper we develop an algebraic model that clarifies the distinction between *routing tables* and *forwarding tables*. We hope that this suggests new approaches to the design of routing protocols.

1 Introduction

Internet data traffic traverses a sequence of links and routers as it travels from source to destination. Routers employ *forwarding tables* to control traffic at each step. Typically, forwarding tables are constructed automatically from *routing tables*, which in turn are generated by routing protocols that dynamically discover network paths.

We attempt to clarify the distinction between forwarding and routing from a high-level perspective, ignoring implementation details. To model routing, we use an algebraic approach based on idempotent semirings (see for example [1]). For this paper, a (network-wide) routing table is simply a matrix $\mathbf{R}$ that satisfies an equation

$$\mathbf{R} = (\mathbf{A} \otimes \mathbf{R}) \oplus \mathbf{I},$$

where $\mathbf{A}$ is an adjacency matrix associated with a graph weighted over a (well-behaved) semiring S. Various algorithms, distributed or not, can be used to compute a routing table $\mathbf{R} = \mathbf{A}^*$ from the adjacency matrix. Each entry $\mathbf{R}(i,\ j)$ is (implicitly) associated with a set of optimal paths from node i to node j.

In Section 2 we model a forwarding table as a matrix $\mathbf{F}$ where each entry $\mathbf{F}(i,\ d)$ is (implicitly) associated with a set of paths from node i to destination d. Here destinations are assumed to be in a namespace disjoint from nodes. We then treat the construction of a forwarding table $\mathbf{F}$ as the process of solving an equation

$$\mathbf{F} = (\mathbf{A} \triangleright \mathbf{F}) \square \mathbf{M},$$

where $\mathbf{F}$ and $\mathbf{M}$ contain entries in a semi-module $(\square,\ \triangleright)$ over the semiring S. Entries in the *mapping table* $\mathbf{M}(i,\ d)$ contain metrics associated with the attachment of external destination d to infrastructure node i.

The solution $\mathbf{F} = \mathbf{R} \triangleright \mathbf{M}$ tells us how to combine routing and mapping to produce forwarding. We present several semi-module constructions that model common Internet forwarding idioms such as *hot-* and *cold-potato* forwarding. Section 3 shows how

R. Berghammer et al. (Eds.): RelMiCS/AKA 2009, LNCS 5827, pp. 29–43, 2009.

mapping tables can themselves be generated from forwarding tables. This provides a model of one simple type of *route redistribution* between distinct routing protocols.

In Section 4 we discuss how this model is related to current rethinking of the Internet's addressing architecture (see for example John Day's book [2]), and to existing problems with route redistribution [3,4,5]. For completeness, Appendix A supplies definitions of semirings and semi-modules.

2 Routing versus Forwarding

With Internet technologies we can make a distiction between *routing* and *forwarding*. We will consider routing to be a function that establishes and maintains available paths within a specified routing domain. How such paths are actually used to carry traffic is for us a question of forwarding.

Of course, routing and forwarding are intimately related, and in practice the two terms are often used as if they were synonyms. Indeed, in the simplest case the distinction may seem pointless: when the forwarding function causes traffic to flow on exactly the paths provided by the routing function. However, even in this simple case the distinction must be made because of the possibility of multiple *equal cost* paths within a network. There are many possible choices for forwarding with equal cost paths, such as randomly choosing a path or dynamically balancing load between paths.

In this section we model a network's *infrastructure* as a directed graph $G = (V, E)$. Given a pair of nodes $i, j \in V$, routing computes a set of paths in G that can be used to transit data from i to j. We model routing with a $V \times V$ *routing matrix* $\mathbf{R}$. Entry $\mathbf{R}(i, j)$ in fact corresponds to the minimal-cost path *weight* from i to j, although under certain assumptions (see later) it is straightforward to recover the associated paths.

In addition, we suppose that there is a set of *external destinations* D that are independent of the network. Destinations can be *directly attached* to any number of nodes in the network. We model the attachment information using a $V \times D$ *mapping matrix* $\mathbf{M}$. Forwarding then consists of finding minimal-cost paths from nodes $i \in V$ to destinations $d \in D$. We model forwarding using a $V \times D$ *forwarding matrix* $\mathbf{F}$. We shall see that there are several different ways to combine routing and mapping matrices to produce a forwarding matrix, with each such method potentially leading to a different set of forwarding paths; the examples in this section all share the same routing matrix, yet have distinct forwarding paths.

2.1 Algebraic Routing

In this section we provide a basic overview of algebraic routing with semirings. Let $S = (S, \oplus, \otimes)$ be an idempotent semiring (Appendix A.1). Associate arcs with elements of the semiring using a weight function $w \in E \to S$. Let $\mathbf{A}$ be the $V \times V$ adjacency matrix induced by w. Denote the set of all paths in G from node i to node j by $P(G, i, j)$. Given a path $\mu = \langle u_0, u_1, \cdots, u_n \rangle \in P(G, i, j)$ with $u_0 = i$ and $u_n = j$, define the weight of μ as $w(\mu) = w(u_0, u_1) \otimes w(u_1, u_2) \otimes \cdots w(u_{n-1}, u_n)$. For paths $\mu, \nu \in P(G, i, j)$, summarise their weights as $w(\mu) \oplus w(\nu)$. Define the *shortest-path weight* from i to j as

$$\delta(i,\, j) = \bigoplus_{p \in P(G,\, i,\, j)} w(p).$$

We seek to find a $V \times V$-matrix $\mathbf{R}$ satisfying $\mathbf{R}(i,\, j) = \delta(i,\, j)$. We term this matrix a *routing solution* because it models the shortest-path weights across a network's infrastructure. How might we compute the value of this matrix? It is straightforward to show that such a matrix $\mathbf{R}$ also satisfies the *routing equation*

$$\mathbf{R} = (\mathbf{A} \otimes \mathbf{R}) \oplus \mathbf{I}. \tag{1}$$

Now, define the *closure* of $\mathbf{A}$ as $\mathbf{A}^* = \mathbf{I} \oplus \mathbf{A} \oplus \mathbf{A}^2 \oplus \cdots$. It is well-known that if this closure exits, then it satisfies equation 1 (see the classic reference [6], or the recent survey of the area [1]). Various sufficient conditions can be assumed to hold on the semiring S which imply that $\mathbf{A}^*$ will always exist (and so the set of adjacency matrices becomes a Kleene algebra), but we will not explore these conditions here.

Note that we use the standard notation of identifying the operators from the semiring S with those from the same semiring lifted to operate over (square) matrices with elements from S. The additive operator for the lifted semiring is defined as

$$(\mathbf{X} \oplus \mathbf{Y})(i,\, j) = \mathbf{X}(i,\, j) \oplus \mathbf{Y}(i,\, j),$$

whilst the multiplicative operator is defined as

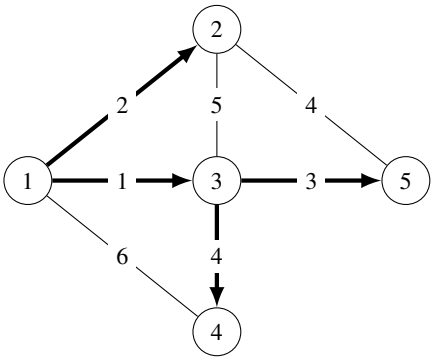
$$(\mathbf{X} \otimes \mathbf{Y})(i,\, j) = \bigoplus_{k \in V} \mathbf{X}(i,\, k) \otimes \mathbf{Y}(k,\, j).$$

Hence we see that the latter operator in fact uses both the underlying $\oplus$ and $\otimes$ operators from S. This distinction will become more significant when we consider lifting certain semi-modules to operate over matrices.

The example in Figure 1 illustrates the algebraic approach. Figure 1(a) presents a simple five node graph with integer labels and Figure 1(b) shows the associated adjacency matrix. We assume that arc weights are symmetric. We wish to compute *shortest-distances* between each pair of nodes and therefore we compute the closure using the semiring $\mathrm{MinPlus} = (\mathbb{N}^\infty,\, \min,\, +)$, where $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$. The resulting matrix is given in Figure 1(c). It is straight-forward to recover the corresponding paths because $\mathrm{MinPlus}$ is *selective*. That is, for all $x,\, y \in S$ we have $x \oplus y \in \{x,\, y\}$. Hence the computed weights actually correspond to the weights of individual paths. The bold arrows in Figure 1(a) denote the shortest-paths tree rooted at node 1; the corresponding path weights are given in the first row of the matrix in Figure 1(c).

2.2 Importing External Destinations

As before, suppose that our network is represented by the graph $G = (V,\, E)$, labelled with elements from the semiring S. Let the external nodes be chosen from some set D, satisfying $V \cap D = \emptyset$. Attach external nodes to G using the *attachment edges* $E' \subseteq V \times D$. In the simplest case, the edges in E' have weights from S, although in



(a) Labelled graph

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} \infty & 2 & 1 & 6 & \infty \\ 2 & \infty & 5 & \infty & 4 \\ 1 & 5 & \infty & 4 & 3 \\ 6 & \infty & 4 & \infty & \infty \\ \infty & 4 & 3 & \infty & \infty \end{bmatrix} \end{array}$$

(b) Adjacency matrix

$$\mathbf{R} = \mathbf{A}^* = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 5 & 4 \\ 2 & 0 & 3 & 7 & 4 \\ 1 & 3 & 0 & 4 & 3 \\ 5 & 7 & 4 & 0 & 7 \\ 4 & 4 & 3 & 7 & 0 \end{bmatrix} \end{array}$$

(c) Routing matrix

Fig. 1. Algebraic routing example using the $\mathrm{MinPlus}$ semiring

the next section we show how to relax this assumption. Let the $V \times D$ *mapping matrix* $\mathbf{M}$ represent the attachment edges.

We now wish to compute the $V \times D$ matrix $\mathbf{F}$ of shortest-path weights from nodes in V to nodes in D. We term $\mathbf{F}$ a *forwarding solution* because it comprises the information required to reach destinations, instead of other infrastructure nodes. We compute $\mathbf{F}$ by post-multiplying the routing solution $\mathbf{R}$ by the mapping matrix $\mathbf{M}$. That is, for $i \in V$ and $d \in D$, we have

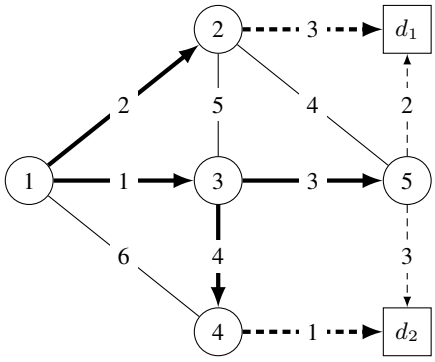
$$\mathbf{F}(i,\, d) = (\mathbf{R} \otimes \mathbf{M})(i,\, d) = \bigoplus_{k \in V} \mathbf{R}(i,\, k) \otimes \mathbf{M}(k,\, d) = \bigoplus_{k \in V} \delta(i,\, k) \otimes \mathbf{M}(k,\, d).$$

Hence we see that $\mathbf{F}(i,\, d)$ corresponds to the shortest total path length from i to d. In other words, $\mathbf{F}$ solves the *forwarding equation*

$$\mathbf{F} = (\mathbf{A} \otimes \mathbf{F}) \oplus \mathbf{M}. \tag{2}$$

Note that we are able to change the value of $\mathbf{M}$ and recompute $\mathbf{F}$ without recomputing $\mathbf{R}$. From an Internet routing perspective this is an important property; if the external information is dynamically computed (by another routing protocol, for example) then it may frequently change, and in such instances it is desirable to avoid recomputing routing solutions.

We illustrate this model of forwarding in Figure 2. The labelled graph of Figure 2(a) is based upon that in Figure 1(a), with the addition of two external nodes: d_1 and d_2. The adjacency matrix $\mathbf{A}$ remains as before, whilst the mapping matrix $\mathbf{M}$, given in Figure 2(b), contains the attachment information for d_1 and d_2. The forwarding solution $\mathbf{F}$ that results from multiplying $\mathbf{R}$ by $\mathbf{M}$ is given in Figure 2(c). Again, it is easy to verify that the elements of $\mathbf{F}$ do indeed correspond to the weights of the shortest paths from nodes in V to nodes in D.



(a) Graph with external information

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} \infty & \infty \\ 3 & \infty \\ \infty & \infty \\ \infty & 1 \\ 2 & 3 \end{bmatrix} \end{array}$$

(b) Mapping matrix

$$\mathbf{R} \otimes \mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} 5 & 6 \\ 3 & 7 \\ 5 & 5 \\ 9 & 1 \\ 2 & 3 \end{bmatrix} \end{array}$$

(c) Forwarding matrix

Fig. 2. Example of combining routing and mapping to create *forwarding*

2.3 A General Import Model Using Semi-modules

Within Internet routing, it is common for the entries in routing and forwarding tables to have distinct types, and for these types to be associated with distinct order relations. We therefore generalise the import model of the previous section to allow this possibility. In particular, we show how to solve this problem using algebraic structures known as *semi-modules* (Appendix A.2).

Assume that we are using the semiring $S = (S, \oplus, \otimes)$ and suppose that we wish to construct forwarding matrices with elements from the idempotent, commutative semigroup $N = (N, \square)$. Furthermore, suppose that the mapping matrix $\mathbf{M}$ contains entries over N. In order to compute forwarding entries, it is necessary to combine routing entries with mapping entries, as before. However, we can no longer use the multiplicative operator from S because the mapping entries are of a different type. Therefore we introduce an operator $\triangleright \in (S \times N) \to N$ for this purpose. We can now construct forwarding entries as

$$\mathbf{F}(i, d) = (\mathbf{R} \triangleright \mathbf{M})(i, d) = \bigsquare_{k \in V} \mathbf{R}(i, k) \triangleright \mathbf{M}(k, d). \tag{3}$$

It is also possible to equationally characterize the resulting forwarding entries, as before. Assume that $\mathbf{R}$ is a routing solution i.e. it satisfies Equation 1. Then, providing that the algebraic structure $N = (N, \square, \triangleright)$ is a semi-module, $\mathbf{F} = \mathbf{R} \triangleright \mathbf{M}$ is a solution to the forwarding equation

$$\mathbf{F} = (\mathbf{A} \triangleright \mathbf{F}) \square \mathbf{M}.$$

In other words, we can solve for $\mathbf{F}$ with $\mathbf{F} = \mathbf{A}^* \triangleright \mathbf{M}$. Significantly, we are able to use semi-modules to model the mapping information whilst still retaining a semiring model of routing.

We now develop two important semi-module constructions that model the most common manner in which routing and mapping are combined: the *hot-potato* and *cold-potato* semi-modules. First define an *egress* node for a destination d as a node k within

the routing domain that is directly attached to d. Hot-potato forwarding to d first selects paths to the closest egress nodes for d and then breaks ties using the mapping information. In contrast, cold-potato forwarding first selects paths to the egress nodes for d with the most preferred mapping values, and then breaks ties using the routing distances.

We now formally define the hot-potato semi-module. Let $S = (S, \oplus_S, \otimes_S)$ be an idempotent semiring with $(S, \oplus_S)$ selective and let $T = (T, \oplus_T)$ be a monoid. The hot-potato semi-module over S is defined as

$$\mathrm{Hot}(S, T) = ((S \times T) \cup \{\infty\}, \vec{\oplus}, \triangleright_{\mathrm{fst}}),$$

where $s_1 \triangleright_{\mathrm{fst}} (s, t) = (s_1 \otimes_S s, t)$ and $s_1 \triangleright_{\mathrm{fst}} \infty = \infty$. The *left* lexicographic product semigroup $((S \times T) \cup \{\infty\}, \vec{\oplus})$ is defined in Appendix A.3. In common with semirings, we can lift semi-modules to operate over (non-square) matrices. When lifting the hot-potato semi-module, we rename the multiplicative operator from $\triangleright_{\mathrm{fst}}$ to $\triangleright_{\mathrm{hp}}$. This is because the lifted multiplicative operator no longer simply applies its left argument to the first component of its right argument. In fact, we shall shortly see that the cold-potato semi-module uses the same underlying multiplicative operator but with a different additive operator, and therefore has a different lifted multiplicative operator.

The behaviour of the hot-potato semi-module can be algebraically characterised as follows. Suppose that for all $j \in V$ and $d \in D$, $\mathbf{M}(j, d) \in \{(1_S, t), \infty_T\}$ where 1_S is the multiplicative identity for S and t is some element of T. Then from Equation 3 it is easy to check that

$$(\mathbf{R} \triangleright_{\mathrm{hp}} \mathbf{M})(i, d) = \vec{\bigoplus_{\substack{j \in V \\ M(j,d) = (1_S, t)}}} (\mathbf{R}(i, j), t).$$

That is, as desired, the mapping metric is simply used to tie-break over otherwise minimal-weight paths to the edge of the routing domain.

We illustrate the hot-potato model of forwarding in Figure 3. This example uses the semi-module $\mathrm{Hot}(\mathrm{MinPlus}, \mathrm{Min})$, where $\mathrm{Min} = (\mathbb{N}^{\infty}, \min)$. The graph of Figure 3(a) is identical to that of Figure 2(a), but now the attachment arcs of d_1 and d_2 are weighted with elements of the hot-potato semi-module. The associated mapping matrix is given in Figure 3(b), whilst the resulting forwarding table is shown in Figure 3(c). Note that 0 is the multiplicative identity of the $(\min, +)$ semiring. Comparing this example to Figure 2, we see that node 1 reaches d_2 via egress node 5 instead of node 4. This is because the mapping information is only used for tie-breaking, instead of being directly combined with the routing distance. Also, in this particular example, it is never the case that there are multiple paths of minimum cost to egress nodes, and therefore no tie-breaking is performed by the mapping information.

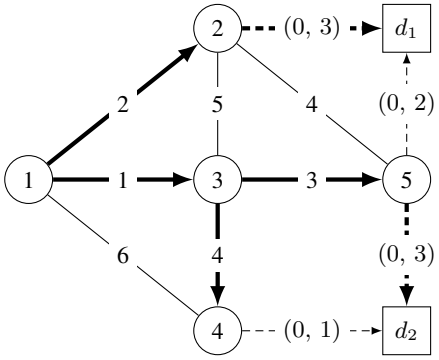
Turning to cold-potato forwarding, the associated semi-module again combines routing and attachment information using the lexicographic product, but with priority now given to the attachment component. As before, let $S = (S, \oplus_S, \otimes_S)$ be a semiring and $T = (T, \oplus_T)$ be a monoid, but now with T idempotent and selective. The cold-potato semi-module over S is defined as

$$\mathrm{Cold}(S, T) = ((S \times T) \cup \{\infty\}, \overleftarrow{\oplus}, \triangleright_{\mathrm{fst}}).$$

Note that the *right* lexicographic product semigroup $(S \times T, \overleftarrow{\oplus})$ is defined in Appendix A.3. Again, when lifting the cold-potato semi-module to operate over matrices we rename the multiplicative operator from $\triangleright_{\mathrm{fst}}$ to $\triangleright_{\mathrm{cp}}$.

Figure 4 illustrates the cold-potato model of forwarding. This example uses the cold-potato semi-module $\mathrm{Cold}(\mathrm{MinPlus},\ \mathrm{Min})$, but is otherwise identical to Figure 3. It is easy to verify that priority is now given to the mapping information when selecting egress nodes.

Within Internet routing, hot-potato forwarding corresponds to choosing the closest egress point from a given routing domain. This is the default behaviour for the Border Gateway Protocol (BGP) routing protocol [7] because it tends to minimise resource usage for outbound traffic within the domain. In contrast, cold-potato forwarding allows the mapping facility to select egress nodes, and hence can lead to longer paths being chosen within the domain. As a result, cold-potato forwarding is less commonly observed



(a) Graph with external information

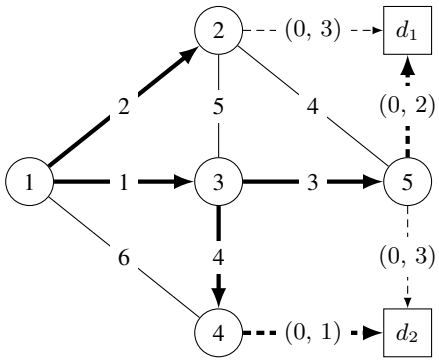
$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} \infty & \infty \\ (0,\,3) & \infty \\ \infty & \infty \\ \infty & (0,\,1) \\ (0,\,2) & (0,\,3) \end{bmatrix} \end{array}$$

(b) Mapping matrix

$$\mathbf{R} \triangleright_{\mathrm{hp}} \mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} (2,\,3) & (4,\,3) \\ (0,\,3) & (4,\,3) \\ (3,\,2) & (3,\,3) \\ (7,\,2) & (0,\,1) \\ (0,\,2) & (0,\,3) \end{bmatrix} \end{array}$$

(c) Forwarding matrix

Fig. 3. Example of *hot-potato* forwarding



(a) Graph with external information

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} \infty & \infty \\ (0,\,3) & \infty \\ \infty & \infty \\ \infty & (0,\,1) \\ (0,\,2) & (0,\,3) \end{bmatrix} \end{array}$$

(b) Mapping matrix

$$\mathbf{R} \triangleright_{\mathrm{cp}} \mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} (4,\,2) & (5,\,1) \\ (4,\,2) & (9,\,1) \\ (3,\,2) & (4,\,1) \\ (7,\,2) & (0,\,1) \\ (0,\,2) & (7,\,1) \end{bmatrix} \end{array}$$

(c) Forwarding matrix

Fig. 4. Example of *cold-potato* forwarding

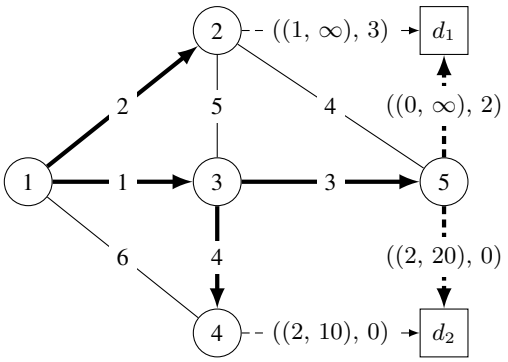
in general on the Internet. However, one specific use is within client-provider peering relations in order to minimise the use of the client's network resources for inbound traffic (at the possible expense of increased resource usage on the provider's network).

2.4 Idealized OSPF: An Example of Combined Mappings

We now present a highly-idealized account that attempts to tease out an algebraic description of the construction of forwarding tables in the OSPF routing protocol [8]. The specification of this protocol [9] runs for 244 pages and is primarily focused on implementation details. For simplicity we ignore OSPF areas.

Destinations are attached in three different ways in OSPF. Type 0 (our terminology) destinations are directly attached to a node, while Type 1 and Type 2 destinations (terminology of [9]) represent two ways of attaching external destinations. These may be statically-configured or learned via other routing protocols. The OSPF specification defines the relative preference for destination types to be used in constructing a forwarding table: Type 0 are preferred to Type 1, and Type 1 are preferred to Type 2.

In addition, Type 2 destinations are associated with a metric that is to be inspected before the internal routing metric In other words, cold-potato forwarding is used for Type 2 destinations. We generalize OSPF and assume that Type 2 metrics come from a commutative, idempotent monoid. $U = (U, \oplus_U)$. We use the network in Figure 5(a) as an ongoing example; here we have $U = (\mathbb{N}^\infty, \max)$, and we think of this as a *bandwidth* metric (note that here $\infty_U = 0$).



(a) Graph with external information

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} \infty & \infty \\ ((1,\infty),3) & \infty \\ \infty & \infty \\ \infty & ((2,10),0) \\ ((0,\infty),2) & ((2,20),0) \end{bmatrix} \end{array} \qquad \mathbf{R} \rhd_{\mathrm{cp}} \mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} ((0,\infty),6) & ((2,20),4) \\ ((0,\infty),6) & ((2,20),8) \\ ((0,\infty),5) & ((2,20),3) \\ ((0,\infty),9) & ((2,20),7) \\ ((0,\infty),2) & ((2,20),0) \end{bmatrix} \end{array}$$

(b) Mapping matrix (c) Forwarding matrix

Fig. 5. Example of *idealized-OSPF* forwarding

We use the following set into which we embed each destination type,

$$W = ((\{0,\, 1,\, 2\} \times U) \times \mathbb{N}^{\infty}) \cup \{\infty\}.$$

Each destination type is embedded into W as follows:

Type	Metric	Embedding
0	$m \neq \infty$	$((0,\, \infty_U),\, m)$
1	$m \neq \infty$	$((1,\, \infty_U),\, m)$
2	$u \neq \infty_U$	$((2,\, u),\, 0)$

The elements of W are then ordered using the (right) lexicographic product (see Appendix A.3),

$$\vec{\oplus} = (\{1,\, 2,\, 3\}, \min) \vec{\times} (U, \oplus_U) \vec{\times} (\mathbb{N}^{\infty},\, \min).$$

Hence the order amongst metrics of the same destination type remains unchanged, whilst the order between different destination types respects the ordering defined within the OSPF specification.

Assume we start with one mapping matrix for each type of destination (how these matrices might actually be produced is ignored),

$\mathbf{M}_0$

	d_1	d_2
1	∞	∞
2	∞	∞
3	∞	∞
4	∞	∞
5	$((0,\, \infty),\, 2)$	∞

$\mathbf{M}_1$

	d_1	d_2
1	∞	∞
2	$((1,\, \infty),\, 3)$	∞
3	∞	∞
4	∞	∞
5	$((1,\, \infty),\, 17)$	∞

$\mathbf{M}_2$

	d_1	d_2
1	∞	∞
2	$((2,\, 40),\, 0)$	∞
3	∞	∞
4	∞	$((2,\, 10),\, 0)$
5	$((2,\, 10),\, 0)$	$((2,\, 20),\, 0)$

We then construct a combined mapping matrix $\mathbf{M}$ by summing the individual matrices as $\mathbf{M} = \mathbf{M}_0 \,\vec{\oplus}\, \mathbf{M}_1 \,\vec{\oplus}\, \mathbf{M}_2$. The resulting mapping matrix is shown in Figure 5(b).

We define the OSPF semi-module as

$$\mathrm{OSPF}(U) = (W,\, \vec{\oplus},\, \triangleright_{\mathrm{snd}})$$

where

$$m \triangleright_{\mathrm{snd}} ((l,\, u),\, m') = ((l,\, u),\, m + m')$$
$$m \triangleright_{\mathrm{snd}} \infty = \infty.$$

The OSPF semi-module is a variant of the cold-potato semi-module; here, instead of combining routing data with the first component of the mapping information and using the right lexicographic order, we instead combine it with the second component and use the left lexicographic order. Hence we refer to the lifted multiplicative operator as $\triangleright_{\mathrm{cp}}$.

Figure 5(c) illustrates the resulting forwarding matrix, $\mathbf{F} = \mathbf{R} \triangleright_{\mathrm{cp}} \mathbf{M}$. For d_1, we see that the Type 0 route is given priority over the Type 1 route. In contrast, there are two Type 2 routes for d_2, and hence the bandwidth component is used as a tie-breaker.

3 Simple Route Redistribution

In this section we show how to allow forwarding between multiple domains by generalizing the import model from Section 2 (here, we limit ourselves to modelling the case where there are two routing domains, although it is possible to generalize to a greater number). In particular, we show how the forwarding matrix from one domain can be used within the mapping matrix of another. This models *redistribution*, where a routing solution from one routing protocol is used within another. Additionally, we demonstrate that it is possible for each routing/forwarding domain to use a different semiring/semi-module pair.

Begin by assuming that there are two routing domains, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Also, assume that there is a set of destinations, D, with V_1, V_2 and D pairwise disjoint. Let G_1 be connected to G_2 with the attachment arcs $E_{1,2} \subseteq V_1 \times V_2$, represented as the $V_1 \times V_2$ *bridging matrix* $\mathbf{B}_{1,2}$. Similarly, let G_2 be connected to D with the attachment arcs $E_{2,d} \subseteq V_2 \times D$, represented as the $V_2 \times D$ attachment matrix $\mathbf{M}_2$. Let $\mathbf{F}_2$ be the forwarding matrix for G_2. We demonstrate how to construct a forwarding matrix from V_1 to D.

We shall use Figure 6 as a running example. Figure 6(a) illustrates two graphs, G_1 and G_2. The second graph, G_2, is directly connected to destinations d_1 and d_2 and therefore we are able to compute the forwarding matrix for G_2 using the method from Section 2. We model the routing in G_2 using the bandwidth semiring $\mathrm{MaxMin} = (\mathbb{N}^\infty, \max, \min)$ and the forwarding using the cold-potato semi-module $\mathrm{Cold}(\mathrm{MaxMin}, \mathrm{Min})$, where $\mathrm{Min} = (\mathbb{N}^\infty, \min)$. The mapping matrix is given in Figure 6(b), whilst the routing and forwarding matrices are given in Figure 6(c) and Figure 6(d) respectively.

In order to compute a forwarding matrix from V_1 to D, we must first construct a mapping matrix $\mathbf{M}_1$ from V_1 to D by combining the forwarding matrix $\mathbf{F}_2$ from G_2 with the bridging matrix $\mathbf{B}_{1,2}$. Let the forwarding in G_2 be modelled using the semi-module $N_2 = (N_2, \square_2, \triangleright_2)$, and that the bridging matrix is modelled using the semigroup $(N_1, \square_1)$. We construct a *right* semi-module $(N_1, \square_1, \triangleleft_1)$ over N_2 i.e. with $\triangleleft_1 \in (N_2 \times N_1) \to N_1$. Then we compute the mapping matrix from G_1 as $\mathbf{M}_1 = \mathbf{B}_{1,2} \triangleleft_1 \mathbf{F}_2$.

Returning to Figure 6, the bridging matrix $\mathbf{B}_{1,2}$ is illustrated in Figure 6(e). We combine the forwarding matrix $\mathbf{F}_2$ with $\mathbf{B}_{1,2}$ using the right version of the semi-module

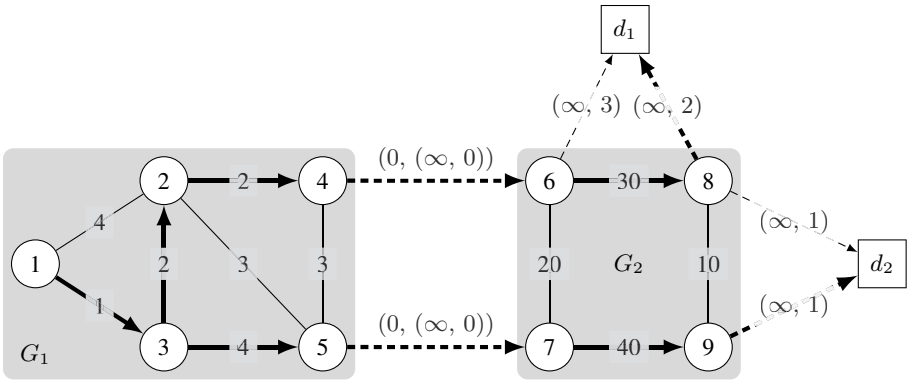
$$\mathrm{Hot}(\mathrm{MinPlus}, \mathrm{Cold}(\mathrm{MaxMin}, \mathrm{Min})).$$

The resulting mapping matrix $\mathbf{M}_1$ is illustrated in Figure 6(f).

Finally, we must combine the mapping matrix $\mathbf{M}_1$ with the routing solution $\mathbf{R}_1$. Suppose that $\mathbf{R}_1$ has been computed using the semiring S. Then we construct a *left* semi-module $(N_1, \square_1, \triangleright_1)$ over S. Compute the forwarding matrix for G_1 as

$$\mathbf{F}_1 = \mathbf{R}_1 \triangleright_1 \mathbf{M}_1 = \mathbf{R}_1 \triangleright_1 (\mathbf{B}_{1,2} \triangleleft_1 \mathbf{F}_2)$$

Hence we see that we have in fact used a pair of semi-modules with identical additive components: a left semi-module $(N_1, \square_1, \triangleleft_1)$ over N_2 and a right semi-module $(N_1, \square_1, \triangleright_1)$ over S.



(a) Multiple graphs with external information

$$\mathbf{M}_2 = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} (\infty, 3) & \infty \\ \infty & \infty \\ (\infty, 2) & (\infty, 1) \\ \infty & (\infty, 1) \end{bmatrix} \end{array}$$

(b) G_2 mapping matrix

$$\mathbf{R}_2 = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccc} 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & 20 & 30 & 20 \\ 20 & \infty & 20 & 40 \\ 30 & 20 & \infty & 20 \\ 20 & 40 & 20 & \infty \end{bmatrix} \end{array}$$

(c) G_2 routing matrix

$$\mathbf{F}_2 = \mathbf{R}_2 \rhd_{\mathrm{cp}} \mathbf{M}_2 = \begin{array}{c} \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} (30, 2) & (30, 1) \\ (20, 2) & (40, 1) \\ (\infty, 2) & (\infty, 1) \\ (20, 2) & (\infty, 1) \end{bmatrix} \end{array}$$

(d) G_2 forwarding matrix

$$\mathbf{B}_{1,2} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 6 & & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ (0, (\infty, 0)) & \infty & \infty & \infty \\ \infty & (0, (\infty, 0)) & \infty & \infty \end{bmatrix} \end{array}$$

(e) G_1 to G_2 bridging matrix

$$\mathbf{M}_1 = \mathbf{B}_{1,2} \lhd_{\mathrm{hp}} \mathbf{F}_2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} \infty & \infty \\ \infty & \infty \\ \infty & \infty \\ (0, (30, 2)) & (0, (30, 1)) \\ (0, (20, 2)) & (0, (40, 1)) \end{bmatrix} \end{array}$$

(f) G_1 mapping matrix

$$\mathbf{R}_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 1 & 5 & 5 \\ 3 & 0 & 2 & 2 & 3 \\ 1 & 2 & 0 & 4 & 4 \\ 5 & 2 & 4 & 0 & 3 \\ 5 & 3 & 4 & 3 & 0 \end{bmatrix} \end{array}$$

(g) G_1 routing matrix

$$\mathbf{F}_1 = \mathbf{R}_1 \rhd_{\mathrm{hp}} \mathbf{M}_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} (5, (30, 2)) & (5, (40, 1)) \\ (2, (30, 2)) & (2, (30, 1) \\ (4, (30, 2)) & (4, (40, 1) \\ (0, (30, 2)) & (0, (30, 1) \\ (0, (20, 2)) & (0, (40, 1) \end{bmatrix} \end{array}$$

(h) G_1 forwarding matrix

Fig. 6. Example of *simple route redistribution*

Completing the example of Figure 6, the routing matrix for G_1 is computed using the semiring $\mathrm{MinPlus}$. The resulting matrix $\mathbf{R}_1$ is shown in Figure 6(g). We combine $\mathbf{R}_1$ with the mapping matrix $\mathbf{M}_1$ using the left version of the semi-module

$$\mathrm{Hot}(\mathrm{MinPlus},\, \mathrm{Cold}(\mathrm{MaxMin}, \mathrm{Min})).$$

The resulting forwarding table $\mathbf{F}_1$ is given in Figure 6(h). The bold arrows in Figure 6(a) denote the forwarding paths from node 1 to destinations d_1 and d_2. Note that the two egress nodes (4 and 5) from G_1 are at identical distances from 1, and therefore the bandwidth components from G_2 are used as tie-breakers. This results in a different egress point for each destination.

This model is significant because it is the first algebraic account of route redistribution – an area that is currently treated as a 'black art' even within the Internet routing community (for example, there are only informal guidelines on how to avoid redistribution anomalies such as loops and oscillations). We hope that our approach can be generalized to provide a basis for understanding existing redistribution techniques, and also for developing new approaches to protocol inter-operations.

4 Related Work and Open Problems

4.1 Locators and Identifiers

Our term *mapping table* has been borrowed (slightly loosely) from recent work attempting to differentiate between infrastructure addresses (called locators) and end-user addresses (called identifiers), as with the Locator/ID Separation Protocol (LISP) [10]. This effort has been motivated by a perceived need to reduce the size of routing and forwarding tables in the Internet's backbone (the world of inter-domain routing [11]).

Restated in our abstract setting, the essential problem that LISP is attempting to solve is that a mapping table $\mathbf{M}$ may be many orders of magnitude larger than the routing table $\mathbf{R}$, leading to a very large forwarding table $\mathbf{F} = \mathbf{R} \triangleright \mathbf{M}$. Since there is no separation between mapping and routing today, such table growth is in fact becoming a real operational problem. (Note that we are using the terms *routing table* and *forwarding table* in a rather unconventional, network-wide, sense. In a distributed setting, the entries $\mathbf{F}(i,\ \_)$ make up the forwarding table at node i.)

LISP proposes that forwarding tables $\mathbf{F}$ be only partially constructed using an on-demand approach – an entry $\mathbf{F}(i,\ d)$ is not constructed until router i receives traffic destined for d. This in turn requires some type of distributed mapping service, for which there are several proposals currently under consideration.

In this paper we have used the separation of locators and identifiers to provide an algebraic view of the distinction between routing and forwarding tables. This model suggests that the Locator/ID split might be usefully applied to intra-domain routing.

4.2 Route Redistribution

Examples of somewhat *ad hoc* mechanisms and techniques added to routing are *route redistribution* for distributing routes between distinct routing protocols (as already discussed), and *administrative distance* (discussed below). Recent research has documented their widespread use and illustrated routing anomalies that can arise as a result [3,4,5]. From our point of view, that work represents a *bottom-up* approach that starts with the complex implementation details of current legacy software. We hope that we have initiated a complementary, *top-down*, approach. The algebraic model has the advantage of

making clear *what problem* is being solved, as distinct from *what algorithm* is being implemented to solve a problem. We assert that the Internet routing literature is severely hobbled by the way that these distinct issues are often hopelessly tangled together.

Our model suggests that new protocols should be designed with a clear distinction between routing, mapping, and forwarding. Furthermore, mechanisms for constructing mapping tables and forwarding tables should be elevated from proprietary implementations to first-class status and standardized.

4.3 Loss of Distributivity

We may attempt to solve the equation $\mathbf{F} = (\mathbf{A} \rhd \mathbf{F}) \square \mathbf{M}$ using an iterative method,

$$\begin{aligned} \mathbf{F}^{[0]} &= \mathbf{M}, \\ \mathbf{F}^{[k+1]} &= (\mathbf{A} \rhd \mathbf{F}^{[k]}) \square \mathbf{M}. \end{aligned}$$

When $\mathbf{A}^*$ exists, and $(N,\ \square,\ \rhd)$ is a semi-module over S, then it is not too hard to see that $\lim_{k \to \infty} \mathbf{F}^{[k]} = \mathbf{A}^* \rhd \mathbf{M}$. However, when modeling current Internet routing protocols several problems may be encountered.

The first is that S may not in fact be a semiring due to violations of the distributivity laws. The situation may not be as hopeless at it might seem. Recent research [12] has pointed out that distributivity that is so essential to semiring theory may have to be abandoned to model some types of Internet routing. Even without distributivity, it may be possible to use an iterative method to arrive at a (locally optimal) routing solution [13] .

On the other hand, it may be that S is a well-behaved semiring, but that the structure $(N,\ \square,\ \rhd)$ is not a semi-module over S. Again, this seems to arise with violations of semi-module distributivity. We conjecture that it would be straightforward to extend the results of [13] to iterations of $\mathbf{F}^{[k]}$. That is, that if (1) the natural order is a total order, (2) $m < \infty_N \implies m < a \rhd m$ and (3) only simple paths are considered, then the iterative method will converge to a (locally optimal) solution to the equation $\mathbf{F} = (\mathbf{A} \rhd \mathbf{F}) \square \mathbf{M}$.

The lack of a global notion of optimality may in fact be perfectly reasonable in the wide Internet where competing Internet Service Providers need to share routes but their local commercial relationships prevent agreement as to what represents a *best* route. However, we suspect that a less obvious source of this type of routing may have evolved with administrative distance.

4.4 Administrative Distance

Administrative distance [3,4,5] is used for determining which entries are placed in a forwarding table when distinct protocols running on the same router have routes to common destinations. Algebraic modeling of administrative distance remains open for a careful formal treatment.

One fundamental problem seems to be that capturing the way routers currently implement administrative distance leads to algebraic structures that are not distributive. In fact, we conjecture that the technique is inherently non-distributive in some sense.

If this is the case, then we can model current routing as implementing an iterative method attempting to find a fixed-point over a non-distributive structure. Perhaps the

way forward is again to elevate this procedure from proprietary code to the level of a first class protocol and design constraints sufficient to guarantee convergence.

Acknowledgements

We are grateful for the support of the Engineering and Physical Sciences Research Council (EPSRC, grant EP/F002718/1) and a grant from Cisco Systems. We also thank M. Abdul Alim, Arthur Amorim, Alexander Gurney, Vilius Naudžiūnas, Philip Taylor, and the conference reviewers for their helpful feedback.

References

1. Gondran, M., Minoux, M.: Graphs, Dioids, and Semirings: New Models and Algorithms. Springer, Heidelberg (2008)
2. Day, J.: Patterns in Network Architectures: A return to fundamentals. Prentice Hall, Englewood Cliffs (2008)
3. Le, F., Xie, G., Zhang, H.: Understanding route redistribution. In: Proc. Inter. Conf. on Network Protocols (2007)
4. Le, F., Xie, G., Pei, D., Wang, J., Zhang, H.: Shedding light on the glue logic of the internet routing architecture. In: Proc. ACM SIGCOMM (2008)
5. Le, F., Xie, G., Zhang, H.: Instability free routing: Beyond one protocol instance. In: Proc. ACM CoNext (December 2008)
6. Carré, B.: Graphs and Networks. Oxford University Press, Oxford (1979)
7. Rekhter, Y., Li, T.: A Border Gateway Protocol. RFC 1771 (BGP version 4) (March 1995)
8. Moy, J.: OSPF: Anatomy of an Internet Routing Protocol. Addison-Wesley, Reading (1998)
9. Moy, J.: OSPF version 2. RFC 2328 (1998)
10. Farinacci, D., Fuller, V., Meyer, D., Lewis, D.: Locator/ID separation protocol (LISP). draft-ietf-lisp-02.txt. Work In Progress (2009)
11. Halabi, S., McPherson, D.: Internet Routing Architectures, 2nd edn. Cisco Press (2001)
12. Sobrinho, J.L.: An algebraic theory of dynamic network routing. IEEE/ACM Transactions on Networking 13(5), 1160–1173 (2005)
13. Griffin, T.G., Gurney, A.J.T.: Increasing bisemigroups and algebraic routing. In: 10th International Conference on Relational Methods in Computer Science (RelMiCS10) (April 2008)
14. Gurney, A.J.T., Griffin, T.G.: Lexicographic products in metarouting. In: Proc. Inter. Conf. on Network Protocols (October 2007)

A Basic Definitions

Our definitions of semirings and semi-modules are fairly standard, taken from [1]. The definitions of lexicographic operations are from [14].

A.1 Semirings

A semiring is a structure $S = (S, \oplus, \otimes)$ where $(S, \oplus)$ is a commutative semigroup, $(S, \otimes)$ is semigroup, and the following conditions hold: (1) $\otimes$ distributes over $\oplus$,

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z), \quad (y \oplus z) \otimes x = (y \otimes x) \oplus (z \otimes x),$$

(2) there exists an identity for $\oplus$, $0_S \in S$, and an identity for $\otimes$, $1_S \in S$, and (3) it is assumed that 0_S is an annihilator for $\otimes$ i.e. $0_S \otimes x = x \otimes 0_S = 0_S$.

For routing, we are normally working with idempotent semirings where $x \oplus x = x$. In this case $(S, \oplus)$ is a semi-lattice, and we use one of the natural orders – $x \leq^L_\oplus y \equiv x = x \oplus y$ and $x \leq^R_\oplus y \equiv y = x \oplus y$. For routing, we will stick with the order $\leq^L_\oplus$ since it corresponds well with the notion of least cost paths. For this reason we use ∞_S rather than 0_S, since $x \leq^L_\oplus \infty_S$ for all $x \in S$.

A.2 Semi-modules

Let $S = (S, \oplus, \otimes)$ be a semiring. A (left) semi-module over S is a structure $N = (N, \square, \triangleright)$, where $(N, \square)$ is a commutative semigroup and $\triangleright$ is a function $\triangleright \in (S \times N) \to N$ that satisfies the following distributivity laws,

$$x \triangleright (m \square n) = (x \triangleright m) \square (x \triangleright n), \quad (x \oplus y) \triangleright m = (x \triangleright m) \square (y \triangleright m).$$

In addition, we assume that the identity for $\square$ exists, 0_N, and that $0_S \triangleright m = 0_N$, $x \triangleright 0_N = 0_N$, and $1_S \triangleright m = m$. Again, in our applications $\square$ is often idempotent, and we use the notation ∞_N rather than 0_N.

A.3 Lexicographic Product

Suppose that we have two semigroups $S = (S, \oplus_S)$ and $T = (T, \oplus_T)$, with S *selective* (i.e. for all $s_1, s_2 \in S$ we have $s_1 \oplus s_2 \in \{s_1, s_2\}$). Then the left lexicographic product of S and T is defined to be the semigroup $S \vec{\times} T = ((S \times T) \cup \{\infty\}, \vec{\oplus})$, where for $(s_1, t_1), (s_2, t_2) \in S \times T$ we have

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

The right lexicographic product $\overleftarrow{\times}$ is similar, except that the semigroup T is assumed to be selective and the order of comparison is reversed.