An Algebraic Approach to Internet Routing

Part III

Timothy G. Griffin

timothy.griffin@cl.cam.ac.uk
Computer Laboratory
University of Cambridge, UK

Departamento de Ingeniería Telemática
Escuela Politécnica Superior
Universidad Carlos III de Madrid
16, 17, 18 March, 2009
Outline for Wednesday

- A mini-metalanguage for routing algebras
- The Metarouting Toolkit (prototype)
- On algebraic metalanguage design
- Min-set constructions and multi-path routing
- A word about hot and cold potatoes
A simple grammar for a mini-metalanguage

Our mini-metalanguage will describe routing algebras

- \((S, \oplus, F \subseteq S \rightarrow S)\)
- \(\oplus\) is commutative, idempotent, and has identity \(\alpha\).

\[
\begin{align*}
\text{base} &::= \text{sp} \\
& \quad \mid \text{bw} \\
& \quad \mid \text{rel} \\
\text{algebra} &::= \text{term} \\
& \quad \mid \text{right} \text{term} \\
& \quad \mid \text{left} \text{term} \\
& \quad \mid \text{lex\_product} \text{term} \ldots \text{term} \\
& \quad \mid \text{function\_union} \text{term} \ldots \text{term} \\
\text{term} &::= \text{base} \\
& \quad \mid (\text{algebra})
\end{align*}
\]
The Semantics

For category base

- $[\text{sp}]^B = (\mathbb{N} \cup \{\infty\}, \min, F_+)$
- $[\text{bw}]^B = (\mathbb{N} \cup \{\infty\}, \max, F_{\min})$
- $[\text{rel}]^B = ([0, 1], \max, F_{\times})$

For category term

- $[b]^T = [b]^B$
- $[(a)]^T = [a]^A$
The Semantics

For category *algebra*

- \([t]^A = [t]^B\)
- \([\text{right } t]^A = (S, \oplus, \{i\})\)
  - where \([t]^T = (S, \oplus, F)\)
- \([\text{left } t]^A = (S, \oplus, K(S))\)
  - where \([t]^T = (S, \oplus, F)\)
The Semantics

\[ \text{lex\_product } t \] \( \bar{\mathcal{A}} \) = \( [t]^{\overline{T}} \)

\[ \text{lex\_product } t \ t' \] \( \bar{\mathcal{A}} \) = \((S, \oplus, F) \times (T, \circ, G) = (S \times T, \oplus \times \circ, F \times G) \)

- where \( [t]^{\overline{T}} = (S, \oplus, F) \)
- and \( [t']^{\overline{T}} = (T, \circ, G) \)

\[ \text{lex\_product } t \ t' \ldots t'' \] \( \bar{\mathcal{A}} \) = \((S, \oplus, F) \times (T, \circ, G) = (S \times T, \oplus \times \circ, F \times G) \)

- where \( [t]^{\overline{T}} = (S, \oplus, F) \)
- and \( \text{lex\_product } t' \ldots t'' \] \( \bar{\mathcal{A}} \) = \((T, \circ, G) \)
The Semantics

- \([\text{function}_\text{union} \, t]^A = [t]^T\)
- \([\text{function}_\text{union} \, t \, t']^A = (S, \oplus, F) +_m (S, \oplus, G) = (S, \oplus, F \cup G)\)
  - where \([t]^T = (S, \oplus, F)\)
  - and \([t']^T = (S, \oplus, G)\)
- \([\text{function}_\text{union} \, t \, t' \ldots t'']^A = (S, \oplus, F) +_m (S, \oplus, G) = (S, \oplus, F \cup G)\)
  - where \([t]^T = (S, \oplus, F)\)
  - and \([\text{function}_\text{union} \, t' \ldots t'']^A = (S, \oplus, G)\)
Some interesting properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M</strong></td>
<td>$\forall a, b \in S \forall f \in F : f(a \oplus b) = f(a) \oplus f(b)$</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>$\forall a, b \in S \forall f \in F - {\omega} : f(a) = f(b) \implies a = b$</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td>$\forall a, b \in S \forall f \in F : f(a) = f(b)$</td>
</tr>
<tr>
<td><strong>I</strong></td>
<td>$\forall a \in S \forall f \in F : a \neq \alpha \implies a &lt;_{L} f(a)$</td>
</tr>
<tr>
<td><strong>ND</strong></td>
<td>$\forall a \in S \forall f \in F : a \leq_{L} f(a)$</td>
</tr>
</tbody>
</table>
We know a few rules ...

(some of the) rules needed for global optimality

\[ M(\text{right}(S)) \]
\[ M(\text{left}(S)) \]
\[ C(\text{right}(S)) \]
\[ K(\text{left}(S))M(S \not
rightarrow T) \iff M(S) \land M(T) \land (C(S) \lor K(T)) \]
\[ M(S +_m T) \iff M(S) \land M(T) \]
... and a few more rules

(some of the) rules needed for local optimality (and for loop-freedom in next-hop forwarding)

\[ I(S \times T) \iff I(S) \lor (ND(S) \land I(T)) \]
\[ ND(S \times T) \iff I(S) \lor (ND(S) \land ND(T)) \]
\[ I(S +_m T) \iff I(S) \land I(T) \]
\[ ND(S +_m T) \iff ND(S) \land ND(T) \]
We can turn rules into bottom-up methods

Example: The $\iff$ rule

$$M(S \times T) \iff M(S) \land M(T) \land (C(S) \lor K(T))$$

becomes a bottom-up method for deriving property $M$ or property $\neg M$ for any expression

$$e = \text{lex_product } t_1 \ t_2$$
We know everything about our base algebras

<table>
<thead>
<tr>
<th></th>
<th>M</th>
<th>C</th>
<th>K</th>
<th>I</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>bw</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>rel</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Now, for each algebra expression $a$ defined by our mini-metalanguage and each property $P$, we can determine in a bottom-up manner whether

$$P([a]^A)$$

or

$$\neg P([a]^A)$$

holds.

No proofs required at algebra specification time!
### A few examples

<table>
<thead>
<tr>
<th>Function</th>
<th>M</th>
<th>C</th>
<th>K</th>
<th>I</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>lex_product sp bw</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>(lex_product sp sp</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>lex_product bw sp</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>lex_product rel bw</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>lex_product rel bw sp</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
- **Specification**: Algorithms are currently picked from a menu, while the routing language is specified in terms of the Routing Algebra Meta-Language (RAML).

- **Errors**: Each algorithm is associated with properties it requires of a routing language (Example: Dijkstra requires a total order on metrics). Properties are automatically derived from RAML expressions. An error is reported when there is a mismatch.
Meet the Metarouters!
From left to right ...

- Philip Taylor
  - Router configuration languages, vectoring protocols
- John Billings
  - Compilation, route redistribution, off-line algorithms
- M. Abdul Alim
  - Link state protocols, route redistribution
- Vilius Naudziunias
  - Automating theorem proving at system design-time
- Tim Griffin
  - Confusion
- Balraj Singh
  - Metaforwarding
- Alex Gurney
  - Algebraic theory
Our evolving metalanguage

- Our current metalanguage is much larger than the mini-metalanguage.
- Dozens of constructors, dozens of properties.
- Hundreds of rules.
  - Automating the tedium of specification correctness!
Let’s implement a simple scoped-product example

\[
<\text{edist}=3, \ \text{epath}=[\text{‘A’}], \ \text{idist}=7, \ \text{ipath}=[\text{‘X’, ‘Y’}]>
\]

```
route-policy
  set internal idist 30
  set internal ipath ‘X’
end-policy
```

```
route-policy
  set external edist 30
  set external epath ‘A’
  set external idist 1
  set external ipath empty
end-policy
```
The external algebra

let inter_region = lex_product
  <
edist    : lte_plus,  
epath     : simple_paths,  
idist     : left lte_plus,  
ipath     : left simple_paths
  >
The internal algebra

let intra_region =
  lex_product
  <
    edist : right lte_plus,
    epath : right simple_paths,
    idist : lte_plus,
    ipath : simple_paths
  >
let regions =
    function_union
<
    external : inter_region,
    internal : intra_region
>

The complete algebra
Example: regions

- Compile to C++
- Plug into e.g generalized BGP algorithm
- Deploy on routers
- Or create offline simulator
Example: generated code (you are not expected to understand it!)

```c
struct times_out
{
    ty11 operator()(ty7 node, ty12 export_, ty6 signature_outer_or_error)
    {
        ty11 var73;
        switch (signature_outer_or_error.tag_)
        {
            case ty11::CONST: break; // Const
            case ty11::REST:
            {
                ty5 signature(signature_outer_or_error.value_);
                ty11 var75;
                switch (export_.tag_)
                {
                    case 1:
                    {
                        Unit x2(*export_.v1_);
                        IntBigPos var80(signature.v1_);
                        String var83(node.v1_);
                        ty1 var85(signature.v2_);
                        ty1 var82(ListSimpCons()(var83, var85)); [...] 
                    } [...] 
                    var73 = var75;
                    break;
                }
                return var73;
            }
        }
    }
};
```
The metalanguage spans multiple classes of algebraic structures

The Quadrants

<table>
<thead>
<tr>
<th>NW</th>
<th>NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bisemigroups (S, ⊕, ⊗)</td>
<td>Order Semigroups (S, ≤, ⊗)</td>
</tr>
<tr>
<td>SW</td>
<td>SE</td>
</tr>
<tr>
<td>Semigroup Transforms (S, ⊕, F)</td>
<td>Order Transforms (S, ≤, F)</td>
</tr>
</tbody>
</table>
Moving around

Operations for translations between quadrants are in the metalanguage.

A few examples

\[ (S, \oplus, \otimes) \xrightarrow{\text{natord}} (S, \leq, \otimes) \]
\[ \text{cayley} \]

\[ (S, \oplus, F) \xrightarrow{\text{natord}} (S, \leq, F) \]
\[ \text{cayley} \]
Properties get dragged along

\[(a \neq 0 \implies a = a \oplus (b \otimes a)) \land \quad \text{natord} \quad \implies a \neq T \implies a < b \otimes a\]

\[(b \otimes a = a \oplus (b \otimes a) \implies a = 0)\]

\[
\text{cayley}
\]

\[(a \neq 0 \implies a = a \oplus f(a)) \land \quad \text{natord} \quad \implies a \neq T \implies a < f(a)\]

\[(f(a) = a \oplus f(a) \implies a = 0)\]

\[
\text{cayley}
\]

\[
\text{natord}
\]
New, experimental constructors for “min-sets”

- For explicit multi-path routing.

**Definition (First, Derived Order Relations)**

\[
\begin{align*}
a < b & \equiv a \preceq b \land \neg (a \preceq b) \quad \text{a is (strictly) less than b} \\
a \sim b & \equiv a \preceq b \land b \preceq a \quad \text{a is equivalent to b} \\
a \simeq b & \equiv a \preceq b \lor b \preceq a \quad \text{a is comparable with b} \\
a \# b & \equiv \neg (a \preceq b) \land \neg (b \preceq a) \quad \text{a is incomparable with b.}
\end{align*}
\]
Direct Product Order

**Definition (Direct Product)**

Let \((S, \preceq_S)\) and \((T, \preceq_T)\) be preordered sets. Then their direct product is denoted \((S, \preceq_S) \times (T, \preceq_T) = (S \times T, \preceq)\), where

\[(s_1, t_1) \preceq (s_2, t_2) \iff s_1 \preceq_S s_2 \land t_1 \preceq_T t_2.\]

**Lemma**

\[(a_1, b_1) \sim (a_2, b_2) \iff a_1 \sim_A a_2 \land b_1 \sim_B b_2\]

\[(a_1, b_1) \# (a_2, b_2) \iff \left(\begin{array}{c}
  a_1 \# a_2 \lor \\
  b_1 \# b_2 \lor \\
  (a_2 < a_1 \land b_1 < b_2) \lor \\
  (b_2 < b_1 \land a_1 < a_2)
\end{array}\right)\]
Direct product example

\[
D \times 1 = (D, 1) = (D, 0)\]

\[
B \times 1 = (B, 1) = (B, 0)\]

\[
A \times 1 = (A, 1) = (A, 0)\]

\[
C \times 1 = (C, 1) = (C, 0)\]
Lexicographic Product Order

Definition (Lexicographic Product)

Let \((S, \preceq_S)\) and \((T, \preceq_T)\) be preordered sets. Then their \textbf{Lexicographic product} is denoted \((S, \preceq_S) \times (T, \preceq_T) = (S \times T, \preceq),\) where

\[(s_1, t_1) \preceq (s_2, t_2) \iff s_1 \prec_S s_2 \lor (s_1 \sim_S s_2 \land t_1 \preceq_T t_2).\]

Lemma

\[(a_1, b_1) \sim (a_2, b_2) \iff a_1 \sim_A a_2 \land b_1 \sim_B b_2\]
\[(a_1, b_1) \not\!\sim (a_2, b_2) \iff a_1 \not\!\sim_A a_2 \lor (a_1 \sim_A a_2 \land b_1 \not\!\sim_B b_2).\]
Lexicographic product example

\[ A \times B \times C \times D = \]

\[ \begin{align*}
(D,1) \\
(D,0)
\end{align*} \]

\[ \begin{align*}
(C,1) \\
(C,0)
\end{align*} \]

\[ \begin{align*}
(A,1) \\
(A,0)
\end{align*} \]

\[ \begin{align*}
(B,1) \\
(B,0)
\end{align*} \]
Minimal Sets

**Definition (Min-sets)**

Suppose that \((S, \preceq)\) is a pre-ordered set. Let \(A \subseteq S\) be finite. Define

\[
\min_{\preceq}(A) \equiv \{ a \in A \mid \forall b \in A : \neg (b < a) \}
\]

\[
P(S, \preceq) \equiv \{ A \subseteq S \mid A \text{ is finite and } \min_{\preceq}(A) = A \}
\]

**Definition (Min-Set Semigroup)**

Suppose that \((S, \preceq)\) is a pre-ordered set. Then

\[
P_{\min}^\cup(S, \preceq) = (P(S, \preceq), \oplus_{\min})
\]

is the semigroup where

\[
A \oplus_{\min} B \equiv \min_{\preceq}(A \cup B).
\]
Min-Set-Map construction

Definition

Suppose that $S = (S, \preceq, F)$ a routing algebra in the style of Sobrinho [Sob03, Sob05]. Then

$$\text{minsetmap}(S) \equiv (\mathcal{P}(S, \preceq), \oplus_{\text{min}}, F_{\text{min}})$$

where $F_{\text{min}} \preceq = \{g_f \mid f \in F\}$ and

$$g_f(A) \equiv \min \preceq(\{f(a) \mid a \in A\}).$$
Let’s turn to BGP MED’s — First, hot potato
Cold Potato

The (4) represents a MED value.
The System MED-EVIL [MGWR02, Sys].

The values (0) and (1) represent MED values sent by AS 4. The other values are IGP link weights.
Best route selection at nodes \( A \) and \( B \).

- \( r_C, r_D \) and \( r_E \) denote routes received from routers C, D, and E, respectively.
- \( A \) receives route \( r_E \) through route reflector \( B \).
- \( B \) receives routes \( r_C \) and \( r_D \) through route reflector \( A \).

<table>
<thead>
<tr>
<th>( u )</th>
<th>( S )</th>
<th>BGP best of ( S ) at ( u )</th>
<th>due to</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>( {r_C, r_D} )</td>
<td>( r_D )</td>
<td>IGP</td>
</tr>
<tr>
<td>( A )</td>
<td>( {r_D, r_E} )</td>
<td>( r_E )</td>
<td>MED</td>
</tr>
<tr>
<td>( A )</td>
<td>( {r_E, r_C} )</td>
<td>( r_C )</td>
<td>IGP</td>
</tr>
<tr>
<td>( A )</td>
<td>( {r_C, r_D, r_E} )</td>
<td>( r_C )</td>
<td>MED, IGP</td>
</tr>
<tr>
<td>( B )</td>
<td>( {r_D, r_E} )</td>
<td>( r_E )</td>
<td>MED</td>
</tr>
<tr>
<td>( B )</td>
<td>( {r_E, r_C} )</td>
<td>( r_C )</td>
<td>IGP</td>
</tr>
</tbody>
</table>
There is not stable routing!

Assume $A$ always has routes $r_C$ and $r_D$, so only two cases:

- $A$ knows the routes $\{r_C, r_D, r_E\}$ and so selects $r_C$. This implies that $B$ has chosen $r_E$, and this is a contradiction, since $B$ would have $\{r_E, r_C\}$ and select $r_C$.

- $A$ has only $\{r_C, r_D\}$ and selects $r_D$. Since $A$ does not learn a route from $B$, we know that $B$ must have selected $r_C$. This is a contradiction since $B$ would learn $r_D$ from $A$ and then pick $r_E$. 
What’s going on with MED?

- Assume MEDs are represented by pairs of the form \((a, m)\), where \(a\) is an ASN and \(m\) is an integer metric.
- The partial order on MEDs is defined as

\[(\alpha_1, m) \preceq_M (\alpha_2, n) \equiv \alpha_1 = \alpha_2 \land m \preceq n.\]

- We can think abstractly of BGP routes as elements of

\[(P, \preceq_P) \times (M, \preceq_M) \times (S, \preceq_S),\]

where \((P, \preceq_P)\) represents the prefix of attributes considered before MED, and \((S, \preceq_S)\) represents the suffix of attributes considered after MED.
What is going on?

Suppose that we have the lexicographic product,

\[(A, \preceq_A) \times (B, \preceq_B) \equiv (A \times B, \preceq),\]

and that \(W\) is a finite subset of \(A \times B\). We would like to explore efficient (and correct) methods for computing the min-set \(\min_{\preceq}(W)\).

Let \(\sim_A\) and \(\sim_B\) be the preorders on \(A\) and \(B\) for which all elements are related.

Pipeline method

We say the pipeline method is correct when

\[
\min_{\preceq_A \times \preceq_B} (W) = \min_{\sim_A \times \sim_B} (\min_{\preceq_A \times \sim_B} (W)).
\]
Claim

The pipeline method is correct if and only if no two elements of $B$ are strictly ordered, or no two elements of $A$ are incomparable.

Proof: For the interesting direction, suppose that $A$ does contain two elements $a_1$ and $a_2$ with $a_1 \nsubseteq a_2$, and $B$ does contain two elements $b_1$ and $b_2$ with $b_1 <_B b_2$. Then

$$\min_{\preceq_A \times \preceq_B} \{(a_1, b_1), (a_2, b_2)\} = \{(a_1, b_1), (a_2, b_2)\}$$

but

$$\min_{\omega_A \times \preceq_B} (\min_{\preceq_A \times \omega_B} \{(a_1, b_1), (a_2, b_2)\})$$

$$= \min_{\omega_A \times \preceq_B} \{(a_1, b_1), (a_2, b_2)\}$$

$$= \{(a_1, b_1)\}.$$


Cisco Systems.
Endless BGP convergence problem in Cisco IOS software releases.
Field Note, October 10 2001,