

An Algebraic Approach to Internet Routing

Lectures 15 — 16

Timothy G. Griffin

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

Michaelmas Term
2009

Defining and implementing a new routing protocol is difficult!

- The space is large
- The proofs are difficult
- Correctness conditions hard to get right

Could the design process be partially automated?

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 α .

```
base ::= sp  
      | bw  
      | rel
```

```
algebra ::= term  
          | right term  
          | left term  
          | lex_product term ... term  
          | function_union term ... term
```

```
term ::= base  
        | (algebra)
```

The Semantics

For category *base*

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

For category *term*

- $\llbracket \mathbf{b} \rrbracket^{\mathcal{T}} = \llbracket \mathbf{b} \rrbracket^{\mathcal{B}}$
- $\llbracket (\mathbf{a}) \rrbracket^{\mathcal{T}} = \llbracket \mathbf{a} \rrbracket^{\mathcal{A}}$

The Semantics

For category *algebra*

- $\llbracket t \rrbracket^{\mathcal{A}} = \llbracket t \rrbracket^{\mathcal{B}}$
- $\llbracket \text{right } t \rrbracket^{\mathcal{A}} = (\mathcal{S}, \oplus, \{i\})$
 - ▶ where $\llbracket t \rrbracket^{\mathcal{T}} = (\mathcal{S}, \oplus, F)$
- $\llbracket \text{left } t \rrbracket^{\mathcal{A}} = (\mathcal{S}, \oplus, K(\mathcal{S}))$
 - ▶ where $\llbracket t \rrbracket^{\mathcal{T}} = (\mathcal{S}, \oplus, F)$

The Semantics

- $\llbracket \text{lex_product } t \rrbracket^A = \llbracket t \rrbracket^T$
- $\llbracket \text{lex_product } t \ t' \rrbracket^A = (\mathbf{S}, \oplus, \mathbf{F}) \vec{\times} (T, \odot, \mathbf{G}) = (\mathbf{S} \times T, \oplus \vec{\times} \odot, \mathbf{F} \times \mathbf{G})$
 - ▶ where $\llbracket t \rrbracket^T = (\mathbf{S}, \oplus, \mathbf{F})$
 - ▶ and $\llbracket t' \rrbracket^T = (T, \odot, \mathbf{G})$
- $\llbracket \text{lex_product } t \ t' \dots t'' \rrbracket^A = (\mathbf{S}, \oplus, \mathbf{F}) \vec{\times} (T, \odot, \mathbf{G}) = (\mathbf{S} \times T, \oplus \vec{\times} \odot, \mathbf{F} \times \mathbf{G})$
 - ▶ where $\llbracket t \rrbracket^T = (\mathbf{S}, \oplus, \mathbf{F})$
 - ▶ and $\llbracket \text{lex_product } t' \dots t'' \rrbracket^A = (T, \odot, \mathbf{G})$

The Semantics

- $\llbracket \text{function_union } t \rrbracket^A = \llbracket t \rrbracket^T$
- $\llbracket \text{function_union } t \ t' \rrbracket^A = (\mathcal{S}, \oplus, F) +_m (\mathcal{S}, \oplus, G) = (\mathcal{S}, \oplus, F \cup G)$
 - ▶ where $\llbracket t \rrbracket^T = (\mathcal{S}, \oplus, F)$
 - ▶ and $\llbracket t' \rrbracket^T = (\mathcal{S}, \oplus, G)$
- $\llbracket \text{function_union } t \ t' \ \dots \ t'' \rrbracket^A = (\mathcal{S}, \oplus, F) +_m (\mathcal{S}, \oplus, G) = (\mathcal{S}, \oplus, F \cup G)$
 - ▶ where $\llbracket t \rrbracket^T = (\mathcal{S}, \oplus, F)$
 - ▶ and $\llbracket \text{function_union } t' \ \dots \ t'' \rrbracket^A = (\mathcal{S}, \oplus, G)$

Some interesting properties

Property	Definition
M	$\forall a, b \in S \forall f \in F : f(a \oplus b) = f(a) \oplus f(b)$
C	$\forall a, b \in S \forall f \in F - \{\omega\} : f(a) = f(b) \implies a = b$
K	$\forall a, b \in S \forall f \in F : f(a) = f(b)$
I	$\forall a \in S \forall f \in F : a \neq \alpha \implies a <_{\oplus}^L f(a)$
ND	$\forall a \in S \forall f \in F : a \leq_{\oplus}^L f(a)$

We know a few rules ...

(some of the) rules needed for global optimality

$$M(\mathbf{right}(S))$$

$$M(\mathbf{left}(S))$$

$$C(\mathbf{right}(S))$$

$$K(\mathbf{left}(S))M(S \overset{\rightarrow}{\times} T) \iff M(S) \wedge M(T) \wedge (C(S) \vee K(T))$$

$$M(S +_m T) \iff M(S) \wedge M(T)$$

... and a few more rules

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

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

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

$$I(\mathbf{S} +_m \mathbf{T}) \iff I(\mathbf{S}) \wedge I(\mathbf{T})$$

$$\text{ND}(\mathbf{S} +_m \mathbf{T}) \iff \text{ND}(\mathbf{S}) \wedge \text{ND}(\mathbf{T})$$

We can turn rules into bottom-up methods

Example : The \iff rule

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

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

$$e = \text{lex_product } t_1 \ t_2$$

if derive properties for t_1	and derive properties for t_2	then derive property for e
M, C	M	M
M	M, K	M
$\neg M$	$\neg M$	$\neg M$
$\neg C$	$\neg K$	$\neg M$

Magic

We know everything about our base algebras

	M	C	K	I	ND
sp	yes	yes	no	yes	yes
bw	yes	no	no	no	yes
rel	yes	yes	no	no	yes

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

$$P(\llbracket a \rrbracket^A)$$

or

$$\neg P(\llbracket a \rrbracket^A)$$

holds.

No proofs required at algebra specification time!

A few examples

	M	C	K	I	ND
lex_product sp bw	yes	no	no	yes	yes
lex_product sp sp	yes	yes	no	yes	yes
lex_product bw sp	no	no	no	yes	yes
lex_product rel bw	yes	no	no	no	yes
lex_product rel bw sp	yes	no	no	yes	yes

BGP-like Example

```
function_union
<internal:
  lex_product
  < ecomm: right cpp,
    epath: right paths,
    idist: sp,
    ipath: paths >,
external:
  lex_product
  < ecomm: cpp,
    epath: paths,
    idist: left sp,
    ipath: left paths > >
```

Extracted “Code”

$$\begin{aligned} & \text{import on internal} \\ & \text{inl}(\perp, \perp, v, (i, j)) \triangleright (ec, ep, d, ip) \\ = & (ec, ep, v + d, (i, j)) \triangleright_{\text{paths}} l \end{aligned}$$

$$\begin{aligned} & \text{import on external} \\ & \text{inr}(x, (m, n), v, l) \triangleright (ec, ep, d, ip) \\ = & (x \triangleright_{\text{cpp}} ec, (m, n)) \triangleright_{\text{paths}} ep, v, l \end{aligned}$$