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 α.

$$
\begin{array}{rcl}
\textit{base} & ::= & \texttt{sp} \\
 & \mid & \texttt{bw} \\
 & \mid & \texttt{rel} \\
 & & \\
\textit{algebra} & ::= & \textit{term} \\
 & \mid & \texttt{right}\ \textit{term} \\
 & \mid & \texttt{left}\ \textit{term} \\
 & \mid & \texttt{lex\_product}\ \textit{term} \ldots \textit{term} \\
 & \mid & \texttt{function\_union}\ \textit{term} \ldots \textit{term} \\
 & & \\
\textit{term} & ::= & \textit{base} \\
 & \mid & (\textit{algebra})
\end{array}
$$

The Semantics

For category *base*

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

For category *term*

- $[\![\mathtt{b}]\!]^{\mathcal{T}} = [\![\mathtt{b}]\!]^{\mathcal{B}}$
- $[\![(\mathtt{a})]\!]^{\mathcal{T}} = [\![\mathtt{a}]\!]^{\mathcal{A}}$

The Semantics

For category *algebra*

- $[\![\mathtt{t}]\!]^{\mathcal{A}} = [\![\mathtt{t}]\!]^{\mathcal{B}}$
- $[\![\mathtt{right\ t}]\!]^{\mathcal{A}} = (S,\ \oplus,\ \{i\})$
  - where $[\![\mathtt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$
- $[\![\mathtt{left\ t}]\!]^{\mathcal{A}} = (S,\ \oplus,\ K(S))$
  - where $[\![\mathtt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$

The Semantics

- $[\![\texttt{lex\_product t}]\!]^{\mathcal{A}} = [\![\texttt{t}]\!]^{\mathcal{T}}$
- $[\![\texttt{lex\_product t t}']\!]^{\mathcal{A}} = (S,\ \oplus,\ F)\ \vec{\times}\ (T,\ \odot,\ G) = (S \times T,\ \oplus\ \vec{\times}\ \odot,\ F \times G)$
  - where $[\![\texttt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$
  - and $[\![\texttt{t}']\!]^{\mathcal{T}} = (T, \odot\ \ G)$
- $[\![\texttt{lex\_product t t}' \ldots \texttt{t}'']\!]^{\mathcal{A}} = (S,\ \oplus,\ F)\ \vec{\times}\ (T,\ \odot,\ G) = (S \times T,\ \oplus\ \vec{\times}\ \odot,\ F \times G)$
  - where $[\![\texttt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$
  - and $[\![\texttt{lex\_product t}' \ldots \texttt{t}'']\!]^{\mathcal{A}} = (T, \odot\ \ G)$

The Semantics

- $[\![\texttt{function\_union t}]\!]^{\mathcal{A}} = [\![\texttt{t}]\!]^{\mathcal{T}}$
- $[\![\texttt{function\_union t t}']\!]^{\mathcal{A}} = (S,\ \oplus,\ F) +_{\mathrm{m}} (S,\ \oplus,\ G) = (S,\ \oplus,\ F \cup G)$
  - where $[\![\texttt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$
  - and $[\![\texttt{t}']\!]^{\mathcal{T}} = (S, \oplus\ \ G)$
- $[\![\texttt{functon\_union t t}' \ldots \texttt{t}'']\!]^{\mathcal{A}} = (S,\ \oplus,\ F) +_{\mathrm{m}} (S,\ \oplus,\ G) = (S,\ \oplus,\ F \cup G)$
  - where $[\![\texttt{t}]\!]^{\mathcal{T}} = (S,\ \oplus,\ F)$
  - and $[\![\texttt{functon\_union t}' \ldots \texttt{t}'']\!]^{\mathcal{A}} = (S, \oplus\ \ G)$

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	$\forall a \in S\ \forall f \in F : a \neq \alpha \implies a <_{\oplus}^{\mathrm{L}} f(a)$
ND	$\forall a \in S\ \forall f \in F : a \leq_{\oplus}^{\mathrm{L}} f(a)$

We know a few rules ...

(some of the) rules needed for global optimality

$$
\begin{array}{rcl}
\mathrm{M}(\mathbf{right}(S)) & & \\
\mathrm{M}(\mathbf{left}(S)) & & \\
\mathrm{C}(\mathbf{right}(S)) & & \\
\mathrm{K}(\mathbf{left}(S))\mathrm{M}(S \vec{\times} T) & \iff & \mathrm{M}(S) \wedge \mathrm{M}(T) \wedge (\mathrm{C}(S) \vee \mathrm{K}(T)) \\
\mathrm{M}(S +_{\mathrm{m}} T) & \iff & \mathrm{M}(S) \wedge \mathrm{M}(T)
\end{array}
$$

... and a few more rules

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

$$\begin{aligned}
\mathrm{I}(S \vec{\times} T) &\iff \mathrm{I}(S) \vee (\mathrm{ND}(S) \wedge \mathrm{I}(T)) \\
\mathrm{ND}(S \vec{\times} T) &\iff \mathrm{I}(S) \vee (\mathrm{ND}(S) \wedge \mathrm{ND}(T)) \\
\mathrm{I}(S +_{\mathrm{m}} T) &\iff \mathrm{I}(S) \wedge \mathrm{I}(T) \\
\mathrm{ND}(S +_{\mathrm{m}} T) &\iff \mathrm{ND}(S) \wedge \mathrm{ND}(T)
\end{aligned}$$

We can turn rules into bottom-up methods

Example : The $\iff$ rule

$$\textsc{m}(S \vec{\times} T) \iff \textsc{m}(S) \wedge \textsc{m}(T) \wedge (\textsc{c}(S) \vee \textsc{k}(T))$$

becomes a bottom-up method for deriving property $\textsc{m}$ or property $\neg\textsc{m}$ 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
¬M		¬M
	¬M	¬M
¬C	¬K	¬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

$$\mathsf{P}([\![a]\!]^{\mathcal{A}})$$

or

$$\neg\mathsf{P}([\![a]\!]^{\mathcal{A}})$$

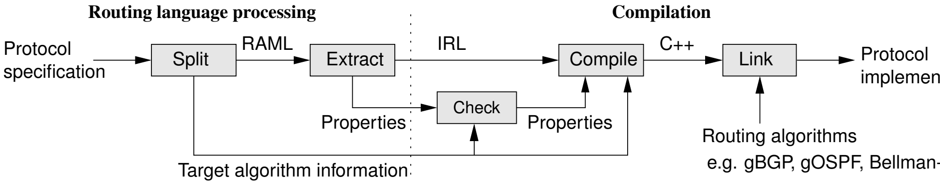
holds.

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

(Prototype) Metarouting System



- 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 mis-match.

Meet the Metarouters!

From left to right ...

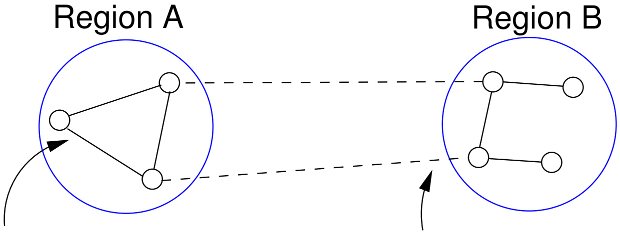
- Philip Taylor
  - Router configuration languages, vectoring protocols
- John Billings
  - Compilation, route redistribution, off-line algorithms
- M. Abdul Alim
  - Link state protocols, route redistribution
- Vilius Naudziunas
  - Automating theorem proving at system design-time
- Tim Griffin
  - 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

```
<edist=3, epath=['A'], idist=7, ipath=['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
   >
```

The complete algebra

```
let regions =
   function_union
   <
     external : inter_region,
     internal : intra_region
   >
```

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!)

```
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

NW	NE
Bisemigroups	Order Semigroups
$(S, \oplus, \otimes)$	$(S, \leq, \otimes)$
SW	**SE**
Semigroup Transforms	Order Transforms
$(S, \oplus, F)$	$(S, \leq, F)$

Moving around

Operations for translations between quadrants are in the metalanguage.

A few examples

$$\begin{array}{ccc} (S,\ \oplus,\ \otimes) & \xmapsto{\text{natord}} & (S,\ \leq,\ \otimes) \\ \Big\downarrow \text{cayley} & & \Big\downarrow \text{cayley} \\ (S,\ \oplus,\ F) & \xmapsto{\text{natord}} & (S,\ \leq,\ F) \end{array}$$

Properties get dragged along

$$
\begin{tikzcd}
\begin{array}{c}(a \neq 0 \implies a = a \oplus (b \otimes a)) \wedge \\ (b \otimes a = a \oplus (b \otimes a) \implies a = 0)\end{array} \arrow[r, mapsto, "\text{natord}"] \arrow[d, mapsto, "\text{cayley}"] & a \neq \top \implies a < b \otimes a \arrow[d, mapsto, "\text{cayley}"] \\
\begin{array}{c}(a \neq 0 \implies a = a \oplus f(a)) \wedge \\ (f(a) = a \oplus f(a) \implies a = 0)\end{array} \arrow[r, mapsto, "\text{natord}"] & a \neq \top \implies a < f(a)
\end{tikzcd}
$$

New, experimental constuctors for “min-sets”

- For explicit multi-path routing.

Definition (First, Derived Order Relations)

$$
\begin{array}{rcll}
a < b & \equiv & a \lesssim b \wedge \neg(a \lesssim b) & a \text{ is (strictly) less than } b \\
a \sim b & \equiv & a \lesssim b \wedge b \lesssim a & a \text{ is equivalent to } b \\
a \asymp b & \equiv & a \lesssim b \vee b \lesssim a & a \text{ is comparable with } b \\
a \sharp b & \equiv & \neg(a \lesssim b) \wedge \neg(b \lesssim a) & a \text{ is incomparable with } b.
\end{array}
$$

Direct Product Order

Definition (Direct Product)

Let $(S, \lesssim_S)$ and $(T, \lesssim_T)$ be preordered sets. Then their direct product is denoted $(S, \lesssim_S) \times (T, \lesssim_T) = (S \times T, \lesssim)$, where

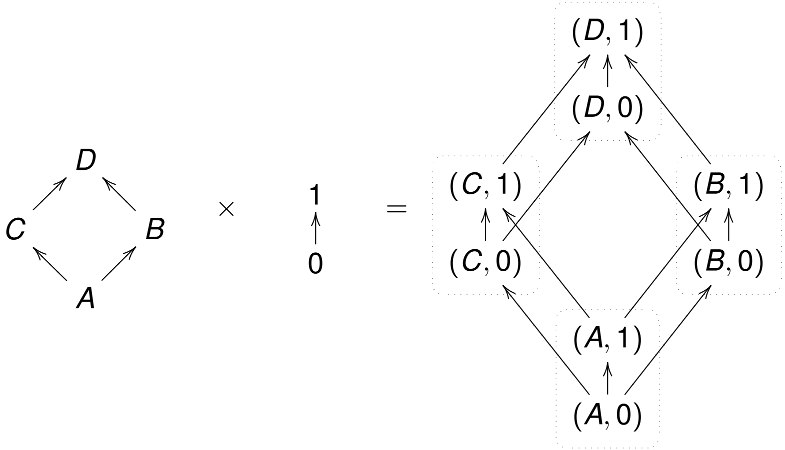
$$(s_1, t_1) \lesssim (s_2, t_2) \iff s_1 \lesssim_S s_2 \wedge t_1 \lesssim_T t_2.$$

Lemma

$$(a_1, b_1) \sim (a_2, b_2) \iff a_1 \sim_A a_2 \wedge b_1 \sim_B b_2$$

$$(a_1, b_1) \sharp (a_2, b_2) \iff \left(\begin{array}{l} a_1 \sharp a_2 \vee \\ b_1 \sharp b_2 \vee \\ (a_2 < a_1 \wedge b_1 < b_2) \vee \\ (b_2 < b_1 \wedge a_1 < a_2) \end{array} \right)$$

Direct product example



Lexicographic Product Order

Definition (Lexicographic Product)

Let $(S, \lesssim_S)$ and $(T, \lesssim_T)$ be preordered sets. Then their Lexicographic product is denoted $(S, \lesssim_S) \vec{\times} (T, \lesssim_T) = (S \times T, \lesssim)$, where

$$(s_1, t_1) \lesssim (s_2, t_2) \iff s_1 <_S s_2 \vee (s_1 \sim_S s_2 \wedge t_1 \lesssim_T t_2).$$

Lemma

$$\begin{aligned} (a_1, b_1) \sim (a_2, b_2) &\iff a_1 \sim_A a_2 \wedge b_1 \sim_B b_2 \\ (a_1, b_1) \sharp (a_2, b_2) &\iff a_1 \sharp_A a_2 \vee (a_1 \sim_A a_2 \wedge b_1 \sharp_B b_2). \end{aligned}$$

Lexicographic product example

$$\begin{tikzcd} & D & \\ C \arrow[ur] & & B \arrow[ul] \\ & A \arrow[ul] \arrow[ur] & \end{tikzcd} \quad \vec{\times} \quad \begin{tikzcd} 1 \\ 0 \arrow[u] \end{tikzcd} \quad = \quad \begin{tikzcd} & (D,1) & \\ & (D,0) \arrow[u] & \\ (C,1) \arrow[ur] & & (B,1) \arrow[ul] \\ (C,0) \arrow[u] & & (B,0) \arrow[u] \\ & (A,1) \arrow[ul] \arrow[ur] & \\ & (A,0) \arrow[u] & \end{tikzcd}$$

Minimal Sets

Definition (Min-sets)

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

$$\begin{array}{rcl} \min_{\lesssim}(A) & \equiv & \{a \in A \mid \forall b \in A : \neg(b < a)\} \\ \mathcal{P}(S, \lesssim) & \equiv & \{A \subseteq S \mid A \text{ is finite and } \min_{\lesssim}(A) = A\} \end{array}$$

Definition (Min-Set Semigroup)

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

$$\mathcal{P}^{\cup}_{\min}(S, \lesssim) = (\mathcal{P}(S, \lesssim), \oplus^{\lesssim}_{\min})$$

is the semigroup where

$$A \oplus^{\lesssim}_{\min} B \equiv \min_{\lesssim}(A \cup B).$$

Min-Set-Map construction

Definition

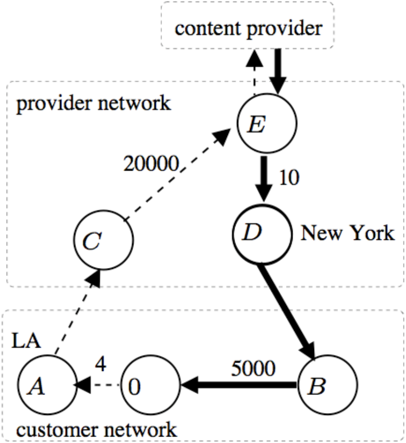
Suppose that $S = (S,\ \lesssim,\ F)$ a routing algebra in the style of Sobrinho [Sob03, Sob05]. Then

$$\mathbf{minsetmap}(S) \equiv (\mathcal{P}(S,\ \lesssim),\ \oplus^{\lesssim}_{\text{min}},\ F^{\lesssim}_{\text{min}})$$

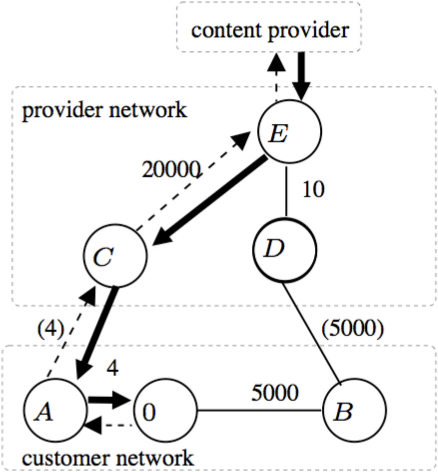
where $F^{\lesssim}_{\text{min}} = \{g_f \mid f \in F\}$ and

$$g_f(A) \equiv \min_{\lesssim}(\{f(a) \mid a \in A\}).$$

Let's turn to BGP MED's — First, hot potato

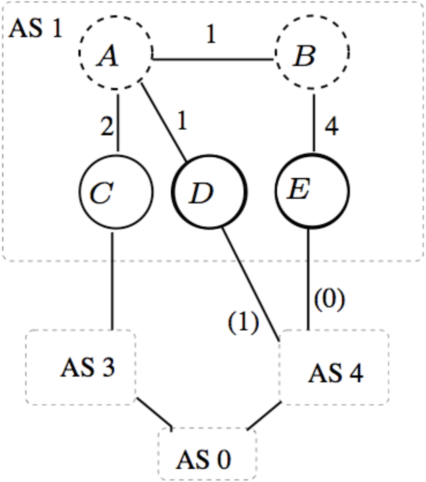


Cold Potato



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

u	S	BGP best of S at u	due to
A	$\{r_C, r_D\}$	r_D	IGP
A	$\{r_D, r_E\}$	r_E	MED
A	$\{r_E, r_C\}$	r_C	IGP
A	$\{r_C, r_D, r_E\}$	r_C	MED, IGP
B	$\{r_D, r_E\}$	r_E	MED
B	$\{r_E, r_C\}$	r_C	IGP

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

$$(\alpha_1,\ m) \lesssim_M (\alpha_2,\ n) \equiv \alpha_1 = \alpha_2 \wedge m \lesssim n.$$

- We can think abstractly of BGP routes as elements of

$$(P,\ \lesssim_P) \vec{\times} (M,\ \lesssim_M) \vec{\times} (S,\ \lesssim_S),$$

where $(P,\ \lesssim_P)$ represents the *prefix* of attributes considered before MED, and $(S,\ \lesssim_S)$ represents the *suffix* of attributes considered after MED.

What is going on?

Suppose that we have the lexicographic product,

$$(A, \lesssim_A) \vec{\times} (B, \lesssim_B) \equiv (A \times B, \lesssim),$$

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 $\min_{\lesssim}(W)$.

- 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

$$\min_{\lesssim_A \vec{\times} \lesssim_B}(W) = \min_{\sim_A \vec{\times} \lesssim_B}(\min_{\lesssim_A \vec{\times} \sim_B}(W)).$$

Pipeline

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

$$\min_{\lesssim_A \vec{\times} \lesssim_B} \{(a_1, b_1), (a_2, b_2)\} = \{(a_1, b_1), (a_2, b_2)\}$$

but

$$\begin{aligned}
& \min_{\omega_A \times \lesssim_B} \left(\min_{\lesssim_A \times \omega_B} \{(a_1, b_1), (a_2, b_2)\} \right) \\
= & \min_{\omega_A \times \lesssim_B} \{(a_1, b_1), (a_2, b_2)\} \\
= & \{(a_1, b_1)\}.
\end{aligned}$$

Bibliography I

[MGWR02] D. McPherson, V. Gill, D. Walton, and A. Retana.
BGP persistent route oscillation condition.
Internet Draft
`draft-ietf-idr-route-oscillation-01.txt` ,
Work In Progress, 2002.

[Sob03] Joao Luis Sobrinho.
Network routing with path vector protocols: Theory and applications.
In *Proc. ACM SIGCOMM*, September 2003.

[Sob05] Joao Luis Sobrinho.
An algebraic theory of dynamic network routing.
IEEE/ACM Transactions on Networking, 13(5):1160–1173, October 2005.

Bibliography II

[Sys] Cisco Systems.
Endless BGP convergence problem in Cisco IOS software releases.
Field Note, October 10 2001,
http://www.cisco.com/warp/public/770/fn12942.html.