Query Language Semantics and Compilation

October 20, 2015

1 Syntax

Naturals, fields, and packets

$$
\begin{aligned}
n &::= 0|1|2|\dots \\
f &::= f_1|\cdots|f_k \\
pk &::= \{f_1 = n_1, \dots, f_k = n_k\} \\
net &::= (in, out, p, t)
\end{aligned}
$$

Predicates

$$
\begin{aligned}
a, b ::= &\ 1 \\
&|0 \\
&|f = n \\
&|a \vee b \\
&|a \wedge b \\
&|\neg a
\end{aligned}
$$

Queries

$$
\begin{aligned}
q, q' ::= &\ (a, b) \\
&|q + q' \\
&|q \cdot q' \\
&|q^*
\end{aligned}
$$

2 Semantics

Language interpretation of NetKAT This is similar to R in the original NetKAT paper.

$$R(p) = \text{range}([\![\text{p}]\!])$$

Language interpretation of query language Recall that a network topology t has the following form

$$\text{dup} \cdot [sw_1 : pt_1] \to [sw_2 : pt_2] \cdot \text{dup}$$

which is short hand for

$$\text{dup} \cdot \text{sw} = sw_1 \cdot \text{pt} = pt_1 \cdot \text{sw} \leftarrow sw_2 \cdot \text{pt} \leftarrow pt_2 \cdot \text{dup}$$

Notice in particular that there is a dup at the beginning and end of t. Now consider the network $in \cdot (p \cdot t)^* \cdot p \cdot out$. Every time a packet traverses an edge in the network and is processed by $p \cdot t$, two packets are prepended to the packet history, and this pair of packets represents the edge traversed.

For example, consider the simple linear network, l presented in the original NetKAT paper. I'll denote the packet $\{\text{sw} = s, \text{pt} = n\}$ by s_n.

$$[\![l]\!](\langle A_1 \rangle) = B_2 :: B_1 :: A_2$$

The pair of packets $B_1 :: A_2$ represents the edge from switch A to switch B. In general, we can represent a path through the network as a history and a set of paths as a set of histories.

We think of our query language as regular expressions over the alphabet of edges. As we just showed, we can represent an edge as a pair of packets. Here, we use a pair of packet predicates.

$$\begin{aligned}
L((a,b)) &= \{pk_b :: \langle pk_a \rangle \mid pk_a \in R(a), pk_b \in R(b)\} \\
L(q + q') &= L(q) \cup L(q') \\
L(q \cdot q) &= \{h' :: h \mid h \in L(q), h' \in L(q')\} \\
L(q^*) &= \bigcup_{i=0}^{\infty} q^i
\end{aligned}$$

Here are some example queries:

- No paths: $(0, 0)$
- All paths: $(1, 1)^*$
- Paths of length n: $(1, 1)^n$
- A single path: $(c_4, d_3) \cdot (d_2, e_4) \cdot (e_2, b_4) \cdot (b_1, a_2)$
- All paths that include a given edge: $(1, 1)^* \cdot (a_1, b_2) \cdot (1, 1)^*$
- All paths that include a given path: $(1, 1)^* \cdot (c_4, d_3) \cdot (d_2, e_4) \cdot (e_2, b_4) \cdot (b_1, a_2) \cdot (1, 1)^*$
- All paths including a bidirectional edge: $(1, 1)^* \cdot ((a_1, b_2) + (b_2, a_1)) \cdot (1, 1)^*$
- All paths traversed by HTTP traffic: $(\text{typ} = \text{http}, \text{typ} = \text{http})^*$
- All paths from host A to host Z: $(\text{sw} = A, \text{sw} = Z) + ((\text{sw} = A, 1) \cdot (1, 1)^* \cdot (1, \text{sw} = Z))$

3 Compilation

Remember that a network $in \cdot (p \cdot t)^* \cdot p \cdot out$ unwraps to something like

$$in \cdot (p \cdot dup \cdot t' \cdot dup)^* \cdot p \cdot out$$

where t' is $\Phi(t)$. We can again unwrap this into a bunch of terms:

$$\begin{gathered} in \cdot p \cdot out \\ in \cdot (p \cdot dup \cdot t' \cdot dup) \cdot p \cdot out \\ in \cdot (p \cdot dup \cdot t' \cdot dup) \cdot (p \cdot dup \cdot t' \cdot dup) \cdot p \cdot out \\ in \cdot (p \cdot dup \cdot t' \cdot dup) \cdot (p \cdot dup \cdot t' \cdot dup) \cdot (p \cdot dup \cdot t' \cdot dup) \cdot p \cdot out \\ \cdots \end{gathered}$$

In order to match a specific path, $(a_1, b_2) \cdot (b_3, c_4)$ for example, we simply replace the dups with our predicates. If there are too few or too many dups, we drop the term.

$$\begin{gathered} in \cdot p \cdot out \qquad \text{(too few dups)} \\ in \cdot (p \cdot a_1 \cdot t' \cdot b_2) \cdot p \cdot out \qquad \text{(too few dups)} \\ in \cdot (p \cdot a_1 \cdot t' \cdot b_2) \cdot (p \cdot b_3 \cdot t' \cdot c_4) \cdot p \cdot out \qquad \text{(good!)} \\ in \cdot (p \cdot a_1 \cdot t' \cdot b_2) \cdot (p \cdot b_3 \cdot t' \cdot c_4) \cdot (p \cdot dup \cdot t' \cdot dup) \cdot p \cdot out \qquad \text{(too many dups)} \\ \cdots \end{gathered}$$

Intuitively, we're checking that a network accepts a set of paths by unwrapping the query into the network replacing dups with predicates.

Compilation helper

$$\begin{aligned} H_{net}((a, b)) &= p \cdot a \cdot t' \cdot b \\ H_{net}(q + q') &= H_{net}(q) + H_{net}(q') \\ H_{net}(q \cdot q') &= H_{net}(q) \cdot H_{net}(q') \\ H_{net}(q^*) &= H_{net}(q)^* \end{aligned}$$

Compilation

$$C_{net}(q) = in \cdot H_{net}(q) \cdot p \cdot out$$

4 Correctness

For a given NetKAT term p, let $A_p = \{\alpha \mid \epsilon_{\alpha,\beta}(p)\}$ and $B_p = \{\beta \mid \epsilon_{\alpha,\beta}(p)\}$. That is, A_p and B_p are the sets of αs and βs with 1's in the E-matrix of p.

We want $C_{net}(q)$ to satisfy the following:

$$L(q) \cap R(\Phi(net)) = \bigcup_{\alpha \in A_{C_{net}(q)}} [\![\mathrm{net}]\!](\alpha)$$

Intuitively it says to take all the paths matched by our query, $L(q)$, and intersect them with all paths permitted by the network, $R(\Phi(net))$. This set is generated by the packets matching all the αs in the E-matrix of our compiled term $C_{net}(q)$.