A *packet* is a map $\alpha : \{\text{fields}\} \to \{\text{values}\}$. There are a fixed finite number of fields in a packet. Fields are denoted $f, g, \ldots$. The value of the field f in packet α is denoted $\alpha.f$.

Packets can be modified using a rebinding operator. The packet $\alpha[m/f]$ is obtained by rebinding field f to value m in α. Formally,

$$\alpha[m/f].g = \begin{cases} \alpha.g & \text{if } f \neq g \text{ (that is, if } f \text{ and } g \text{ are different fields)} \\ m & \text{if } f = g. \end{cases}$$

Packets are in one-to-one correspondence with atoms (complete tests). The packet with values $\alpha.f = n_f$ corresponds to the atom (complete test) $\bigwedge_f f = n_f$. We can therefore identify packets with atoms. This allows us to interpret α as either an atom or its corresponding packet, depending on context.

A *packet history* is a nonnull list of packets $\alpha_1 :: \cdots :: \alpha_n$. The *head packet* is α_1. Packet histories are denoted h.

NFAs

Apply the Kleene construction to get an NFA M_e from a NetKAT expression e. Assume wlog that De Morgan has been applied to tests so that all negations are applied only to primitive tests. Thus edge labels are either assignments $f \leftarrow n$, primitive tests $f = m$ or their negations $f \neq m$, or dup. Call this alphabet Σ. We have a nondeterministic automaton (Q, τ, S, F) where:

- Q is a finite set of states
- $\tau \subseteq Q \times \Sigma \times Q$ is the transition relation
- $S \subseteq Q$ are the start states
- $F \subseteq Q$ are the accept states.

Transducer Semantics

Can be viewed operationally or denotationally.

Operational View

Start with a single nonempty packet history placed on a nondeterminstically chosen start state. Now suppose we have a packet history on a state s. If $s \in F$, we can nondeterministically decide to stop, in which case we include the current packet history in the output set. Otherwise, we can move along any nondeterministically chosen enabled edge out of s, modifying the packet history as determined by the label of the edge. Whether an edge is enabled depends on the label and the current head packet.

- Assignment edges are always enabled. If the edge is an assignment edge $s \xrightarrow{f \leftarrow m} t$ and the current packet history is $\alpha :: h$, then we can move from s to t, rebinding the field f of the head packet to the value m. The new packet history is $\alpha[m/f] :: h$.

- Dup edges are always enabled. If the edge is a dup edge $s \xrightarrow{\mathsf{dup}} t$ and the current packet history is $\alpha :: h$, then we can move from s to t, duplicating the head packet. The new packet history is $\alpha :: \alpha :: h$.

- A test edge $s \xrightarrow{f=m} t$ or $s \xrightarrow{f \neq m} t$ is enabled if the test is true for the head packet of the current packet history. If the current packet history is $\alpha :: h$, then the edge $s \xrightarrow{f=m} t$ is enabled if $\alpha.f = m$ and $s \xrightarrow{f \neq m} t$ is enabled if $\alpha.f \neq m$. If the edge is enabled, we can move along the edge to state t, but we do not alter the packet history.

Let $L(N_e)(h)$ denote the output set starting with input history h. This is the the set of packet histories that can ever be "output" (can ever occupy a final state) on input h. Then $L(M_e) : H \to 2^H$. Note that $L(M_e)(h)$ only depends on the head packet of h; thus if α is a single packet, then

$$L(M_e)(\alpha :: h) = \{x :: h \mid x \in L(M_e)(\alpha)\},$$

or in other words, $x :: h \in L(M_e)(\alpha :: h)$ if and only if $x \in L(M_e)(\alpha)$.

Denotational View

We can represent the output set of $L(M_e)(h)$ in terms of a least fixpoint of a monotone map on state labelings of type $\ell : Q \to 2^H$, ordered by pointwise set inclusion; that is, $\ell \leq \ell'$ if $\ell(s) \subseteq \ell'(s)$ for all $s \in Q$. The labeling we are interested in is the least labeling $\ell : Q \to 2^H$ such that:

- $h \in \ell(s)$ for all $s \in S$
- if $\alpha :: h \in \ell(s)$ and $s \xrightarrow{f \leftarrow m} t \in \tau$, then $\alpha[m/f] :: h \in \ell(t)$
- if $\alpha :: h \in \ell(s)$ and $s \xrightarrow{\mathsf{dup}} t \in \tau$, then $\alpha :: \alpha :: h \in \ell(t)$
- if $\alpha :: h \in \ell(s)$, $s \xrightarrow{f=m} t \in \tau$, and $\alpha.f = m$, then $\alpha :: h \in \ell(t)$
- if $\alpha :: h \in \ell(s)$, $s \xrightarrow{f \neq m} t \in \tau$, and $\alpha.f \neq m$, then $\alpha :: h \in \ell(t)$.

Then $L(M_e)(h) = \bigcup_{t \in F} \ell(t)$.

Theorem 0.1 $[\![e]\!] = L(M_e)$.

Language Recognition Semantics for NFAs

Inputs to the automaton are strings in $At \cdot (P \cdot \mathsf{dup})^* \cdot P$. Such a string x is *accepted* if there is a path from some $s \in S$ to some $t \in F$ with label y such that the inequality $x \leq y$ follows from the axioms of NetKAT. The label of a directed path is the concatenation of the edge labels along the path in order. Let $G(M_e)$ denote the set of accepted strings.

We have overloaded the symbol G—recall that it also denotes the inductively defined map from NetKAT expressions to subsets of $At \cdot (P \cdot \mathsf{dup})^* \cdot P$.

Theorem 0.2 $G(e) = G(M_e)$.

Relation between Transducer and Language Semantics

Let α_i be packets, $1 \leq i \leq m$. Let p_i be the corresponding complete assignments; that is, if $\alpha_i.f = n_f$ for all fields f, then p_i is the complete assignment that assigns $f \leftarrow n_f$ for all fields f.

Theorem 0.3 *The following are equivalent:*

(i) $\alpha_m :: \cdots :: \alpha_1 \in L(M_e)(\alpha_0)$

(ii) $\alpha_m :: \cdots :: \alpha_1 \in [\![e]\!](\alpha_0)$

(iii) $\alpha_0 p_1 \, \mathsf{dup} \, p_2 \, \mathsf{dup} \cdots \mathsf{dup} \, p_m \in G(M_e)$

(iv) $\alpha_0 p_1 \, \mathsf{dup} \, p_2 \, \mathsf{dup} \cdots \mathsf{dup} \, p_m \in G(e)$.

DFAs

Left-to-Right

A left-to-right DFA is a coalgebra for the functor $FS = S^{At \times P} \times 2^{At \times P}$ with distinguished start state $s \in S$. The structure map is pair of functions

$$\delta : S \to S^{At \times P} \qquad\qquad \varepsilon : S \to 2^{At \times P}.$$

We write $\delta_{\alpha p \, \mathsf{dup}}(q)$ for $\delta(q)(\alpha, p)$ and $\varepsilon_{\alpha p}(q)$ for $\varepsilon(q)(\alpha, p)$, so

$$\delta_{\alpha p \, \mathsf{dup}} : S \to S \qquad\qquad \varepsilon_{\alpha p} : S \to 2.$$

Inputs to the automaton are strings of the form

$$\alpha p_1 \, \mathsf{dup} \, p_2 \, \mathsf{dup} \cdots \mathsf{dup} \, p_m$$

in $At \cdot (P \cdot \mathsf{dup})^* \cdot P$. Acceptance is defined in terms of a coinductively defined predicate $\mathsf{Accept} : S \times At \cdot (P \cdot \mathsf{dup})^* \cdot P \to 2$.

$$\mathsf{Accept}(q, \alpha p \, \mathsf{dup} \, x) = \mathsf{Accept}(\delta_{\alpha p \, \mathsf{dup}}(q), \alpha_p x)$$
$$\mathsf{Accept}(q, \alpha p) = \varepsilon_{\alpha p}(q).$$

A string $x \in At \cdot (P \cdot \mathsf{dup})^* \cdot P$ is *accepted* by the automaton if $\mathsf{Accept}(s, x)$.

Right-to-Left

A right-to-left DFA is a coalgebra for the functor $FS = S^P \times 2^{At \times P}$ with distinguished start state $s \in S$. The structure map is a pair of functions

$$\delta : S \to S^P \qquad\qquad \varepsilon : S \to 2^{At \times P}.$$

We write $\delta_{\mathsf{dup}\,p}(q)$ for $\delta(q)(p)$ and $\varepsilon_{\alpha p}(q)$ for $\varepsilon(q)(\alpha, p)$, so

$$\delta_{\mathsf{dup}\,p} : S \to S \qquad\qquad \varepsilon_{\alpha p} : S \to 2.$$

Inputs to the automaton are strings of the form

$$\alpha p_1 \,\mathsf{dup}\, p_2 \,\mathsf{dup} \cdots \mathsf{dup}\, p_m$$

in $At \cdot P \cdot (\mathsf{dup} \cdot P)^*$. Acceptance is defined in terms of a coinductively defined predicate $\mathsf{Accept} : S \times At \cdot P \cdot (\mathsf{dup} \cdot P)^* \to 2$.

$$\begin{aligned} \mathsf{Accept}(q, x \,\mathsf{dup}\, p) &= \mathsf{Accept}(\delta_{\mathsf{dup}\,p}(q), x) \\ \mathsf{Accept}(q, \alpha p) &= \varepsilon_{\alpha p}(q). \end{aligned}$$

A string $x \in At \cdot P \cdot (\mathsf{dup} \cdot P)^*$ is *accepted* by the automaton if $\mathsf{Accept}(s, x)$.

Conversion of NFAs to DFAs?