Configuration

$\texttt{SERVER}$ refers to any class of elements where this operation is defined:

- getId: $\texttt{SERVER} \to \mathbb{N}$.

We impose the restriction that this function is inyective.

$\texttt{ENTITY}$ refers to any class of elements where this operation is defined:

- getId: $\texttt{ENTITY} \to \mathbb{N}$.

We impose the restriction that this function is inyective.

Elements of type $\texttt{FEL}$ are subsets of $\texttt{EVENT}$ and they are operated as priority queues. The priority of a given event is given by it's time of ocurrence and a total relation defined over event types:

- $\texttt{insertEvent}$: $\texttt{FEL} \to (\texttt{EVENT} \to \texttt{FEL})$.
- $\texttt{deleteIncoming}$: $\texttt{FEL} \to \texttt{FEL}$.
- $\texttt{peekIncoming}$: $\texttt{FEL} \to \texttt{EVENT}$.

Elements of type $\texttt{CHANNEL}$ are subsets of $\mathbb{N} \times \texttt{ENTITY}$ and are operated as priority queues:

- $\texttt{insertEntity}$: $\texttt{CHANNEL} \to (\mathbb{N} \to (\texttt{ELEMENT} \to \texttt{CHANNEL}))$.
- $\texttt{peekTopEntity}$: $\texttt{CHANNEL} \to \texttt{ELEMENT}$.
- $\texttt{deleteTopEntity}$: $\texttt{CHANNEL} \to \texttt{CHANNEL}$.

A configuration is a tuple

$$(\texttt{Servers}, \texttt{Entities}, \texttt{Channels}, \texttt{isServing}, \texttt{waits}, \texttt{listens}, L)$$

where:

- $\texttt{Servers}$ is a collection of elements of type $\texttt{SERVER}$.
- $\texttt{Entities}$ is a collection of elements of type $\texttt{ENTITY}$.
- $\texttt{Channels}$ is a collection of elements of type $\texttt{CHANNEL}$.
- $\texttt{isServing} \subseteq \texttt{Servers} \times \texttt{Entities}$ is the relationship that indicates which entity is being served by which server.
- $\texttt{waits} \subseteq \texttt{Entities} \times \texttt{Channels}$ is the relationship that indicates in which channel an entity is waiting.
- $\texttt{attends} \subseteq \texttt{Servers} \times \texttt{Channels}$ is the relationship that indicates which channels a given server is currently listening.
- L is an element of type $\texttt{FEL}$.

Rules of Transition

It would be desirable that each user could decide how the system evolves over time according to the diferent events that occur. This is why we introduce rules of transition, which are made up of two parts: The first part is a first order formula that indicates the conditions that a configuration must satisfy before applying the transition. The second part defines the new configuration of the system after the application. Thus, we can see a rule of transistion as a partial function `Rule`: $\texttt{CONFIGURATION} \rightarrow \texttt{CONFIGURATION}$. We will now show examples of such rules for a given system.

Example of a traditional system

We will consider a traditional system made up of n servers and one channel, and where the only events that we consider are arribals and exits. The initial configuration of this system is:

$$\langle \{s_1, \ldots, s_n\}, \emptyset, \{Q\}, \{\texttt{ARRIBAL}, \texttt{OUT}\}, \emptyset, \emptyset, \{(s_1, Q), \ldots, (s_n, Q)\}, \emptyset \rangle$$

$$\frac{\texttt{isEmpty}(L) = \texttt{TRUE} \qquad e\colon \texttt{ENTITY} \wedge e \notin \texttt{Entities}}{\texttt{Entities}' = \texttt{Entities} \cup \{e\} \wedge L' = L \cup \{(\texttt{ARRIBAL}, k, e)\}}$$