Continuation	$\mathcal{K}$	$\stackrel{\text{def}}{=}$	$\mathcal{S} : E$
Parallel Threads	$\mathcal{P}$	$::=$	$\mathcal{P} \| \mathcal{P} \mid \mathcal{K}$
Thread Environment	$\mathcal{S}$	$\stackrel{\text{def}}{=}$	$(\mathcal{K}_s; \mathcal{K}_c; \mathcal{T}; \mathcal{C}_s; \mathcal{C}_r)$
Logical time	$\mathcal{T}$	$::=$	$\mathbb{Z} \mid 1, 2, 3... \mid \mathbb{M}$
Choice Continutation	$\mathcal{K}_c$	$::=$	$\mathcal{S} : E$
Send Channel IDs	$\mathcal{C}_s$	$::=$	$\cdot \mid \mathcal{C}_s, x_{ch} \mapsto \mathcal{T}$
Receive Channel IDs	$\mathcal{C}_r$	$::=$	$\cdot \mid \mathcal{C}_r, x_{ch} \mapsto \mathcal{T}$
Continuation Stack	$\mathcal{K}_s$	$::=$	$\cdot \mid \mathcal{K}_s, \mathcal{T} \mapsto \mathcal{K}$
Channel State	Δ	$::=$	$\cdot \mid \Delta, x_{ch} \mapsto ch$
Channel	ch	$\stackrel{\text{def}}{=}$	(s, c, r, v)
Evaluation Context	E	$::=$	$\cdot \mid sync\ \mathcal{K} \mid E(e)$
Terms	e	$::=$	$send\ x_{ch}\ e \mid recv\ x_{ch} \mid yield \mid newChan \mid choose\ e\ e \mid backtrack \mid return\ e$ $\mid par\ e\ e \mid \lambda\ x.e \mid e\ e$
Values	v	$::=$	$\langle\rangle \mid 0, 1, \ldots$

Figure 1: Syntax

$$
\begin{array}{lllll}
send(t): & s = c = r \wedge t > s & \rightarrow & s' = t \\
sendBack(t): & t < s \wedge & & \\
 & (s = c = r \vee & & \\
 & (s > c \wedge r = c) \vee & & \\
 & (s < c \wedge r < c \wedge s > c)) & \rightarrow & s' = t \\
recv(t): & s > c \wedge r = c \wedge t > r & \rightarrow & r' = t
\end{array}
$$

Figure 2: State Predicates

If we can specify the predicate on the states in this way, it helps when I specify the semantics. I'm not sure if SAL can handle this.

$\boxed{\Delta : \mathcal{P} \Rightarrow \Delta' : \mathcal{P}'}$

$$\text{(send)}\ \frac{send(T)\ \textit{holds for}\ s, c, r}{\Delta, x_{ch} \mapsto (s, c, r, \emptyset) : \mathcal{P} \| \mathcal{S} : E(send\ x\ v) \Rightarrow \Delta, x_{ch} \mapsto (s', c, r, v) : \mathcal{P} \| \mathcal{S}' : E(\langle\rangle)}$$

$$\textit{where}\ \mathcal{S} = (\mathcal{K}_s; \mathcal{K}_c; \mathcal{T}; \mathcal{C}_s; \mathcal{C}_r), \mathcal{S}' = (\mathcal{K}_s, \mathcal{T} \mapsto \mathcal{K}; \mathcal{K}_c; \mathcal{T}'; \mathcal{C}_s, x_{ch} \mapsto \mathcal{T}; \mathcal{C}_r)$$

Figure 3: Semantics

The idea here is to define the semantics of the language in terms of the state predicates. This is just a single example, and I think I'm missing some details, but hopefully it gets the idea across. We can perform a *send* if the *send* predicate holds, and when we do, we add a new continuation to the stack, advance the logical time, put the sent value on the channel, etc.