Thread	$\mathcal{T}$	$\stackrel{\text{def}}{=}$	$[\mathcal{I}; \mathcal{T}; \mathcal{K}; \mathcal{L}_s; \mathcal{L}_r; \mathcal{S}]\, E$
Parallel Threads	$\mathcal{P}$	$::=$	$\mathcal{P} \| \mathcal{P} \mid \mathcal{T}$
Process ID	$\mathcal{I}$	$\stackrel{\text{def}}{=}$	*A representation process IDis*
Logical time	$\mathcal{T}$	$\stackrel{\text{def}}{=}$	*A representation of the logical time of the thread*
Stable time	$\mathcal{S}$	$\stackrel{\text{def}}{=}$	*The logical time to backtrack to if we encounter a backtrack*
Send Channel IDs	$\mathcal{L}_s$	$::=$	$\cdot \mid \mathcal{L}_s, x$
Receive Channel IDs	$\mathcal{L}_r$	$::=$	$\cdot \mid \mathcal{L}_r, x$
Continuation Stack	$\mathcal{K}$	$::=$	$\cdot \mid \mathcal{K}, \mathcal{T}$
Global Channel State	Δ	$::=$	$\cdot \mid \Delta, x \mapsto ch$
Channel	ch	$\stackrel{\text{def}}{=}$	$(stime, rtime, ctime)$
Evaluation Context	E	$::=$	$\cdot \mid sync\ E \mid E(e)$
Terms	e	$::=$	$send \mid recv\ x \mid yield \mid newChan \mid choose \mid backtrack \mid return\ v$
Values	v	$::=$	$\langle\rangle \mid 0, 1, \ldots$

Figure 1: Syntax

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

For brevity, here are some discussion level macros:
$\Gamma = [\mathcal{I};\mathcal{T};\mathcal{K};\mathcal{L}_s;\mathcal{L}_r;\mathcal{S}]$
$\Gamma_1 = [\mathcal{I}_1;\mathcal{T}_1;\mathcal{K}_1;\mathcal{L}_{s1};\mathcal{L}_{r1};\mathcal{S}_1]$
$\Gamma_2 = [\mathcal{I}_2;\mathcal{T}_2;\mathcal{K}_2;\mathcal{L}_{s2};\mathcal{L}_{r2};\mathcal{S}_2]$
$update(\Delta, \mathcal{T}_s, \mathcal{L}_s, \mathcal{T}_r, \mathcal{L}_r) =$
$\{x \mapsto (\mathcal{T}_s, rtime(ch), ctime(ch)) \mid x \in \mathcal{L}_s \wedge ch = \Delta(x)\} \cup$
$\{x \mapsto (stime(ch), \mathcal{T}_r, ctime(ch)) \mid x \in \mathcal{L}_r \wedge ch = \Delta(x)\} \cup$
$\{x \mapsto ch \mid x \mapsto ch \in \Delta \wedge x \notin \{dom(\mathcal{L}_r) \cup dom(\mathcal{L}_r)\}\}$

$$\text{(NewChan)}\ \frac{}{\Delta : \mathcal{P}\|\Gamma E(newChan) \Rightarrow \Delta, x \mapsto ch : \mathcal{P}\|\Gamma E(x)}\ \textit{fresh}\ \text{ch,x}$$

$$\text{Comm}\ \frac{\begin{array}{c} ctime(ch) \equiv stime(ch) \equiv rtime(ch) \\ \mathcal{T}' = max(\mathcal{T}_1, \mathcal{T}_2) + 1 \qquad k_1 = \Gamma_1 E_1(send\ v\ x) \qquad k_2 = \Gamma_2 E_2(recv\ x) \\ \widehat{\Gamma_1} = \left[\mathcal{I}_1;\mathcal{T}';\mathcal{K}, k_1;\mathcal{L}_s, x;\mathcal{L}_r;\mathcal{S}\right] \qquad \widehat{\Gamma_2} = \left[\mathcal{I}_2;\mathcal{T}';\mathcal{K}, k_2;\mathcal{L}_s;\mathcal{L}_r, x;\mathcal{S}\right] \end{array}}{\Delta, x \mapsto ch : \mathcal{P}\|\Gamma_1 E_1(send\ v\ x)\|\Gamma_2 E_2(recv\ x) \Rightarrow \Delta, x \mapsto ch' : \mathcal{P}'\|\widehat{\Gamma_1} E_1(\langle\rangle)\|\widehat{\Gamma_2} E_2(v)}$$

$$\text{(Choose)}\ \frac{k = \Gamma E(e_2) \qquad \hat{\Gamma} = [\mathcal{I};\mathcal{T};\mathcal{K}, k;\mathcal{L}_s;\mathcal{L}_r;\mathcal{T}]}{\Delta : \mathcal{P}\|\Gamma E(choose\ e_1\ e_2) \Rightarrow \Delta : \mathcal{P}\|\hat{\Gamma} E(e_1)}$$

$$\text{(Yield)}\ \frac{}{\Delta : \mathcal{P}\|\Gamma E(yield) \Rightarrow \Delta : \Gamma E(\langle\rangle)\|\mathcal{P}}$$

$$\text{(Backtrack)}\ \frac{\begin{array}{c} \exists k \in \mathcal{K}. k = \hat{\Gamma} E'\ \ where\ \hat{\Gamma} = \left[\mathcal{I};\mathcal{S};\mathcal{K}';\mathcal{L}_s';\mathcal{L}_r';\mathcal{S}'\right] \\ \Delta' = update(\Delta, \mathcal{L}_s'(x), \mathcal{L}_s', \mathcal{L}_r'(x), \mathcal{L}_s') \end{array}}{\Delta : \mathcal{P}\|\Gamma E(backtrack) \Rightarrow \Delta' : \mathcal{P}\|\hat{\Gamma} sync\ E'}$$

$$\text{(Return-End)}\ \frac{\forall x \in \{dom(\mathcal{L}_s) \cup dom(\mathcal{L}_r)\}. \exists ch \in \Delta(x). rtime(ch) \equiv stime(ch) \equiv \mathbb{M}}{\Delta : \mathcal{P}\|\Gamma E(return\ v) \Rightarrow \Delta : \mathcal{P}\|\Gamma E(v)}$$

$$\text{(Return-Wait)}\ \frac{\Delta' = update(\Delta, \mathbb{M}, \mathcal{L}_s, \mathbb{M}, \mathcal{L}_r)}{\Delta : \mathcal{P}\|\Gamma E(return\ v) \Rightarrow \Delta' : \mathcal{P}\|\Gamma E(return\ v)}$$

$$\text{(Return-Back)}\ \frac{\begin{array}{c} (\exists x \in dom(\mathcal{L}_r). \exists ch = \Delta(x). stime(ch) < ctime(ch). \mathcal{T}' = stime(ch)) \vee \\ (\exists x \in dom(\mathcal{L}_s). \exists ch = \Delta(x). rtime(ch) < ctime(ch). \mathcal{T}' = rtime(ch)) \\ \hat{\Gamma} = \left[\mathcal{I};\mathcal{T};\mathcal{K};\mathcal{L}_s;\mathcal{L}_r;\mathcal{T}'\right] \end{array}}{\Delta : \mathcal{P}\|\Gamma E(return\ v) \Rightarrow \Delta : \mathcal{P}\|\hat{\Gamma} E(backtrack)}$$

$$\text{(Sync)}\ \frac{\begin{array}{c} \forall x \in \{dom(\mathcal{L}_s) \cup dom(\mathcal{L}_r)\}. \exists ch = \Delta(x). stime(ch) \equiv rtime(ch) < ctime(ch) \\ \Delta' = \{x \mapsto (stime(ch), rtime(ch), stime(ch)) \mid x \in \{dom(\mathcal{L}_s) \cup dom(\mathcal{L}_r)\} \wedge ch = \Delta(x)\} \end{array}}{\Delta : \mathcal{P}\|\Gamma sync\ E \Rightarrow \Delta' : \mathcal{P}\|\Gamma E}$$

Figure 2: Semantics

The NewChan rule simply adds a new channel to the global channel store, and returns a new identifier for it. Note that the semantics of the language currently lack a binding form that actually allows this identifier to be used (since threads require the identifier), but such an extension is trivial.

The Comm rule allows two processes to communicate synchronously, if the times on the channel are all in sync. This indicates that none of the threads involved are currently trying to backtrack. It updates the channels times to be the max of the sender or receivers time plus 1, saves the current continuation to the continuation stack, and updates the time of both threads.

Choose creates a new continuation whose stable time is the stable time prior to the new choice point, and whose continuation is the second choice. This ensure that backtracking once takes the second choice, and backtracking again goes to the prior stable time.

Yield simply forces a context switch if the system cannot support all threads running in parallel. Of course, a scheduler would also be required in such a situation, which is not formalized here.

Backtrack finds the continuation whose logical time matches the stable time of the thread, restores the timestamps on all the channels it has used to the times stored for each in the past continuation. Finally, it transfers to a 'sync' context, which waits for neighbors to backtrack and resynchronized the channel times. Note: if new channels have been created between the current context and the last stable time, this doesn't currently handle them. I'm not entirely sure how to handle such channels just yet, but this situation can be avoided and the system still be useful.

Return-End provides the conditions for terminating. Both the sender and receiver on all channels the thread is involved in must 'release' the channel. This occurs by each setting their time on the channel to some maximal time value, $\mathbb{M}$

Return-Wait updates all the channels the thread has used with the $\mathbb{M}$ value, and continues to wait for something in the global context to change before it can make progress.

Return-Back provides for backtracking of neighbor threads. The thread must wait on all channels it has used, and if the threads partner has set it's time backwards (that is, the neighbors time is less than the channels time), then the thread must backtrack and join it's partner.

Finally, Sync, as mentioned previously, allows a backtracking thread to wait for it's neighbors to join it at a previous logical time, before resynchronizing the channel.

The formal language lacks *par*, functions, binding forms, and some other uninteresting details. However, what's here should provide enough interesting semantics for now.