MODULE *ELZ*

EXTENDS *Integers*

This model describes the semantics of rendezvous signal exchange intended for C programs verification tools. These operations should not be considered as regular operators for execution.

CONSTANTS	*Processes*,	The numbver of processes (threads actually) that can communicate. *\
	Signals,	The set of names of signal that can be sent by any thread. *\
	WorkingSet	Values that can be assigned at any working step *\

VARIABLES	*SigStates*,	Describes the state of each signal transition. *\
	ProcStates,	Process states: working or ready. *\
	ProcValues,	Current values stored per process *\
	SigStorage,	Describes the state of each process. *\
	ProcSignals	Specifies the signal that expected to be received *\

$$vars \triangleq \langle SigStates,\ ProcStates,\ ProcValues,\ SigStorage,\ ProcSignals \rangle$$

At the very beginning, each process has a 0 value, and all processes have the same signal as a chosen one.

$$\begin{array}{lll} Init & \triangleq & \land\ SigStates = [s \in Signals \mapsto \text{“idle”}] \\ & & \land\ SigStorage\ = [s \in Signals \mapsto 0] \\ & & \land\ ProcStates\ = [p \in Processes \mapsto \text{“working”}] \\ & & \land\ ProcValues = [p \in Processes \mapsto 0] \\ & & \land\ ProcSignals = [p \in Processes \mapsto 0] \end{array}$$

At the very beginning, each process has a 0 value, and all processes have the same signal as a chosen one.

At the working step, a process may change its values nondeterministically.

$$\begin{array}{lll} Working(p) & \triangleq & \land\ ProcStates[p] = \text{“working”} \\ & & \land\ \exists\, i \in WorkingSet :\ \ ProcValues' = [ProcValues\ \text{EXCEPT}\ ![p] = i] \\ & & \land\ \text{UNCHANGED}\ \langle ProcStates,\ SigStates,\ SigStorage,\ ProcSignals \rangle \end{array}$$

Flag action shows other processes that this process wants to receive a message.

$$\begin{array}{lll} Flag(p,\ s) & \triangleq & \land\ ProcStates[p] = \text{“working”} \\ & & \land\ ProcStates' = [ProcStates\ \text{EXCEPT}\ ![p] = \text{“ready”}] \\ & & \land\ SigStates[s] = \text{“idle”} \\ & & \land\ SigStates' = [SigStates\ \text{EXCEPT}\ ![s] = \text{“waiting”}] \\ & & \land\ \text{UNCHANGED}\ \langle ProcValues,\ SigStorage,\ ProcSignals \rangle \end{array}$$

Send value to any process that waits signal “s”.

$$\begin{array}{lll} Send(p,\ s) & \triangleq & \land\ ProcStates[p] = \text{“working”} \\ & & \land\ SigStates[s] = \text{“waiting”} \end{array}$$

$$\begin{array}{l}
\land \mathit{SigStates}' = [\mathit{SigStates} \text{ EXCEPT } ![s] = \text{“set”}] \\
\land \mathit{SigStorage}' = [\mathit{SigStorage} \text{ EXCEPT } ![s] = \mathit{ProcValues}[p]] \\
\land \text{UNCHANGED } \langle \mathit{ProcStates},\ \mathit{ProcValues},\ \mathit{ProcSignals} \rangle
\end{array}$$

A process can receive the value that was sent by another process.

$$\mathit{Receive}(p,\ s) \triangleq \begin{array}[t]{l}
\land \mathit{ProcStates}[p] = \text{“ready”} \\
\land \mathit{SigStates}[s] = \text{“set”} \\
\land \mathit{ProcSignals}[p] = s \\
\land \mathit{ProcStates}' = [\mathit{ProcStates} \text{ EXCEPT } ![p] = \text{“working”}] \\
\land \mathit{ProcValues}' = [\mathit{ProcValues} \text{ EXCEPT } ![p] = \mathit{SigStorage}[s]] \\
\land \mathit{SigStates}' = [\mathit{SigStates} \text{ EXCEPT } ![s] = \text{“idle”}] \\
\land \text{UNCHANGED } \langle \mathit{SigStorage},\ \mathit{ProcSignals} \rangle
\end{array}$$

This action is an artificial one and intended for choosing another signal by a process. It is crucial preventing the change of a process'es signal when it is in the “ready” state.

$$\mathit{ChangeSignal}(p) \triangleq \begin{array}[t]{l}
\land \mathit{ProcStates}[p] = \text{“working”} \\
\land \exists\, s \in \mathit{Signals} : \mathit{ProcSignals}' = [\mathit{ProcSignals} \text{ EXCEPT } ![p] = s] \\
\land \text{UNCHANGED } \langle \mathit{SigStates},\ \mathit{ProcStates},\ \mathit{ProcValues},\ \mathit{SigStorage} \rangle
\end{array}$$

This action is an artificial one and intended for choosing another signal by a process. It is crucial preventing the change of a process'es signal when it is in the “ready” state.

Each step a process may either work, change its signal, flag and then receive a signal or send any signal.

$$\mathit{Next} \triangleq \exists\, p \in \mathit{Processes} : \begin{array}[t]{l}
\lor \mathit{Working}(p) \\
\lor \mathit{ChangeSignal}(p) \\
\lor \exists\, s \in \mathit{Signals} : \begin{array}[t]{l}
\lor \mathit{Flag}(p,\ s) \\
\lor \mathit{Send}(p,\ s) \\
\lor \mathit{Receive}(p,\ s)
\end{array}
\end{array}$$

The formula describes the behaviour of the model.

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next}]_{vars}$$

The formula below is a type invariant.

$$\mathit{TypeOK} \triangleq \begin{array}[t]{l}
\land \mathit{SigStates} \in [\mathit{Signals} \rightarrow \{\text{“idle”},\ \text{“waiting”},\ \text{“set”}\}] \\
\land \mathit{SigStorage} \in [\mathit{Signals} \rightarrow \mathit{WorkingSet} \cup \{0\}] \\
\land \mathit{ProcStates} \in [\mathit{Processes} \rightarrow \{\text{“working”},\ \text{“ready”}\}] \\
\land \mathit{ProcValues} \in [\mathit{Processes} \rightarrow \mathit{WorkingSet} \cup \{0\}] \\
\land \mathit{ProcSignals} \in [\mathit{Processes} \rightarrow \mathit{Signals} \cup \{0\}]
\end{array}$$

The formula below shows the correspondence between signals and processes.

$$\mathit{PropPending} \triangleq \exists\, x \in \mathit{Processes} : \mathit{ProcStates}[x] = \text{“ready”} \equiv \exists\, s \in \mathit{Signals} : \mathit{SigStates}[s] \neq \text{“idle”}$$

\ * Modification History
\ * Last modified *Mon Feb* 10 17:00:44 *MSK* 2020 by *zakharov*

\ * Created *Fri Feb* 07 12:23:21 *MSK* 2020 by *zakharov*