We use A Task structure {stat, result} to keep the status (*running* or *done*) and the result (⊥ or *val*) of a task and associate each task with a task identifier. The tasks are global variables can be created when a procedure is posted or asynchronously called. We mark the execution frames of tasks with the corresponding task identifiers. These task identifiers are used while creating, waiting for and returning a task.

CALL

$$\frac{l' \in e(g,l)}{\langle g, \langle l, S[call\ x \coloneqq p(e)]\rangle w, m\rangle \ \rightarrow \langle g, \langle l', s_p\rangle \langle l, S[x \coloneqq\ \star]\rangle w, m\rangle}$$

RETURN-CALL

$$\frac{v \in e(g,l)}{\langle g, \langle l, return\ e\rangle \langle l', S[x \coloneqq\ \star]\rangle w, m\rangle \ \rightarrow \langle g, \langle l', S[x \coloneqq v]\rangle w, m\rangle}$$

POST

$$\frac{l' \in e(g,l) \qquad g' = g \cup \{t.stat = running,\ t.result = \bot\} \qquad w' = \langle l', s_p\rangle^t}{\langle g, \langle l, S[post\ t \coloneqq p(e)]\rangle w, m\rangle \ \rightarrow \langle g', \langle l, S[skip]\rangle w, give(m, w')\rangle}$$

RETURN-POST

$$\frac{g' = g[\ t \rightarrow\ \{t.stat = done,\ t.result = e\}]}{\langle g, \langle l, return\ e\rangle^t, m\rangle \ \rightarrow \langle g', \varepsilon, m\rangle}$$

ASYNC

$$\frac{l' \in e(g,l) \qquad g' = g \cup \{t.stat = running,\ t.result = \bot\}}{\langle g, \langle l, S[async\ t \coloneqq p(e)]w\rangle, m\rangle \ \rightarrow \langle g', \langle l', s_p\rangle^t \langle l, S[skip]w\rangle, m\rangle}$$

RETURN-ASYNC

$$\frac{l' \in e(g,l) \qquad g' = g[\ t \rightarrow\ \{t.stat = done,\ t.result = e\}]}{\langle g, \langle l, return\ e\rangle^t \langle l', w\rangle, m\rangle \ \rightarrow \langle g', \langle l', w\rangle, m\rangle}$$

POST-WAIT-CONTINUE

$$\frac{l' \in e(g,l) \qquad \{t.stat = done, t.result = v\} \in g}{\langle g, \langle l, S[x \coloneqq wait\ t]w\rangle^{t'}, m\rangle \ \rightarrow \langle g, \langle l, S[x \coloneqq v]w\rangle^{t'}, m\rangle}$$

ASYNC-WAIT-CONTINUE

$$\frac{l' \in e(g,l) \qquad \{t.stat = done, t.result = v\} \in g}{\langle g, \langle l, S[x \coloneqq wait\ t]w\rangle^{t'} \langle l', w'\rangle^{t''}, m\rangle \ \rightarrow \langle g, \langle l, S[x \coloneqq v]w\rangle^{t'} \langle l', w'\rangle^{t''}, m\rangle}$$

ASYNC-WAIT (async call returns to its caller)

$$\frac{l' \in e(g,l) \qquad \{t.stat = running,\ t.result = \bot\} \in g \qquad w'' = \langle l, S[x \coloneqq wait\ t]w\rangle)^{t'}}{\langle g, \langle l, S[x \coloneqq wait\ t]w\rangle^{t'} \langle l', w'\rangle^{t''}, m\rangle \ \rightarrow \langle g', \langle l', w'\rangle^{t''}, give(m, w''\rangle}$$

POST-WAIT (no caller, dispatch a new task)

$$\frac{\{t.stat = running,\ t.result = \bot\} \in g \qquad w' = \langle l, S[x \coloneqq wait\ t]w\rangle^{t'}}{\langle g, \langle l, S[x \coloneqq wait\ t]w\rangle^{t'}, m\rangle \ \rightarrow \langle g, \varepsilon, give(m, w')\rangle}$$

How to introduce delays?

An idea:

- I assume that a task waiting for a delayed task also delays. Otherwise, I will wait in the same statement even if it is scheduled (ASYNC-WAIT or POST-WAIT). It should execute the transition in DELAY in order to be able to continue its execution.
  (A scheduler executes the non-delaying tasks first. When all tasks are completed/delayed, it takes and executes the tasks in delayed buffer)

DELAY

$$\frac{\langle m',w\rangle \in take(m,g)}{\langle g,\varepsilon,m\rangle \rightarrow \langle g,\varepsilon,delay(m',w)\rangle}$$