termvar, x, f				
nat, n				
val, v	::=			values
	|	x		variable
	|	n		integer
	|	(v_1, v_2)		value pair
	|	inl v		
	|	inr v		
	|	**fun** $f(x)e$	bind x in e	function definition
exp, e	::=			expressions
	|	v		value
	|	$(\|e_1, e_2\|)$		parallel pair
	|	let $x = e_1$ in e_2		let expression
	|	fst v		
	|	snd v		
	|	**case** v **of** $\{$ **inl** $x_1.e_1$, **inr** $x_2.e_2\}$	bind x_1 in e_1	case expression
			bind x_2 in e_2	
	|	$v_1\, v_2$		function application
	|	$e[v/x]$	M	substitution
	|	$C[e]$	M	context application
C	::=			evaluation context
	|	$(\|\_\_, e_2\|)$		pair L
	|	$(\|e_1, \_\_\|)$		pair L
	|	**let** $x = \_\_$ **in** e_2		let L
id	::=			fork/join id
	|	n		
pid	::=			processor id
	|	n		
task, t	::=			tasks for deques
	|	**Left** *id* e		
	|	**Right** *id* e		
current_task, *ct*	::=			
	|	**Current** $e\ s$		
	|	**Recovered**		
deque, d	::=			deque
	|	$[]$		empty
	|	$[t\|d]$		t as the first element
	|	$d@t$		t as the last element
res, r	::=			results (either successful or faulty)
	|	v		successful result as a value
	|	**BOT** (e)		

	$\|$	$\mathbf{LETL}\,(r, x, e)$		
	$\|$	$\mathbf{LETR}\,(r)$		
	$\|$	$\mathbf{PTUPLE}\,(r_1, r_2)$		
	$\|$	$r_1 \# r_2$	M	combine results
$stack,\ s$	::=			stack
	$\|$	$\mathbf{Init}\ id$		
	$\|$	$\mathbf{Left}\ id$		
	$\|$	$\mathbf{Right}\ id$		
	$\|$	$\mathbf{Top}$		
	$\|$	$\mathbf{Cons}\ C\ s$		
	$\|$	$\mathbf{Fail}$		
$branch,\ b$	::=			values to store in result map
	$\|$	$\mathbf{Neither}\ s$		
	$\|$	$\mathbf{Left}\ v\ s$		
	$\|$	$\mathbf{Right}\ v\ s$		
	$\|$	$\mathbf{Finished}\ v$		
$result\_map,\ rm$	::=			map for joins
	$\|$	NO_RESULT		
	$\|$	$rm_1[id := b]$		
	$\|$	$rm_1 \backslash id$		
$current\_exp,\ ce$	::=			
	$\|$	$\mathbf{START}\ e$		
	$\|$	$\mathbf{RUN}\ e$		
	$\|$	$\mathbf{FAILED}$		
$proc\_map,\ pm$	::=			
	$\|$	$pm[pid := (ce, s, d, ct)]$		
$terminals$	::=			
	$\|$	$\Downarrow$		reduce
	$\|$	$\Downarrow^f$		faulty reduce
	$\|$	$\Downarrow^r$		recover
	$\|$	$\Rightarrow$		step
	$\|$	$\Rightarrow^1$		single processor step
	$\|$	$\Rightarrow^m$		multi processor step
$formula$	::=			
	$\|$	$judgement$		
	$\|$	$s_1 \neq s_2$	M	
	$\|$	$r_1 = r_2$	M	
	$\|$	$pm[pid] == (ce, s, d, ct)$	M	
	$\|$	$rm[id] == (b)$	M	
	$\|$	$\mathbf{fresh}\ id\ \mathbf{in}\ rm$	M	

	$\|$	**last** $d\ t$	M	
	$\|$	**removelast** $d_1\ d_2$	M	
	$\|$	**True**	M	
	$\|$	**False**	M	
	$\|$	$(formula)$	M	
	$\|$	$formula_1 \vee formula_2$	M	
	$\|$	$formula_1 \wedge formula_2$	M	
	$\|$	not_value(r)	M	
	$\|$	exp_not_value(e)	M	
not_value_funs	::=			
	$\|$	not_value(r)		
	$\|$	exp_not_value(e)		
res_combine_funs	::=			
	$\|$	$r_1 \# r_2 === r$		combine results
OpSemJudg	::=			
	$\|$	$e_1 \Rightarrow e_2$		small step
	$\|$	$ce_1, s_1, d_1, ct_1 \Rightarrow^1 ce_2, s_2, d_2, ct_2$		single-processor small step
	$\|$	$pm_1, rm_1 \Rightarrow^m pm_2, rm_2$		multi-processor small step
	$\|$	$e \Downarrow v$		e reduces to v (fault-free)
	$\|$	$e \Downarrow^f r$		e reduces to r (fault-prone)
	$\|$	$r_1 \Downarrow^r r_2$		r_1 reduces to r_2 (recovery)
judgement	::=			
	$\|$	*OpSemJudg*		
user_syntax	::=			
	$\|$	*termvar*		
	$\|$	*nat*		
	$\|$	*val*		
	$\|$	*exp*		
	$\|$	*C*		
	$\|$	*id*		
	$\|$	*pid*		
	$\|$	*task*		
	$\|$	*current_task*		
	$\|$	*deque*		
	$\|$	*res*		
	$\|$	*stack*		
	$\|$	*branch*		
	$\|$	*result_map*		
	$\|$	*current_exp*		
	$\|$	*proc_map*		
	$\|$	*terminals*		
	$\|$	*formula*		

$\boxed{\mathsf{not\_value}(r)}$

$$\begin{aligned}
&\mathsf{not_value}(v) \equiv \textbf{False}\\
&\mathsf{not_value}(\textbf{BOT}\,(e)) \equiv \textbf{True}\\
&\mathsf{not_value}(\textbf{LETL}\,(r, x, e)) \equiv \textbf{True}\\
&\mathsf{not_value}(\textbf{LETR}\,(r)) \equiv \textbf{True}\\
&\mathsf{not_value}(\textbf{PTUPLE}\,(r_1, r_2)) \equiv \textbf{True}
\end{aligned}$$

$\boxed{\mathsf{exp\_not\_value}(e)}$

$$\begin{aligned}
&\mathsf{exp_not_value}(v) \equiv \textbf{False}\\
&\mathsf{exp_not_value}((|e_1, e_2|)) \equiv \textbf{True}\\
&\mathsf{exp_not_value}(\mathrm{let}\ x = e_1\ \mathrm{in}\ e_2) \equiv \textbf{True}\\
&\mathsf{exp_not_value}(\mathrm{fst}\ v) \equiv \textbf{True}\\
&\mathsf{exp_not_value}(\mathrm{snd}\ v) \equiv \textbf{True}\\
&\mathsf{exp_not_value}(\textbf{case}\,v\,\textbf{of}\,\{\,\textbf{inl}\,x_1.e_1,\ \textbf{inr}\,x_2.e_2\}) \equiv \textbf{True}\\
&\mathsf{exp_not_value}(v_1\,v_2) \equiv \textbf{True}
\end{aligned}$$

$\boxed{r_1 \# r_2}$ combine results

$$\begin{aligned}
&v_1 \# v_2 \equiv (v_1, v_2)\\
&r_1 \# r_2 \equiv \textbf{PTUPLE}\,(r_1, r_2)
\end{aligned}$$

$\boxed{e_1 \Rightarrow e_2}$ small step

$$\frac{}{v \Rightarrow v}\ \text{STEP\_VALUE}$$

$$\frac{e_1 \Rightarrow e_1'}{\mathrm{let}\ x = e_1\ \mathrm{in}\ e_2 \Rightarrow \mathrm{let}\ x = e_1'\ \mathrm{in}\ e_2}\ \text{STEP\_LET\_FIRST}$$

$$\frac{}{\mathrm{let}\ x = v_1\ \mathrm{in}\ e_2 \Rightarrow e_2[v_1/x]}\ \text{STEP\_LET\_SECOND}$$

$$\frac{v_1 = \textbf{fun}\ f(x)e}{v_1\,v_2 \Rightarrow e[v_2/x][v_1/f]}\ \text{STEP\_APP}$$

$$\frac{v = (v_1, v_2)}{\mathrm{fst}\ v \Rightarrow v_1}\ \text{STEP\_FIRST}$$

$$\frac{v = (v_1, v_2)}{\mathrm{snd}\ v \Rightarrow v_2}\ \text{STEP\_SECOND}$$

$$\frac{v = \mathrm{inl}\ v_1}{\textbf{case}\,v\,\textbf{of}\,\{\,\textbf{inl}\,x_1.e_1,\ \textbf{inr}\,x_2.e_2\} \Rightarrow e_1[v_1/x_1]}\ \text{STEP\_CASE\_LEFT}$$

$$\frac{v = \mathrm{inr}\ v_2}{\textbf{case}\,v\,\textbf{of}\,\{\,\textbf{inl}\,x_1.e_1,\ \textbf{inr}\,x_2.e_2\} \Rightarrow e_2[v_2/x_2]}\ \text{STEP\_CASE\_RIGHT}$$

$$\frac{e_1 \Rightarrow e_1'}{(|e_1, e_2|) \Rightarrow (|e_1', e_2|)}\ \text{STEP\_PAR\_LEFT}$$

$$\frac{e_2 \Rightarrow e_2'}{(|e_1, e_2|) \Rightarrow (|e_1, e_2'|)}\ \text{STEP\_PAR\_RIGHT}$$

$$\frac{}{(|v_1, v_2|) \Rightarrow (v_1, v_2)}\ \text{STEP\_PAR\_JOIN}$$

$\boxed{ce_1, s_1, d_1, ct_1 \Rightarrow^1 ce_2, s_2, d_2, ct_2}$ single-processor small step

$$\frac{e_1 \Rightarrow e_2}{\textbf{RUN}\ e_1, s, d, ct \Rightarrow^1 \textbf{RUN}\ e_2, s, d, ct} \quad \text{SPSTEP\_EVAL}$$

$$\frac{}{\textbf{START}\ e, s, d, ct \Rightarrow^1 \textbf{RUN}\ e, s, d, \textbf{Current}\ e\ s} \quad \text{SPSTEP\_RUN}$$

$$\frac{\mathsf{exp\_not\_value}(e)}{\textbf{RUN}\ C[e], s, d, ct \Rightarrow^1 \textbf{RUN}\ e, \textbf{Cons}\ C\ s, d, ct} \quad \text{SPSTEP\_PUSH\_CONTEXT}$$

$$\frac{}{\textbf{RUN}\ v, \textbf{Cons}\ C\ s, d, ct \Rightarrow^1 \textbf{RUN}\ C[v], s, d, ct} \quad \text{SPSTEP\_APPLY\_CONTEXT}$$

$$\frac{}{\textbf{RUN}\ v, \textbf{Top}, [\textbf{Right}\ id\ e|d], ct \Rightarrow^1 \textbf{START}\ e, \textbf{Right}\ id, d, ct} \quad \text{SPSTEP\_POP\_TASK}$$

$$\frac{}{\textbf{RUN}\ e, s, d, ct \Rightarrow^1 \textbf{FAILED}, \textbf{Fail}, d, ct} \quad \text{SPSTEP\_FAIL}$$

$\boxed{pm_1, rm_1 \Rightarrow^m pm_2, rm_2}$ multi-processor small step

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ (|e_1, e_2|), s, d, ct) \\ \textbf{fresh}\ id\ \textbf{in}\ rm \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{START}\ e_1, \textbf{Left}\ id, [\textbf{Right}\ id\ e_2|d], ct)], rm[id := \textbf{Neither}\ s]} \quad \text{MPSTEP\_FORK}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{FAILED}, \textbf{Fail}, d, \textbf{Current}\ e\ s) \\ pm[pid_2] == (\textbf{RUN}\ v, \textbf{Top}, [], ct) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{FAILED}, \textbf{Fail}, d, \textbf{Recovered})][pid_2 := (\textbf{START}\ e, s, d, ct)], rm} \quad \text{MPSTEP\_RECOVER}$$

$$\frac{\begin{array}{l} pm[pid_1] == (ce_1, s_1, d_1, ct_1) \\ pm[pid_2] == (ce_2, s_2, d_2@t, ct_2) \\ s_1 \neq \textbf{Fail} \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (ce_1, s_1, [t|d_1], ct_1)][pid_2 := (ce_2, s_2, d_2, ct_2)], rm} \quad \text{MPSTEP\_STEAL}$$

$$\frac{\begin{array}{l} ce_1, s_1, d_1, ct_1 \Rightarrow^1 ce_2, s_2, d_2, ct_2 \\ pm[pid_1] == (ce_1, s_1, d_1, ct_1) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (ce_2, s_2, d_2, ct_2)], rm} \quad \text{MPSTEP\_LOCAL}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ v, \textbf{Init}\ id, d, ct) \\ \textbf{fresh}\ id\ \textbf{in}\ rm \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{RUN}\ v, \textbf{Top}, d, ct)], rm[id := \textbf{Finished}\ v]} \quad \text{MPSTEP\_FINISH}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ v, \textbf{Left}\ id, d, ct) \\ rm[id] == (\textbf{Neither}\ s) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{RUN}\ v, \textbf{Top}, d, ct)], rm[id := \textbf{Left}\ v\ s]} \quad \text{MPSTEP\_LEFT\_FIRST}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ v, \textbf{Left}\ id, d, ct) \\ rm[id] == (\textbf{Right}\ v_2\ s) \\ v_3 = (v, v_2) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{RUN}\ v_3, s, d, ct)], rm[id := \textbf{Finished}\ v_3]} \quad \text{MPSTEP\_LEFT\_LAST}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ v, \textbf{Right}\ id, d, ct) \\ rm[id] == (\textbf{Neither}\ s) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{RUN}\ v, \textbf{Top}, d, ct)], rm[id := \textbf{Right}\ v\ s]} \quad \text{MPSTEP\_RIGHT\_FIRST}$$

$$\frac{\begin{array}{l} pm[pid_1] == (\textbf{RUN}\ v, \textbf{Right}\ id, d, ct) \\ rm[id] == (\textbf{Left}\ v_1\ s) \\ v_3 = (v_1, v) \end{array}}{pm, rm \Rightarrow^m pm[pid_1 := (\textbf{RUN}\ v_3, s, d, ct)], rm[id := \textbf{Finished}\ v_3]} \quad \text{MPSTEP\_RIGHT\_LAST}$$

$\boxed{e \Downarrow v}$ e reduces to v (fault-free)

$$\frac{}{v \Downarrow v} \quad \textsc{REDUCE\_INIT}$$

$$\frac{\begin{array}{c} e_1 \Downarrow v_1 \\ e_2[v_1/x] \Downarrow v_2 \end{array}}{\text{let } x = e_1 \text{ in } e_2 \Downarrow v_2} \quad \textsc{REDUCE\_LET}$$

$$\frac{\begin{array}{c} v_1 = \textbf{fun } f(x)e \\ e[v_2/x][v_1/f] \Downarrow v \end{array}}{v_1\, v_2 \Downarrow v} \quad \textsc{REDUCE\_APP}$$

$$\frac{v = (v_1, v_2)}{\text{fst } v \Downarrow v_1} \quad \textsc{REDUCE\_FIRST}$$

$$\frac{v = (v_1, v_2)}{\text{snd } v \Downarrow v_2} \quad \textsc{REDUCE\_SECOND}$$

$$\frac{\begin{array}{c} v = \text{inl } v_1 \\ e_1[v_1/x_1] \Downarrow v_3 \end{array}}{\textbf{case}\, v\, \textbf{of}\, \{\, \textbf{inl}\, x_1.e_1,\ \textbf{inr}\, x_2.e_2\} \Downarrow v_3} \quad \textsc{REDUCE\_CASELEFT}$$

$$\frac{\begin{array}{c} v = \text{inr } v_2 \\ e_2[v_2/x_2] \Downarrow v_3 \end{array}}{\textbf{case}\, v\, \textbf{of}\, \{\, \textbf{inl}\, x_1.e_1,\ \textbf{inr}\, x_2.e_2\} \Downarrow v_3} \quad \textsc{REDUCE\_CASERIGHT}$$

$$\frac{\begin{array}{c} e_1 \Downarrow v_1 \\ e_2 \Downarrow v_2 \end{array}}{(|e_1, e_2|) \Downarrow (v_1, v_2)} \quad \textsc{REDUCE\_PTUPLE}$$

$\boxed{e \Downarrow^f r}$ e reduces to r (fault-prone)

$$\frac{}{v \Downarrow^f v} \quad \textsc{REDUCEF\_INIT}$$

$$\frac{\begin{array}{c} e_1 \Downarrow^f v_1 \\ e_2[v_1/x] \Downarrow^f v_2 \end{array}}{\text{let } x = e_1 \text{ in } e_2 \Downarrow^f v_2} \quad \textsc{REDUCEF\_LET}$$

$$\frac{\begin{array}{c} v_1 = \textbf{fun } f(x)e \\ e[v_2/x][v_1/f] \Downarrow^f r \end{array}}{v_1\, v_2 \Downarrow^f r} \quad \textsc{REDUCEF\_APP}$$

$$\frac{v = (v_1, v_2)}{\text{fst } v \Downarrow^f v_1} \quad \textsc{REDUCEF\_FIRST}$$

$$\frac{v = (v_1, v_2)}{\text{snd } v \Downarrow^f v_2} \quad \textsc{REDUCEF\_SECOND}$$

$$\frac{\begin{array}{c} v = \text{inl } v_1 \\ e_1[v_1/x_1] \Downarrow^f r \end{array}}{\textbf{case}\, v\, \textbf{of}\, \{\, \textbf{inl}\, x_1.e_1,\ \textbf{inr}\, x_2.e_2\} \Downarrow^f r} \quad \textsc{REDUCEF\_CASELEFT}$$

$$\frac{\begin{array}{c} v = \text{inr } v_2 \\ e_2[v_2/x_2] \Downarrow^f r \end{array}}{\textbf{case}\, v\, \textbf{of}\, \{\, \textbf{inl}\, x_1.e_1,\ \textbf{inr}\, x_2.e_2\} \Downarrow^f r} \quad \textsc{REDUCEF\_CASERIGHT}$$

$$\frac{e_1 \Downarrow^f r_1 \quad e_2 \Downarrow^f r_2 \quad r = r_1 \# r_2}{(|e_1, e_2|) \Downarrow^f r} \quad \textsc{reducef\_ptuple}$$

$$\frac{}{e \Downarrow^f \mathbf{BOT}\,(e)} \quad \textsc{reducef\_bottom}$$

$$\frac{e_1 \Downarrow^f r_1 \quad \mathsf{not\_value}(r_1)}{\text{let } x = e_1 \text{ in } e_2 \Downarrow^f \mathbf{LETL}\,(r_1, x, e_2)} \quad \textsc{reducef\_let}\mathrm{FL}$$

$$\frac{e_1 \Downarrow^f v_1 \quad e_2[v_1/x] \Downarrow^f r \quad \mathsf{not\_value}(r)}{\text{let } x = e_1 \text{ in } e_2 \Downarrow^f \mathbf{LETR}\,(r)} \quad \textsc{reducef\_let}\mathrm{FR}$$

$\boxed{r_1 \Downarrow^r r_2}$ r_1 reduces to r_2 (recovery)

$$\frac{}{v \Downarrow^r v} \quad \textsc{recover\_init}$$

$$\frac{e \Downarrow^f r}{\mathbf{BOT}\,(e) \Downarrow^r r} \quad \textsc{recover\_bottom}$$

$$\frac{r \Downarrow^r v \quad e[v/x] \Downarrow^f v_2}{\mathbf{LETL}\,(r, x, e) \Downarrow^r v_2} \quad \textsc{recover\_let}\mathrm{LS}$$

$$\frac{r \Downarrow^r r_1 \quad \mathsf{not\_value}(r_1)}{\mathbf{LETL}\,(r, x, e) \Downarrow^r \mathbf{LETL}\,(r_1, x, e)} \quad \textsc{recover\_let}\mathrm{LFL}$$

$$\frac{r \Downarrow^r v \quad e[v/x] \Downarrow^f r_2 \quad \mathsf{not\_value}(r_2)}{\mathbf{LETL}\,(r, x, e) \Downarrow^r \mathbf{LETR}\,(r_2)} \quad \textsc{recover\_let}\mathrm{LFR}$$

$$\frac{r \Downarrow^r v}{\mathbf{LETR}\,(r) \Downarrow^r v} \quad \textsc{recover\_let}\mathrm{RS}$$

$$\frac{r \Downarrow^r r_1 \quad \mathsf{not\_value}(r_1)}{\mathbf{LETR}\,(r) \Downarrow^r \mathbf{LETR}\,(r_1)} \quad \textsc{recover\_let}\mathrm{RF}$$

$$\frac{r_1 \Downarrow^r r_1' \quad r_2 \Downarrow^r r_2' \quad r = r_1' \# r_2'}{\mathbf{PTUPLE}\,(r_1, r_2) \Downarrow^r r} \quad \textsc{recover\_ptuple}$$

```
Definition rules:        53 good    0 bad
Definition rule clauses: 128 good    0 bad
```