Preservation

Part 1

Assume:

$\vdash H : \star$

$H \vdash F : \sigma$

$H ; a \vdash F$ ok

$H, F \longrightarrow H', F'$

Show:

$\vdash H' : \star$

$H' \vdash F' : \sigma$

$H' ; a \vdash F'$ ok

Shown in LaCasa

Part 2

$\vdash H : \star$

$H \vdash FS$

$H ; a \vdash FS$ ok

$H, FS \longrightarrow H', FS'$

$\vdash H' : \star$

$H' \vdash FS'$

$H' ; b \vdash FS'$ ok

Shown in LaCasa

Part 3: Tasklet reduction

Assume:

1) $\vdash H : *$
2) $H \vdash TS$
3) $H \vdash TS$ ok
4) $H, \{T\} \uplus TS \rightsquigarrow H', TS'$

Show:

$\vdash H' : *$ ✓ ⟶ Trivial as $H = H'$ or proof identical to Galasa

I $H' \vdash TS'$

II $H' \vdash TS'$ ok

Cases of $\rightsquigarrow$

E-ASYNC
E-FINISH1
E-FINISH2
E-TASK-DONE
E-BOX
E-CAPTURE
E-SWAP

E-Async

$L' = [x \to o]$ $P = \emptyset$ $\Gamma' = x : C$

- $\Gamma ; \text{ocap} \vdash t : \widetilde{C}$ by 2, T-ASYNC
- $H \vdash \Gamma' ; L'$ by WF-ENV ?
- $\vdash \Gamma' ; L' ; P$ by WF-PERM
- ⇒ $H \vdash \{T_1\}$

$\Gamma'' = \Gamma \setminus \text{Perm}[Q]$

- $\Gamma'' ; a \vdash s : \sigma$ by 2, T-ASYNC
- $H \vdash \Gamma'' ; L$ by 2, $\Gamma'' \subseteq \Gamma$, WF-ENV ?
- $\vdash H \vdash \Gamma'' ; L ; P \setminus \{p\}$ by WF-PERM ?
- ⇒ $H \vdash \{T_2\}$

⇒ $H \vdash \{T_1, T_2\} \uplus TS$ I

I $H' \vdash TS'$ checks for coherency between types and heap+variable bindings. The other rules are likely easy to prove and any problems require only minor changes to the typing rules.

II $H' \vdash TS'$ ok

H |- {T1, T2} + TS ok
- f < f' trivial, no FINISH gets added
- No two tasks await same id. Trivial

Two cases:
- Async was checked with ocap
-> T1 + T2 are also checked with ocap
-> TS' ok
- Async wasn't checked with ocap
-> T1 is ocap, T2 not
-> TS' ok because we still have only one non-ocap active task

E-FINISH1
f < f' by definition of f' fresh
No two tasks await same id by def of fresh
Current task becomes inactive, awaits new task
Currently ocap -> new also ocap -> no problem
Currently not ocap -> current becomes inactive, new also not ocap
-> no problem

E-FINISH2
f < f' trivial
No two tasks await same id trivial
if task was not ocap -> no non-ocap active tasks exists (only it and its direct ancestors are allowed to be non-ocap. I.e. if no active tasks exist, only this inactive task does) -> This task becomes the new non-ocap active task

E-TASK-DONE
f < f' trivial
No two tasks await same id trivial
Only one non-ocap active task:
If task was non-ocap
-> No active non-ocap task exists
-> Its parent might become next (see E-FINISH2), otherwise everything is ocap

E-BOX/CAPTURE/SWAP
f < f' trivial
No two tasks await same id trivial
only one non-ocap active task trivial. Removing tasks does not invalidate any invariant