# Preservation

## Part 1

| Assume: | Show: |
|---|---|
| $\vdash H : *$ | $\vdash H' : *$ |
| $H \vdash F : \sigma$ | $H' \vdash F' : \sigma$ |
| $H ; a \vdash F$ ok | $H' ; a \vdash F'$ ok |
| $H, F \rightarrow H', F'$ | |

Shown in LaCasa

## Part 2

| | |
|---|---|
| $\vdash H : *$ | $\vdash H' : *$ |
| $H \vdash FS$ | $H' \vdash FS'$ |
| $H ; a \vdash FS$ ok | $H' ; b \vdash FS'$ ok |
| $H, FS \twoheadrightarrow H', FS'$ | |

Shown in LaCasa

## Part 3: Taskset reduction

Assume:

1) $\vdash H : *$
2) $H \vdash TS, WS$
3) $H, TS, WS$ ok
4) $H, TS, WS \rightsquigarrow H', TS', WS'$

Show:

$\vdash H' : *$ ✓ $\longrightarrow$ Trivial as $H = H'$ or proof identical to Lalasa

I $H' \vdash TS', WS'$

II $H', TS', WS'$ ok

Cases of $\rightsquigarrow$

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

E-Async

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

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

$\Gamma'' = \Gamma \setminus Perm[Q]$

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

$\to H \vdash \{T_1, T_2\} \uplus TS, WS'$ 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
ID-ordering trivial
ID-Uniqueness trivial
isolation see other file

H |- {T1, T2} + TS, WS ok
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

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

if task was not ocap -> no non-ocap active tasks exist (only it and its direct ancestors are allowed to be non-ocap. I.e. if no active tasks exist, only this in-active task does) -> This task becomes the new non-ocap active task

E-TASK-DONE

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

only one non-ocap active task trivial.
Removing tasks does not invalidate any invariant

We also have to show preservation of Task-level invariants when applying the frame and frame stack level rules.
idOrdering(WS): WS is not modified by any rule other than those mentioned above
finishUniqueness(WS): Same
allButOneOcap(TS, WS): Framestacks preserve ok-ness, effect is also preserved
isolated(H, TS, WS): See isolation proof