Isolation

Assume:

1) $H, \{T\} \uplus TS \leadsto H, TS'$

2) $isolated(H, \{T\} \uplus TS)$

3) $\vdash H : *$

4) $H \vdash TS$?

This was never explicitly used (or defined). What well-formedness do we actually need?
No tasks awaits a itself (or larger await-cycles) etc. is reasonable but what else?
Is this even necessary for isolation?

Show

$isolated(H, TS')$

Proof by case distinction on used reduction rule "$\leadsto$":

E-FINISH2:

$T = (f, k, \langle FINISH\ f' \rangle \circ FS)$ $\quad T' = (f, k, FS)$ $\quad TS' = \{T'\} \uplus TS$

$\forall (g, k', GS) \in TS.\ isolated(H, \langle FINISH\ f' \rangle \circ FS, GS) \vee awaits(\{T\} \uplus TS, f', g) \vee GS = \langle FINISH\ g' \rangle \circ GS' \wedge awaits(\{T\} \uplus TS), g', f)$ by 2

$isolated(H, \langle FINISH\ f' \rangle \circ FS, GS) \Rightarrow isolated(H, FS, GS)$ by ISO-FS, ACC-FS

$awaits(\{T\} \uplus TS, f', g) \Rightarrow false$ by awaitsBlocksReduction

$awaits(\{T\} \uplus TS, g', f) \Rightarrow awaits(\{T'\} \uplus TS, g', f)$ by $awaits(\{(f, k, FS)\} \uplus TS, g, f) = awaits(\{(f, k', FS')\} \uplus TS, g, f)$

$\forall (g, k, GS) \in TS.\ isolated(H, FS, GS) \vee awaitsT(TS, (g, GS), T_2)$

E-FINISH1:

$T = (f, k, F \circ FS)$ $\quad F = \langle L, let\ x = finish\ \{t\}\ in\ s, P \rangle^l$ $\quad U = (f', true, \langle L, t, P \rangle^\varepsilon)$

$T' = (f, k, \langle FINISH\ f' \rangle^m \circ \langle L, s, P \rangle^l \circ FS)$ $\quad TS' = \{T', U\} \uplus TS$

$isolated(H, \{T\} \uplus TS) \Rightarrow isolated(H, \{U\} \uplus TS)$ by ISO-FS, ACC-FS, FINISH1NewTaskCopiesVars

$isolated(H, \{T\} \uplus TS) \Rightarrow isolated(H, \{T'\} \uplus TS)$ by T'-def

$\langle L, s, P \rangle^l \approx F$

$\langle FINISH\ f' \rangle \circ FS \lessapprox FS$

where $\lessapprox$ means "is isolated from more tasks"/ more constraind

$isolated(H, \{T', U\})$ by FINISH 1-def, ISO-TS $\quad$ (T' awaits U)

$isolated(H, \{T\} \uplus TS) \wedge isolated(H, \{U\} \uplus TS) \wedge isolated(H, \{T, U\}) \Rightarrow isolated(H, \{T, U\} \uplus TS)$

should hold in general
but certainly does here
(Potential problem: addition of
a task adds an await chain.
e.g. T1 awaits T2 awaits T3
Without T2, no await. But adding
it should never break isolation

$isolated(H, \{T', U\} \uplus TS)$

$TS' = \{T'\} \uplus TS$

$\{q \mid accRoot(o,T) \wedge reach(H,o,q)\}$

$= \{q \mid accRoot(o,T') \wedge reach(H,o,q)\}$

= Reachables preserved => Separation preserved
This applies to: E-Null, E-Var, E-Select, E-Assign, E-Return1, E-Return2, E-Open

Same reasoning applies to E-Box, E-Capture, E-Swap
+ they drop some tasks. But:
isolation(H,TS) and (TS' smaller TS) ==> isolation(H,TS')

E-Task-Done similarly

E-Async:
If T + TS isolated ==> T1 + TS isolated and T2 + TS isolated (they each are "smaller" than T as they only contain parts of Ts bindings

T1 + T2 isolated by definition. T1 has no permission for box(x). T2 has only x.

E-Invoke, E-New: New value guaranteed separated from everything else through OCAP.
But its not trivial to show this!