Isolation

Assume: Show

1) $H, TS, ws \rightsquigarrow H', TS', ws'$ $\quad$ isolated(H', TS', ws')

2) isolated(H, TS, ws)

3) $\vdash H : *$

4) $H \vdash TS$

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

└ E-FINISH2: $H, TS, \{(T, f')\} \uplus ws' \rightsquigarrow H, \{T\} \uplus TS, ws'$

$T = (f, k, FS) \quad T' = (g, k', GS)$

$\forall T' \in TS + ws.$ isolated$(H, FS, GS) \vee$ blocks$(ws, g, f) \vee$ blocks(ws, f, g) by 2

((T, f') reduced $\Rightarrow \neg$ blocks(ws, g, f))

$\forall T' \in TS + ws.$ isolated$(H, FS, GS) \vee$ blocks(ws, f, g)

$\Rightarrow$ isolated

E-FINISH1:

$T = (f, k, F \circ FS) \quad F = \langle L, \text{let } x = \text{finish } \{\,\} \text{ in } s, P \rangle'$

$T_1 = (f', \text{true}, \langle L, t, P \rangle')$ $\quad T_2 = (f, k, \langle L[x := \text{null}], s, P \rangle \circ FS)$ $\quad TS' = \{T_1\} \uplus TS$, $ws' = \{(T_2, f')\} \uplus ws$

- isolated(H, TS', ws) by T_1-def $\quad \langle L, t, P \rangle' \approx F$
- isolated(H, TS, ws') by T_2-def $\quad \langle L, s, P \rangle' \approx F$

where $\approx$ means "is isolated as much as"

- isolated$(H, \{T_1\}, \{(T_2, f')\})$ by blocks(ws', f', f)
- isolated$(H, TS', ws) \wedge$ isolated$(H, TS, ws') \wedge$ isolated$(H, \{T_1\}, \{T_2\}) \Rightarrow$ isolated(H, TS', ws')

should hold in general but certainly does here (Potential problem: addition of a task adds a block 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,WS) and (WS' smaller WS) ==> isolation(H,TS,WS')

E-Task-Done similarly

E-Async:
Similar to E-Finish1
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: With the introduced check it should be trivial. Reachset should be identical
E-New: With fixed invoke also fine. New class -> all null. A non-ocap class is only problematic on invokes which we check separately

sep(H, o, o'): reach(H, o, q) and reach(H, o', q') ==> q =/= q'
reachables(H, o): {x. reach(H, o, x)}
reachables(H, FS): Union {reachables(H, o) | o in accRoot(FS)}

isolated(H, FS, FS'):
o in accRoot(FS) and o' in accRoot(FS') ==> sep(H, o, o')
isolatedSet(H, FS, FS'):
o in accRoot(FS) and o' in accRoot(FS')
==> reachables(H, o) intersect reachables(H,o') == emptyset
isolatedSet ==> isolated by definition of reachables:
forall q in reachables(o), q' in reachables(o'). q =/= q'

for one FS. reachables(H, FS) = reachables(H', FS')
and the rest GS. reachables(H, GS) = reachables(H', GS)
then isolatedSet(H, all GS + FS) ==> isolatedSet(H', all GS+FS')

alternative: add new object but object only in one reachables

Issue: E-INVOKE expects ocap. So all-but-one-ocap required for Isolation