A. Proof of Preservation

Theorem. *If* $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,S$ *and* $H\,\mid\,S\,\longrightarrow\,H'\,\mid\,S'$*, then there exist* Σ' *and* Γ' *such that* $\Sigma,\,\Sigma'\,;\,\Gamma,\Gamma'\,\vdash\,H'\,\mid\,S'$.

Suppose that $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,S$ and $H\,\mid\,S\,\longrightarrow\,H'\,\mid\,S'$. We perform a case analysis on the rule used to perform the reduction.

- Rule

$$\frac{\begin{array}{l} F(\mathit{this}) = p \\ H(p) = C\,(f_1 = w_1\,p_1\,;\,..\,;\,f_n = w_n\,p_n) \\ F(x) = w'\,p' \end{array}}{H\,\mid\,\langle\,F\,\mid\,x = \mathit{this}\,.\,f_i\,;\,s\,\rangle\,S\,\longrightarrow\,H\,\mid\,\langle\,F\,[x \mapsto w'\,p_i]\,\mid\,s\,\rangle\,S}$$

Since $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,\langle\,F\,\mid\,x = \mathit{this}\,.\,f_i\,;\,s\,\rangle\,S$ is valid, it holds that $\Gamma\,(\mathit{this}) = C$, $\mathit{fields}\,(C) = t_1\,f_1\,..\,t_k\,f_k$, $\Gamma\,(x) = t'$ and $ti <: t'$. Let Σ' and F' be empty. We must show that $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,\langle\,F\,[x \mapsto w_i\,p_i]\,\mid\,s\,\rangle\,S$. The only non-trivial sub-goal is $\Sigma\,;\,\Gamma\,\vdash\,F\,[x \mapsto w_i\,p_i]$, which in turn requires to prove $\Sigma\,\vdash\,w_i\,p_i : t_i$. We know that $F(y) = p$ and $H(p) = C\,(f_1 = w_1\,p_1\,;\,..\,;\,f_n = w_n\,p_n)$: since $\Sigma\,\vdash\,H$ it holds that $\Sigma\,\vdash\,w_i\,p_i : t_i$.

- Rule

$$\frac{\begin{array}{l} F(\mathit{this}) = p \\ F(x) = w'\,p' \\ H(p) = C\,(f_1 = sv_1\,;\,..\,;\,f_n = sv_n) \\ \mathit{fields}\,(C) = t_1\,f_1\,..\,t_n\,f_n \\ sv = [\![\,t_i\,]\!]\,p' \end{array}}{H\,\mid\,\langle\,F\,\mid\,\mathit{this}\,.\,f_i = x\,;\,s\,\rangle\,S\,\longrightarrow\,H\,[p \mapsto (H\,(p)\,.\,f_i \mapsto sv)]\,\mid\,\langle\,F\,\mid\,s\,\rangle\,S}$$

Since $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,\langle\,F\,\mid\,\mathit{this}\,.\,f_i = x\,;\,s\,\rangle\,S$ is valid, it holds that $\Gamma\,(\mathit{this}) = C$, $\mathit{fields}\,(C) = t_1\,f_1\,..\,t_k\,f_k$, and $\Gamma\,(x) <:\,t_i$. Let Σ' and F' be empty. The non-trivial goal is $\Sigma\,\vdash\,H\,[p \mapsto (H\,(p)\,.\,f_i \mapsto sv)]$, and in particular the two sub-goals $\Sigma(p) = C$ and $\Sigma\,\vdash\,sv : t_i$. The first holds because $H(p) = C\,(f_1 = sv_1\,;\,..\,;\,f_n = sv_n)$ and because $\Sigma\,\vdash\,H$. The second holds because $F(x) = sv$ and $\Sigma\,;\,\Gamma\,\vdash\,F$.

- Rule

$$\frac{\begin{array}{l} F(y) = p \\ \mathbf{ptype}\,(H,\,p) = C \\ \mathbf{mbody}\,(m,\,C) = x_1\,..\,x_n\,.\,s_0\,;\,\mathbf{return}\,x_0 \\ \mathbf{mtype}\,(m,\,C) = t_1\,..\,t_n\,\rightarrow\,t \\ F(y_1) = w_1\,p_1 \quad .. \quad F(y_n) = w_n\,p_n \\ sv_1 = [\![\,t_1\,]\!]\,p_1 \quad .. \quad sv_n = [\![\,t_n\,]\!]\,p_n \\ F(x) = w'\,p' \\ \mathit{cast} = \mathbf{w2c}\,(w') \end{array}}{H\,\mid\,\langle\,F\,\mid\,x = y\,.\,m\,(y_1\,..\,y_n)\,;\,s\,\rangle\,S\,\longrightarrow\,H\,\mid\,\langle\,[]\,[x_1 \mapsto sv_1\,..\,x_n \mapsto sv_n]\,[\mathit{this} \mapsto p]\,\mid\,s_0\,;\,\mathbf{return}\,x_0\,\rangle\,\langle\,F\,\mid\,x = \mathit{cast}\,\mathit{ret}\,;\,s\,\rangle\,S}$$

Since $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,\langle\,F\,\mid\,x = y\,.\,m\,(y_1\,..\,y_n)\,;\,s\,\rangle\,S$ is valid, it holds that $\Gamma\,(y) = C$ and $\mathbf{mtype}\,(m,\,C) = t_1\,..\,t_n\,\rightarrow\,t'$. For all i such that $\mathbf{concr}\,(t_i)$ holds, we have $\Gamma\,\vdash\,y_i <:\,t_i$; otherwise we have $G| - wipi : ti$. Also, let $\Gamma\,(x) = t$: we have $\Sigma\,\vdash\,w'\,p' : t$, and if $\mathbf{concr}\,(t)$, then $t' <: t$. Let Σ' be empty. Let $\Gamma' = x_1 : t_1,\,..,\,x_n : t_n,\,\mathit{this}{:}C,\,\mathit{ret}{:}t'$. After some unfoldings, the non-trivial goals left are

$$\begin{array}{ll} 1) & \Sigma\,;\,\Gamma,\Gamma'\,\vdash\,[]\,[x_1 \mapsto sv_1\,..\,x_n \mapsto sv_n]\,[\mathit{this} \mapsto p] \\ 2) & \Gamma,\Gamma'\,\vdash\,s_0 \\ 3) & \Gamma,\Gamma'\,\vdash\,x = \mathit{cast}\,\mathit{ret} \end{array}$$

For the goal 1), for each $1 \leq i \leq n$, we distinguish two cases. If $\mathbf{concr}\,(t_i)$, since $\Gamma\,\vdash\,y_i <:\,t_i$ and $F(y_i) = p_i$ (the wrapper w_i is empty) and $\Sigma\,;\,\Gamma\,\vdash\,F$, we have $\Sigma\,\vdash\,p_i : t_i$. If not, then by construction of $[\![-]\!]$, it holds $\Sigma\,\vdash\,[\![\,t_i\,]\!]\,p_i : t_i$. We conclude that for all i it holds $\Sigma\,;\,\Gamma,\Gamma'\,\vdash\,[]\,[x_i \mapsto [\![\,t_i\,]\!]\,p_i]$ by construction of Γ'. The constraint $\Sigma\,;\,\Gamma,\Gamma'\,\vdash\,[]\,[\mathit{this} \mapsto p]$ is satisfied instead because $\mathbf{ptype}\,(H,\,p) = C$. The goal follows.

The goal 2) is true because the method body s_0 was well-typed in (a subset of) the environment Γ'.

For the goal 3), we distinguish two cases. If $\mathbf{concr}\,(t)$, then 3) holds because $\Gamma'\,\vdash\,\mathit{ret} : t'$ and $t' <:\,t$, where t is the type of x in Γ. Otherwise, w' is not empty, and by definition of **w2c** and since $\Sigma\,\vdash\,w'\,p' : t$, we conclude $\Gamma,\Gamma'\,\vdash\,x = \mathit{cast}\,\mathit{ret}$.

- Rule

$$\frac{\begin{array}{l} F(y) = (\textbf{like}\ C)\ p \\ \textbf{ptype}\,(H,\ p) = D \\ \textbf{mbody}\,(m,\ D) = x_1 \mathinner{..} x_n \,.\, s_0 \,;\, \textbf{return}\, x_0 \\ \textbf{mtype}\,(m,\ C) = t_1 \mathinner{..} t_n \ \rightarrow\ t \\ \textbf{mtype}\,(m,\ D) = t'_1 \mathinner{..} t'_n \ \rightarrow\ t' \\ \forall i \,.\, t_i <:\ t'_i \ \vee\ t'_i = \textbf{dyn} \\ (\textbf{concr}\,(t)\ \wedge\ \textbf{concr}\,(t'))\ \Rightarrow\ t' <:\ t \\ F(y_1) = w_1\, p_1 \quad .. \quad F(y_n) = w_n\, p_n \\ sv_1 = [\![\, t'_1 \,]\!]\, p_1 \quad .. \quad sv_n = [\![\, t'_n \,]\!]\, p_n \\ F(x) = w'\, p' \\ \mathit{cast} = \textbf{w2c}\,(w') \end{array}}{H \mid \langle\, F \mid x = y \,.\, m\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S \ \longrightarrow\ H \mid \langle\, [\,]\,[x_1 \mapsto sv_1 \mathinner{..} x_n \mapsto sv_n]\,[\mathit{this} \mapsto p] \mid s_0 \,;\, \textbf{return}\, x_0 \,\rangle\, \langle\, F \mid x = \mathit{cast}\,(t)\, \mathit{ret}\,;\ s \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \vdash H \mid \langle\, F \mid x = y \,.\, m\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S$ is valid, it holds that $\Gamma\,(y) = \textbf{like}\ C$. For all i such that $\textbf{concr}\,(t_i)$ holds, we have $\Gamma \vdash y_i <:\ t_i$; otherwise we have $\Sigma \vdash w_i\, p_i : t_i$ (and the w_i wrapper is not empty). Let $\Gamma' = x_1 : t'_1, .., x'_n : t'_n, \mathit{this} : D, \mathit{ret}{:}t$. After some unfoldings, the non-trivial goals left are

$$\begin{array}{ll} 1) & \Sigma\,;\,\Gamma, \Gamma' \vdash [\,]\,[x_1 \mapsto sv_1 \mathinner{..} x_n \mapsto sv_n]\,[\mathit{this} \mapsto p] \\ 2) & \Gamma, \Gamma' \vdash s_0 \\ 3) & \Gamma, \Gamma' \vdash x = \mathit{cast}\,(t)\, \mathit{ret} \end{array}$$

For the goal 1), for each $1 \leq i \leq n$, we distinguish two cases. If $\textbf{concr}\,(t_i)$, since $\Gamma \vdash y_i <:\ t_i$ and $F(y_i) = p_i$ (the wrapper w_i is empty) and $\Sigma\,;\,\Gamma \vdash F$, we have $\Sigma \vdash p_i : t_i$ and in turn $\Sigma \vdash p_i : t'_i$. If not, then by construction of $[\![-]\!]$, it holds $\Sigma \vdash [\![\, t_i \,]\!]\, p_i : t_i$. We conclude that for all i it holds $\Sigma\,;\,\Gamma, \Gamma' \vdash [\,]\,[x_i \mapsto [\![\, t'_i \,]\!]\, p_i]$ by construction of Γ'. The constraint $\Sigma\,;\,\Gamma, \Gamma' \vdash [\,]\,[\mathit{this} \mapsto p]$ is satisfied instead because $\textbf{ptype}\,(H,\ p) = C$. The goal follows.

The goal 2) is true because the method body s_0 was well-typed in (a subset of) the environment Γ'.

For the goal 3), we distinguish two cases. If $\textbf{concr}\,(t) \wedge \textbf{concr}\,(t')$, then let t'' be the type of x in Γ. 3) holds because $\Gamma' \vdash \mathit{ret} : t$ and $t' <: t <: t''$. Otherwise, w' is not empty, and by definition of $\textbf{w2c}$ and since $\Sigma \vdash w'\, p' : t$, we conclude $\Gamma, \Gamma' \vdash x = \mathit{cast}\,(t)\, \mathit{ret}$.

XXXXXXXXXXXXXXXXXXXXX

Since $\Sigma\,;\,\Gamma \vdash H \mid \langle\, F \mid x = y \,.\, m\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S$ is valid, it holds that $\Gamma\,(y) = \textbf{like}\ C$, $\textbf{mtype}\,(m,\ C) = t_1 \mathinner{..} t_n \ \rightarrow\ t'$, $\Gamma \vdash y_i <:\ t_i$ for all $1 \leq i \leq k$, and $\Gamma\,(x) = t$ where $t' <:\ t$. The condition $\textbf{mtype}\,(m,\ C) = \textbf{mtype}\,(m,\ D)$ guarantees that the method actually invoked offers the same type interface than the method expected. It is then possible to conclude with the same argument of the previous case.

- Rule

$$\frac{\begin{array}{l} F(y) = (\textbf{dyn})\ p \\ \textbf{ptype}\,(H,\ p) = C \\ \textbf{mbody}\,(m,\ C) = x_1 \mathinner{..} x_n \,.\, s_0 \,;\, \textbf{return}\, x_0 \\ \textbf{mtype}\,(m,\ C) = t_1 \mathinner{..} t_n \ \rightarrow\ t \\ F(y_1) = w_1\, p_1 \quad .. \quad F(y_n) = w_n\, p_n \\ \forall i \,.\, \textbf{concr}\,(t_i)\ \Rightarrow\ \textbf{svtype}\,(H,\ w_i\, p_i) <:\ t_i \\ sv_1 = [\![\, t_1 \,]\!]\, p_1 \quad .. \quad sv_n = [\![\, t_n \,]\!]\, p_n \end{array}}{H \mid \langle\, F \mid x = y \,.\, m\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S \ \longrightarrow\ H \mid \langle\, [\,]\,[x_1 \mapsto sv_1 \mathinner{..} x_n \mapsto sv_n]\,[\mathit{this} \mapsto p] \mid s_0 \,;\, \textbf{return}\, x_0 \,\rangle\, \langle\, F \mid x = (\textbf{dyn})\, \mathit{ret}\,;\ s \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \vdash H \mid \langle\, F \mid x = y \,.\, m\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S$ is valid, it holds that $\Gamma\,(x) = \textbf{dyn}$. Let Σ' be empty, and let $\Gamma' = x_1 : t_1,\ ..,\ x_n : t_n,\ \mathit{this}{:}C, \mathit{ret}{:}t'$. After some unfoldings (and ignoring the cases proved by the same argument that the concrete type case) the non-trivial goals left are:

$$\begin{array}{ll} 1) & \Sigma\,;\,\Gamma, \Gamma' \vdash [\,]\,[x_1 \mapsto sv_1 \mathinner{..} x_n \mapsto sv_n]\,[\mathit{this} \mapsto p] \\ 2) & \Gamma, \Gamma' \vdash x = (\textbf{dyn})\, \mathit{ret} \end{array}$$

for some $\Gamma''; \Gamma'''$. Goal 1) holds because for all i such that $\textbf{concr}\,(t_i)$, the proper type constraint is enforced by the dynamic check $\textbf{svtype}\,(H,\ sv_i) <:\ t_i$, while for the other indexes the type constraint is satisfied because of the wrapper $[\![\, t_i \,]\!]$. Goal 2) is true because of the $(\textbf{dyn})$ cast.

- Rule

$$\frac{\begin{array}{l} p \text{ fresh for } H \\ \mathit{fields}\,(C) = t_1\, f_1 \mathinner{..} t_n\, f_n \\ F(y_1) = w_1\, p_1 \quad .. \quad F(y_n) = w_n\, p_n \\ sv_1 = [\![\, t_1 \,]\!]\, p_1 \quad .. \quad sv_n = [\![\, t_n \,]\!]\, p_n \end{array}}{H \mid \langle\, F \mid x = \textbf{new}\ C\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S \ \longrightarrow\ H\,[p \mapsto C\,(f_1 = sv_1\,;\,..\,;\, f_n = sv_n)] \mid \langle\, F\,[x \mapsto p] \mid s \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \vdash H \mid \langle\, F \mid x = \textbf{new}\ C\,(y_1 \mathinner{..} y_n)\,;\ s \,\rangle\, S$ is valid, it holds that $\Gamma\,(x) = C$, $\mathit{fields}\,(C) = t_1\, f_1 \mathinner{..} t_n\, f_n$, and for all i such that $\textbf{concr}\,(t_i)$, $\Gamma \vdash y_i <:\ t_i$. Let $\Sigma' = p : C$ for p fresh, and let $\Gamma' = x : t$. The environments Σ, Σ' and Γ, Γ' are well-formed. After some unfoldings, the non-trivial goals left are:

$$1) \qquad \Sigma,\, \Sigma' \,;\, \Gamma, \Gamma' \;\vdash\; F\,[x \mapsto p]$$
$$2) \qquad \Sigma,\, \Sigma' \;\vdash\; H\,[p \mapsto C\,(f_1 = sv_1 \,;\, ..\,;\, f_n = sv_n)]$$

Goal 1) amounts to show that $\Sigma,\, \Sigma' \;\vdash\; p : C$, which is true because of the static semantics. For goal 2) we must show that $\Sigma,\, \Sigma' \;\vdash\; sv_i : t_i$. This follows from $\Sigma,\, \Sigma' \,;\, \Gamma, \Gamma' \;\vdash\; F, \Gamma \;\vdash\; y_i <:\; t_i$ and $F(y_i) = sv_i$.

- Rule

$$\frac{}{H \mid \langle\, F_0 \mid \mathbf{return}\, x \,\rangle\, \langle\, F_1 \mid s_1 \,\rangle\, S \;\longrightarrow\; H \mid \langle\, F_1\,[ret \mapsto F_0(x)] \mid s_1 \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \;\vdash\; H \mid \langle\, F_0 \mid \mathbf{return}\, x \,\rangle\, \langle\, F_1 \mid s_1 \,\rangle\, S$ is valid, it holds that $\Gamma\,(x) = \Gamma\,(ret)$. Let Σ' and Γ' be empty. Since $\Sigma\,;\,\Gamma \;\vdash\; F_0$ and $\Gamma\,(x) = \Gamma\,(ret)$, it holds that $\Sigma\,;\,\Gamma \;\vdash\; F_1\,[ret \mapsto F_0(x)]$. The result follows.

- Rule

$$\frac{}{H \mid \langle\, F \mid x =\; y\,;\, s \,\rangle\, S \;\longrightarrow\; H \mid \langle\, F\,[x \mapsto F(y)] \mid s \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \;\vdash\; H \mid \langle\, F \mid x \;=\; y\,;\, s \,\rangle\, S$ is valid, it holds that $\Gamma\,(y) \;=\; \Gamma\,(x)$. Let Σ', Γ' be empty. The only non-trivial goal is $\Sigma\,;\,\Gamma \;\vdash\; F\,[x \mapsto F(y)]$. Since $\Sigma\,;\,\Gamma \;\vdash\; F$, we know that $\Sigma \;\vdash\; sv : \Gamma\,(y)$, and the result follows because $\Gamma\,(y) = \Gamma\,(x)$.

- Rule

$$\frac{\begin{array}{l} F(y) = w\,p \\ \mathbf{ptype}\,(H,\, p) = D \\ D <:\; C \end{array}}{H \mid \langle\, F \mid x =\; (C)\,y\,;\, s \,\rangle\, S \;\longrightarrow\; H \mid \langle\, F\,[x \mapsto p] \mid s \,\rangle\, S}$$

Since $\Sigma\,;\,\Gamma \;\vdash\; H \mid \langle\, F \mid x =\; (C)\,y\,;\, s \,\rangle\, S$ is well-typed, it holds that $\Gamma\,(x) =\; C$. It is trivial to satisfy the goal $\Sigma\,;\,\Gamma \;\vdash\; F\,[x \mapsto\; p]$ since $\mathbf{ptype}\,(H,\, p) = D$ and $D <:\; C$. The cases for cast to **like** C and $\mathbf{dyn}$ are similar.

B. Proof of Progress

Theorem. *If a well-typed configuration $\Sigma\,;\,\Gamma \;\vdash\; H \mid \langle\, F \mid s \,\rangle\, S$ is stuck, that is $H \mid \langle\, F \mid s \,\rangle\, S \not\longrightarrow$, then the statement s is of the form $x = y\,.\,m\,(y_1\,..\,y_n)\,;\, s'$ and $\Gamma\,(y)$ is $\mathbf{dyn}$ or **like** C for some C, or s is of the form $x = (C)y; s'$ and $F(y) = w\,p$ with **ptype**$(H,p) \not<:\; C$.*

The proof is by structural induction on the length of s (again, we treat statements as lists).

$x = \mathit{this}\,.\,f_i\,;\, s$ By unfoldings of the initial assumption due to the well-formedness rules (a), $\Sigma \vdash H$, (b) $\Sigma;\Gamma \vdash F$, (c) $\Gamma \vdash x = \mathit{this}.f_i$, and (d) $\Sigma;\Gamma \vdash H \mid S$. By (c), $x, \mathit{this} \in \mathit{dom}(\Gamma)$ so by (b) and $F(x) = sv_1$ and $F(y) = sv_2$. Remark that $sv_2 = p$ (as bindings for this in the stack are created only when a method is invoked). We then also know that $\Gamma(y) = C$, that the field f_i exists in C and that $H(p) = D[...]$, where $D <: C$. This guarantees that the field f_i exists in the object pointed to by p, and the rule RED_FIELD can reduce.

$\mathit{this}\,.\,f_i = x\,;\, s$ Same reasoning as above.

$x = y_0.m(y_1..y_n); s$ By unfoldings of the induction hypothesis due to the well-formedness rules (a) $\Sigma \;\vdash\; H$, (b) $\Sigma;\Gamma \;\vdash\; F$, (c) $\Gamma \;\vdash\; x \;=\; y.m(y_1..y_n); s$, and (d) $\Sigma;\Gamma \;\vdash\; H \mid S$. By (c) and trivial unfoldings, $y_0..y_n \;\in\; \mathit{dom}(\Gamma)$, so by (b), $F(y_i) \;=\; sv_i$ for $i = 1..n$, i.e,. all local variables referred to by the statement exist on the current stack frame.

If $sv_0 = p$, then, by WF-rules for frames $\Gamma(y_0) = C$ for some C and $\Sigma \vdash p : C$. By WF-rule for heaps, $H(p) = C'(...)$ s.t. $C' <: C$, i.e., $\mathbf{ptype}(H, p) \;=\; C'$. By subclassing rules, C has method m implies C' has method m with same signature. Thus, the **mbody** lookup will succeed. And arities will be correct from (c). So the rule RED_CALL_LIKE can reduce. If $sv_0 \;=\; (\mathit{dyn})\; p$, then either the rule RED_CALL_DYN reduces, or one of tests **svtype**$(H, w_i\; p_i)$ fails, and in this case the configuratin is stuck, and the first conclusion of the theorem applies. If $sv_0 \;=\; (\mathit{like}\; C)\; p$, then either the rule RED_CALL_LIKE reduces, or **mtype**(m, C) and **mtype**(m, D) are not compatible, the configuration is stuck, and the first conclusion of the theorem applies.

$\mathbf{skip}\,;\, s$ The statement $\mathbf{skip}$ always reduce.

$x = \mathbf{new}\; C\,(y_1\,..\,y_n)\,;\, s$ By unfoldings of the initial assumption, (a) there exists Γ, Γ' s.t., $\Gamma \vdash t\; x = \mathbf{new}\; C(y_1..y_n)$ and (b) $\Sigma;\Gamma \vdash F$. By (a) NEW and TS-VAR, $y_i \;\in\; \mathit{dom}(\Gamma)$ for $i \;=\; 1..n$. By (b) and WF-SF-STACK-FRAME, $y_i \;\in\; \mathit{dom}(F)$ for $i \;=\; 1..n$. Thus, all the necessary variables are present in F, and rule RED_NEW can reduce.

$x =\; y\,;\, s$ Similar to the case above: the well-formedness constraints ensure that y is defined in the current stackframe.

$x =\; (t)\,y\,;\, s$ By unfoldings of the initial assumption due to the well-formedness rules, (a) exists Γ s.t., $\Gamma \vdash x = (t)\; y \triangleright \Gamma$, and (b) $\Sigma;\Gamma \vdash F$. All matching type rules require (indirectly, via T-VAR) $x \in \mathit{dom}(\Gamma)$ and $y \in \mathit{dom}(\Gamma)$. By (b) and WF-SF-STACK-FRAME, $F(y) = w\; p$ for some $w\; p$. If t is **like** C or $\mathbf{dyn}$, then the rule RED_CAST_OTHER can reduce. If t is a concrete type C and (**ptype**(H,p) $<:\; C$, then the rule RED_CAST_CLASS_OK can reduce; otherwise the configuration is stuck because s is of the form $x = (C)y; s'$ and $F(y) = w\; p$ with **ptype**$(H, p) \not<:\; C$.

$\mathbf{return}\; y$ We can show that $F(y)$ from the well-formedness conditions, following the same reasoning as above. Then, the rule RED-RETURN can reduce.

C. Proof of Compilation

Theorem. *Let $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,S$ be a well-typed source configuration':*

1. *if $H\,\mid\,S\;\longrightarrow\;H'\,\mid\,S'$, then $[\![\,\Gamma,\,H\,\mid\,S\,]\!]\;\longrightarrow\;[\![\,\Gamma,\,H'\,\mid\,S'\,]\!]$;*
2. *conversely, if $[\![\,\Gamma,\,H\,\mid\,S\,]\!]\;\longrightarrow\;H''\,\mid\,S''$, then there exists a well-typed source configuration $\Sigma'\,;\,\Gamma'\,\vdash\,H'\,\mid\,S'$ such that $H\,\mid\,S\;\longrightarrow\;H'\,\mid\,S'$ and $[\![\,\Gamma',\,H'\,\mid\,S'\,]\!] = H''\,\mid\,S''$.*

Given a well-typed source configuration $\Sigma\,;\,\Gamma\,\vdash\,H\,\mid\,S$, we perform a case analysis on the reduction rules that apply.

- Rule

$$\frac{\begin{array}{l} F(\mathit{this}) = p \\ F(x) = w'\,p' \\ H(p) = C\,(f_1 = sv_1\,;\,..\,;\,f_n = sv_n) \\ \mathit{fields}\,(C) = t_1\,f_1\,..\,t_n\,f_n \\ sv = [\![\,t_i\,]\!]\,p' \end{array}}{H\,\mid\,\langle\,F\,\mid\,\mathit{this}\,.\,f_i = x\,;\,s\,\rangle\,S\;\longrightarrow\;H\,[p\mapsto(H\,(p)\,.\,f_i\mapsto sv)]\,\mid\,\langle\,F\,\mid\,s\,\rangle\,S}$$

  In the compiled configuration $[\![\,\Gamma,\,H\,\mid\,\langle\,F\,\mid\,\mathit{this}\,.\,f_i\;=\;x\,;\,s\,\rangle\,S\,]\!]$, it holds that $F(\mathit{this})\;=\;p$, $F(x)\;=\;p'$, $H(p)\;=\;C(f1\;=\;p1;..;fn = pn)$ where $sv_i = w_i\,p_i$ for some w_i. The compiled configuration then reduces to the compilation of $H\,[p\mapsto(H\,(p)\,.\,f_i\mapsto sv)]\,\mid\,\langle\,F\,\mid\,s\,\rangle\,S$.

  Conversely, if the compiled configuration reduces, since compiled reductions are deterministic, the source configuration can reduce via RED_ASSIGN, and the simulation diagram commutes.

- Rules RED_NEW, RED_COPY, RED_RETURN, RED_CAST_CLASS_OK, RED_CAST_OTHER, and RED_CALL follow using the same argument.
- Rule

$$\frac{\begin{array}{l} F(y) = p \\ \mathbf{ptype}\,(H,\,p) = C \\ \mathbf{mbody}\,(m,\,C) = x_1\,..\,x_n\,.\,s_0\,;\,\mathbf{return}\,x_0 \\ \mathbf{mtype}\,(m,\,C) = t_1\,..\,t_n\;\rightarrow\;t \\ F(y_1) = w_1\,p_1\quad..\quad F(y_n) = w_n\,p_n \\ sv_1 = [\![\,t_1\,]\!]\,p_1\quad..\quad sv_n = [\![\,t_n\,]\!]\,p_n \\ F(x) = w'\,p' \\ \mathit{cast} = \mathbf{w2c}\,(w') \end{array}}{H\,\mid\,\langle\,F\,\mid\,x = y\,.\,m\,(y_1\,..\,y_n)\,;\,s\,\rangle\,S\;\longrightarrow\;H\,\mid\,\langle\,[]\,[x_1\mapsto sv_1\,..\,x_n\mapsto sv_n]\,[\mathit{this}\mapsto p]\,\mid\,s_0\,;\,\mathbf{return}\,x_0\,\rangle\,\langle\,F\,\mid\,x = \mathit{cast}\,\mathit{ret}\,;\,s\,\rangle\,S}$$

  In the compiled configuration, it holds that $F(y)\;=\;p$ and $F(y_i)\;=\;p_i$. The statement $x\;=\;y\,.\,m\,(y_1\,..\,y_n)$ has been compiled to $x\;=\;y\,@\,m\,(y_1\,..\,y_n)$, because since the configuration was well-typed and $F(y)\;=\;p$, it must hold that $\Gamma\,(y)\;=\;C$. The compiled configuration can reduce via REC_CALL_TARGET into the compiled outcome.

  Conversely, if the target configuration reduces, since the source configuration was well-typed and compiled reductions are deterministic, the source configuration can reduce via RED_CALL, and the simulation diagram commutes.

- Rule

$$\frac{\begin{array}{l} F(y) = (\textbf{like}\;C)\,p \\ \mathbf{ptype}\,(H,\,p) = D \\ \mathbf{mbody}\,(m,\,D) = x_1\,..\,x_n\,.\,s_0\,;\,\mathbf{return}\,x_0 \\ \mathbf{mtype}\,(m,\,C) = t_1\,..\,t_n\;\rightarrow\;t \\ \mathbf{mtype}\,(m,\,D) = t'_1\,..\,t'_n\;\rightarrow\;t' \\ \forall i\,.\,t_i <:\,t'_i\;\vee\;t'_i = \mathbf{dyn} \\ (\mathbf{concr}\,(t)\;\wedge\;\mathbf{concr}\,(t'))\;\Rightarrow\;t' <:\,t \\ F(y_1) = w_1\,p_1\quad..\quad F(y_n) = w_n\,p_n \\ sv_1 = [\![\,t'_1\,]\!]\,p_1\quad..\quad sv_n = [\![\,t'_n\,]\!]\,p_n \\ F(x) = w'\,p' \\ \mathit{cast} = \mathbf{w2c}\,(w') \end{array}}{H\,\mid\,\langle\,F\,\mid\,x = y\,.\,m\,(y_1\,..\,y_n)\,;\,s\,\rangle\,S\;\longrightarrow\;H\,\mid\,\langle\,[]\,[x_1\mapsto sv_1\,..\,x_n\mapsto sv_n]\,[\mathit{this}\mapsto p]\,\mid\,s_0\,;\,\mathbf{return}\,x_0\,\rangle\,\langle\,F\,\mid\,x = \mathit{cast}\,(t)\,\mathit{ret}\,;\,s\,\rangle\,S}$$

  In the compiled configuration, it holds that $F(y)\;=\;p$ and $F(y_i)\;=\;p_i$. The statement $x\;=\;y\,.\,m\,(y_1\,..\,y_n)$ has been compiled to $x = y\,@\,(\textbf{like}\,C)\;m\,(y_1\,..\,y_n)$, because since the configuration was well-typed and $F(y) = (\textbf{like}\;C)\,p$, it must hold that $\Gamma\,(y) = \textbf{like}\;C$. All the type verifications required by the rule CALL_LIKE_TARGET are satisfied since they were satisfied for the rule CALL_LIKE. The rule CALL_LIKE_TARGET can then reduce into the compiled outcome.

  Conversely, if the target configuration reduces via RED_CALL_LIKE_TARGET, since the source configuration was well-typed and compiled reductions are deterministic, the source configuration can reduce via RED_CALL_LIKE because the type verifications were already satisfied by RED_CAL_LIKE_TARGET, and the simulation diagram commutes.

- The case for the rule CALL_DYN is analogous to that of CALL_LIKE. Again, the key point is that the type verifications performed by CALL_DYN_TARGET have already been performed by CALL_DYN, and vice-versa.