Robbins, Ed and King, Andy and Schrijvers, Tom  (2015) Proof appendix to accompany the paper, "From MinX to MinC: Semantics-Driven Decompilation of Recursive Datatypes".  University of Kent

DOI

Link to record in KAR

http://kar.kent.ac.uk/51459/

Document Version

Author's Accepted Manuscript

Copyright & reuse
Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (e.g Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

Versions of research
The version in the Kent Academic Repository may differ from the final published version. Users are advised to check http://kar.kent.ac.uk for the status of the paper. Users should always cite the published version of record.

Enquiries
For any further enquiries regarding the licence status of this document, please contact: researchsupport@kent.ac.uk

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at http://kar.kent.ac.uk/contact.html
A. Proof Appendix

A.1 Type Safety

We write $\lambda \cdot \beta; \rho \vdash \langle \sigma, \pi, s \rangle \xrightarrow{\ast} \langle \sigma', \pi', s' \rangle$.

• $\Sigma; \lambda; \beta; \rho \vdash \langle \sigma, \pi, s \rangle \xrightarrow{\ast} \langle \sigma', \pi', s' \rangle$

• $\Gamma; \Sigma; \Psi \vdash \rho$

• $\Sigma; \Psi \vdash \sigma; \pi$

then for some $\Psi' \supseteq \Psi$

(a) $\Gamma; \Sigma; \Psi' \vdash \rho$

(b) $\Sigma; \Psi' \vdash \sigma'; \pi'$

(c) $\Gamma; \Sigma; \vdash s'$

Proposition 11 (safety for function definitions).

1. Progress: if

- $\Sigma \vdash f(x : \theta) (y : \theta', l, \lambda_c, j)$

- $\Sigma; \lambda_c; \beta; \rho \vdash \langle \sigma, \pi, \lambda_c(l) \rangle \xrightarrow{\ast} \langle \sigma', \pi', \text{return} \rangle$

- $\Gamma_c = \{ x : \theta, y : \theta' \}$

- $\Gamma_c; \Sigma; \Psi \vdash \rho$

- $\Sigma; \Psi \vdash \sigma; \pi$

then

(a) $\Sigma; \lambda_c; \beta; \rho \vdash \langle \sigma, \pi, \lambda_c(l) \rangle \xrightarrow{\ast} \langle \sigma', \pi', \text{return} \rangle$

(b) $\Sigma; \Psi' \vdash \sigma'; \pi'$

Proposition 11 (safety for function definitions).

1. Progress: if

- $\Sigma \vdash f(x : \theta) (y : \theta', l, \lambda_c, j)$

- $\Sigma; \lambda_c; \beta; \rho \vdash \langle \sigma, \pi, \lambda_c(l) \rangle \xrightarrow{\ast} \langle \sigma', \pi', \text{return} \rangle$

- $\Gamma_c = \{ x : \theta, y : \theta' \}$

- $\Gamma_c; \Sigma; \Psi \vdash \rho$

- $\Sigma; \Psi \vdash \sigma; \pi$

then

(a) $\Sigma; \lambda_c; \beta; \rho \vdash \langle \sigma, \pi, \lambda_c(l) \rangle \xrightarrow{\ast} \langle \sigma', \pi', \text{return} \rangle$

(b) $\Sigma; \Psi' \vdash \sigma'; \pi'$

Proof 1. Propositions 8, 9, 10 and 11 proved together by mutual structural induction on the typing judgements for $\ell, e, s$ and $c_d$.

- By case analysis on $\Gamma_c; \Sigma; \vdash \ell : \theta$ in Fig. 4. To show 1b or conversely 1a, 2a, 2b and 2c hold for proposition 8. Observe that 2a holds if $\Psi' \supseteq \Psi$.

1. Let $\ell = x$. By rule t-var $\lambda_c; \beta; \rho \vdash \langle \sigma, \pi, x \rangle \xrightarrow{\ast} \langle \sigma, \pi, a \rangle$

where $a = \rho(x)$ hence 1a holds. Put $\Psi' = \Psi$. Since $\Gamma_c; \Sigma; \Psi \vdash \rho$ it follows $\Sigma; \Psi' \vdash \rho(x) : \theta* \ast 2c$ holds. Moreover $\Sigma; \Psi' \vdash \sigma; \pi$ and 2b holds.

2. Let $\ell : \theta = s : \tau$. Since $\Gamma_c; \Sigma; \Psi \vdash \rho$ it follows $a = \rho(x) \neq 0$. By rule l-attr $\lambda_c; \beta; \rho \vdash \langle \sigma, \pi, x \rangle \xrightarrow{\ast} \langle \sigma, \pi, a \rangle$

thus 1a holds. Put $\Psi' = \Psi$. By rule t-attr $\Gamma_c; \Sigma; \Psi \vdash \sigma; \pi$ it follows $\Sigma; \Psi \vdash \sigma(a) : \tau*$ and by rule st-addr $\Sigma; \Psi \vdash \sigma(a) : \tau*$ thus $\Sigma; \Psi \vdash \sigma(a) : \tau*$ and 2c holds. Moreover $\Sigma; \Psi' \vdash \sigma; \pi$ and 2b holds.

3. Let $\ell : \theta = x : c : \theta_c$. Since $\Gamma_c; \Sigma; \Psi \vdash \rho$ let $a = \rho(x) \neq 0$ and let $v = \sigma(a) + \varepsilon$. If $\rho(x) = 0$ or $v \notin \cup \pi$ then 1b holds. Otherwise $\lambda_c; \beta; \rho \vdash \langle \sigma, \pi, x \rangle \xrightarrow{\ast} \langle \sigma, \pi, a \rangle$ and 1a holds. Put $\Psi' = \Psi$. By rule t-fld $\Gamma_c; \Sigma; \Psi \vdash \sigma; \pi$ it follows $\Sigma; \Psi \vdash \sigma \vdash \pi \vdash \sigma \vdash \pi : \ast$ and by rule st-addr $\rho(x) : \ast$ in $\Sigma$ and $\Sigma; \Psi \vdash \sigma; \pi$ it follows $\Sigma; \Psi \vdash \sigma \vdash \pi \vdash \sigma \vdash \pi : \ast$ and by rule t-var $\lambda_c; \beta; \rho \vdash \langle \sigma, \pi, x \rangle \xrightarrow{\ast} (\sigma, \pi, a)$. By rule st-addr $\sigma \vdash \pi \vdash \sigma \vdash \pi : \ast$ and 2c holds. Moreover $\Sigma; \Psi' \vdash \sigma; \pi$ and 2b holds.
Let $\Sigma; \Psi \vdash (x' \in e' \mid \psi)$. By rule t-addr $\Sigma; \Psi \vdash e' : t$ hence 2c. Also by rule vt-bot $\Sigma; \Psi \vdash \bot : \tau$ by and rule st-comp $\Sigma; \Psi; \sigma', \pi + a : \tau$ hence $\Sigma; \Psi; \sigma', \pi$ and 2b holds.

6. Let $e : \theta$ be new struct $N : N^* \text{ and } n = |\Sigma(N)|$. By rule e-str $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, \psi \mid e \rangle) \to (\langle \sigma', \pi' \mid v \rangle)$, where $\sigma' = \sigma \circ (a \mapsto \bot \cup \{a, a + n - 1\})$ and $\pi' = \pi \cup \{a, a + n - 1\}$. Put $\Sigma' = \Sigma \cup \{a : N, a + 1 : \theta_1, \ldots, a + n - 1 : \theta_n, \}$. By rule t-addr $\Sigma; \Psi \vdash e' : t$ hence 2c holds.

7. Let $e : \theta$ be new decl $e \mid \psi$. By rule t-new-addr $\Gamma; \Sigma \vdash e : t$ hence by induction:
   - Either $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e \rangle) \to \psi$. By rule e-str $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e \rangle) \to \psi$. Hence 1b.
   - Or $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e \rangle) \to (\langle \sigma', \pi' \mid v \rangle)$. By rule e-str $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e \rangle) \to (\langle \sigma', \pi' \mid v \rangle)$ and by rule st-comp $\Sigma; \Psi \vdash (\langle \sigma', \pi' \rangle) : \theta[\psi]$. By rule t-addr $\Sigma; \Psi \vdash (\langle \sigma', \pi' \rangle) : \theta[\psi]$ and by rule t-addr $\Sigma; \Psi \vdash (\langle \sigma', \pi' \rangle) : \theta[\psi]$ and by rule st-comp $\Sigma; \Psi \vdash (\sigma', \pi' \mid v) : \theta[\psi]$. Hence 1a.
   - By induction there exists $\psi' \geq \Sigma$ such that $\Sigma; \Psi \vdash (\sigma', \pi')$. By rule e-str $\Sigma; \bar{\rho} \vdash (\langle \sigma', \pi', e \rangle) \to (\langle \sigma'', \pi'' \mid v \rangle)$, where $\sigma'' = \sigma \circ (a \mapsto \bot \cup \{a, a + n - 1\})$ and $\pi'' = \pi \cup \{a, a + n - 1\}$. Put $\Sigma'' = \Sigma \cup \{a : N \cup \{a + 1 : \theta_1, \ldots, a + n - 1 : \theta_n, \}$. By rule t-addr $\Sigma; \Psi \vdash e' \mid \psi$. Hence 2c holds.

8. Let $e : \theta = (e_1 \oplus e_2) : t$. By rule t-addr $\Gamma; \Sigma \vdash e_1 : t$ and $\Gamma; \Sigma \vdash e_2 : t$. Hence by induction:
   - Either $\Sigma; \bar{\rho} \vdash (\langle \sigma, e_1 \rangle) \to \psi$. By rule e-op-err $\Sigma; \bar{\rho} \vdash (\langle \sigma, e_1 \rangle) \to \psi$. Hence 1b.
   - Or $\Sigma; \bar{\rho} \vdash (\langle \sigma', \pi' \mid e_2 \rangle) \to \psi$. By rule e-str $\Sigma; \bar{\rho} \vdash (\langle \sigma', \pi' \mid e_2 \rangle) \to \psi$. Hence 1b.
   - Or $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e_1 \rangle \to (\langle \sigma', \pi' \mid v_1 \rangle)$ and $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e_1 \rangle \to (\langle \sigma', \pi' \mid v_2 \rangle)$. By rule e-op-err $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e_1 \rangle \to (\langle \sigma', \pi' \mid v_1 \rangle)$. Hence 1a.
   - By induction $\Sigma; \Psi \vdash (\langle \sigma, \pi, e_1 \rangle \to (\langle \sigma', \pi' \mid v_1 \rangle)$. By rule t-addr $\Sigma; \Psi \vdash e \mid \psi$. By rule e-op-err $\Sigma; \bar{\rho} \vdash (\langle \sigma, \pi, e_1 \rangle \to (\langle \sigma', \pi' \mid v_2 \rangle)$. By rule t-addr $\Sigma; \Psi \vdash e \mid \psi$. Hence 2c holds.

9. Let $e : \theta = (e_1 \oplus e_2) : \tau[\psi]$. Similar to previous case.
10. Let \( e : \theta = f(\bar{c}) \); \( \theta ; e \). By rule t-call \( \Gamma ; ; \Sigma \vdash e : \theta \) where \( \phi_i (f) = f(x : \theta ; y : \theta ; i, \lambda, e) \) and \( \Sigma \vdash \theta ; e \). With respect to \( e \), there are two possibilities:

- Either for some \( c : \Sigma ; \bar{c} ; \rho \vdash (\sigma_1, \pi_1, \sigma_i, \pi_i) \) err. Then by rule e-call-err it follows that 1b holds.

- Or for all \( c : \Sigma ; \bar{c} ; \rho \vdash (\sigma_1, \pi_1, \sigma_i, \pi_i) \) and by the inductive hypothesis \( \Sigma ; \Psi, \theta ; i \) and \( \Sigma ; \Psi \vdash \sigma_1, \pi_1 \). Let \( \Psi' = \Psi \cup \{ (e : \theta, \pi') \} \). Then it is easy to verify \( \Sigma ; \Psi' ; \sigma_1, \pi_1, \sigma_i, \pi_i \) and \( \Gamma ; ; \Sigma \vdash \rho' \). By the progress induction hypothesis we then have for \( s \):

- Either \( \Sigma ; \bar{c} ; \rho, \pi, \rho' \vdash (\sigma', \pi_n, \lambda_0 (l)) \) return. Hence 1a.

- Otherwise 1b.

Preservation follows from the induction hypotheses for all \( e, \pi \), and \( s \).

* By case analysis on \( \Gamma ; ; \Sigma \vdash s \) in Fig. 4. To show that either 1b or conversely 1a, 2a, 2b, and 2c of Proposition 10 hold. Observe that 2a holds if \( \Psi \not\subset \Sigma \).

1. Let \( \Gamma ; \Sigma \vdash s \). From the induction hypothesis for \( \ell \), either \( \Sigma ; \bar{c} ; \rho \vdash (\sigma', \pi', a) \) err, and hence 1b, or \( \Sigma ; \bar{c} ; \rho \vdash (\sigma, \pi, \ell) \rightarrow (\sigma', \pi', a) \). In the latter case, we have either \( \Sigma ; \bar{c} ; \rho \vdash (\sigma', \pi', a) \rightarrow (\sigma', \pi', a) \) err, and hence 1b, or \( \Sigma ; \bar{c} ; \rho \vdash (\sigma', \pi', a) \rightarrow (\sigma', \pi', a) \). By s-ass we then have \( \Sigma ; \bar{c} ; \rho, \pi, \ell \vdash (s', \pi', a) \rightarrow (s', \pi', a) \) where \( s'' = s \) and hence 1a.

We get \( \Gamma ; \Sigma \vdash s \) from t-assign. Hence 2c. From the induction hypotheses for \( \ell \) we get type preservation \( \Sigma ; \Psi'' \vdash a : \theta \star s \) and \( \Psi ; \varpi \vdash v : \theta \) and type consistency \( \Pi ; \psi ; \sigma'' : \varpi' \). Hence, through rule vt-addr we know that \( (a : \theta ; e) \in \Psi'' \). From rule t-ass we know \( \Sigma \vdash \theta < \theta \). Hence, through rule vt-sub we have \( \Sigma ; \Psi'' \vdash \varpi' : \varpi' \). Since \( \sigma''(a) = v \) we have hence by rule st-comp \( \Sigma ; \Psi''; \sigma'' ; \rho' ; \sigma_i, \pi_i \) a \( \theta : \theta \). Hence \( \Sigma ; \Psi'' \vdash \sigma'' ; \sigma' \). Thus 2b.

2. Let \( \Gamma ; \Sigma \vdash (s) \). Then:

- Either \( \Sigma ; \bar{c} ; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', e) \) err. Hence 1b.

- Or \( \Sigma ; \bar{c} ; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', e) \). Then:

  - Either \( v = \bot \). Hence 1b.

  - Or \( v = 0 \). Then by rule s-if-false \( \Sigma ; \lambda_0 (l) ; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', s') \). Hence 1a. We call this scenario 1.

  - Or \( v = 0 \). Then by rule s-if-true \( \Sigma ; \lambda_0 (l) ; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', s) \). Hence 1a. We call this scenario 2.

In scenario 1 we have from t-if \( \Gamma ; \Sigma \vdash s \). Hence 2c. In scenario 2 we have that \( s' \in range(\lambda_0) \). Hence \( \Gamma ; \Sigma \vdash s' \). Hence 2c. In both scenarios we have from the induction hypothesis for \( e \) that \( \Sigma ; \Psi' \vdash \sigma' ; \pi' \). Hence 2b.

3. Let \( \Gamma ; \Sigma \vdash \rho \). Then either \( l \notin dom(\lambda_0) \) and thus \( \Sigma ; \lambda_0 (l) \vdash (\sigma, \pi, goto) \rightarrow (\sigma', \pi', \rho) \). Hence 1b. Alternatively \( \lambda_0 (l) = \emptyset \). Then by rule s-goto \( \Sigma ; \lambda_0 (l) \vdash (\sigma, \pi, goto) \rightarrow (\sigma, \pi, \lambda_0 (l)) \). Hence 1a.

From \( \Gamma ; \Sigma \vdash s \) it follows that \( \Gamma ; \Sigma \vdash s \). Hence 2c. Let \( \Psi \) err. Then 2b.

4. Let \( \Gamma ; \Sigma \vdash \rho \). Then 1c. Also vacuously 2c and 2b.

* Proposition 11 follows by the repeated application of Proposition 10 combining progress and preservation at every step. Besides the givens of Proposition 2, Proposition 10 also requires \( \Gamma ; \Sigma \vdash \lambda_0 \). This is given by rule t-def which is the only possible way that the well-typing of the function definition could have been constructed.

A.2 Well-Typed Decomposition

Proposition 12 (well-typed instruction decomposition). If \( \mu c ; \Gamma ; \Sigma \vdash i : e \) then for some \( \theta_1 \) and \( \theta_2 \)

1. \( \Gamma ; \Sigma \vdash \theta_1 : \theta_2 \).

2. \( \Gamma ; \Sigma \vdash e : e_1 \).

3. \( \Sigma \vdash \theta_2 < \theta_1 \).

Proof 2. The proof proceeds by case analysis on the inference rules of the instruction translation relation.

1. Case tr-\( \tau^* \). Let \( \theta_1 = \theta_2 = \theta \). From tr-\( \tau^* \) we have \( (x : \theta) \in \Sigma \). Then by rule t-var \( \Gamma ; \Sigma \vdash x : \theta \). Hence 1.

2. Case tr-var. Let \( \theta_1 = \theta_2 = \theta \). From tr-var we have \( (x : \theta) \in \Sigma \). Then by rule t-var \( \Gamma ; \Sigma \vdash y : \theta \). Hence 2.

3. Case tr-comp. Let \( \theta_1 = \theta_2 = \theta \). From tr-comp we have \( (x : \theta) \in \Sigma \). Then by rule t-comp \( \Gamma ; \Sigma \vdash x + y : \theta \). Hence 3.

4. Case tr-if. Let \( \theta_1 = \theta_2 = \theta \). From tr-if we have \( (x : \theta) \in \Sigma \). Then by rule t-if \( \Gamma ; \Sigma \vdash x : \theta \). Hence 4.

5. Case tr-ass. Let \( \theta_1 = \theta_2 = \theta \). From tr-ass we have \( (x : \theta) \in \Sigma \). Then by rule t-ass \( \Gamma ; \Sigma \vdash x : \theta \). Hence 5.

6. Case tr-call. Let \( \theta_1 = \theta_2 = \theta \). From tr-call we have \( (x : \theta) \in \Sigma \). Then by rule t-call \( \Gamma ; \Sigma \vdash x : \theta \). Hence 6.

7. Case tr-mov. Let \( \theta_1 = \theta_2 = \theta \). From tr-mov we have \( (x : \theta) \in \Sigma \). Then by rule t-mov \( \Gamma ; \Sigma \vdash x : \theta \). Hence 7.

8. Case tr-mov-n. Let \( \theta_1 = \theta_2 = \theta \). From tr-mov-n we have \( (x : \theta) \in \Sigma \). Then by rule t-mov-n \( \Gamma ; \Sigma \vdash x : \theta \). Hence 8.
11. Case tr-mov-r1. From tr-mov-r1 we have \((x : \theta_1) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta_1\). Hence 1. From tr-mov-r1 we have \((y : \theta_2)[*] \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \theta_2[*]\). Also by rule t-\(\Sigma\); \(\Sigma \vdash 0 : \emptyset\). Then by rule t-arf \(\Gamma_c; \Sigma \vdash \emptyset[y], \Sigma \vdash 0 : \theta_2\). Hence 2. From tr-mov-r1 we have \(\Sigma \vdash \theta_2 : \emptyset\). Hence 3. 

12. Case tr-mov-r2. From tr-mov-r2 we have \((x : \theta_1) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta_1\). Also by rule t-\(\Sigma\); \(\Sigma \vdash 0 : \emptyset\). Hence 1. From tr-mov-r2 we have \((y : \theta_2) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \theta_2\). Hence 2. From tr-mov-r2 we have \(\Sigma \vdash \theta_2 : \emptyset\). Hence 3. 

13. Case tr-mov-r3. From tr-mov-r3 we have \((x : \theta) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta\). Hence 1. From tr-mov-r3 we have \((y : N[*]) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : N[*]\). Then by rule t-fldf \(\Gamma_c; \Sigma \vdash y \rightarrow : \emptyset_0\). Hence 1. From tr-mov-r3 we have \((y : \theta) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \theta\). Hence 2. From tr-mov-r3 we have \(\Sigma \vdash \theta : \emptyset\). Hence 3. 

14. Case tr-mov-r4. From tr-mov-r4 we have \((x : N) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : N\). Then by rule t-fldf \(\Gamma_c; \Sigma \vdash x \rightarrow : \theta_0\). Hence 1. From tr-mov-r4 we have \((y : \theta_0) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \theta_0\). Hence 2. From tr-mov-r4 we have \(\Sigma \vdash \theta_0 : \emptyset\). Hence 3. 

15. Case tr-mov-r5. From tr-mov-r5 we have \((x : \theta_1) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta_1\). Hence 1. From tr-mov-r5 we have \((y : \emptyset)[*] \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \emptyset[*]\). Also from tr-mov-r5 we have \(\Gamma_c; \Sigma \vdash m : t\). Then by rule t-arf \(\Gamma_c; \Sigma \vdash y[m] : \theta_0\). Hence 2. From tr-mov-r5 we have \(\Sigma \vdash \theta_0 : \emptyset\). Hence 3. 

16. Case tr-mov-i4. From tr-mov-i4, we have \((x : \theta)[*] \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta\). Hence 1. From tr-mov-i4 we have \((y : \emptyset)[*] \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \emptyset[*]\). Hence 2. From tr-mov-i4 we have \(\Sigma \vdash \emptyset : \emptyset\). Hence 3. 

17. Case tr-call. From tr-call we have \((x : \theta_1) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta_1\). Hence 1. From tr-call we have \((y : \emptyset)[*] \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash y : \emptyset[*]\). Also by rule t-\(\Sigma\); \(\Sigma \vdash 0 : \emptyset\). Then by rule t-arf \(\Gamma_c; \Sigma \vdash \emptyset[y], \Sigma \vdash 0 : \theta_1\). Hence 2. From tr-call we have \(\Sigma \vdash \theta_1 : \emptyset\). Hence 3. 

Proposition 13 (well-typed block declaration). If \(\mu; \nu; \Gamma_c; \Sigma \vdash b \Rightarrow s\) then \(\Gamma_c; \Sigma \vdash s\). 

Proof 3. This proof proceeds by structural induction on the block translation relation.

1. Case tr-instr. From tr-instr we have \(\mu; \nu; \Gamma_c; \Sigma \vdash t \Rightarrow \ell : e\). Hence, by Proposition 12 we have \(\Gamma_c; \Sigma \vdash \ell : \theta_1\), \(\Gamma_c; \Sigma \vdash e : \theta_0\) and \(\Sigma \vdash \theta_0 \Rightarrow \theta_1\). Also by rule tr-instr we have \(\mu; \nu; \Gamma_c; \Sigma \vdash b \Rightarrow s\). Hence by the induction hypothesis we have \(\Gamma_c; \Sigma \vdash s\). Then by rule t-ass we have \(\Gamma_c; \Sigma \vdash \ell : e; s\).

2. Case tr-if. If tr-if we have \((x : \theta) \in \Gamma_c\). Then by rule t-var \(\Gamma_c; \Sigma \vdash x : \theta_0\). Also from tr-if we have \(\mu; \nu; \Gamma_c; \Sigma \vdash b \Rightarrow s\). Hence, from the induction hypothesis we have \(\Gamma_c; \Sigma \vdash s\). Then the proposition follows from rule t-if.

3. Case tr-goto. This follows from rule t-goto.

4. Case tr-ret. This follows from rule t-ret.

Proposition 14 (well-typed definition declaration). If \(\Sigma \vdash d_2 \Rightarrow d_1\) then \(\Sigma \vdash d_1\).

Proof 4. We show that the four preconditions to rule t-def are satisfied:

1. From rule t-def we know that \(\Gamma_c = (x : \emptyset_0, y : \emptyset[\ell])\).

2. From rule t-def we know that \(a \in \text{dom}(\lambda_0)\) and \(\ell = \mu_3(a)\). Hence \(\ell \in \text{range}(\mu_3)\).

3. From rule t-def we know that \(r_{\text{if}} \in \mathcal{F}_0\). We also know that

4. From rule t-def we know that \(\forall \ell \rightarrow \ell \in \mu_3 : \mu_3; \nu_3; \Gamma_c \vdash \lambda_0 : \ell_0\). From Proposition 13 we then know that \(\forall \ell \in \text{range}(\mu_3) : \Gamma_c \vdash \lambda_0 : \ell_0\). From rule t-def we know that \(\text{range}(\mu_3) = \text{dom}(\lambda_3)\).

5. Hence \(\ell \in \text{dom}(\lambda_3)\).

A.3 Semantics Preservation

Instructions We prove Propositions 7 and 6 together.

Proof 5. The proof proceeds by case analysis on the derivation of the judgement \(\mu_3; \nu; \Gamma_c; \Sigma \vdash t \Rightarrow \ell : e\).

(a) This case is not possible. Rule ex-\(\Sigma\)-r always applies.

(b) In this case rules ex-\(\Sigma\)-r is used for progress on \(v : R \Rightarrow (H, R, \text{pa}_v, r_v, r_v \times \ell) \Rightarrow (R', H')\). Here \(R' = R \cup \text{non}(r_v \mapsto b_v \# (b_v \times \ell))\) where \(b_v = R_0(b_v)\) and \(b_v = R_0(b_v)\). Similarly, through rule l-var \(\Sigma; \bar{b}_v \# \rho \Rightarrow (\sigma, \pi, x) \Rightarrow (\sigma, \pi, a)\) with \(a = \rho(x)\). Also through rules e-op, c-val, l-var and e-const we obtain \(\Sigma; \bar{b}_v \# \rho \Rightarrow (\sigma, \pi, x \times \ell(y \times m)) \Rightarrow (\sigma, \pi, v)\) where \(v = v_0 \times \mu(v_0 \times m), v_0 = \sigma(a), a = \rho(y)\) and \(v_0 = \sigma(a)\).

From rule tr-\(\Sigma\)-r we know \((r_v : x) \in \mu_3\). Hence from the related registers we know \(\mu_3 \vdash b_0 \Rightarrow v_0\). Similarly, we know \(v_0 \Rightarrow b_0 \Rightarrow v_0\). Then from \((x : \emptyset)[*] \in \Gamma_c\).
Γ, and the store typing of σ it follows that \( v_s = n_s \) and from the success of the addition, it also follows that \([n_s, n_s \oplus (v_y \cdot m)] \subseteq r \). Hence, also from the store typing all \( m \) values at the addresses in this range have type \( \theta \). From the related heaps it then follows that \( c/m = sizeof(\theta) \) that \( \mu_s \vdash (b_i \oplus c \ (b_j \cdot e \ c)) \leftrightarrow (v \oplus (v_y \cdot m)) \). Hence, the update registers are still related.

2. Case tr-\( \oplus \)-r². Then \( \ell = (op_m r_i, r_j, e) \), \( \ell = e = x \oplus (y \cdot e) \).

(a) This case is not possible. Rule ex-\( \oplus \)-r² always applies.

(b) In this case rules ex-\( \oplus \)-r² is used for progress on \( \ell : R \vdash \langle H, R, op_m r_i, r_j, e \rangle \rightarrow (H, R') \). Here \( R' = R \circ w \ {r_i \mapsto b_i \oplus (b_j \cdot e \ c)} \) where \( b_i = R_{0,w}(r_i) \) and \( b_j = R_{0,w}(r_j) \).

Similarly, through rule l-var \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, x) \rightarrow (\sigma, \pi, a) \) with \( a = \rho(x) \). Also through rules e-op, e-ival, l-var and e-const we obtain \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, (x \oplus (y \cdot m))) \rightarrow (\sigma, \pi, v) \) where \( v = v_0 \oplus (v_y \cdot m) \) and \( v_0 = \sigma(a), \). Also through rule l-var we obtain \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma, \pi, a) \).

From rule tr-\( \oplus \)-r² we know \((r_i, x)_w \in \mu_r\). Hence from the related registers we know \( \mu_s \vdash (b_i \oplus (b_j \cdot e \ c)) \leftrightarrow (v \oplus (v_y \cdot m)) \). Hence, the update registers are still related.

3. Case tr-\( \ominus \)-rc. Then \( \ell = (op_m r_i, c), \ell = e = x \ominus c \).

(a) This case is not possible. Rule ex-\( \ominus \)-rc always applies.

(b) In this case rules ex-\( \ominus \)-rc is used for progress on \( \ell : R \vdash \langle H, R, op_m r_i, c \rangle \rightarrow (H, R') \). Here \( R' = R \circ w \ {r_i \mapsto b_i \ominus c} \) where \( b_i = R_{0,w}(r_i) \).

Similarly, through rule l-var \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, (x \ominus (y \cdot m))) \rightarrow (\sigma, \pi, v) \) where \( v = (v_0 \ominus (v_y \cdot m)) \) and \( v_0 = \sigma(a) \). Also through rules e-op, e-ival, l-var and e-const we obtain \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, c) \rightarrow (\sigma, \pi, a) \).

From rule tr-\( \ominus \)-rc we know \((r_i, x)_w \in \mu_r\). Hence from the related registers we know \( \mu_s \vdash (b_i \ominus c) \leftrightarrow (v \ominus (v_y \cdot m)) \). Hence, the update registers are still related.

4. Case tr-\( \ominus \)-rc. Then \( \ell = (op_m r_i, c), \ell = e = x \ominus m \).

(a) This case is not possible. Rule ex-\( \ominus \)-rc always applies.

(b) In this case rules ex-\( \ominus \)-rc is used for progress on \( \ell : R \vdash \langle H, R, op_m r_i, c \rangle \rightarrow (H, R') \). Here \( R' = R \circ w \ {r_i \mapsto b_i \ominus (b_j \cdot e \ c)} \) where \( b_i = R_{0,w}(r_i) \).

Similarly, through rule l-var \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, x) \rightarrow (\sigma, \pi, a) \) with \( a = \rho(x) \). Also through rules e-op, e-ival, l-var and e-const we obtain \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, (x \ominus (y \cdot m))) \rightarrow (\sigma, \pi, v) \) where \( v = (v_0 \ominus (v_y \cdot m)) \) and \( v_0 = \sigma(a) \).

From rule tr-\( \ominus \)-rc we know \((r_i, x)_w \in \mu_r\). Hence from the related registers we know \( \mu_s \vdash (b_i \ominus (b_j \cdot e \ c)) \leftrightarrow (v \ominus (v_y \cdot m)) \). Hence, the update registers are still related.

5. Case tr-\( \ominus \)-rc. Then \( \ell = (op_m^2 r_i, r_j), \ell = e = x \ominus y \).

(a) This case is not possible. Rule ex-\( \ominus \)-rc always applies.

(b) In this case rules ex-\( \ominus \)-rc is used for progress on \( \ell : R \vdash \langle H, R, op_m^2 r_i, r_j \rangle \rightarrow (H, R') \). Here \( R' = R \circ w \ {r_i \mapsto b_i \ominus (b_j \cdot e \ c)} \) where \( b_i = R_{0,w}(r_i) \) and \( b_j = R_{0,w}(r_j) \).

Similarly, through rule l-var \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, x) \rightarrow (\sigma, \pi, a) \) with \( a = \rho(x) \). Also through rules e-op, e-ival, l-var and e-const we obtain \( \Sigma; \bar{\rho} \cdot \rho \vdash (\sigma, \pi, (x \ominus (y \cdot m))) \rightarrow (\sigma, \pi, v) \) where \( v = (v_0 \ominus (v_y \cdot m)) \) and \( v_0 = \sigma(a) \).

From rule tr-\( \ominus \)-rc we know \((r_i, x)_w \in \mu_r\). Hence from the related registers we know \( \mu_s \vdash (b_i \ominus (b_j \cdot e \ c)) \leftrightarrow (v \ominus (v_y \cdot m)) \). Hence, the update registers are still related.
10. Case tr-mov-ir. Then $\iota = (\text{mov}_w, r_i, [r_j])$, $\ell = x$ and $e = y[0]$.
   (a) This case is possible iff $R(r_i) = 0$ or $R(r_j) = 1$. Because of the related registers and, from rule tr-mov-ir, $(r_i : y) \in \mu_r$, we have $\mu_\rho \vdash R(r_j) \leftrightarrow (\rho(x))$. In either of the cases for $R(r_j)$ we also have $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi) \Rightarrow \sigma y[0]$. 
   (b) In this case rules ex-mov-ri is used for progress on $\mu$: $\overline{R} \vdash \langle H, \overline{R}, \text{mov}_w, r_i, r_j \rangle \Rightarrow \langle H', R' \rangle$. Here $R' = R_0 \cup \{r_i \mapsto b_2\}$ where $b_2 = H^w(b_1)$ and $b_1 = R(r_j)$.
   Similarly, through rule l-var we obtain $(r_j : y) \in \mu_r$. Hence from the related registers we know $\mu_\rho \vdash b_2 \sim w_1$. From related stores, we also know $\mu_\rho \vdash b_2 \sim v_1$. Also from rule tr-mov-ir we know $(r_i : x) \in \mu_r$. Hence, the registers are related. After the update we can see that they are still related.

11. Case tr-mov-ri+. Then $\iota = (\text{mov}_w, r_i, [r_j])$, $\ell = x$ and $e = y \rightarrow 0$.
   (a) This case is possible iff $R(r_i) = 0$ or $R(r_j) = 1$. Because of the related registers and, from rule tr-mov-ri+, $(r_i : y) \in \mu_r$, we have $\mu_\rho \vdash R(r_j) \leftrightarrow (\rho(x))$. In either of the cases for $R(r_j)$ we also have $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi) \Rightarrow \sigma y$. 
   (b) In this case rules ex-mov-ri is used for progress on $\mu$: $\overline{R} \vdash \langle H, \overline{R}, \text{mov}_w, r_i, r_j \rangle \Rightarrow \langle H', R' \rangle$. Here $R' = R_0 \cup \{r_i \mapsto b_2\}$ where $b_2 = H^w(b_1)$ and $b_1 = R(r_j)$.
   Similarly, through rule l-var we obtain $(r_j : y) \in \mu_r$. Hence from the related registers we know $\mu_\rho \vdash b_2 \sim v_1$. From related stores, we also know $\mu_\rho \vdash b_2 \sim v_1$. Also from rule tr-mov-ir we know $(r_i : x) \in \mu_r$. Hence, the registers are related. After the update we can see that they are still related.

12. Case tr-mov-ir2. Then $\iota = (\text{mov}_w, [r_i], \ell) = x[0]$ and $e = y[0]$.
   (a) This case is possible iff $R(r_i) = 0$ or $R(r_j) = 1$. Because of the related registers and, from rule tr-mov-ir2, $(r_i : x) \in \mu_r$, we have $\mu_\rho \vdash R(r_i) \leftrightarrow (\rho(x))$. In either of the cases for $R(r_j)$ we also have $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, x) \Rightarrow \sigma x[0]$.
   (b) In this case rules ex-mov-ir is used for progress on $\mu$: $\overline{R} \vdash \langle H, \overline{R}, \text{mov}_w, [r_i] \rangle \Rightarrow \langle H', R' \rangle$. Here $R' = H \cup \{r_i \mapsto b_2\}$ where $b_2 = H^w(b_1)$ and $b_1 = R(r_i)$.
   Similarly, through rule l-var we obtain $(r_i : x) \in \mu_r$. Hence from the related registers we know $\mu_\rho \vdash b_2 \sim w_1$. From related stores, we also know $\mu_\rho \vdash b_2 \sim v_1$. Since $(x : \theta_1) \in \Gamma$, we know that $v_1$ is an address. Because of related heaps, we then know that $(b_1, v_1) w_{\theta_1}$.
   After the update we can see that they are still related.
From rule tr-mov-ri$^+$ we know $(r_i : y)_x \in \mu_T$. Hence from the related registers we know $\mu_T \vdash \Delta R \omega \sigma(y)$. From the translation rule we also have $(y : \theta)[s] \in \Gamma_c$. Because of the progress, it means that $[a', a'' + m] \subseteq \pi$. Because of the related heaps and well-typed store it follows that $\mu_T \vdash \beta R \omega \Delta$. Also from rule tr-mov-ri$^+$, we know $(r_i : x)_w \in \mu_T$. After the update we can see that they are still related.

16. Case tr-mov-ri$^+$. Then $e = (\text{mov}_w r_i[r_j + c], \ell = x)$ and $e = y \rightarrow m$.

(a) This case is possible iff $R(r_i) = 0, R(r_j) = \bot$ or $(R(r_j) + c) \notin \text{dom}(H)$. Because of the related registers and heaps, and from rule tr-mov-ri$^+$, we have $\mu_T \vdash R(r_i) \rightarrow \sigma(y)$. In either of the first two cases for $R(r_i)$ we also have $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x \rightarrow m) \Rightarrow \bot$. In the last case, because of related heaps, it also has to be that $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x \rightarrow m) \Rightarrow \bot$.

(b) In this case rules ex-mov-$\forall$ is used for progress on $e: \Delta R \vdash \langle H, R, \text{mov}_w r_i[r_j + c] \rangle \Rightarrow \langle H', R' \rangle$. Here $H' = H \circ \{ R(r_i) \rightarrow c + n \rightarrow R_{\text{mov}(r_i)} \}$.

Similarly, through rule l-var $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x \rightarrow m) \Rightarrow \bot$.

From the translation rule we also have $(x : N^x) \in \Gamma_c$. Because of the progress, it means that $[a', a'' + m] \subseteq \pi$. Because of the related heaps and well-typed store it follows that $\mu_T \vdash \beta R \omega \sigma(y)$. Also from rule tr-mov-ri$^+$, we know $(r_i : x)_w \in \mu_T$. After the update we can see that they are still related.

17. Case tr-mov-ri$^+$ r. Then $e = (\text{mov}_w [r_i + c], r_j, \ell = x[m]$ and $e = y$.

(a) This case is possible iff $R(r_i) = 0, R(r_j) = \bot$ or $(R(r_j) + c) \notin \text{dom}(H)$. Because of the related registers and heaps, and from rule tr-mov-ri$^+$, we have $\mu_T \vdash R(r_j) \rightarrow \sigma(y)$. In either of the first two cases for $R(r_j)$ we also have $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x[m]) \Rightarrow \bot$. In the last case, because of related heaps, it also has to be that $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x[m]) \Rightarrow \bot$.

(b) In this case rules ex-mov-$\forall$ is used for progress on $e: \Delta R \vdash \langle H, R, \text{mov}_w [r_i + c] \rangle \Rightarrow \langle H', R' \rangle$. Here $H' = H \circ \{ R(r_i) \rightarrow \sigma(y) \}$.

Similarly, through rule l-var $\Sigma; \mu_T \rho \vdash (\sigma, \pi, x[m]) \Rightarrow \bot$.

From the translation rule we also have $(x : N^x) \in \Gamma_c$. Because of the progress, it means that $[a', a'' + m] \subseteq \pi$. Because of the related heaps and well-typed store it follows that $\mu_T \vdash \beta R \omega \sigma(y)$. Also from rule tr-mov-ri$^+$, we know $(r_i : x)_w \in \mu_T$. Hence from the related registers we know $\mu_T \vdash R(r_i) \rightarrow \sigma(y)$. From the translation rule we also have $(x : \theta)[s] \in \Gamma_c$. Because of the progress, it means that $[a', a'' + m] \subseteq \pi$. Because of the related heaps and well-typed store it follows that $\mu_T \vdash \beta R \omega \Delta$. Also from rule tr-mov-ri$^+$, we know $(r_i : x)_w \in \mu_T$. Hence $\mu_T \vdash R_{\text{mov}(r_i)} \omega \Delta$. After the update we can see that $(R(r_i) + c)$ and $a' + m$ are still related.

18. Case tr-mov-ri$^+$. Then $e = (\text{mov}_w [r_i + c], r_j, \ell = x \rightarrow m$ and $e = y$.

(a) This case is possible iff $R(r_i) = 0, R(r_j) = \bot$ or $(R(r_j) + c) \notin \text{dom}(H)$. Because of the related registers and heaps,
Similarly, through rule l-var $\Sigma; \bar{\rho}, \rho \vdash \langle \sigma, \pi, x \rangle \xrightarrow{\ell} \langle \sigma, \pi, d' \rangle$ where $d' = \rho(x)$. Also through rule c-ar we obtain $\Sigma; \bar{\rho}, \rho \vdash \langle \sigma, \pi, \text{new } \theta(m) \rangle \xrightarrow{\ell} \langle \sigma', \pi, d'' \rangle$ where $d'' = \rho(x) + \frac{1}{\bar{\sigma}}^{m-1}$. The new memory relations are straightforward.

23. Case tr-call. This case follows coinductively.

**Basic Blocks** The two propositions for basic blocks are the following.

**Proposition 15** (Preservation of Progress for Basic Blocks). If
- $\mu_\lambda; \mu_\tau; \Gamma_c; \Sigma \vdash b \xrightarrow{\ell} s$
- $\forall (a : l) \in \mu_\lambda : \mu_\lambda; \mu_\tau; \Gamma_c; \Sigma \vdash \lambda_a(l) \xrightarrow{\ell} \lambda_a(l)$
- $\Gamma_c; \Sigma; \Psi \vdash \rho$
- $\Sigma; \Psi \vdash \sigma; \pi$
- $\mu_a; \nu_a; \pi; \bar{\rho}, \rho \vdash H \xrightarrow{\ell} \sigma$
- $\mu_a; \bar{\mu}_a; \mu_\tau; \sigma \vdash \bar{R}, R \xrightarrow{\ell} \bar{\rho}, \rho$
- $\lambda_a; \bar{R} \vdash \langle H, R, b \rangle \xrightarrow{\ell} \langle H', R', b' \rangle$

then
- $\Sigma; \lambda_a; \bar{\rho}, \rho \vdash \langle \sigma, \pi, s \rangle \xrightarrow{\ell} \text{err or}$
- $\Sigma; \lambda_a; \bar{\rho}, \rho \vdash \langle \sigma', \pi', s' \rangle$.

**Proposition 16** (Preservation of Related Memory for Basic Blocks). If
- $\mu_\lambda; \mu_\tau; \Gamma_c; \Sigma \vdash b \xrightarrow{\ell} s$
- $\forall (a : l) \in \mu_\lambda : \mu_\lambda; \mu_\tau; \Gamma_c; \Sigma \vdash \lambda_a(l) \xrightarrow{\ell} \lambda_a(l)$
- $\Gamma_c; \Sigma; \Psi \vdash \rho$
- $\Sigma; \Psi \vdash \sigma; \pi$
- $\mu_a; \nu_a; \pi; \bar{\rho}, \rho \vdash H \xrightarrow{\ell} \sigma$
- $\mu_a; \mu_\tau; \mu_\tau; \sigma \vdash \bar{R}, R \xrightarrow{\ell} \bar{\rho}, \rho$
- $\lambda_a; \bar{R} \vdash \langle H, R, b \rangle \xrightarrow{\ell} \langle H', R', b' \rangle$
- $\Sigma; \lambda_a; \bar{\rho}, \rho \vdash \langle \sigma, \pi, s \rangle \xrightarrow{\ell} \langle \sigma', \pi', s' \rangle$

then for some $\mu'_a \geq \mu_a$ and $\nu'_a \geq \nu_a$:
- $\mu'_a; \bar{\mu}'_a; \mu_\tau; \sigma' \vdash \bar{R}, R \xrightarrow{\ell} \bar{\rho}, \rho$
- $\mu'_a; \nu'_a; \pi; \bar{\rho}, \rho \vdash H' \xrightarrow{\ell} \sigma'$

**Proof 6.** The proof is straightforward.

**Function Definitions** The two propositions for function definitions are the following.

**Proposition 17** (Preservation of Progress for Function Definitions). If
- $\Sigma \vdash \langle f, \bar{r}_x, \bar{r}_x, a, \lambda_c, j \rangle \xrightarrow{\ell} f(\bar{r}_x \xrightarrow{\ell} \bar{y}, \bar{y}, l, \lambda_c, j)$
- $\mu_v = \{ \bar{r}_x \xrightarrow{\ell} \bar{y}, \bar{y} \xrightarrow{\ell} \bar{y} \}$
- $\Gamma_v = \{ \bar{x} : \bar{\theta}, \bar{y} : \bar{\theta} \}$
- $\Gamma_c; \Sigma; \Psi \vdash \rho$
- $\Sigma; \Psi \vdash \sigma; \pi$
- $\mu_a; \nu_a; \pi; \bar{\rho}, \rho \vdash H \xrightarrow{\ell} \sigma$
- $\mu_a; \mu_\tau; \mu_\tau; \sigma \vdash \bar{R}, R \xrightarrow{\ell} \bar{\rho}, \rho$
- $\lambda_a; \bar{R} \vdash \langle H, R, \lambda_v(a) \rangle \xrightarrow{\ell} \langle H', R', b' \rangle$

then
- $\Sigma; \lambda_a; \bar{\rho}, \rho \vdash \langle \sigma, \pi, \lambda_v(l) \rangle \xrightarrow{\ell} \text{err or}$
- $\Sigma; \lambda_a; \bar{\rho}, \rho \vdash \langle \sigma', \pi', s' \rangle$.

**Proposition 18** (Preservation of Related Memory for Function Definitions). If
- $\mu_\lambda; \mu_\tau; \Gamma_c; \Sigma \vdash b \xrightarrow{\ell} s$