Citation for published version

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
A. Proof Appendix

A.1 Type Safety

We write $\Sigma; \Psi \vdash \sigma; \pi$ to signify that
$$\forall (a : \theta) \in \Psi \cdot \Sigma; \Psi; \sigma; \pi \vdash a : \theta$$

We also write $\Gamma; \Sigma; \Psi \vdash \rho$ to signify that
$$\forall (x : \theta) \in \Gamma \cdot \Sigma; \Psi; \rho(x) : \theta \land \rho(x) \neq 0$$

Moreover, we write $\Gamma; \Sigma \vdash \lambda_c$ to signify that
$$\forall s \in \text{range}(\lambda_c) \cdot \Gamma; \Sigma \vdash s$$

Proposition 8 (safety for lvalue evaluation).

1. Progress: if
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   • $\Gamma; \Sigma \vdash \ell : \theta$
   then
     (a) $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, \ell) \rightsquigarrow (\sigma', \pi', a)$ or
     (b) $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, \ell) \rightarrow \text{err}$.

2. Preservation: if
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   • $\Gamma; \Sigma \vdash \ell : \theta$
   • $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, \ell) \rightarrow (\sigma', \pi', \alpha)$
   then for some $\Psi' \supseteq \Psi$
     (a) $\Gamma; \Sigma; \Psi' \vdash \rho$
     (b) $\Sigma; \Psi' \vdash \sigma'; \pi'$
     (c) $\Sigma; \Psi' \vdash a : \theta^*$

Proposition 9 (safety for expression evaluation).

1. Progress: if
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   • $\Gamma; \Sigma \vdash e : \theta$
   then
     (a) $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', v)$ or
     (b) $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, e) \rightarrow \text{err}$.

2. Preservation: if
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   • $\Gamma; \Sigma \vdash e : \theta$
   • $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, e) \rightarrow (\sigma', \pi', v)$
   then for some $\Psi' \supseteq \Psi$
     (a) $\Gamma; \Sigma; \Psi' \vdash \rho$
     (b) $\Sigma; \Psi' \vdash \sigma'; \pi'$
     (c) $\Sigma; \Psi' \vdash v : \theta$

Proposition 10 (safety for statement evaluation).

1. Progress: if
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   • $\Gamma; \Sigma \vdash s$
   • $\Gamma; \Sigma \vdash \lambda_c$
   then
     (a) $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, s) \rightarrow (\sigma', \pi', s')$ or
     (b) $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, s) \rightarrow \text{err}$ or
     (c) $s = \text{return}$.

2. Preservation: if
   • $\Gamma; \Sigma \vdash s$
   • $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, s) \rightarrow (\sigma', \pi', s')$
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   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
   • $\Sigma \vdash f(\overline{x}) : (\theta, \ldots, \lambda_c, j)$
   • $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, \lambda_c(l)) \rightarrow^* (\sigma', \pi', \text{return})$
   • $\Gamma = \{ x : \theta, y : \theta \}$
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   then
     (a) $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, \lambda_c(l)) \rightarrow^* (\sigma', \pi', \text{return})$
     (b) $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, \lambda_c(l)) \rightarrow^* \text{err}$ (we assume this sub-sums divergence).

2. Preservation: if
   • $\Sigma \vdash f(\overline{x}) : (\theta, \ldots, \lambda_c, j)$
   • $\Sigma; \lambda_c; \bar{\rho}; \rho \vdash (\sigma, \pi, \lambda_c(l)) \rightarrow^* (\sigma', \pi', \text{return})$
   • $\Gamma = \{ x : \theta, y : \theta \}$
   • $\Gamma; \Sigma; \Psi \vdash \rho$
   • $\Sigma; \Psi \vdash \sigma; \pi$
   then for some $\Psi' \supseteq \Psi$
     (a) $\Gamma; \Sigma; \Psi' \vdash \rho$
     (b) $\Sigma; \Psi' \vdash \sigma'; \pi'$

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

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

1. Let $\ell = x$. By rule l-var $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, x) \rightarrow (\sigma, \pi, o)$ where $o = \rho(x)$ hence 1a holds. Put $\Psi' = \Psi$. Since $\Sigma; \Psi \vdash \rho$ it follows $\Sigma; \Psi' \vdash \rho(x) : \theta^* \land 2c$ holds. Moreover $\Sigma; \Psi' \vdash \sigma; \pi \land 2b$ holds.

2. Let $\ell = \theta = \text{ex} : \tau$. Since $\Gamma; \Sigma; \Psi \vdash \rho$ it follows $a = \rho(x) \neq 0$. By rule l-ptr $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, \text{ex}) \rightarrow (\sigma, \pi, \sigma(a))$ thus 1a holds. Put $\Psi' = \Psi$. By rule t-var $\Gamma; \Sigma \vdash \text{ex} : \tau^*$ and by $\Gamma; \Sigma; \Psi \vdash \rho$ it follows $\Sigma; \Psi \vdash \sigma : \tau *$. By rule st-comp $\Sigma; \Psi \vdash \sigma(a) : \tau$ thus $\Sigma; \Psi \vdash \sigma(a) : \tau^* \land 2c$ holds. Moreover $\Sigma; \Psi' \vdash \sigma; \pi \land 2b$ holds.

3. Let $\ell : \theta = \text{ex} : e : \theta_c$. Since $\Gamma; \Sigma; \Psi \vdash \rho$ let $a = \rho(x) \neq 0$ and let $e = \sigma(a) \lor c$. If $\rho(x) = 0$ or $e \notin \cup \pi$ then 1b holds. Otherwise $\Sigma; \bar{\rho}; \rho \vdash (\sigma, \pi, x \rightarrow c) \rightarrow (\sigma, \pi, v)$ and 1a holds. Put $\Psi' = \Psi$. By rule t-fld $\Gamma; \Sigma \vdash \text{ex} : \tau^*$ and by rule r-var $\Sigma \vdash (x : \tau^*) \rightarrow (\sigma, \pi, \sigma(a))$ thus 1a holds. Put $\Psi' = \Psi$. By rule r-addr $\rho(x) : \text{ex}^* \in \Psi$ and by $\Sigma; \Psi \vdash \sigma; \pi$ it follows $\Sigma; \Psi \vdash \sigma(\rho(x)) : \text{ex}^*$. By rule st-comp $\Sigma; \Psi \vdash \sigma(a) : \tau^*$ thus $\Sigma; \Psi \vdash \sigma(a) : \tau^* \land 2b$ holds.
Let $e : \theta \vdash \sigma \vdash r(x)$, by rule e-addr. Hence 1b.

6. Let $\epsilon = \theta$ new structure $N : N$ and $n = |\Sigma(N)|$. By rule e-str $\Sigma^* \vdash \sigma, \pi, new\ structure\ N \vdash \sigma', \pi'$ where $\sigma' = \sigma \circ (a \mapsto \|N\| + 1)$ and $\pi' = \pi \cup \{a, a + n - 1\}$. Put $\Psi' = \Psi \cup \{a : N, a + 1 : \theta_1, \ldots, a + n - 1 : \theta_n, \ldots\}$. By rule vt-add $\Sigma^* : \psi : a : N$ hence 2c holds.

7. Let $e : \theta \vdash \psi(e) : \theta[e]$. By rule t-new $\Gamma_e : \psi : a : \theta[e]$ hence 2c.

8. Let $e : \theta \vdash \psi(e) : \theta[e]$ by rule t-addr $\Sigma^* : \psi : a : \theta[e]$ hence 2c.

9. Let $e : \theta \vdash \psi(e) : \theta[e]$ similar to previous case.

---

**Figure 13:** Well-formed type declarations of MNC programs
Let $e : \theta = f(\vec{e}) : \theta_0$. By rule t-call $\Gamma ; \Sigma \vdash e_1 : \theta_1'$ where $\phi_i(f) = f(x, \theta) : \theta_i'$. Let $\Sigma \vdash \theta_0 < \theta_1$. With respect to $e_1$, there are two possibilities:

- Either for some $i : \Sigma, \bar{\rho}, \rho \vdash (\sigma_i, \pi_i, e_i) \to \varepsilon$, then by rule e-call-e it follows that 1b holds.
- Or for all $i : \Sigma, \bar{\rho}, \rho \vdash (\sigma_i, \pi_i, e_i) \to (\sigma_i, \pi_i, e_i)$ and by the inductive hypothesis $\Sigma, \Psi \vdash \theta_i ; v_i$ and $\Sigma, \Psi, \pi_i \vdash \sigma_i = \varepsilon$. We get $\Sigma, \Psi, \rho \vdash \sigma_i'. \pi_i'. \varepsilon$. Then it is easy to verify $\Sigma, \Psi' \vdash \sigma_i', \pi_i', \rho_i'$. By the progress induction hypothesis we then have for $s$:
  - Either $\Sigma, \bar{\rho}, \rho, \rho_i' \vdash (\sigma_i', \pi_i, \lambda_i(l)) \to \varepsilon^*(\sigma', \pi', \lambda_i(l), \text{return})$. Hence 1a.
  - Otherwise 1b.

Preservation follows from the induction hypotheses for all $e_i$ and $s$.

* By case analysis on $\Gamma ; \Sigma \vdash e$ in Fig. 4. To show that either 1b or conversely 1a, 2a, 2b and 2c of Proposition 10 hold. Observe that 2a holds if $\Psi \supseteq \Psi$.

1. Let $\Gamma ; \Sigma \vdash \ell : (e) ; s$. From the induction hypothesis for $\ell$, either $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, \ell) \to \varepsilon$ and hence 1b, or $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, \ell) \to (\sigma', \pi', \alpha)$. In the latter case, we have either $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, \ell) \to (\sigma', \pi', \alpha)$, or $\Sigma, \bar{\rho}, \rho \vdash (\sigma', \pi', \alpha) \to (\sigma', \pi', \alpha')$. By s-ass we then have $\Sigma, \bar{\rho}, \rho, \rho_i' \vdash (\sigma_i', \pi_i', \lambda_i, l) \to (\sigma_i', \pi_i', \lambda_i, l)$. Then it is easy to verify $\Sigma, \Psi' \vdash (\sigma_i', \pi_i', \lambda_i, l) \to (\sigma', \pi', \lambda_i(l), \text{return})$. Hence 1a.

2. Let $\Gamma ; \Sigma \vdash (e) ; s$. Then
  - Either $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, e) \to \varepsilon$. Hence 1b.
  - Or $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, e) \to (\sigma', \pi', \varepsilon)$. Then
    - Either $s = \perp$. Hence 1b.
    - Or $s = \emptyset$. Then by rule s-if-false $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, (e \text{ goto } l)) \to (\sigma', \pi', s')$. Hence 1a. We call this scenario 1.
    - Or $s \neq \emptyset$. Then by rule s-if-true $\Sigma, \bar{\rho}, \rho \vdash (\sigma, \pi, (e \text{ goto } l)) \to (\sigma', \pi', s')$. 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 $s'$ range $(\lambda_i, \lambda_i)$. Hence $\Sigma \vdash \sigma'$. Hence 2c. In both scenarios we have from the induction hypothesis for $e$ that $\Sigma, \Psi' \vdash \sigma' ; \pi'$. Hence 2b.

3. Let $\Gamma ; \Sigma \vdash \ell \text{ goto } l$. Then either $\ell \in \text{dom}(\lambda_i)$ and thus $\Sigma, \lambda_i ; \bar{\rho}, \rho \vdash (\sigma, \pi, \text{ goto } l) \to \varepsilon$. Hence 1b. Alternatively $\lambda_i(l) = \perp$. Then by rule s-goto $\Sigma, \lambda_i ; \bar{\rho}, \rho \vdash (\sigma, \pi, \text{ goto } l) \to (\sigma, \pi, \sigma_i ; \lambda_i)$. Hence 1a.

4. Let $\Gamma ; \Sigma \vdash \ell \text{ return}$. Hence 1c. Also vacuously 2c and 2b.

* Proposition 11 follows by the repeated application of Proposition 10 combining progress and preservation at every step.
11. Case tr-mov-ri₂. From tr-mov-ri₂ we have \( (x : \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y : \theta_0 \), hence 1. From tr-mov-ri₂ we have \( y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ri₂ we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

12. Case tr-mov-ir₂. From tr-mov-ir₂ we have \( (x : \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ir₂ we have \( y : \theta_0 \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y : \theta_0 \). Hence 2. From tr-mov-ir₂ we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

13. Case tr-mov-ris. From tr-mov-ris we have \( (x : \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y : \theta_0 \). Hence 1. From tr-mov-ris we have \( y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ris we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

14. Case tr-mov-ir. From tr-mov-ir we have \( (x : \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y : \theta_0 \). Hence 1. From tr-mov-ir we have \( y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ir we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

15. Case tr-mov-ri₁. From tr-mov-ri₁ we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-mov-ri₁ we have \( y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ri₁ we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

16. Case tr-mov-ir₁. From tr-mov-ir₁ we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-mov-ir₁ we have \( y : \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y : [0] \). Hence 2. From tr-mov-ir₁ we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

17. Case tr-mov-r. From tr-mov-r we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-mov-r we have \( y ; \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y ; [0] \). Hence 2. From tr-mov-r we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

18. Case tr-mov-r. From tr-mov-r we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-mov-r we have \( y ; \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y ; [0] \). Hence 2. From tr-mov-r we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

19. Case tr-alloc. From tr-alloc we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-alloc we have \( y ; \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y ; [0] \). Hence 2. From tr-alloc we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

20. Case tr-alloc. From tr-alloc we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-alloc we have \( y ; \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y ; [0] \). Hence 2. From tr-alloc we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

21. Case tr-alloc. From tr-alloc we have \( (x ; \theta_0) \in \Gamma_c \). Then by rule t-var \( \Gamma_c ; \Sigma \vdash y ; \theta_0 \). Hence 1. From tr-alloc we have \( y ; \theta_0 \). Also by rule t-ar \( \Gamma_c ; \Sigma \vdash y ; [0] \). Hence 2. From tr-alloc we have \( \Sigma ; \theta_0 \in \theta_1 \), hence 3.

22. Case tr-call. From tr-call we have \( (u ; \theta_

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 \( \Gamma_c ; \Sigma \vdash \theta \vdash \phi \). (a) This case is not possible. Rule ex-\( \theta \)-r almost always applies.

(b) In this case rules ex-\( \theta \)-r is used for progress on \( v \).
Γ, and the store typing of σ it follows that vs = ns, and from the success of the addition, it also follows that [n., n.s ⊕ (v·s * m)] ⊆ τ. Hence, also from the store typing all m values at the addresses in this range have type θ. From the related heaps it then follows with c/m = sizeof(θ) that μo ⊢ (b⊥ +4 (b⊥ + ε c)) ⇐ (v ⊕ ε (v·s * m)). Hence, the update registers are still related.

2. Case tr-⊕*2. Then η = (opw r1, r1, ε), ℓ = x and ε = x ⊕ (y * ε). (a) This case is not possible. Rule ex-⊕*4 always applies. (b) In this case rules ex-⊕*4 is used for progress on: ⊢ (H, R, opw r1, r1, ε) → (H, R'). Here R' = R o w {r1 → b⊥ + (b⊥ + ε c)} where b⊥ = R0.w(r1) and b⊥ = R0.w(r1).

Similarly, through rule l-var Σ; µo ⊢ (σ, π, x) → (σ, π, a) with a = ρ(x). Also through rules e-op, e-ival, l-var and e-const we obtain Σ; µo ⊢ (σ, π, (x ⊕ (y * m))) → (σ, π, v) where v = b⊥ + (v·s * m), vs = σ(a), a = ρ(y) and v = σ(a'). From rule tr-⊕*2 we know (r1: x)w ∈ µr. Hence from the related registers we know μo ⊢ b⊥ + ε v. It then follows that μo ⊢ (b⊥ +4 (b⊥ + ε c)) ⇐ (v ⊕ ε (v·s * m)). Hence, the update registers are still related.

3. Case tr-⊕. Then η = (opw r1, r1), ℓ = x and x = x ⊕ ε. (a) This case is not possible. Rule ex-⊕.4 always applies. (b) In this case rules ex-⊕.4 is used for progress on: ⊢ (H, R, opw r1, r1) → (H, R'). Here R' = R o w {r1 → b⊥ + (b⊥ + ε c)} where b⊥ = R0.w(r1).

Similarly, through rule l-var Σ; µo ⊢ (σ, π, x) → (σ, π, a) with a = ρ(x). Also through rules e-op, e-ival, l-var and e-const we obtain Σ; µo ⊢ (σ, π, (x ⊕ (x orb))) → (σ, π, v) where v = b⊥ + (v·s * m) and v = ρ(a). From rule tr-⊕ we know (r1: x)w ∈ µr. Hence from the related registers we know μo ⊢ b⊥ +4 ε v. Then from (x : t) ∈ Γc, (y : t) ∈ Γc and w = sizeof(t) it follows that μo ⊢ (b⊥ +4 (b⊥ + ε c)) ⇐ (v ⊕ ε (v·s * m)). Hence, the update registers are still related.

4. Case tr-◆. Then η = (opw r1, r1), ℓ = x and x = x ⊕ m. (a) This case is not possible. Rule ex-◆.4 always applies. (b) In this case rules ex-◆.4 is used for progress on: ⊢ (H, R, opw r1, r1) → (H, R'). Here R' = R o w {r1 → b⊥ +4 c} where b⊥ = R0.w(r1).

Similarly, through rule l-var Σ; µo ⊢ (σ, π, x) → (σ, π, a) with a = ρ(x). Also through rules e-op, e-ival, l-var and e-const we obtain Σ; µo ⊢ (σ, π, (x ⊕ m)) → (σ, π, v) where v = b⊥ + (v·s * m) and v = σ(a). From rule tr-◆ we know (r1: x)w ∈ µr. Hence from the related registers we know μo ⊢ b⊥ +4 ε v. Then from (x : t) ∈ Γc and the store typing of σ it follows that v = vs and from the success of the addition, it also follows that [n., n.s ⊕ m] ⊆ τ. Hence, also from the store typing all m values at the addresses in this range have type θ. From the related heaps it then follows with c/m = sizeof(θ) that μo ⊢ (b⊥ +4 c) ⇐ (v ⊕ ε (v·s * m)). Hence, the update registers are still related.

5. Case tr-◆. Then η = (opw r1, r1), ℓ = x and x = x ⊕ y. (a) This case is not possible. Rule ex-◆.4 always applies. (b) In this case rules ex-◆.4 is used for progress on: ⊢ (H, R, opw r1, r1) → (H, R'). Here R' = R o w {r1 → b⊥ +4 c} where b⊥ = R0.w(r1) and b⊥ = R0.w(r1).

Similarly, through rule l-var Σ; µo ⊢ (σ, π, x) → (σ, π, a) with a = ρ(x). Also through rules e-op, e-ival, l-var and e-constant we obtain Σ; µo ⊢ (σ, π, (x ⊕ y)) → (σ, π, v) where v = b⊥ + (v·s * m), vs = σ(a), a = ρ(y) and v = σ(a'). From rule tr-◆ we know (r1: x)w ∈ µr. Hence from the related registers we know μo ⊢ b⊥ +4 v. By similar reasoning we know μo ⊢ b⊥ +4 v. Then from (x : t) ∈ Γc, (y : t) ∈ Γc and w = sizeof(t) it follows that μo ⊢ (b⊥ +4 (b⊥ + ε c)) ⇐ (v ⊕ ε (v·s * m)). Hence, the update registers are still related.
13. Case tr-mov-ir2. Then $\ell = (\text{mov}_w \ [r_i], r_j)$, $\ell = x[0]$ and $e = y[0]$.

(a) This case is possible iff $R(r_i) = 0$ or $R(r_j) = \bot$. Because of the related registers and, from rule tr-mov-ir2, $(r_i, x) \in \mu_r$, we have $\mu_r \vdash R(r_i) \iff \sigma(p(x))$. In either of the cases for $R(r_j)$ we also have $\Sigma; \rho \vdash (\sigma, x) \to \varepsilon$.

(b) In this case rules ex-mov-ir is used for progress on $\ell$: $R \vdash \{H, R, \text{mov}_w \ [r_i], r_j \} \to \{H', R'\}$. Here $H' = H \circ \{b_1, \ldots, b_i + (w - 1) \to b_2\}$ where $b_1 = R(r_i)$ and $b = R_0w(r_j)$.

Similarly, through rule l-var we know $(r_j : y) \in \mu_r \mu$. Hence from the related registers we know $\mu_r \vdash b_2 \equiv v_2$. From related stores, we also know $\mu_r \vdash b_2 \equiv v_2$. Also from rule tr-mov-ir2 we know $(r_i : x) \in \mu_r \mu$. Hence, the registers are related. After the update we can see that they are still related.

14. Case tr-mov-ir3. Then $\ell = (\text{mov}_w \ [r_i], r_j)$, $\ell = x[0]$ and $e = y$.

(a) This case is possible iff $R(r_i) = 0$ or $R(r_j) = \bot$. Because of the related registers and, from rule tr-mov-ir3, $(r_i, x) \in \mu_r$, we have $\mu_r \vdash R(r_i) \iff \sigma(p(x))$. In either of the cases for $R(r_j)$ we also have $\Sigma; \rho \vdash (\sigma, x) \to \varepsilon$.

(b) In this case rules ex-mov-ir is used for progress on $\ell$: $R \vdash \{H, R, \text{mov}_w \ [r_i], r_j \} \to \{H', R'\}$. Here $H' = H \circ \{b_1, \ldots, b_i + (w - 1) \to b_2\}$ where $b_1 = R(r_i)$ and $b = R_0w(r_j)$.

Similarly, through rule l-var we know $(r_j : y) \in \mu_r \mu$. Hence from the related registers we know $\mu_r \vdash b_2 \equiv v_2$. From related stores, we also know $\mu_r \vdash b_2 \equiv v_2$. Also from rule tr-mov-ir3 we know $(r_i : x) \in \mu_r \mu$. Hence, the registers are related. After the update we can see that they are still related.

15. Case tr-mov-ir4. Then $\ell = (\text{mov}_w \ r_i, [r_i + c])$, $\ell = x[0]$ and $e = y[m]$.

(a) This case is possible iff $R(r_i) = 0$, $R(r_j) = \bot$ or $(R(r_j) + c) \not\in \text{dom}(H)$. Of the related registers and heaps, and from rule tr-mov-ir4 $(r_i, y) \in \mu_r$, we have $\mu_r \vdash R(r_i) \iff \sigma(p(y))$. In either of the first two cases for $R(r_j)$ we also have $\Sigma; \rho \vdash (\sigma, y[m]) \to \varepsilon$. In the last case, because of related heaps, it also has to be that $\Sigma; \rho \vdash (\sigma, y[m]) \to \varepsilon$.

(b) In this case rules ex-mov-ir is used for progress on $\ell$: $R \vdash \{H, R, \text{mov}_w \ r_i, [r_i + c] \} \to \{H', R'\}$. Here $R' = R_0w \vdash \{r_i \to b\}$ where $b = H' \circ (b')$ and $b = R(r_j) + c$.

Similarly, through rule l-var we know $(\sigma, y[m]) \in \mu_r \mu$. Hence, $\mu_r \vdash b \equiv v_2$. Also from rule tr-mov-ir3 we know $(r_i : x) \in \mu_r \mu$. Because of related heaps, we then know that $(b_1, v_1) \in \mu_\mu$. After the update we can see that they are still related.
From rule tr-mov-ri+ we know \( (r_i : y)_w \in \mu_R \). Hence from the related registers we know \( \mu_w \vdash \tilde{b} \sim \sigma y \). From the translation rule we also have \( (y : \theta)[y] \in \Gamma_c \). Because of the progress, it means that \([\sigma y, \sigma a' + m] \subseteq \pi \). Because of the related heaps and well-typed store it follows that \( \mu_w \vdash \tilde{b} \sim \sigma y \). Also from rule tr-mov-ri+ we know \( (r_i : x)_w \in \mu_R \). After the update we can see that they are still related.

16. Case tr-mov-ri+2. \( \ell = (\text{move}_{\text{w}} \ r_i, [r_j + c], \ell = x \rightarrow m \) and \( e = y \rightarrow m \).

(a) This case is possible iff \( R(r_i) = 0, R(r_i) = \perp \) or \( (R(r_i)) + c \) \( \notin \text{dom}(H) \). Because of the related registers and heaps, and from rule tr-mov-ri+2 \( (r_i : y)_4 \in \mu_R \), we have \( \mu_w \vdash R(r_i) \rightsquigarrow \sigma y \). In either of the first two cases for \( R(r_i) \) we also have \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, y \mid e \rangle \rightsquigarrow \text{err} \). In the last case, because of related heaps, it also has to be that \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, x \rightarrow m \rangle \rightsquigarrow \text{err} \).

(b) In this case rules ex-mov-ri+ is used for progress on \( e \): \( \tilde{R} \vdash \langle H, R, \text{mov}_w, r_i, [r_j + c] \rangle \rightarrow \langle H', R' \rangle \). Here \( H' = H \circ \langle H(R(r_i)) + c \rangle + n \rightarrow R_{\text{mov}+1}(r_i) \rangle \). Similarly, through rule l-var \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, x \rightarrow m \rangle \rightsquigarrow \langle \sigma, \pi, a \rangle \). Also through rules e-lval and l-var we obtain \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, y \rangle \rightarrow \langle \sigma, \pi, v \rangle \) where \( v = \sigma(a') \) and \( a' = \rho(y) \).

17. Case tr-mov-ri+. \( \ell = (\text{move}_{\text{w}} \ r_i + c, r_j, \ell = x \mid m \) and \( e = y \rightarrow m \).

(a) This case is possible iff \( R(r_i) = 0, R(r_i) = \perp \) or \( (R(r_i)) + c \) \( \notin \text{dom}(H) \). Because of the related registers and heaps, and from rule tr-mov-ri+ \( (r_i : x)_4 \in \mu_R \), we have \( \mu_w \vdash R(r_i) \rightsquigarrow \sigma y \). In either of the first two cases for \( R(r_i) \) we also have \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, x \mid m \rangle \rightsquigarrow \text{err} \). In the last case, because of related heaps, it also has to be that \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, y \rightarrow m \rangle \rightsquigarrow \text{err} \). From rule tr-mov-ri+2 we know \( (r_j : y)_4 \in \mu_R \). Hence from the related registers we know \( \mu_w \vdash \tilde{b} \sim \sigma a' \). From the translation rule we also have \( \langle y : \theta \rangle \in \Gamma_c \) and \( \Sigma(N) = \langle \theta_0, \ldots, \theta_n \rangle \). Because of the progress, it means that \([\sigma a', \sigma a' + m] \subseteq \pi \). Because of the related heaps and well-typed store it follows that \( \mu_w \vdash \tilde{b} \sim \sigma a' \). Also from rule tr-mov-ri+ we know \( (r_j : x)_w \in \mu_R \). After the update we can see that they are still related.

18. Case tr-mov-ri+2. \( \ell = (\text{move}_{\text{w}} \ r_i + c, r_j, \ell = x \rightarrow m \) and \( e = y \rightarrow m \).

(a) This case is possible iff \( R(r_i) = 0, R(r_i) = \perp \) or \( (R(r_i)) + c \) \( \notin \text{dom}(H) \). Because of the related registers and heaps, and from rule tr-mov-ri+2 \( (r_i : x)_4 \in \mu_R \), we have \( \mu_w \vdash R(r_i) \rightsquigarrow \sigma x \). In either of the first two cases for \( R(r_i) \) we also have \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, x \rightarrow m \rangle \rightsquigarrow \text{err} \). In the last case, because of related heaps, it also has to be that \( \Sigma; \tilde{\rho} \vdash \langle \sigma, \pi, x \rightarrow m \rangle \rightsquigarrow \text{err} \).
Similarly, through rule l-var \( \Sigma, \rho \vdash (\sigma, \pi, x) \xrightarrow{\delta} (\sigma, \pi, a') \) where \( a' = \rho(x) \). Also through rule e-ar we obtain
\( \Sigma, \rho \vdash (\sigma, \pi, \text{new } \theta(m)) \xrightarrow{\delta} (\sigma', \pi, a') \) where \( \sigma' = \sigma \circ \lceil a'' + 1 \rceil_{m-1} \).

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
- \( \mu_{\lambda}; \mu_{\Gamma}; \Gamma; \Sigma \vdash b \xrightarrow{\delta} s \)
- \( \forall (a : l) \in \mu_{\lambda}; \mu_{\Gamma}; \Gamma; \Sigma \vdash \lambda_\nu(a) \xrightarrow{\delta} \lambda_\nu(l) \)
- \( \Gamma; \Sigma; \Psi \vdash \rho \)
- \( \Sigma; \Psi \vdash \sigma; \pi \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Gamma \xrightarrow{\delta} \sigma \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Sigma \xrightarrow{\delta} \sigma \)
- \( \lambda_\nu; \Gamma \vdash (H, R, \lambda_\nu(a)) \xrightarrow{\delta} (H', R', b') \)

then
- \( \Sigma; \lambda_\nu; \bar{\rho}; \rho \vdash (\sigma, \pi, s) \xrightarrow{\delta} \text{err or} \)
- \( \Sigma; \lambda_\nu; \bar{\rho}; \rho \vdash (\sigma', \pi', s') \).

**Proposition 16** (Preservation of Related Memory for Basic Blocks). If
- \( \mu_{\lambda}; \mu_{\Gamma}; \Gamma; \Sigma \vdash b \xrightarrow{\delta} s \)
- \( \forall (a : l) \in \mu_{\lambda}; \mu_{\Gamma}; \Gamma; \Sigma \vdash \lambda_\nu(a) \xrightarrow{\delta} \lambda_\nu(l) \)
- \( \Gamma; \Sigma; \Psi \vdash \rho \)
- \( \Sigma; \Psi \vdash \sigma; \pi \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Gamma \xrightarrow{\delta} \sigma \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Sigma \xrightarrow{\delta} \sigma \)
- \( \lambda_\nu; \Gamma \vdash (H, R, \lambda_\nu(a)) \xrightarrow{\delta} (H', R', b') \)

then for some \( \mu' \geq \mu \) and \( \nu' \geq \nu \):
- \( \mu'_{\lambda}; \mu'_{\Gamma}; \pi; \bar{\rho}; \rho \vdash \Gamma \xrightarrow{\delta} \sigma \)
- \( \mu'_{\lambda}; \nu'_{\lambda}; \pi; \bar{\rho}; \rho \vdash H' \xrightarrow{\delta} \sigma' \)

**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
- \( \Sigma \vdash f(x : \bar{x}, y : \bar{y}, a, \lambda_\nu(j)) \xrightarrow{\delta} f(x : \bar{x}, y : \bar{y}, l, \lambda_\nu(j)) \)
- \( \mu_{\nu} = \{ \bar{x} \xrightarrow{\delta} \bar{x}, \bar{y} \xrightarrow{\delta} \bar{y} \} \)
- \( \Gamma_{\nu} = \{ x : \bar{x}, y : \bar{y} \} \)
- \( \Gamma; \Sigma; \Psi \vdash \rho \)
- \( \Sigma; \Psi \vdash \sigma; \pi \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Gamma \xrightarrow{\delta} \sigma \)
- \( \mu_{\lambda}; \nu_{\lambda}; \pi; \bar{\rho}; \rho \vdash \Sigma \xrightarrow{\delta} \sigma \)
- \( \lambda_\nu; \Gamma \vdash (H, R, \lambda_\nu(a)) \xrightarrow{\delta} (H', R', b') \)

then
- \( \Sigma; \lambda_\nu; \bar{\rho}; \rho \vdash (\sigma, \pi, \lambda_\nu(l)) \xrightarrow{\delta} \text{err or} \)
- \( \Sigma; \lambda_\nu; \bar{\rho}; \rho \vdash (\sigma', \pi', s') \).

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