Data-flow analyses as effects and graded monads (additional proofs)

Andrej Ivašković
Department of Computer Science and Technology, University of Cambridge, UK
andrej.ivaskovic@cst.cam.ac.uk

Alan Mycroft
Department of Computer Science and Technology, University of Cambridge, UK
alan.mycroft@cst.cam.ac.uk

Dominic Orchard
School of Computing, University of Kent, UK
d.a.orchard@kent.ac.uk

The following document is an extended appendix for the paper of the same name which appears in FSCD 2020. This document provides full proofs that were not included in the appendix due to space limitations. Proofs of some auxiliary lemmas are mechanised in Agda, which is included with this material as `GradedMonad.agda`, tested with Agda version 2.6.0.1.

1 Type soundness of the graded monadic McCarthy transformation

▶ **Lemma 2** (paper). *For all* Φ, ℓ, S, τ

$$\Phi \vdash (\ell : S) : \tau \,\&\, \Phi(\ell) \;\wedge\; \forall \ell' \in \mathsf{dom}(\Phi).(g_{\ell'} : \mathsf{MultiState}^{\Phi(\ell')}_{\mathsf{triv}}\ \mathsf{Int})$$
$$\implies \quad [\![\ell : S]\!]_{GM} : \mathsf{MultiState}^{\Phi(\ell)}_{\mathsf{triv}}\ \tau$$

Proof. We proceed by induction on the structure of CFG effect-system derivations.

- Case (ASSIGN)

$$(\textsc{assign})\ \frac{\Phi(\ell) = \langle\!\langle \ell : X := e \rangle\!\rangle_{\mathrm{TF}} \rhd \Phi(\ell')}{\Phi \vdash (\ell : X := e;\ \mathsf{goto}\ \ell') : \mathsf{Int} \,\&\, \Phi(\ell)}$$

For liveness: $\langle\!\langle \ell : X := e \rangle\!\rangle_{\mathrm{TF}} = \lambda s.\ (s \setminus \mathit{kill}(\ell : X := e)) \cup \mathit{gen}(\ell : X := e)$.
We consider the cases for e specified in the translation (other cases are similar):

- $e = Y + Z$
  Thus, $\langle\!\langle \ell : X := Y + Z \rangle\!\rangle_{\mathrm{TF}} = \lambda s.\ (s \setminus \{X\}) \cup \{Y, Z\}$.
  The translation is as follows:

$$[\![\ell : X := Y + Z;\ \mathsf{goto}\ \ell']\!]_{\mathrm{GM}} = (g_\ell = \mathsf{do}\ \{y \leftarrow \mathsf{getY};\ z \leftarrow \mathsf{getZ};\ \mathsf{putX}\ (y + z);\ g_{\ell'}\})$$

  Since $g_{\ell'}$ has type $\mathsf{MultiState}^{\Phi(\ell')}_{\mathsf{triv}}\ \mathsf{Int}$ (second premise) then the above translation has type $\mathsf{MultiState}^{\phi}_{\mathsf{triv}}\ \mathsf{Int}$ where ϕ is equal to:

$$\phi = \mathit{gen}_Y \rhd \mathit{gen}_Z \rhd \mathit{kill}_X \rhd \Phi(\ell') = (\lambda s.\ (s \setminus \{X\}) \cup \{Y, Z\}) \rhd \Phi(\ell')$$

  which matches the CFG effect information $\Phi(\ell)$ for the ASSIGN rule with $e = Y + Z$.
- $e = k$ (where k is an integer constant)
  Thus, $\langle\!\langle \ell : X := k \rangle\!\rangle_{\mathrm{TF}} = \lambda s.\ (s \setminus \{X\})$.
  The translation is as follows:

$$[\![\ell : X := k;\ \mathsf{goto}\ \ell']\!]_{\mathrm{GM}} = (g_\ell = \mathsf{do}\ \{\mathsf{putX}\ k;\ g_{\ell'}\})$$

Since $g_{\ell'}$ has type $\mathsf{MultiState}^{\Phi(\ell')}_{\mathsf{triv}}\ \mathsf{Int}$ (second premise) then the above translation above translation has type $\mathsf{MultiState}^{\phi}_{\mathsf{triv}}\ \mathsf{Int}$ where ϕ is equal to:

$$\phi = \mathit{kill}_X \rhd \Phi(\ell') = (\lambda s.\ s \setminus \{X\}) \rhd \Phi(\ell')$$

which matches the CFG effect information $\Phi(\ell)$ for the ASSIGN rule with $e = k$.

- Case (HALT)

$$(\text{HALT})\ \frac{\Phi(\ell) = \langle\!\langle v \rangle\!\rangle_{\mathrm{TF}}}{\Phi \vdash (\ell : \mathsf{halt}\ v) : \mathsf{Int}\ \&\ \Phi(\ell)}$$

First, for liveness $\langle\!\langle v \rangle\!\rangle_{\mathrm{TF}} = \lambda s.\ s \cup \mathit{fv}(v)$.
We consider the two possible cases.

- $v = k$ (where k is an integer constant)
  In this case, $\mathit{fv}(k) = \emptyset$, so $\Phi(\ell) = \langle\!\langle k \rangle\!\rangle_{\mathrm{TF}} = \mathit{id}$.
  The translation is simple: $[\![\ell : \mathsf{halt}\ k]\!]_{\mathrm{GM}} = (g_\ell = \mathsf{return}\ k)$.
  The type of $\mathsf{return}\ k$ is $\mathsf{MultiState}^{\mathit{id}}_{\mathsf{triv}}\ \mathsf{Int}$.
  Since $\Phi(\ell) = \mathit{id}$, we are done.
- $v = X$
  In this case $\mathit{fv}(X) = \{X\}$, hence $\Phi(\ell) = \langle\!\langle X \rangle\!\rangle_{\mathrm{TF}} = \lambda s.\ s \cup \{X\} = \mathit{gen}_X$.
  The translation is:

  $$[\![\ell : \mathsf{halt}\ X]\!]_{\mathrm{GM}} = (g_\ell = \mathsf{do}\ \{x \leftarrow \mathsf{getX};\ \mathsf{return}\ x\})$$

  The type of g_ℓ is $\mathsf{MultiState}^{\phi}_{\mathsf{triv}}\ \mathsf{Int}$, where $\phi = \mathit{gen}_X \rhd \mathit{id} = \mathit{gen}_X$. This matches $\Phi(\ell)$, which we have shown is also $\mathit{gen}_(X)$.

- Case (IF)

$$(\text{IF})\ \frac{\Phi(\ell) = \langle\!\langle v \rangle\!\rangle_{\mathrm{TF}} \rhd (\Phi(\ell_1) \sqcup \Phi(\ell_2))}{\Phi \vdash (\ell : \mathsf{if}\ v \geq 0\ \mathsf{then\ goto}\ \ell_1\ \mathsf{else\ goto}\ \ell_2) : \mathsf{Int}\ \&\ \Phi(\ell)}$$

As a consequence of the second premise, we infer the types of g_{ℓ_1} and g_{ℓ_2} are $\mathsf{MultiState}^{\Phi(\ell_1)}_{\mathsf{triv}}\ \mathsf{Int}$ and $\mathsf{MultiState}^{\Phi(\ell_2)}_{\mathsf{triv}}\ \mathsf{Int}$, respectively.
By assumption we also have $\Phi(\ell) = \langle\!\langle v \rangle\!\rangle_{\mathrm{TF}} \rhd (\Phi(\ell_1) \sqcup \Phi(\ell_2))$.
Consider two cases for v:

- $v = k$ (where k is an integer constant)
  In this case $\langle\!\langle v \rangle\!\rangle_{\mathrm{TF}} = \mathit{id}$, so $\Phi(\ell) = \Phi(\ell_1) \sqcup \Phi(\ell_2)$.
  The conditional branch gets translated as follows:

  $$[\![\ell : \mathsf{if}\ k \geq 0\ \mathsf{then\ goto}\ \ell_1\ \mathsf{else\ goto}\ \ell_2]\!]_{\mathrm{GM}} = (g_\ell = \mathsf{if}\ k \geq 0\ \mathsf{then\ sub}\ g_{\ell_1}\ \mathsf{else\ sub}\ g_{\ell_2})$$

  Recall that, for a general graded monad M:

  $$\mathsf{sub} : \forall r \forall s \forall \alpha.\ M^r \alpha \to M^s \alpha \qquad \mathit{if}\ r \sqsubseteq s$$

  Therefore, in the case of this graded monad, $\mathsf{sub}\ g_{\ell_1}$ and $\mathsf{sub}\ g_{\ell_2}$ can be typed as:

  $$\mathsf{sub}\ g_{\ell_1} : \mathsf{MultiState}^{\Phi(\ell_1) \sqcup \Phi(\ell_2)}_{\mathsf{triv}}\ \mathsf{Int}$$
  $$\mathsf{sub}\ g_{\ell_2} : \mathsf{MultiState}^{\Phi(\ell_1) \sqcup \Phi(\ell_2)}_{\mathsf{triv}}\ \mathsf{Int}$$

  since the premise tells us that $\Phi(\ell_1) \sqcup \Phi(\ell_2)$ is defined and is the upper-bound of both $\Phi(\ell_1)$ and $\Phi(\ell_2)$ by the usual properties of the partial order.
  The type of g_ℓ is then $\mathsf{MultiState}^{\phi}_{\mathsf{triv}}\ \mathsf{Int}$, where $\phi = \Phi(\ell_1) \sqcup \Phi(\ell_2)$, which matches $\Phi(\ell)$.

- $v = X$
  In this case $\langle\!\langle v \rangle\!\rangle_{\mathrm{TF}} = gen_X$, so $\Phi(\ell) = gen_X \rhd (\Phi(\ell_1) \sqcup \Phi(\ell_2))$.
  The conditional branch gets translated as follows:

$$\begin{aligned}&[\![\ell : \mathsf{if}\ X \geq 0\ \mathsf{then\ goto}\ \ell_1\ \mathsf{else\ goto}\ \ell_2]\!]_{\mathrm{GM}} \\ &\quad = \quad (g_\ell = \mathsf{do}\ \{\ x \leftarrow \mathsf{getX}; \\ &\qquad\qquad\qquad \mathsf{if}\ x \geq 0\ \mathsf{then\ sub}\ g_{\ell_1}\ \mathsf{else\ sub}\ g_{\ell_2}\})\end{aligned}$$

  The type of g_ℓ is then $\mathsf{MultiState}^{\phi}_{\mathsf{triv}}\ \mathsf{Int}$, where $\phi = gen_X \rhd (\Phi(\ell_1) \sqcup \Phi(\ell_2))$ following the same reasoning as in the previous case with regards sub, which matches $\Phi(\ell)$.

□

2 Properties of operations over transfer functions involved in the graded monad construction.

We first give some intermediate lemmas about transfer functions (as generated by our analysis) which are used in proving the rest of the results here. These are mechanised in `GradedMonad.agda`.

▶ **Lemma 1.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \in reads(\phi) \implies x \in reads(\phi \rhd \phi')$$

or said another way (expanding the definition of reads):

$$\phi(\emptyset) \subseteq (\phi \rhd \phi')(\emptyset)$$

Proof. Mechanised in Agda.

□

▶ **Lemma 2.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \in footprint(\phi) \implies x \in footprint(\phi \rhd \phi')$$

Proof. Mechanised in Agda.

□

▶ **Lemma 3.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \in footprint(\phi) \implies x \in footprint(\phi' \rhd \phi)$$

(note this is the symmetric case to the above).

Proof. Mechanised in Agda.

□

▶ **Lemma 4.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \notin footprint(\phi) \wedge x \notin footprint(\phi') \iff x \notin footprint(\phi \rhd \phi')$$

Proof. Mechanised in Agda.

□

▶ **Lemma 5.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ and for all ϕ'' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi''}\ \tau$ then:*

$$\forall x.\ x \notin \mathit{footprint}(\phi \rhd \phi') \wedge x \in \mathit{reads}(\phi \rhd \phi' \rhd \phi'') \implies x \in \mathit{reads}(\phi' \rhd \phi'')$$

Proof. Mechanised in Agda.

□

▶ **Lemma 6.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \notin \mathit{footprint}(\phi) \wedge x \in \mathit{reads}(\phi') \implies x \in \mathit{reads}(\phi \rhd \phi')$$

Proof. Mechanised in Agda.

□

▶ **Lemma 7.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ and for all ϕ'' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi''}\ \tau$ then:*

$$\forall x.\ x \notin \mathit{footprint}(\phi) \wedge x \in \mathit{reads}(\phi' \rhd \phi'') \implies x \in \mathit{reads}(\phi \rhd \phi' \rhd \phi'')$$

Proof. Mechanised in Agda.

□

▶ **Lemma 8.** *For all ϕ such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi}\ \tau$ and for all ϕ' such that $\exists \ell, S, \tau$ where $[\![(l : S)]\!]_{GM} : \mathsf{MultiState}^{\phi'}\ \tau$ then:*

$$\forall x.\ x \notin \mathit{footprint}(\phi) \wedge x \notin \mathit{reads}(\phi \rhd \phi') \implies x \notin \mathit{reads}(\phi')$$

Proof. Mechanised in Agda. Note this is a trivial consequence of Lemma 6 using classical logic, but is stated here as a separate result and proved constructively.

□

3 Results on state-management operations that provide the graded monad axioms

For reference, the following repeats the core definitions used in the graded monad construction (show in the appendix of the paper).

▶ **Definition 9.** *Firstly, recall the notion of the* reads *set of variables for a computation (use to compute the domain of the input state) and the* footprint *set of variables for a computation (used to compute the domain of the output state):*

$$\begin{aligned} \mathit{reads}(\phi) &= \phi(\emptyset) \\ \mathit{footprint}(\phi) &= \phi(\emptyset) \cup (\mathit{Vars} \setminus \phi(\mathit{Vars})) \end{aligned}$$

Next, the following computes a restriction on an input state:

$$\downarrow_{\phi,\phi'} : \mathsf{Store}(\mathit{reads}(\phi \rhd \phi')) \to \mathsf{Store}(\mathit{reads}(\phi)) = \lambda s.s_{|\phi(\emptyset)}$$

The following operation merges two stores, by padding the second parameter store (the output of a computation labelled with transfer function ϕ) with values from the first parameter (the input of a composite computation labelled by transfer function $\phi \rhd \phi'$), to provide the input for the computation labelled with ϕ':

$$\blacktriangleleft_{\phi,\phi'} : \mathsf{Store}(\mathit{reads}(\phi \rhd \phi')) \times \mathsf{Store}(\mathit{footprint}(\phi)) \to \mathsf{Store}(\mathit{reads}(\phi'))$$
$$= \lambda(s, s').\{ \begin{cases} x \mapsto s'(x) & x \in \mathit{footprint}(\phi) \\ x \mapsto s(x) & x \in \mathit{reads}(\phi \rhd \phi') \wedge x \notin \mathit{footprint}(\phi) \end{cases} \}$$

where $x \in \mathit{reads}(\phi')$. Thus values in the second parameter store take precedence.

The last operation is used to merge the output store of two computations by padding the second parameter store with values from the first parameter store:

$$\triangleleft_{\phi,\phi'} : \mathsf{Store}(\mathit{footprint}(\phi)) \times \mathsf{Store}(\mathit{footprint}(\phi')) \to \mathsf{Store}(\mathit{footprint}(\phi \rhd \phi'))$$
$$= \lambda(s, s').\{ \begin{cases} x \mapsto s'(x) & x \in \mathit{footprint}(\phi') \\ x \mapsto s(x) & x \in \mathit{footprint}(\phi) \wedge x \notin \mathit{footprint}(\phi') \end{cases} \}$$

where $x \in \mathit{footprint}(\phi \rhd \phi')$. Thus values in the second parameter store take precedence.

▶ **Proposition 5** (Restriction right unit). *$\forall \phi$ and $s \in \mathsf{Store}(\mathit{reads}(\phi))$ then $\downarrow_{\phi,id} s \equiv s$*

Proof. Expanding the definition of restriction in the setting of this premise we then have:

$$\downarrow_{\phi,id} s = s_{|(\phi(\emptyset))} : \mathsf{Store}(\mathit{reads}(\phi))$$

Since $s \in \mathsf{Store}(\mathit{reads}(\phi))$ then restricting the domain of s to its own domain is the identity, i.e., $\downarrow_{\phi,id} s = s$.

□

▶ **Proposition 6** (Merge $\triangleleft$ right unit). *$\forall \phi$ and $s \in \mathsf{Store}(\mathit{footprint}(\phi))$ then $s \triangleleft_{\phi,id} \hat{\emptyset} \equiv s$*

Proof. We have that $\mathit{footprint}(id) = id(\emptyset) \cup (\mathit{Vars} \setminus id(\mathit{Vars})) = \emptyset$.

Expanding the definition of merge $\triangleleft$ in the setting of this premise, we have:

$$s \triangleleft_{\phi,id} \hat{\emptyset} = \{ \begin{cases} x \mapsto \hat{\emptyset}(x) & x \in \mathit{footprint}(\emptyset) \\ x \mapsto s(x) & x \in \mathit{footprint}(\phi) \wedge x \notin \mathit{footprint}(\emptyset) \end{cases} \}$$

Since $\mathit{footprint}(\emptyset) = \emptyset$, then $x \in \mathit{footprint}(\emptyset) \equiv \mathit{false}$ thus we can rewrite the above to:

$$s \triangleleft_{\phi,id} \hat{\emptyset} = \{ \begin{cases} x \mapsto \hat{\emptyset}(x) & \mathit{false} \\ x \mapsto s(x) & x \in \mathit{footprint}(\phi) \wedge x \notin \emptyset \end{cases} \}$$
$$= \{ \begin{cases} x \mapsto s(x) & x \in \mathit{footprint}(\phi) \end{cases} \}$$

As $\mathsf{dom}(s) = \mathit{footprint}(\phi)$ (from the premise) then $s \triangleleft_{\phi,id} \hat{\emptyset} = s$.

□

▶ **Proposition 7** (Merge $\blacktriangleleft$ right unit). *$\forall \phi'$ and $s \in \mathsf{Store}(\mathit{reads}(\phi'))$ then $s \blacktriangleleft_{id,\phi'} \hat{\emptyset} \equiv s$*

Proof. As in the previous proposition, we have that: $\mathit{footprint}(id) = \emptyset$ (hence the type-soundness of the propositions statement here).

Using this fact, and expanding the definition of merge $\blacktriangleleft$ as applied in the premise, gives:

$$s \blacktriangleleft_{id,\phi'} \hat{\emptyset} = \{ \begin{cases} x \mapsto \hat{\emptyset}(x) & x \in \emptyset \\ x \mapsto s(x) & x \in \mathit{reads}(\phi') \wedge x \notin \emptyset \end{cases} \}$$

Since $x \in \emptyset \equiv \mathit{false}$, and by the left identity property of the transfer function pomonoid, we can rewrite the above to:

$$\begin{aligned} s \blacktriangleleft_{id,\phi'} \hat{\emptyset} &= \{ \begin{cases} x \mapsto \hat{\emptyset}(x) & \mathit{false} \\ x \mapsto s(x) & x \in \mathit{reads}(\phi') \end{cases} \} \\ &= \{ \{ x \mapsto s(x) \quad x \in \mathit{reads}(\phi') \} \end{aligned}$$

As $\mathsf{dom}(s) = \mathit{reads}(\phi')$ (which matches the domain of the result here) then this is the identity and thus $s \blacktriangleleft_{id,\phi'} \hat{\emptyset} \equiv s$

□

▶ **Proposition 8** (Merge $\triangleleft$ left unit). *$\forall \phi'$ and $s \in \mathsf{Store}(\mathit{footprint}(\phi'))$ then $\hat{\emptyset} \triangleleft_{id,\phi'} s \equiv s$*

Proof. Since $\mathit{footprint}(id) = \emptyset$, expanding the definitions for the premise we get:

$$\hat{\emptyset} \triangleleft_{id,\phi'} s = \{ \begin{cases} x \mapsto s(x) & x \in \mathit{footprint}(\phi') \\ x \mapsto \hat{\emptyset}(x) & x \in \emptyset \wedge x \notin \mathit{footprint}(\phi') \end{cases} \}$$

As $x \in \mathit{footprint}(id) \equiv x \in \emptyset \equiv \mathit{false}$ then the above reduces to:

$$\hat{\emptyset} \triangleleft_{id,\phi'} s = \{ \{ x \mapsto s(x) \quad x \in \mathit{footprint}(\phi') \}$$

Since $\mathsf{dom}(s') = \mathit{footprint}(\phi)$, which matches the domain of the result, then the above collapses to $\hat{\emptyset} \triangleleft_{id,\phi'} s = s$.

□

▶ **Proposition 9** (Restriction closure). *$\forall \phi, \phi', \phi''$ and $s \in \mathsf{Store}(\mathit{reads}(((\phi \rhd \phi') \rhd \phi')))$ then:* $\downarrow_{\phi,\phi'} (\downarrow_{\phi \rhd \phi', \phi''} s) \equiv \downarrow_{\phi, \phi' \rhd \phi''} s$

Proof. The typing of the state in this lemma relies on the associativity of $\rhd$.

Expanding the definition of restriction on the equality here, we need to show that:

$$\begin{aligned} \downarrow_{\phi,\phi'} (\downarrow_{\phi \rhd \phi', \phi''} s) &= (s_{|(\phi \rhd \phi')(\emptyset)})_{|(\phi(\emptyset))} \\ \downarrow_{\phi, \phi' \rhd \phi''} s &= s_{|\phi(\emptyset)} \end{aligned}$$

where $s_{|(\phi \rhd \phi')(\emptyset)} : \mathsf{Store}(\mathit{reads}(\phi \rhd \phi'))$.

By Lemma 1, we have that: $\mathit{reads}(\phi) \subseteq \mathit{reads}(\phi \rhd \phi') \subseteq \mathit{reads}(\phi \rhd \phi' \rhd \phi'')$.

Therefore, by the property of function restriction that $(f_{|Y})_{|X} = f_{|X}$ if $Y \subseteq X$, then we have $(s_{|(\phi \rhd \phi')(\emptyset)})_{|(\phi(\emptyset))} \equiv s_{|\phi(\emptyset)}$.

□

▶ **Proposition 10** (Merge $\lhd$ associativity). *$\forall \phi, \phi', \phi''$ and $s \in \mathsf{Store}(\mathit{footprint}(\phi))$ $s' \in \mathsf{Store}(\mathit{footprint}(\phi'))$ $s'' \in \mathsf{Store}(\mathit{footprint}(\phi''))$ then:*
$(s \lhd_{\phi,\phi'} s') \lhd_{(\phi \rhd \phi'),\phi''} s'' \equiv s \lhd_{\phi,\phi' \rhd \phi''} (s' \lhd_{\phi',\phi''} s'')$.

Proof. As an abbreviation we write $\mathsf{fp}(\phi) = \mathit{footprint}(\phi)$. We also use that $A \wedge B \wedge (A \Rightarrow B) \equiv A \wedge (A \Rightarrow B)$ (labelled *eqv* in the following proofs) to simplify the cases here.

Expanding the definition, we have that:

$$
\begin{aligned}
& (s \lhd_{\phi,\phi'} s') \lhd_{(\phi \rhd \phi'),\phi''} s'' \\
= \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ \{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \end{cases} \}(x) & x \in \mathsf{fp}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi'') \end{cases} \} \\
= \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi'') \end{cases} \} \\
\{\mathit{eqv}\text{+}L.3\} = \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi'') \end{cases} \} \\
\{\mathit{eqv}\text{+}L.2\} = \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \end{cases} \} \\
\{L.4\} = \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi' \rhd \phi'') \end{cases} \} \qquad (*)
\end{aligned}
$$

$$
\begin{aligned}
& s \lhd_{\phi,\phi' \rhd \phi''} (s' \lhd_{\phi',\phi''} s'') \\
= \{ & \begin{cases} \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \end{cases} & x \in \mathsf{fp}(\phi' \rhd \phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi' \rhd \phi'') \end{cases} \} \\
= \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \wedge x \in \mathsf{fp}(\phi' \rhd \phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \wedge x \in \mathsf{fp}(\phi' \rhd \phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi' \rhd \phi'') \end{cases} \} \\
\{\mathit{eqv}\text{+}L.3\} = \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \wedge x \in \mathsf{fp}(\phi' \rhd \phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi' \rhd \phi'') \end{cases} \} \\
\{\mathit{eqv}\text{+}L.2\} = \{ & \begin{cases} x \mapsto s''(x) & x \in \mathsf{fp}(\phi'') \\ x \mapsto s'(x) & x \in \mathsf{fp}(\phi') \wedge x \notin \mathsf{fp}(\phi'') \\ x \mapsto s(x) & x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi' \rhd \phi'') \end{cases} \} \\
= (*)
\end{aligned}
$$

□

▶ **Proposition 11** (Merge $\triangleleft$/$\blacktriangleleft$ associativity). *$\forall \phi, \phi', \phi''$ and $s \in \mathsf{Store}(\mathit{reads}((\phi \rhd \phi') \rhd \phi''))$ and $s' \in \mathsf{Store}(\mathit{footprint}(\phi))$ and $s'' \in \mathsf{Store}(\mathit{footprint}(\phi'))$ then:*

$$s \blacktriangleleft_{(\phi \rhd \phi'),\phi''} (s' \triangleleft_{\phi,\phi'} s'') \equiv (s \blacktriangleleft_{\phi,\phi' \rhd \phi''} s') \blacktriangleleft_{\phi',\phi''} s''$$

Proof. Expand the LHS and RHS by the definitions (where in the first expansion we include the domain restriction of $\blacktriangleleft$, in the condition) (and we write $\mathsf{rd}(\phi)$ for $\mathit{reads}(\phi)$ for brevity):

$$\begin{aligned}
& s \blacktriangleleft_{(\phi \rhd \phi'),\phi''} (s' \triangleleft_{\phi,\phi'} s'') \\
&= \{ \begin{cases} x \mapsto \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \end{cases} (x) & x \in \mathsf{fp}(\phi \rhd \phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \\
&= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \\
\{eqv{+}L.3\} &= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \wedge x \in \mathsf{fp}(\phi \rhd \phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \\
\{eqv{+}L.2\} &= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \quad (*)
\end{aligned}$$

$$\begin{aligned}
& (s \blacktriangleleft_{\phi,\phi' \rhd \phi''} s') \blacktriangleleft_{\phi',\phi''} s'' \\
&= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto \{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \end{cases} (x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi') \end{cases} \} \\
&= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi') \end{cases} \} \\
\{L.4\} &= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \end{cases} \} \\
\{eqv{+}L.5\} &= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \\
\{eqv{+}L.6\} &= \{ \begin{cases} x \mapsto s''(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi') \\ x \mapsto s'(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \notin \mathsf{fp}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi \rhd \phi') \end{cases} \} \\
&= (*)
\end{aligned}$$

□

▶ **Proposition 12** (Merge ◀/restriction commutativity). *$\forall \phi, \phi', \phi''$ and $s \in \mathsf{Store}(\mathit{reads}((\phi \rhd \phi') \rhd \phi''))$ and $s' \in \mathsf{Store}(\mathit{footprint}(\phi))$ then: $(\downarrow_{\phi \rhd \phi', \phi''} s) \blacktriangleleft_{\phi,\phi'} s' \equiv \downarrow_{\phi',\phi''} (s \blacktriangleleft_{\phi,\phi' \rhd \phi''} s')$*

Proof. Recall the property of standard set-theoretic function restriction that if $x \in A \implies f_{|A}(x) \equiv f(x)$. Call this (*).

Expanding the definitions of the left and the right hand sides of the goal, we rewrite them both into equivalent forms. We also include the domain restriction of $\blacktriangleleft$,:

$$
\begin{aligned}
&(\downarrow_{\phi \rhd \phi', \phi''} s) \blacktriangleleft_{\phi,\phi'} s' : \mathsf{Store}(\mathit{reads}(\phi')) \\
=&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s_{|(\phi \rhd \phi')(\emptyset)}(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{rd}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi) \end{cases} \} \\
\{(*)\} =&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{rd}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi) \end{cases} \} (**)
\end{aligned}
$$

$$
\begin{aligned}
&\downarrow_{\phi',\phi''} (s \blacktriangleleft_{\phi,\phi' \rhd \phi''} s') : \mathsf{Store}(\mathit{reads}(\phi')) \\
=&(\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \end{cases} \})_{|\phi'(\emptyset)} \\
=&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \in \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi') \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi') \end{cases} \} \\
\{eqv{+}L.1\} =&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \in \mathsf{rd}(\phi \rhd \phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi') \end{cases} \} \\
\{eqv{+}L.7\} =&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi' \rhd \phi'') \wedge x \notin \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi') \end{cases} \} \\
\{eqv{+}L.1\} =&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \notin \mathsf{fp}(\phi) \wedge x \in \mathsf{rd}(\phi') \end{cases} \} \\
\{eqv{+}L.6\} =&\{ \begin{cases} x \mapsto s'(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{fp}(\phi) \\ x \mapsto s(x) & x \in \mathsf{rd}(\phi') \wedge x \in \mathsf{rd}(\phi \rhd \phi') \wedge x \notin \mathsf{fp}(\phi) \end{cases} \} \\
=&(**)
\end{aligned}
$$

□