Effective Entailment Checking for Separation Logic with Inductive Definitions: Supplementary Material

Jens Katelaan[1], Christoph Matheja[2], and Florian Zuleger[1]

[1] TU Wien, Vienna, Austria
[2] RWTH Aachen University, Aachen, Germany

1 Overview of the Supplementary Material

- In Section 2 we introduce some additional notation used in the proofs.
- In Section 3 we show properties of heap graphs and the rename, forget, and composition operations that will be useful for proving the correctness of our approach.
- In Section 4 we relate the separation-logic semantics to the heap-graph operations.
- In Section 5 we justify our restriction to SIDs without equalities between variables, parameter repetitions or unsatisfiable unfoldings.
- Section 6 contains the proofs of all lemmas and theorems stated in Section 4 of the main paper.
- Section 7 contains the proofs of all lemmas and theorems stated in Section 5 of the main paper.
- In Section 8 we explain the extension our profile-based entailment checking to entailment queries between arbitrary symbolic heaps (as opposed to just predicate calls).
- Section 9 contains the SID definitions for all benchmarks used in Table 1 in the main paper.
- In Section 10 we present our full experimental results.

1.1 Proof Index

Table 1 (on the next page) will help you find the lemmas and theorems from the main paper in this document.

2 Additional Notation

We will use the following additional notation in this document.

- Let f be a partial function $\mathrm{elems}(f) := \mathrm{dom}(f) \cup \mathrm{img}(f)$.
- We define $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\langle z_1, \ldots, z_n \rangle) := \langle f(z_1), \ldots, f(z_n) \rangle$, where

$$f \colon \mathbf{Var} \to \mathbf{Var}, \quad z \mapsto \begin{cases} \mathbf{y}[i] & \text{if } \mathbf{x}[i] = z \\ z & \text{otherwise .} \end{cases}$$

Table 1: Finding lemmas and theorems in the supplementary material.

Result	In the paper	In this document
SL semantics vs. heap-graph operations	Lemma 1	Lemma 14 (p. 6)
Our restricted SIDs are a normal form of $\mathrm{SL}_{\mathrm{btw}}$	Section 3	Section 5 (p. 6)
$\mathsf{profile}_{\Phi}$ is a sound abstraction	Lemma 2	Lemma 15 (p. 7)
$\mathbf{Profiles}^{\mathbf{y}}(\Phi)$ is finite	Lemma 3	Corollary 1 (p. 9)
$\mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{y})$ is computable	Lemma 4	Lemma 18 (p. 9)
$\mathsf{profile}_{\Phi}$ is a homomorphism	Theorem 2	Theorem 4 (p. 16)
$\mathsf{abstractSID}$ terminates	Section 5	Lemma 34 (p. 18)
pred_1 entails pred_2 iff all pred_1-profiles contain pred_2	Theorem 3	Theorem 6 (p. 19)
$\mathsf{abstractSID}$ is correct	Theorem 4	Theorem 5 (p. 18)
$\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$ is decidable for $\mathrm{SL}_{\mathrm{btw}}$	Corollary 1	Corollary 2 (p. 20)
$\varphi \models_{\Phi} \psi$ for established φ, ψ is decidable for $\mathrm{SL}_{\mathrm{btw}}$	Section 5	Section 8 (p. 20)

- **HG** is the set of all heap graphs; **CHG** is the set of all concrete heap graphs, i.e., heap graphs $\mathcal{M}$ with $\mathsf{calls}_{\mathcal{M}} = \emptyset$; and **AHG** is the set of all abstract heap graphs, i.e., heap graphs with $\mathsf{calls}_{\mathcal{M}} \neq \emptyset$.
- We collect all concrete heap graphs whose free variables are a subset of $\mathbf{y}$ in $\mathbf{CHG}^{\mathbf{y}}$, i.e. $\mathbf{CHG}^{\mathbf{y}} := \{\mathcal{M} \in \mathbf{CHG} \mid \mathsf{FV}_{\mathcal{M}} \subseteq \mathbf{y}\}$.
- We write $\mathcal{M}_1 \subseteq \mathcal{M}$ iff $\mathcal{M}_1$ is a *subgraph* of $\mathcal{M}$, i.e., iff there exists an $\mathcal{M}_2$ such that $\mathcal{M}_1 \bullet \mathcal{M}_2 \cong \mathcal{M}$.
- Let $\mathcal{M} \in \mathbf{HG}$. Let calls be a set of predicate calls. We define $\mathcal{M} - \mathsf{calls} := \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}_{\mathcal{M}} \setminus \mathsf{calls} \rangle$ and $\mathcal{M} + \mathsf{calls} := \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}_{\mathcal{M}} \cup \mathsf{calls} \rangle$.
- We denote by $x \rightarrow^*_{\mathcal{M}} y$ that $\langle x, y \rangle$ is in the reflexive-transitive closure of $\mathsf{Ptr}_{\mathcal{M}}$.
- We write $\varphi = \exists \mathbf{y} \ . \ \Sigma * \Gamma$ to denote a symbolic heap with zero or more $*$-separated points-to assertions Σ and zero or more $*$-separated predicate calls Γ.

3 Properties of Heap Graphs

3.1 Properties of the Heap-Graph Operations $\bullet$, $\mathsf{rename}_{\mathbf{x},\mathbf{y}}$ and $\mathsf{forget}_{\mathbf{x}}$

Lemma 1 (Commutativity and associativity of $\bullet$). *Let $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ be heap graphs. Then $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}_2 \bullet \mathcal{M}_1$ and $(\mathcal{M}_1 \bullet \mathcal{M}_2) \bullet \mathcal{M}_3 = \mathcal{M}_1 \bullet (\mathcal{M}_2 \bullet \mathcal{M}_3)$.*

Proof. As usual, we interpret the above statements up to isomorphism of heap graphs. The result then follows immediately from the associativity and commutativity of set union, $\cup$, used to define $\bullet$.

Lemma 2 (Rename is invertible). *Let $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$ be defined. Then*

$$\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})) \cong \mathcal{M}.$$

Proof. Let $\mathcal{M}' := \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$. Recall that by definition, the two operations $\mathsf{rename}_{\mathbf{x},\mathbf{y}}$ and $\mathsf{rename}_{\mathbf{y},\mathbf{x}}$ correspond to functions f and f' such that $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) = f(\mathcal{M})$ and $\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathcal{M}') = f'(\mathcal{M}')$. By examining the definition of these functions, it is easy to see that the functions $\hat{f} : \mathsf{vars}(\mathcal{M}) \to \mathsf{vars}(\mathcal{M}')$ and $\hat{f}' : \mathsf{vars}(\mathcal{M}') \to \mathsf{vars}(\mathcal{M})$ obtained by restricting f and f' to $\mathsf{vars}(\mathcal{M})$ and $\mathsf{vars}(\mathcal{M}')$ are bijective. $f \circ f'$ thus is the identity function for all variables $x \in \mathsf{vars}(\mathcal{M})$. Therefore, $f(f'(\mathcal{M})) = \mathcal{M}$.

Lemma 3. *Let* $\mathcal{M} = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$ *and let* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$ *be defined. Then* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_k)$

Proof. We can assume w.l.o.g. that $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_k$. (I.e., we can replace the $\mathcal{M}_i$ by appropriate isomorphic models prior to applying the composition operation rather than during the composition operation.) The result thus follows from the observation that $f(\mathcal{M}_1) \cup f(\mathcal{M}_2) = f(\mathcal{M}_1 \cup \mathcal{M}_2)$.

Lemma 4. *Let* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$. *Then there exist* $\mathcal{M}'_1, \ldots, \mathcal{M}'_k$ *such that* $\mathcal{M} = \mathcal{M}'_1 \bullet \cdots \bullet \mathcal{M}'_k$ *and for all* $1 \leq i \leq k$, $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}'_i) = \mathcal{M}_i$.

Proof. Combine Lemmas 2 and 3.

Lemma 5. *Let* $\mathcal{M} = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$. *Let* $\mathbf{x} \in \mathbf{Var}^*$ *be such that for every* $x \in \mathbf{x}$ *there exists at most one* $1 \leq i \leq k$ *such that* $x \in \mathrm{elems}(\mathsf{Ptr}_{\mathcal{M}})$. *Then* $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) = \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k)$.

Proof. If we remove $\mathbf{x}$ from all $\mathsf{FV}_{\mathcal{M}_i}$, it is also removed from

$$\mathsf{FV}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k)} = \mathsf{FV}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1)} \cup \cdots \cup \mathsf{FV}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k)}.$$

Modulo renaming auxiliary variables, $\mathsf{Ptr}_{\mathcal{M}} = \mathsf{Ptr}_{\mathcal{M}_1} \cup \cdots \cup \mathsf{Ptr}_{\mathcal{M}_k}$

Since, by assumption, every $x \in \mathbf{x}$ occurs in at most one $\mathsf{Ptr}_{\mathcal{M}_i}$, there is only one heap graph $\mathcal{M}_i$ in which $\mathbf{x}$ may be renamed after forgetting x. Thus $\mathsf{Ptr}_{\mathsf{forget}_x(\mathcal{M}'_i \bullet \mathcal{M}'_j)} = \mathsf{Ptr}_{\mathsf{forget}_x(\mathcal{M}'_i)} \cup \mathsf{Ptr}_{\mathsf{forget}_x(\mathcal{M}'_j)}$ for all $\mathcal{M}'_i \cong \mathcal{M}_i, \mathcal{M}'_j \cong \mathcal{M}_j$, $1 \leq i \neq j \leq k$. The result follows.

Lemma 6. *Let* $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$. *Then there exist* $\mathcal{M}'_1, \ldots, \mathcal{M}'_k$ *such that* $\mathcal{M} = \mathcal{M}'_1 \bullet \cdots \bullet \mathcal{M}'_k$ *and for all* $1 \leq i \leq k$, $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}'_i) = \mathcal{M}_i$.

Proof. Simply let $\mathcal{M}'_i := \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \cup \mathbf{x}, \mathsf{calls}_{\mathcal{M}} \rangle$. Clearly, $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}'_i) = \mathcal{M}_i$. Also, $\mathsf{FV}_{\mathcal{M}} = (\mathsf{FV}_{\mathcal{M}_1} \cup \cdots \cup \mathsf{FV}_{\mathcal{M}_k}) \cup \mathbf{x} = (\mathsf{FV}_{\mathcal{M}_1} \cup \mathbf{x}) \cup \cdots \cup (\mathsf{FV}_{\mathcal{M}_k} \cup \mathbf{x}) = \mathsf{FV}_{\mathcal{M}'_1} \cup \cdots \cup \mathsf{FV}_{\mathcal{M}'_k}$. Hence $\mathcal{M} = \mathcal{M}'_1 \bullet \cdots \bullet \mathcal{M}'_k$.

3.2 Rooted Heap Graphs

At several points in this document we will exploit that all models of predicates of SIDs that fall into the $\mathrm{SL}_{\mathrm{btw}}$ fragment are *rooted*.

Definition 1 (Rooted heap graph). *A heap graph* $\mathcal{M}$ *with* $\mathsf{Ptr}_{\mathcal{M}} \neq \emptyset$ *is rooted if there exists a variable* $x \in \mathsf{FV}_{\mathcal{M}}$ *such that for all* $y \in \mathrm{elems}(\mathsf{Ptr}_{\mathcal{M}})$, $x \to^*_{\mathcal{M}} y$. *We call* x *the* root *of* $\mathcal{M}$ *and say that* $\mathcal{M}$ *is* rooted *in* x.

Lemma 7 (SIDs only have rooted models.). *Let $\varPhi$ be an SID that satisfies connectivity and progress. Let $\mathcal{M} \in \mathbf{HG}$. If $\mathsf{Ptr}_{\mathcal{M}} \neq \emptyset$ and $\mathcal{M} \models_{\varPhi} \mathsf{pred}(\mathbf{x})$ for some $\mathsf{pred} \in \mathbf{Preds}(\varPhi)$, $\mathbf{x} \in \mathbf{Var}^*$ then there is a parameter $x \in \mathbf{x}$ such that $\mathcal{M}$ is rooted in x.*

Proof. We proceed by induction on the number n of rule applications used to derive that $\mathcal{M} \models_{\varPhi} \mathsf{pred}(\mathbf{x})$.

- $n = 1$: Since $\mathsf{Ptr}_{\mathcal{M}} \neq \emptyset$, there is a rule $(\mathsf{pred} \Leftarrow x \to \mathbf{z_0}) \in \mathbf{Rules}(\varPhi)$ such that $\mathcal{M} \models_{\varPhi} \exists x \to \mathbf{z_0}$. Note that the rule contains exactly one pointer because $\varPhi$ satisfies progress. Consequently, $\mathcal{M}$ is rooted in x. Moreover, by definition of the semantics of points-to assertions, $x \in \mathbf{x}$.
- $n > 1$: $\mathcal{M}$ is derived via a rule $(\mathsf{pred} \Leftarrow \exists \mathbf{y} \ . \ x \to \mathbf{z_0} * \mathsf{pred}_1(\mathbf{z_1}) * \cdots * \mathsf{pred}_k(\mathbf{z_k})) \in \mathbf{Rules}(\mathsf{pred})$ and hence of the form (cf. Lemma 14) $\mathsf{forget}_{\mathbf{y}}(\mathcal{M}_0 \bullet \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k)$, where $\mathcal{M}_0 \models_{\varPhi} x \to \mathbf{z_0}$ and, for $1 \leq i \leq k$, $\mathcal{M}_i \models_{\varPhi} \mathsf{pred}_i(\mathbf{z_i})$. By induction, all $\mathcal{M}_i$, $1 \leq i \leq k$, are rooted. Let $x_i \in \mathbf{z_i}$ be the root of $\mathcal{M}_i$. Since $\varPhi$ satisfies connectivity, it is guaranteed that $\{x_1, \ldots, x_k\} \subseteq \mathbf{z_0}$. Hence $\mathcal{M}$ is rooted in x. Because of connectivity, $x \notin \mathbf{y}$ and thus $x \in \mathbf{x}$.

4 Correspondence Between Separation-Logic Semantics and Heap-Graph Operations

We first note that *substituting* a predicate call with a model of that call is the same as *composing* the original heap graph (without the call) with a model of the call. We will need this auxiliary result in later proofs.

Lemma 8. *Let $\varPhi$ be an SID. Let $\mathcal{M}_1 \in \mathbf{AHG}$ and $\mathcal{M}_2 \in \mathbf{HG}$ such that $\mathcal{M}_1 \bullet \mathcal{M}_2$ is defined. Furthermore, assume $\mathcal{M}_1 \models_{\varPhi} \mathsf{pred}_1(\mathbf{x_1})$, $\mathsf{pred}_2(\mathbf{x_2}) \in \mathsf{calls}_{\mathcal{M}_1}$, $\mathcal{M}_2 \models_{\varPhi} \mathsf{pred}_2(\mathbf{x_2})$, and $\mathsf{FV}_{\mathcal{M}_2} = \mathbf{x_2}$. Then $((\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) \bullet \mathcal{M}_2) \models_{\varPhi} \mathsf{pred}_1(\mathbf{x_1})$.*

Proof. Let n be the number of rule applications that were applied to derive that $\mathcal{M}_2 \models_{\varPhi} \mathsf{pred}_2(\mathbf{x_2})$. We proceed by induction on n.

- If $n = 1$, then $\mathcal{M}_2 = \langle \emptyset, \mathbf{x_2}, \{\mathsf{pred}_2(\mathbf{x_2})\} \rangle$ for appropriate $\mathbf{z_2} \supseteq \mathbf{x_2}$. In that case
$$(\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) \bullet \mathcal{M}_2 = \mathcal{M}_1$$
and the result follows trivially.
- If $n > 1$, there exists a rule $(\mathsf{pred}_2 \Leftarrow \varphi) \in \mathbf{Rules}(\varPhi)$, $\varphi = \exists \mathbf{y} \ . \ \varSigma * \varGamma$, such that $\mathcal{M}_2 \models_{\varPhi} \varphi[\mathsf{fv}(\mathsf{pred}_2)/\mathbf{x_2}]$.
Let $\mathbf{z} \in \mathbf{Var}^*$ and $\varphi' := \varphi[\mathbf{y}/\mathbf{z}] = \varSigma' * \varGamma'$ such that $\mathcal{M}_i \models_{\varPhi} \varphi'$. Such a $\mathbf{z}$ must exist by definition of the semantics of quantifiers.
Let $m := |\varGamma'|$. Consider the decomposition $\mathcal{M}_2 = \mathcal{M}_2^0 \bullet \cdots \bullet \mathcal{M}_2^m$ such that $\mathcal{M}_i^0 \models_{\varPhi} \varSigma'$ and $\mathcal{M}_i^j$, $j \geq 1$, is such that $\mathcal{M}_i^j \models_{\varPhi} \varGamma'[j]$. Again, this decomposition must exist by the semantics. Observe that
$$(\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) \bullet \mathcal{M}_2 = (\mathcal{M}_1 - \mathsf{pred}_2(\mathbf{x_2})) \bullet \mathcal{M}_2^0 \bullet \cdots \bullet \mathcal{M}_2^m$$

By associativity of $\bullet$,

$$(\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) \bullet \mathcal{M}_2 = ((\mathcal{M}_1 - \mathsf{pred}_2(\mathbf{x_2})) \bullet \mathcal{M}_2^0) \bullet \mathcal{M}_2^1 \bullet \cdots \bullet \mathcal{M}_2^m$$

Hence also

$$((\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) + \Gamma') \bullet \mathcal{M}_2 = (((\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) + \Gamma') \bullet \mathcal{M}_2^0) \bullet \mathcal{M}_2^1 \bullet \cdots \bullet \mathcal{M}_2^m$$

Let $\mathcal{M}'_1 := ((\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) + \Gamma') \bullet \mathcal{M}_2^0$ Note that $\mathcal{M}'_1 \models_{\Phi} \mathsf{pred}_1(\mathbf{x_1})$, because it is like $\mathcal{M}_1$ except that $\mathsf{pred}_2(\mathbf{x_2})$ has been replaced with $\mathcal{M}_2^0 + \Gamma'$, which is a model of $\Sigma' * \Gamma'$.

Observe further that for each $\mathcal{M}_2^i$, $i > 0$, $\mathcal{M}_2^i \models_{\Phi} \Gamma'[i]$; and this heap graph was derived via strictly fewer rule applications than $\mathcal{M}_2$. Thus we can successively apply the induction hypothesis: First to $\mathcal{M}'_1$ and $\mathcal{M}_2^1$, then to $((\mathcal{M}'_1 - \{\Gamma'[1]\}) \bullet \mathcal{M}_2^1)$ and $\mathcal{M}_2^2$, etc., until we have processed all m calls, thus obtaining that $(\mathcal{M}_1 - \{\mathsf{pred}_2(\mathbf{x_2})\}) \bullet \mathcal{M}_2 \models_{\Phi} \mathsf{pred}_1(\mathbf{x_1})$.

We next study how the heap-graph operations and the separation-logic semantics are related.

Lemma 9. *Let $\mathcal{M} \in \mathbf{HG}$ with $x \notin \mathsf{FV}_{\mathcal{M}}$. Then $\mathcal{M} \models_{\Phi} \exists x \;.\; \varphi$ iff there exists a heap graph $\mathcal{M}'$ such that $\mathcal{M}' \models \varphi$ and $\mathcal{M} = \mathsf{forget}_x(\mathcal{M}')$.*

Proof. $\Rightarrow$ Let $\mathcal{M} \models_{\Phi} \exists x \;.\; \varphi$. By definition of the semantics of quantifiers, there exists a $y \in \mathbf{Var}$ such that $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \cup \{y\}, \mathsf{calls}_{\mathcal{M}} \rangle \models_{\Phi} \varphi[x/y]$. Since $x \notin \mathsf{FV}_{\mathcal{M}}$ we can rename y to x in $\mathcal{M}$ to obtain a heap graph $\mathcal{M}''$ that is isomorphic to $\mathcal{M}$ such that $\langle \mathsf{Ptr}_{\mathcal{M}''}, \mathsf{FV}_{\mathcal{M}''} \cup \{x\}, \mathsf{calls}_{\mathcal{M}''} \rangle \models_{\Phi} \varphi[x/x] = \varphi$. Set $\mathcal{M}' := \langle \mathsf{Ptr}_{\mathcal{M}''}, \mathsf{FV}_{\mathcal{M}''} \cup \{x\}, \mathsf{calls}_{\mathcal{M}''} \rangle$. $\mathcal{M} = \mathsf{forget}_x(\mathcal{M}')$.

$\Leftarrow$ Assume that there exists a heap graph $\mathcal{M}'$ such that $\mathcal{M}' \models \varphi$ and $\mathcal{M} = \mathsf{forget}_x(\mathcal{M}')$. By the definition of forget, $\mathcal{M} = \langle \mathsf{Ptr}_{\mathcal{M}'}, \mathsf{FV}_{\mathcal{M}'} \setminus \{x\}, \mathsf{calls}_{\mathcal{M}'} \rangle$ and hence $\mathcal{M}' = \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \cup \{x\}, \mathsf{calls}_{\mathcal{M}} \rangle$. In other words, there exists an $x \in \mathbf{Var}$ (namely x) such that $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \cup \{x\}, \mathsf{calls}_{\mathcal{M}} \rangle \models_{\Phi} \varphi[x/x]$. Thus $\mathcal{M} \models_{\Phi} \exists x \;.\; \varphi$.

Lemma 10. *If $\mathcal{M} \models_{\Phi} \varphi$ then $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) \models_{\Phi} \exists \mathbf{x} \;.\; \varphi$*

Proof. Assume that all variables in $\mathbf{x}$ occurs in φ. (For variables that do not occur in φ the result holds trivially.) Recall that $\mathsf{Ptr}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M})} = \mathsf{Ptr}_{\mathcal{M}}$ and $\mathsf{calls}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M})} = \mathsf{calls}_{\mathcal{M}}$. Since $\mathbf{x} \subseteq \mathsf{FV}_{\mathcal{M}}$, $\mathsf{FV}_{\mathsf{forget}_{\mathbf{x}}(\mathcal{M})} \cup \mathbf{x} = \mathsf{FV}_{\mathcal{M}}$. Thus

$$\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}_{\mathcal{M}} \rangle \models_{\Phi} \varphi$$

implies that

$$\left\langle \mathsf{Ptr}_{\mathsf{forget}_x(\mathcal{M})}, \mathsf{FV}_{\mathsf{forget}_x(\mathcal{M})} \cup \mathbf{x}, \mathsf{calls}_{\mathsf{forget}_{\mathbf{x}}\mathcal{M}} \right\rangle \models_{\Phi} \varphi[\mathbf{x}/\mathbf{x}].$$

By definition of the semantics of exists (substituting $\mathbf{x}$ for itself), we obtain that $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) \models_{\Phi} \exists \mathbf{x} \;.\; \varphi$.

Lemma 11. $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ *iff there exists a heap graph* $\mathcal{M}'$ *such that* $\mathcal{M}' \models \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ *and* $\mathcal{M} \cong \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}),\mathbf{x}}(\mathcal{M}')$.

Proof. $\Rightarrow$ Assume $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$. There are two cases.

- There exists $\mathbf{z} \supseteq \mathbf{x}$ such that $\mathcal{M} \cong \langle \emptyset, \mathbf{z}, \{\mathsf{pred}(\mathbf{x})\}\rangle$. In this case, let $\mathcal{M}' := \langle \emptyset, \mathsf{rename}_{\mathbf{x},\mathsf{fv}(\mathsf{pred})}(\mathbf{z}), \{\mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\}\rangle$. Then $\mathcal{M}' \models \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ and $\mathcal{M} \cong \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}),\mathbf{x}}(\mathcal{M}')$.
- There is a rule $(\mathsf{pred} \Leftarrow \psi) \in \mathbf{Rules}(\Phi)$ such that $\mathcal{M} \models_{\Phi} \psi[\mathsf{fv}(\mathsf{pred})/\mathbf{x}]$. Let $\mathcal{M}' := \mathsf{rename}_{\mathbf{x},\mathsf{fv}(\mathsf{pred})}(\mathcal{M})$. Observe that $\mathcal{M}' \models_{\Phi} \psi[\mathsf{fv}(\mathsf{pred})/\mathsf{fv}(\mathsf{pred})]$ and hence $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$. By Lemma 2, $\mathcal{M} \cong \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}),\mathbf{x}}(\mathcal{M}')$.

$\Leftarrow$ Analogously.

Lemma 12. *Let* $\mathcal{M} \in \mathbf{HG}$. *Let* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{Var}^*$ *be sequences without repetitions. If* $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{z})$ *then* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}))$.

Proof. Assume $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{z})$. According to the semantics of predicate calls, either

- There exists $\mathbf{z}' \supseteq \mathbf{z}$ such that $\mathcal{M} \cong \langle \emptyset, \mathbf{z}', \{\mathsf{pred}(\mathbf{z})\}\rangle$. In that case, $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \cong \langle \emptyset, \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}'), \{\mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}))\}\rangle$ and hence $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}))$.
- There is a rule $(\mathsf{pred} \Leftarrow \psi) \in \mathbf{Rules}(\Phi)$ such that $\mathcal{M} \models_{\Phi} \psi[\mathsf{fv}(\mathsf{pred})/\mathbf{z}]$. Then $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(M) \models_{\Phi} \psi[\mathsf{fv}(\mathsf{pred})/\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z})]$ and hence $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}))$.

Note that in particular, if $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ then $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathbf{y})$.

Lemma 13. *Let* $\mathcal{M} \in \mathbf{HG}$. *Let* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{Var}^*$ *be sequences without repetitions. If* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathbf{z})$ *then* $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathbf{z}))$.

Proof. Let $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) \models_{\Phi} \mathsf{pred}(\mathbf{z})$. By Lemma 12, $\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})) \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathbf{z}))$. By Lemma 2, $\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})) \cong \mathcal{M}$. Thus $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathbf{z}))$.

Lemma 14. *Let* $\varphi = \exists \mathbf{y} \,.\, (x_1 \to \mathbf{y}_1) * \cdots * (x_m \to \mathbf{y}_m) * \mathsf{pred}_1(\mathbf{z}_1) * \cdots * \mathsf{pred}_n(\mathbf{z}_n)$ *be a symbolic heap.* $\mathcal{M} \models_{\Phi} \varphi$ *iff there exist* $\mathcal{M}_1, \ldots, \mathcal{M}_{m+n}$ *such that* $\mathcal{M}_i \models_{\Phi} x \to \mathbf{y}_i$ *for* $1 \leq i \leq m$, $\mathcal{M}_{m+i} \models_{\Phi} \mathsf{pred}_{m+i}(\mathsf{fv}(\mathsf{pred}_{m+i}))$ *for* $1 \leq i \leq n$ *and*

$$\begin{aligned}\mathcal{M} = \mathsf{forget}_{\mathbf{y}}(\; &\mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_m \bullet \\ &\mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z}_1}(\mathcal{M}_{m+1}) \bullet \cdots \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_n),\mathbf{z}_n}(\mathcal{M}_{m+n}))\ .\end{aligned}$$

Proof. Follows immediately from Lemmas 9 and 11 together with the semantics of $*$ (which is defined via $\bullet$).

5 A Normal Form of $\mathbf{SL}_{\mathbf{btw}}$

We briefly justify the additional assumptions about the syntax of SIDs that we made at the end of Section 3 in the paper. We then discuss the treatment of pure formulas in HARRSH.

Assumption 1: All unfoldings of all predicates are satisfiable. By [5, Thm. 1 + 3], every SID Φ can be automatically transformed into an SID Φ' whose set of unfoldings is equal to the set of satisfiable unfoldings of Φ. We will call this property *all-satisfiability* later in this draft.

Assumption 2: There are no parameter repetitions. Exploiting the close relationship between SIDs and hyperedge replacement grammars (see e.g. [2,4]), the *well-formedness theorem* for hyperedge replacement grammars [3, Chapter 1, Thm. 4.6] implies that every SID can be converted into an equivalent SID that does not have parameter repetitions. Note that this implies that equalities between variables can always be eliminated from SIDs—if there are no parameter repetitions, equalities between parameters immediately lead to unsatisfiability.

Pure formulas and aliasing in Harrsh. While pure formulas can always be eliminated from an SID, such a translation leads to an increase in the number of predicates in the SID and the number of entailment queries to discharge. To avoid these detrimental effects, Harrsh implements an extension of the $\mathbf{Profiles}(\Phi)$ domain that directly supports pure formulas. In this extended domain, contexts are extended to keep track of guaranteed and forbidden aliasing effects in the underlying models. Harrsh is thus capable of handling $\mathrm{SL}_{\mathrm{btw}}$ SIDs with pure formulas and parameter repetitions *without* a conversion that eliminates these $\mathrm{SL}_{\mathrm{btw}}$ features. In our experimental evaluation of Harrsh, we made heavy use of these features; see Sections 9 and 10.

6 Missing Proofs of Section 4 (Profiles: An Abstraction for Concrete Heap Graphs)

6.1 Notation

We will use some additional notation in the correctness proofs for the profile abstraction:

- $\mathsf{contexts}_{\Phi}(\mathcal{M}) := \{\mathcal{C} \mid \mathcal{C} \text{ is a context of } \mathcal{M}\}$
- $\mathsf{decomps}_{\Phi}(\mathcal{M}) := \{\mathcal{E} \mid \mathcal{E} \text{ is a context decomposition of } \mathcal{M}\}$
- $\mathbf{Decomp}(\Phi) := \bigcup_{\mathcal{M} \in \mathbf{CHG}} \mathsf{decomps}_{\Phi}(\mathcal{M})$
- $\mathbf{Decomp}^{\mathbf{y}}(\Phi) := \{\mathcal{E} \in \mathbf{Decomp}(\Phi) \mid \mathsf{fv}(\mathcal{E}) \subseteq \mathbf{y}\}$.

Unless noted otherwise, we assume that all SIDs considered in this section satisfy all assumptions from Section 3 of the paper, i.e., we assume that they satisfy progress, connectivity, establishment, have only satisfiable unfoldings, and have pairwise different parameters.

6.2 Soundness of the Profile Abstraction

Lemma 15. *Let* $\mathcal{M}, \mathcal{M}' \in \mathbf{CHG}$ *such that* $\mathsf{profile}_{\Phi}(\mathcal{M}) = \mathsf{profile}_{\Phi}(\mathcal{M}')$. *Then, for all* $\mathsf{pred} \in \mathbf{Preds}(\Phi)$, *it holds that* $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ *iff* $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathbf{x})$.

Proof. While it would be possible to show this directly, the easiest way to obtain this result is as direct consequence of Theorem 6 (p. 19): $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ iff $\{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle\} \in \mathsf{profile}_{\Phi}(\mathcal{M})$ iff (because $\mathsf{profile}_{\Phi}(\mathcal{M}) = \mathsf{profile}_{\Phi}(\mathcal{M}')$) $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathbf{x})$.

6.3 Finiteness of the Profile Domain

Recall the notions of isomorphism for contexts, context decompositions and profiles from the main paper. In this document, we use $\cong$ to denote these notions of isomorphism as well as to denote heap-graph isomorphism. That is, we write $\mathcal{C}_1 \cong \mathcal{C}_2$, $\mathcal{E}_1 \cong \mathcal{E}_2$ and $\mathcal{P}_1 \cong \mathcal{P}_2$ to denote that the given contexts, context decompositions and profiles are isomorphic.

We need some auxiliary concepts to show the finiteness of our abstract domain. Let $\mathcal{C} = \langle \mathbf{y}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle$. We define $\mathsf{numcalls}(\mathcal{C}) := 1 + \mathsf{size}(\mathsf{calls})$. Analogously, for context decomposition $\mathcal{E}$, we define $\mathsf{numcalls}(\mathcal{E}) := \sum_{\mathcal{C}\in\mathcal{E}} \mathsf{numcalls}(\mathcal{C})$.

Recall that an SID is *all-satisfiable* if all of its unfoldings are satisfiable.

Lemma 16. *Let Φ be an SID that satisfies progress and connectivity and all-satisfiability. Let $\mathcal{E} \in \mathbf{Decomp}^{\mathbf{y}}(\Phi)$. Then* $\mathsf{numcalls}(\mathcal{E}) \leq 2\,|\mathbf{y}|$.

Proof. Let $\langle \mathbf{y}'', \mathsf{pred}'(\mathbf{z}'), \mathsf{calls}\rangle \in \mathcal{E}$. Let $\mathsf{pred}(\mathbf{z}) \in \mathsf{calls} \cup \{\mathsf{pred}'(\mathbf{z}')\}$. Since Φ satisfies progress and connectivity, we know that in each model of $\mathsf{pred}(\mathbf{z})$ there is a variable among $\mathbf{z}$ that is allocated.

Claim: This variable is among $\mathbf{y}''$. To see this, recall that if there is a heap graph $\mathcal{M}$ such that $\mathcal{E} = \mathsf{decomps}_{\Phi}(\mathcal{M})$, there exist $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that

1. $\mathcal{M} = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$
2. $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ where for each $1 \leq i \leq k$, $\mathcal{C}_i \in \mathsf{contexts}_{\Phi}(\mathcal{M}_i)$

Let $\mathcal{C}_i = \langle \mathbf{y}_i, \mathsf{pred}_i(\mathbf{z}_i), \mathsf{calls}\rangle$ for $\mathsf{calls} = \{\mathsf{pred}_{i,1}(\mathbf{z}_{i,1}), \ldots, \mathsf{pred}_{i,m_i}(\mathbf{z}_{i,m_i})\}$. By definition of contexts and because of all-satisfiability, there exist $\mathcal{M}_{i,1}, \ldots, \mathcal{M}_{i,m_i}$ such that $\mathcal{M}_{i,j} \models_{\Phi} \mathsf{pred}_{i,j}(\mathbf{z}_{i,j})$ and such that $\mathcal{M}_i \bullet \mathcal{M}_{i,1} \bullet \cdots \bullet \mathcal{M}_{i,m_i} \models_{\Phi} \mathsf{pred}_i(\mathbf{z}_i)$.

Because of progress, both $\mathcal{M}_i$ and all $\mathcal{M}_{i,j}$ are nonempty. By Lemma 7 on page 4, each $\mathcal{M}_i$ ($\mathcal{M}_{i,j}$) is rooted in a parameter of pred_i ($\mathsf{pred}_{i,j}$). In other words, one of the variables in $\mathbf{z}_i$ is allocated in $\mathcal{M}_i$ and one of the variables in $\mathbf{z}_{i,j}$ is allocated in each $\mathcal{M}_{i,j}$. By definition of contexts, none of these parameters is an auxiliary variable of $\mathbf{z}_i$ ($\mathbf{z}_{i,j}$). Thus, we conclude that at least $\mathsf{numcalls}(\mathcal{C}_i)$ parameters of $\mathbf{y}_i$ are allocated in all extensions of $\mathcal{M}_i$ to a model of of $\mathsf{pred}_i(\mathbf{z}_i)$.

Now observe that (1) $\mathbf{y}' = \bigcup_i(\mathbf{y_i})$ and (2) every variable among $\mathbf{y}'$ can of course be allocated at most once in the model $\mathcal{M} \bullet \mathcal{M}_{1,1} \bullet \mathcal{M}_{1,2} \bullet \cdots \bullet \mathcal{M}_{k,m_k}$ and thus occur in at most 2 predicate calls in $\mathcal{E}$. (I.e., there are at most one i and at most j such that $y \in \mathbf{y}'$ occurs in $\mathsf{pred}_i(\mathbf{z}_i)$ and in calls_j.) We thus conclude that the *total* number of predicate calls in all contexts in $\mathcal{E}$ is bounded from above by $2\,|\mathbf{y}|$; formally, $\mathsf{numcalls}(\mathcal{E}) \leq 2\,|\mathbf{y}|$.

Lemma 17. *Let $\mathbf{y} \in \mathbf{Var}^*$. The set $\mathbf{Decomp}^{\mathbf{y}}(\Phi)$ is finite (up to isomorphism for context decompositions).*

Proof. – Let $m := |\mathbf{y}|$
- By Lemma 16, we know that $\mathsf{numcalls}(\mathcal{E}) \leq 2m$.
- Let $k := |\mathbf{Preds}(\Phi)|$. There are k^{2m+1} ordered sequences of predicate identifiers of length at most $2m$.
- Let n be the maximum arity of a predicate in $\mathbf{Preds}(\Phi)$.
- Each predicate call thus receives at most n variables. Thus at most $n(m-1)$ auxiliary variables occur in a context decomposition. Since we are only interested in counting context decompositions up to isomorphism, we may assume w.l.o.g. that all auxiliary variables are drawn from a fixed set X of size $n(m-1)$.
- Consequently there are only finitely many ways to turn each sequence of predicate identifiers into a sequence of predicate calls.
- Finally, each context decomposition contains at most $|\mathbf{y}| = m$ contexts and there are only finitely many ways to split any given finite sequence of predicate calls into at most m subsequences.

Corollary 1. *For every SID Φ and variables $\mathbf{x} \in \mathbf{Var}^*$, the set of profiles*

$$\mathbf{Profiles}^{\mathbf{x}}(\Phi) = \{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \text{ concrete heap graph}, \mathsf{fv}(\mathsf{profile}_{\Phi}(\mathcal{M})) \subseteq \mathbf{x}\}$$

is finite up to profile isomorphism.

Proof. Note that $\mathbf{Profiles}^{\mathbf{x}}(\Phi)$ is a subset of the powerset of $\mathbf{Decomp}^{\mathbf{x}}(\Phi)$. The powerset of a finite set is finite. Thus $\mathbf{Profiles}^{\mathbf{x}}(\Phi)$ is finite.

6.4 Computing the Profile of Points-to Assertions

Lemma 18. *Profiles of single allocations, i.e.* $\mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{y})$*, are computable.*

Proof. Let $\mathcal{M} \cong x \rightarrowtail \mathbf{y}$. Observe that if $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathcal{M})$ then $|\mathcal{E}| = 1$. This holds because $\mathcal{M}$ consists of only a single pointer, so there are no decompositions of $\mathcal{M}$ into more than one nonempty part.

In order to compute $\mathsf{profile}_{\Phi}(\mathcal{M})$, we thus just need to compute all contexts of $\mathcal{M}$. To this end, let calls be such that $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}\rangle \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$. Again because $\mathcal{M}$ contains just one pointer, this implies that the SID Φ contains a rule of the form $\exists \mathbf{y}' x' \to \mathbf{z}' * \circledast \mathsf{calls}$, where we write $\circledast \mathsf{calls}$ to denote the separating conjunction of all calls in calls. If this were not the case, then $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}\rangle \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ could not hold.

To compute the profile of $x \rightarrowtail \mathbf{y}$, it is thus sufficient to loop over all rules of the SID and create a (single-element) context decomposition for each rule of whose local allocation $\mathcal{M}$ is a model. This is formalized in the function $\mathsf{pointstoProfile}$ presented as Algorithm 1.

6.5 Abstraction of Rename

Lemma 19. $\mathcal{M} = \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \emptyset\rangle$ *be a concrete heap graph,* $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M})$ *and* $\mathbf{x}, \mathbf{y} \in \mathbf{Var}^*$ *such that* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$ *is defined. Then* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}) \in \mathsf{contexts}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.

Algorithm 1: The algorithm $\mathsf{pointstoProfile}_{\Phi}(x \rightarrowtail \mathbf{y})$ that computes the profile of the model $x \rightarrowtail \mathbf{y}$ (cf. Lemma 18).

```
$\mathcal{P} := \emptyset$;
$\mathcal{M} := x \rightarrowtail \mathbf{y}$;
for $(\mathsf{pred} \Leftarrow \exists \mathbf{y}' \,.\, x' \to \mathbf{z}' * \mathsf{pred}_1(\mathbf{z_1}) * \cdots * \mathsf{pred}_k(\mathbf{z_k})) \in \mathbf{Rules}(\Phi)$ do
    if $\mathcal{M} \models \exists \mathbf{y}' \,.\, x' \to \mathbf{z}'$ then
        $\mathcal{C} := \langle \{x\} \cup \mathbf{y}, \mathsf{pred}(\mathsf{fv}(\mathsf{pred})), \{\mathsf{pred}_1(\mathbf{z_1}), \ldots, \mathsf{pred}_k(\mathbf{z_k})\} \rangle$;
        $\mathcal{P} := \mathcal{P} \cup \{\{\mathcal{C}\}\}$;
return $\mathcal{P}$;
```

Proof. Let pred, $\mathbf{z}$, and calls be such that $\mathcal{C} = \langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{z}), \mathsf{calls} \rangle$. Since $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M})$, for the model $\mathcal{M}' = \langle \mathsf{Ptr}_{\mathcal{M}}, \mathbf{z}, \mathsf{calls} \rangle$ it holds that $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathbf{z})$. Let $\mathcal{M}'_{ren} := \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}')$ and $\mathcal{M}_{ren} := \left\langle \mathsf{Ptr}_{\mathcal{M}'_{ren}}, \mathsf{FV}_{\mathcal{M}'_{ren}}, \emptyset \right\rangle$. By Lemma 12, $\mathcal{M}'_{ren} \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z}))$. By definition of contexts, it thus holds that $\left\langle \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathsf{FV}_{\mathcal{M}}), \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z})), \mathsf{calls}_{\mathcal{M}'_{ren}} \right\rangle \in \mathsf{contexts}_{\Phi}(\mathcal{M}_{ren})$. Now observe that

- $\mathcal{M}_{ren} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$
- $\left\langle \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathsf{FV}_{\mathcal{M}}), \mathsf{pred}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathbf{z})), \mathsf{calls}_{\mathcal{M}'_{ren}} \right\rangle = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C})$

Thus $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}) \in \mathsf{contexts}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.

Lemma 20. *Let* $\mathcal{M} \in \mathbf{CHG}$ *and* $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$. *Then* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}) \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.

Proof. Let $\mathcal{E}' := \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E})$. Since $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$, there exist $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathcal{M} = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$ and $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ for appropriately chosen contexts $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1), \ldots, \mathcal{C}_k \in \mathsf{contexts}_{\Phi}(\mathcal{M}_k)$. By definition of $\mathsf{rename}_{\mathbf{x},\mathbf{y}}$, it holds that

$$\mathcal{E}' = \{\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}_1), \ldots, \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}_k)\}\,.$$

By Lemma 19, $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}_i) \in \mathsf{contexts}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_i))$ for each $1 \leq i \leq k$.

We thus obtain that $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_k))$ and, since (by Lemma 3) $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}_k) = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$, $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.

Lemma 21. *Let* $\mathcal{M} = \langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \emptyset \rangle \in \mathbf{CHG}$ *and let* $\mathbf{x}, \mathbf{y} \in \mathbf{Var}^*$ *such that* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$ *is defined. Let* $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$. *Then there exists a context* $\mathcal{C}' \in \mathsf{contexts}_{\Phi}(\mathcal{M})$ *such that* $\mathcal{C} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}')$.

Proof. Write $\mathcal{M}_{ren} := \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$. Let pred, $\mathbf{z}$, and calls be such that $\mathcal{C} = \langle \mathsf{FV}_{\mathcal{M}_{ren}}, \mathsf{pred}(\mathbf{z}), \mathsf{calls} \rangle$. Since $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M}_{ren})$, for the model

$$\mathcal{M}' = \langle \mathsf{Ptr}_{\mathcal{M}_{ren}}, \mathbf{z}, \mathsf{calls} \rangle$$

it holds that $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathbf{z})$. We now consider the heap graphs obtained by reversing the renaming, i.e., we let $\mathcal{M}'_{rev} := \mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathcal{M}')$ and $\mathcal{M}_{rev} := \langle \mathsf{Ptr}_{\mathcal{M}'_{rev}}, \mathsf{FV}_{\mathcal{M}'_{rev}}, \emptyset \rangle$. By Lemma 13, $\mathcal{M}'_{rev} \models_{\Phi} \mathsf{pred}(\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathbf{z}))$. Observe that $\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathsf{FV}_{\mathcal{M}_{ren}}) = \mathsf{FV}_{\mathcal{M}_{rev}}$ and thus by definition of contexts, it follows for $\mathcal{C}' := \left\langle \mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathsf{FV}_{\mathcal{M}_{ren}}), \mathsf{pred}(\mathsf{rename}_{\mathbf{y},\mathbf{x}}(\mathbf{z})), \mathsf{calls}_{\mathcal{M}'_{rev}} \right\rangle$ that $\mathcal{C}' \in \mathsf{contexts}_{\Phi}(\mathcal{M}_{rev})$. As $\mathcal{M}_{rev} = \mathcal{M}$ (by Lemma 2) and $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}') = \mathcal{C}$, it follows that $\mathcal{C}' \in \mathsf{contexts}_{\Phi}(\mathcal{M})$.

Lemma 22. *Let* $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$. *Then there exists a context decomposition* $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ *such that* $\mathcal{E} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}')$.

Proof. Since $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$, there exist $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}) = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$ and $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ for some appropriately chosen contexts $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1), \ldots, \mathcal{C}_k \in \mathsf{contexts}_{\Phi}(\mathcal{M}_k)$.

By Lemma 4, there exist $\mathcal{M}'_1, \ldots, \mathcal{M}'_k$ such that $\mathcal{M} = \mathcal{M}'_1 \bullet \cdots \bullet \mathcal{M}'_k$ and such that for $1 \leq i \leq k$, $\mathcal{M}_i = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}'_i)$. By Lemma 21, there exist (for all $1 \leq i \leq k$) contexts $\mathcal{C}'_i \in \mathsf{contexts}_{\Phi}(\mathcal{M}'_i)$ such that $\mathcal{C}_i = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{C}'_i)$. Let $\mathcal{E}' := \{\mathcal{C}'_1, \ldots, \mathcal{C}'_k\}$. By construction, $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ and $\mathcal{E} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}')$.

Theorem 1. *Let* $\mathcal{M} \in \mathbf{CHG}$ *and let* $\mathbf{x}, \mathbf{y} \in \mathbf{Var}^*$ *be such that* $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$ *is defined. Then* $\overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M})) = \mathsf{profile}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.

Proof. We show that $\mathcal{E} \in \overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$ iff $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$. We prove each inclusion separately.

- $\subseteq$ Let $\mathcal{E} \in \overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$. By definition of $\overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}$, there exists a context decomposition $\mathcal{E}' \in \mathsf{profile}_{\Phi}(\mathcal{M})$ (and thus $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$) such that $\mathcal{E} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}')$. By Lemma 20, it holds that $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}') \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$, i.e., $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$ and thus $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$.
- $\supseteq$ Let $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$ and thus $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M}))$. By Lemma 22, there exists a context decomposition $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ (and hence $\mathcal{E}' \in \mathsf{profile}_{\Phi}(\mathcal{M})$) such that $\mathcal{E} = \mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{E}')$. Hence, by definition of $\overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}$, $\mathcal{E} \in \overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$

6.6 Abstraction of Forget

For convenience, we introduce shorthand notation for removing variables from contexts and context decompositions. Let $\mathcal{C} = \langle \mathbf{x}, \mathsf{pred}(\mathbf{z}), \mathsf{calls} \rangle$. We denote by $\mathcal{C} - \mathbf{y}$ the context $\langle \mathbf{x} \setminus \mathbf{y}, \mathsf{pred}(\mathbf{z}), \mathsf{calls} \rangle$. Similarly, for a context decomposition $\mathcal{E}$, $\mathcal{E} - \mathbf{y}$ denotes the context decomposition $\{\mathcal{C} - \mathbf{y} \mid \mathcal{C} \in \mathcal{E}\}$. Analogously, we denote by $\mathcal{C} + \mathbf{y}$ the context $\langle \mathbf{x} \cup \mathbf{y}, \mathsf{pred}(\mathbf{z}), \mathsf{calls} \rangle$ and by $\mathcal{E} + \mathbf{y}$ the context decomposition $\{\mathcal{C} - \mathbf{y} \mid \mathcal{C} \in \mathcal{E}\}$.

Lemma 23. *Let* $\mathcal{M} \in \mathbf{CHG}$ *and* $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M})$ *such that* $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{C}) = \emptyset$. *Then* $(\mathcal{C} - \mathbf{x}) \in \mathsf{contexts}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$.

Proof. Let pred, $\mathbf{z}$, and calls be such that $\mathcal{C} = \langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle$. By definition, $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathbf{z}, \mathsf{calls}\rangle \models_{\Phi} \mathsf{pred}(\mathbf{z})$. Since $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{C}) = \emptyset$, in particular $\mathbf{x} \cap \mathbf{z} = \emptyset$ and hence $\mathbf{z} \setminus \mathbf{x} = \mathbf{z}$. Consequently, $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathbf{z} \setminus \mathbf{x}, \mathsf{calls}\rangle \models_{\Phi} \mathsf{pred}(\mathbf{z})$. Thus $\langle \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle \in \mathsf{contexts}_{\Phi}(\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \emptyset\rangle)$.

Observe that (1) $\langle \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle = \mathcal{C} - \mathbf{x}$ and (2) $\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \emptyset\rangle = \mathsf{forget}_{\mathbf{x}}(\mathcal{M})$. The result follows.

Lemma 24. *Let Φ be established. Let $\mathcal{M}, \mathcal{M}_1, \mathcal{M}_2 \in \mathbf{CHG}$ such that $\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}_2$. Let $x \in \mathsf{FV}_{\mathcal{M}} \cap \mathrm{elems}(\mathsf{Ptr}_{\mathcal{M}_1}) \cap \mathrm{elems}(\mathsf{Ptr}_{\mathcal{M}_2})$. Finally, let $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1)$ and $\mathcal{C}_2 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_2)$ be such that $\{\mathcal{C}_1, \mathcal{C}_2\} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$. Then $x \in \mathsf{usedvs}(\mathcal{C}_1) \cup \mathsf{usedvs}(\mathcal{C}_2)$.*

Proof. Let $\mathcal{C}_1 = \langle \mathbf{x}_1, \mathsf{pred}_1(\mathbf{z}_1), \mathsf{calls}_1\rangle$ and $\mathcal{C}_2 = \langle \mathbf{x}_2, \mathsf{pred}_2(\mathbf{z}_2), \mathsf{calls}_2\rangle$, and thus $(\mathcal{M}_1 + \mathsf{calls}_1) \models_{\Phi} \mathsf{pred}_1(\mathbf{z}_1)$ and $(\mathcal{M}_2 + \mathsf{calls}_2) \models_{\Phi} \mathsf{pred}_2(\mathbf{z}_2)$. There must be at least one $i \in \{1,2\}$ such that $x \in (\mathrm{img}(\mathsf{Ptr}_{\mathcal{M}_i}) \setminus \mathrm{dom}(\mathsf{Ptr}_{\mathcal{M}_i}))$—otherwise $\mathcal{M}_1 \bullet \mathcal{M}_2$ would be undefined. Because Φ is established and $(\mathcal{M}_i + \mathsf{calls}_i) \models_{\Phi} \mathsf{pred}_i(\mathbf{z}_i)$, x must be allocated in a model of one of the calls $\mathsf{pred}(\mathbf{z}) \in \mathsf{calls}_i$. For this to be possible, x must occur in $\mathbf{z}$. Thus $x \in \mathsf{usedvs}(\mathcal{C}_i)$ and hence $x \in \mathsf{usedvs}(\mathcal{C}_1) \cup \mathsf{usedvs}(\mathcal{C}_2)$.

Lemma 25. *Let $\mathcal{M} \in \mathbf{CHG}$ and $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ such that $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{E}) = \emptyset$. Then $(\mathcal{E} - \mathbf{x}) \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$.*

Proof. Let $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ such that $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{E}) = \emptyset$. Since $\mathcal{E}$ is a context decomposition of $\mathcal{M}$, there exist $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathcal{M} = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$ and $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ for some $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1), \ldots, \mathcal{C}_k \in \mathsf{contexts}_{\Phi}(\mathcal{M}_k)$.

Since $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{E}) = \emptyset$, it trivially holds for all $1 \leq i \leq k$ that $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{C}_i) = \emptyset$. We thus apply Lemma 23 to each context in $\mathcal{E}$ to obtain that for $1 \leq i \leq k$, $(\mathcal{C}_i - \mathbf{x}) \in \mathsf{contexts}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}_i))$. Thus, $(\mathcal{E} - \mathbf{x}) \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k))$.

It remains to be shown that $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) = \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k)$. Since $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{E}) = \emptyset$, it follows (by a straightforward generalization of the contrapositive of Lemma 24 to decompositions with more than two elements) that every variable in $\mathbf{x}$ is in $\mathrm{elems}(\mathsf{Ptr}_{\mathcal{M}_i})$ for at most one $1 \leq i \leq k$. We can thus apply Lemma 5 and obtain that $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) = \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{forget}_{\mathbf{x}}(\mathcal{M}_k)$. By definition of context decompositions, it follows that $(\mathcal{E} - \mathbf{x}) \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$.

Lemma 26. *Let $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$. Then $(\mathcal{C} + \mathbf{x}) \in \mathsf{contexts}_{\Phi}(\mathcal{M})$.*

Proof. Let $\mathcal{C} = \langle \mathsf{FV}_{\mathcal{M}'}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle$. Write $\mathcal{M}' := \mathsf{forget}_{\mathbf{x}}(\mathcal{M})$.

Use that $\mathsf{FV}_{\mathcal{M}'} = \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}$ to derive that

$$\langle \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle \in \mathsf{contexts}_{\Phi}(\langle \mathsf{Ptr}_{\mathcal{M}'}, \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \emptyset\rangle).$$

Now observe that $\mathsf{Ptr}_{\mathcal{M}} = \mathsf{Ptr}_{\mathcal{M}'}$ and consequently $\langle \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle \in \mathsf{contexts}_{\Phi}(\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}} \setminus \mathbf{x}, \emptyset\rangle)$. Assume w.l.o.g. that $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{C}) = \emptyset$. (You can always replace $\mathcal{C}$ by an isomorphic context that has this property.) By definition of contexts, adding $\mathbf{x}$ to the free variables of both the heap graph and the context yields a context of the extended heap graph. Therefore also $\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle \in \mathsf{contexts}_{\Phi}(\langle \mathsf{Ptr}_{\mathcal{M}}, \mathsf{FV}_{\mathcal{M}}, \mathsf{calls}\rangle)$. The result follows because $\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle = \mathcal{C} + \mathbf{x}$.

Lemma 27. *Let* $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$*. Then there exists a context decomposition* $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ *such that* $\mathcal{E} = \mathsf{forget}_{\mathbf{x}}(\mathcal{E}')$*.*

Proof. Since $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$, there exist $\mathcal{M}_1, \ldots, \mathcal{M}_k$ such that $\mathsf{forget}_{\mathbf{x}}(\mathcal{M}) = \mathcal{M}_1 \bullet \cdots \bullet \mathcal{M}_k$ and $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ for some appropriate contexts $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1), \ldots, \mathcal{C}_k \in \mathsf{contexts}_{\Phi}(\mathcal{M}_k)$.

By Lemma 6, there exist $\mathcal{M}'_1, \ldots, \mathcal{M}'_k$ such that $\mathcal{M} = \mathcal{M}'_1 \bullet \cdots \bullet \mathcal{M}'_k$ and such that for $1 \leq i \leq k$, $\mathcal{M}_i = \mathsf{forget}_{\mathbf{x}}(\mathcal{M}'_i)$. By Lemma 26, there exist for all $1 \leq i \leq k$ contexts $\mathcal{C}'_i \in \mathsf{contexts}_{\Phi}(\mathcal{M}'_i)$ such that $\mathcal{C}_i = \mathsf{forget}_{\mathbf{x}}(\mathcal{C}'_i)$. (Specifically, this holds for $\mathcal{C}'_i = \mathcal{C}_i + \mathbf{x}$). Let $\mathcal{E}' := \{\mathcal{C}'_1, \ldots, \mathcal{C}'_k\}$. By construction, $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ and $\mathcal{E} = \mathsf{forget}_{\mathbf{x}}(\mathcal{E}')$.

Theorem 2. *Let* $\mathcal{M} \in \mathbf{CHG}$ *and* $\mathbf{x} \in \mathbf{Var}^*$ *such that* $\mathsf{forget}_{\mathbf{x}}(\mathcal{M})$ *is defined. Then* $\overline{\mathsf{forget}}_{\mathbf{x}}(\mathsf{profile}_{\Phi}(\mathcal{M})) = \mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$*.*

Proof. We show that $\mathcal{E} \in \overline{\mathsf{forget}}_{\mathbf{x}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$ iff $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$. We prove each inclusion separately.

- $\subseteq$ Let $\mathcal{E} \in \overline{\mathsf{forget}}_{\mathbf{x}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$ and thus $\mathcal{E} \in \mathsf{forget}_{\mathbf{x}}(\mathsf{decomps}_{\Phi}(\mathcal{M}))$. By definition of $\overline{\mathsf{forget}}_{\mathbf{x}}$, there exists an $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ such that $\mathbf{x} \cap \mathsf{usedvs}(\mathcal{E}) = \emptyset$ and $\mathcal{E} = \mathcal{E}' - \mathbf{x}$. By Lemma 25 we obtain $(\mathcal{E}' - \mathbf{x}) \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$, hence $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$, and hence $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$.
- $\supseteq$ Let $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$ and hence $\mathcal{E} \in \mathsf{decomps}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M}))$. By Lemma 27, there exists $\mathcal{E}' \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ such that $\mathcal{E} = \mathsf{forget}_{\mathbf{x}}(\mathcal{E}')$. By definition of $\overline{\mathsf{forget}}_{\mathbf{x}}$, we obtain that $\mathcal{E} \in \overline{\mathsf{forget}}_{\mathbf{x}}(\mathsf{profile}_{\Phi}(\mathcal{M}))$

6.7 Abstraction of Composition

Lemma 28. *Let* $\mathcal{M}_1$*,* $\mathcal{M}_2$ *be concrete heap graphs,* $\mathcal{C}_1 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1)$ *and* $\mathcal{C}_2 \in \mathsf{contexts}_{\Phi}(\mathcal{M}_2)$*. Finally, let* $\mathcal{C}$ *be such that* $\{\mathcal{C}_1, \mathcal{C}_2\} \triangleright \{\mathcal{C}\}$*. Then* $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1 \bullet \mathcal{M}_2)$*.*

Proof. Let $\mathcal{C}_1 = \langle \mathsf{FV}_{\mathcal{M}_1}, \mathsf{pred}_1(\mathbf{z}_1), \mathsf{calls}_1\rangle$ and $\mathcal{C}_2 = \langle \mathsf{FV}_{\mathcal{M}_2}, \mathsf{pred}_2(\mathbf{z}_2), \mathsf{calls}_2\rangle$. Assume w.l.o.g. that $\mathcal{C} = \mathcal{C}_2[\mathcal{C}_1]$. (Otherwise swap $\mathcal{C}_1$ and $\mathcal{C}_2$ and/or replace them with isomorphic contexts such that $\mathcal{C}_2[\mathcal{C}_1]$ is defined. This must be possible because $\{\mathcal{C}_1, \mathcal{C}_2\} \triangleright \{\mathcal{C}\}$.)

By definition of contexts, it holds for $\mathcal{M}'_1 = \langle \mathsf{Ptr}_{\mathcal{M}_1}, \mathsf{FV}_{\mathcal{M}_1}, \mathsf{calls}_1\rangle$ and $\mathcal{M}'_2 = \langle \mathsf{Ptr}_{\mathcal{M}_2}, \mathsf{FV}_{\mathcal{M}_2}, \mathsf{calls}_2\rangle$ that $\mathcal{M}'_1 \models_{\Phi} \mathsf{pred}_1(\mathbf{z_1})$ and $\mathcal{M}'_2 \models_{\Phi} \mathsf{pred}_2(\mathbf{z_2})$. By Lemma 8, $((\mathcal{M}'_2 - \{\mathsf{pred}_1(\mathbf{z_1})\}) \bullet \mathcal{M}'_1) \models_{\Phi} \mathsf{pred}_2(\mathbf{z_2})$.

Observe that by setting the calls of $((\mathcal{M}'_2 - \{\mathsf{pred}_1(\mathbf{z_1})\}) \bullet \mathcal{M}'_1)$ to $\emptyset$ we obtain the concrete heap graph $\mathcal{M}_1 \bullet \mathcal{M}_2$. Thus, by definition of contexts,

$$\langle \mathsf{FV}_{\mathcal{M}_1} \cup \mathsf{FV}_{\mathcal{M}_2}, \mathsf{pred}_2(\mathbf{z_2}), (\mathsf{calls}_2 \setminus \{\mathsf{pred}_1(\mathbf{z}_1)\}) \cup \mathsf{calls}_1\rangle \in \mathsf{contexts}_{\Phi}(\mathcal{M}_1 \bullet \mathcal{M}_2).$$

Now observe that this context is equal to $\mathcal{C}_2[\mathcal{C}_1] = \mathcal{C}$. Thus $\mathcal{C} \in \mathsf{contexts}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.

Lemma 29. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be concrete heap graphs such that $\mathcal{M}_1 \bullet \mathcal{M}_2$ is defined. Let $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$ and $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$. Finally, let $\mathcal{E}$ be such that $\mathcal{E}_1 \cup \mathcal{E}_2 \rhd^* \mathcal{E}$. Then $\mathcal{E} \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.*

Proof. We proceed by induction on the number n of $\rhd$ steps used to derive $\mathcal{E}$ from $\mathcal{E}_1 \cup \mathcal{E}_2$.

- Assume $n = 0$. In that case, $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2$. Assume $\mathcal{E}_1 = \{\mathcal{C}_{1,1}, \dots, \mathcal{C}_{1,k_1}\}$ and $\mathcal{E}_2 = \{\mathcal{C}_{2,1}, \dots, \mathcal{C}_{2,k_2}\}$. Let $\mathcal{M}_{1,1}, \dots, \mathcal{M}_{1,k_1}$ and $\mathcal{M}_{2,1}, \dots, \mathcal{M}_{2,k_2}$ be such that (1) for each i, j, $\mathcal{C}_{i,j} \in \mathsf{contexts}_\Phi(\mathcal{M}_{i,j})$, (2) $\mathcal{M}_1 = \mathcal{M}_{1,1} \bullet \cdots \bullet \mathcal{M}_{1,k_1}$ and (3) $\mathcal{M}_2 = \mathcal{M}_{2,1} \bullet \mathcal{M}_{2,k_2}$. Such decompositions of $\mathcal{M}_1$ and $\mathcal{M}_2$ must exist because $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$ and $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$. It thus follows that $\{\mathcal{C}_{1,1}, \dots, \mathcal{C}_{1,k_1}\} \cup \{\mathcal{C}_{2,1}, \dots, \mathcal{C}_{2,k_2}\} = \mathcal{E}_1 \cup \mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.
- Assume $n > 0$. Let $\mathcal{E}'$ be such that $\mathcal{E}_1 \cup \mathcal{E}_2 \rhd^* \mathcal{E}' \rhd \mathcal{E}$. By induction, $\mathcal{E}' \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.
  Let $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{E}'$ be such that $\mathcal{E} = (\mathcal{E}' \setminus \{\mathcal{C}_1, \mathcal{C}_2\}) \cup \{\mathcal{C}_2[\mathcal{C}_1]\}$. (Where we assume w.l.o.g. that the contexts in $\mathcal{E}'$ have already been replaced by isomorphic contexts such that $\mathcal{C}_2[\mathcal{C}_1]$ is defined.) Now let $\mathcal{M}'_1, \mathcal{M}'_2, \mathcal{M}'_3$ be such that (1) $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}'_1 \bullet \mathcal{M}'_2 \bullet \mathcal{M}'_3$, (2) $\mathcal{C}_1 \in \mathsf{contexts}_\Phi(\mathcal{M}'_1)$, and (3) $\mathcal{C}_2 \in \mathsf{contexts}_\Phi(\mathcal{M}'_2)$. Observe that $\{\mathcal{C}_1, \mathcal{C}_2\} \rhd \{\mathcal{C}_2[\mathcal{C}_1]\}$ and thus by Lemma 28, $\mathcal{C}_2[\mathcal{C}_1] \in \mathsf{contexts}_\Phi(\mathcal{M}'_1 \bullet \mathcal{M}'_2)$. Note further that $\mathcal{E}' \setminus \{\mathcal{C}_1, \mathcal{C}_2\} \in \mathsf{decomps}_\Phi(\mathcal{M}'_3)$. Since $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{M}'_1 \bullet \mathcal{M}'_2 \bullet \mathcal{M}'_3$, it follows that $\mathcal{E} = (\mathcal{E}' \setminus \{\mathcal{C}_1, \mathcal{C}_2\}) \cup \{\mathcal{C}_2[\mathcal{C}_1]\} \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.

Lemma 30. *Let $\mathcal{M}, \mathcal{M}_1, \mathcal{M}_2 \in$ **CHG** such that $\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}_2$ and let $\mathcal{C} \in \mathsf{contexts}_\Phi(\mathcal{M})$. Then there exist $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$ and $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$ such that $\mathcal{E}_1 \cup \mathcal{E}_2 \rhd^* \{\mathcal{C}\}$.*

Proof. Let $\mathcal{C} = \langle \mathsf{FV}_\mathcal{M}, \mathsf{pred}(\mathbf{z}), \mathsf{calls}\rangle$. Let $\mathcal{M}_{1,1}, \dots, \mathcal{M}_{1,m}$ and $\mathcal{M}_{2,1}, \dots, \mathcal{M}_{2,n}$ be such that $\mathcal{M}_1 = \mathcal{M}_{1,1} \bullet \cdots \bullet \mathcal{M}_{1,m}$, $\mathcal{M}_2 = \mathcal{M}_{2,1} \bullet \cdots \bullet \mathcal{M}_{2,n}$ and such that there are no decompositions of $\mathcal{M}_1$ into more than m parts and no decompositions of $\mathcal{M}_2$ into more than n parts.

For each $\mathcal{M}_{i,j}$ there exists a context $\mathcal{C}_{i,j}$ such that $\{\mathcal{C}_{i,j}\} \in \mathsf{decomps}_\Phi(\mathcal{M}_{i,j})$. Such a context must exist as (1) $\mathsf{decomps}_\Phi(\mathcal{M}_{i,j})$ is nonempty because $\mathcal{M}_{i,j} \subseteq \mathcal{M}$ and $\mathsf{decomps}_\Phi(\mathcal{M})$ is nonempty, and (2) $\mathsf{decomps}_\Phi(\mathcal{M}_{i,j})$ can only contain single-element sets because by assumption, it is impossible to decompose $\mathcal{M}_{i,j}$ into multiple parts.

We thus set $\mathcal{E}_1 := \{\mathcal{C}_{1,i} \mid 1 \leq i \leq m\}$ and $\mathcal{E}_2 := \{\mathcal{C}_{2,i} \mid 1 \leq i \leq n\}$. Observe that $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$ and $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$.

Now consider the sequence of rules of the semantics used to derive that the model relationship $\langle \mathsf{Ptr}_\mathcal{M}, \mathbf{z}, \mathsf{calls}\rangle \models_\Phi \mathsf{pred}(\mathbf{z})$ holds. Each $\mathcal{M}_{i,j}$ was derived in a subsequence of this derivation. In other words, the derivation sequence can be decomposed into derivation sequences for all the $\mathcal{M}_{i,j}$.

For each $\mathcal{C}_{i,j} = \left\langle \mathbf{x}_{i,j}, \mathsf{pred}_{i,j}(\mathbf{z}_{i,j}), \mathsf{calls}_{i,j} \right\rangle$, the predicate call $\mathsf{pred}_{i,j}(\mathbf{z}_{i,j})$ is the call at the start of the sub-derivation for $\mathcal{M}_{i,j}$ and the calls $\mathsf{calls}_{i,j}$ correspond to the predicate calls that remain at the end of the sub-derivation of $\mathcal{M}_{i,j}$. For this reason, the context substitution $\mathcal{C}_{i,j}[\mathcal{C}_{k,l}]$ is defined iff $\mathcal{M}_{k,l}$ corresponds to the subsequence that begins in one of the calls in $\mathsf{calls}_{i,j}$.

By successively assembling the derivation sequence for $\mathcal{M}$ out of the derivation sequences for the $\mathcal{M}_{i,j}$ and every time as we add a subgraph $\mathcal{M}_{k,l}$ applying the context substitution of $\mathcal{C}_{k,l}$ into the context constructed so far, we will eventually have processed the entire derivation sequence of $\mathcal{M}$ and, in parallel, will have obtained a sequence of $\triangleright$ steps deriving $\mathcal{E}_1 \cup \mathcal{E}_2 \triangleright^* \{\mathcal{C}\}$.

Note that in Lemma 30, $\mathcal{E}_1$ and $\mathcal{E}_2$ may but need not be single-element context decompositions.

Lemma 31. *Let* $\mathcal{M} = \mathcal{M}_1 \bullet \mathcal{M}_2$ *and* $\mathcal{E} \in \mathsf{decomps}_\Phi(\mathcal{M})$. *Then there exist* $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$ *and* $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$ *such that* $\mathcal{E}_1 \cup \mathcal{E}_2 \triangleright^* \mathcal{E}$.

Proof. Assume $\mathcal{E} = \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$. By definition of context decomposition, there exist $\mathcal{N}_1, \dots, \mathcal{N}_k$ such that (1) $\mathcal{M}_1 \bullet \mathcal{M}_2 = \mathcal{N}_1 \bullet \cdots \bullet \mathcal{N}_k$ and (2) $\mathcal{C}_i \in \mathsf{contexts}_\Phi(\mathcal{N}_i)$ for $1 \leq i \leq k$.

We define sets $\mathcal{E}_{1,1}, \dots, \mathcal{E}_{1,k}$ and $\mathcal{E}_{2,1}, \dots, \mathcal{E}_{2,k}$ as follows.

1. If $\mathcal{N}_i \subseteq \mathcal{M}_1$, $\mathcal{E}_{i,1} := \{\mathcal{C}_i\}$ and $\mathcal{E}_{i,2} = \emptyset$.
2. If $\mathcal{N}_i \subseteq \mathcal{M}_2$, $\mathcal{E}_{i,2} := \{\mathcal{C}_i\}$ and $\mathcal{E}_{i,1} = \emptyset$.
3. If, $\mathcal{N}_i \not\subseteq \mathcal{M}_1$ and $\mathcal{N}_i \not\subseteq \mathcal{M}_2$ we let $\mathcal{N}_i^1$, $\mathcal{N}_i^2$ be such that $\mathcal{N}_i = \mathcal{N}_i^1 \bullet \mathcal{N}_i^2$, $\mathcal{N}_i^1 \subseteq \mathcal{M}_1$, and $\mathcal{N}_i^2 \subseteq \mathcal{M}_2$, i.e., split $\mathcal{N}_i$ into its $\mathcal{M}_1$-part and its $\mathcal{M}_2$-part. We choose $\mathcal{E}_i^1$ and $\mathcal{E}_i^2$ be such that $\mathcal{E}_i^1 \cup \mathcal{E}_i^2 \triangleright^* \{\mathcal{C}_i\}$, which is possible by Lemma 30 (applied to $\mathcal{N}_i, \mathcal{N}_i^1, \mathcal{N}_i^2$ and $\mathcal{C}_i$).

For $\mathcal{E}_1 = \mathcal{E}_{1,1} \cup \cdots \cup \mathcal{E}_{1,k}$ and $\mathcal{E}_2 = \mathcal{E}_{2,1} \cup \cdots \cup \mathcal{E}_{2,k}$ it now holds by construction that $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$, $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$ and $\mathcal{E}_1 \cup \mathcal{E}_2 \triangleright^* \mathcal{E}$.

Theorem 3. *Let* $\mathcal{M}_1$ *and* $\mathcal{M}_2$ *be concrete heap graphs such that* $\mathcal{M}_1 \bullet \mathcal{M}_2$ *be defined. Then* $\mathsf{profile}_\Phi(\mathcal{M}_1) \mathbin{\overline{\bullet}} \mathsf{profile}_\Phi(\mathcal{M}_2) = \mathsf{profile}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.

Proof. We show that $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1) \mathbin{\overline{\bullet}} \mathsf{profile}_\Phi(\mathcal{M}_2)$ iff $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$. We prove each inclusion separately.

$\subseteq$ Let $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1) \mathbin{\overline{\bullet}} \mathsf{profile}_\Phi(\mathcal{M}_2)$. Hence there exist $\mathcal{E}_1 \in \mathsf{profile}_\Phi(\mathcal{M}_1)$ and $\mathcal{E}_2 \in \mathsf{profile}_\Phi(\mathcal{M}_2)$ such that $\mathcal{E}_1 \cup \mathcal{E}_2 \triangleright^* \mathcal{E}$. By Lemma 28, $\mathcal{E} \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$. Thus $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$.

$\supseteq$ Let $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$ and thus $\mathcal{E} \in \mathsf{decomps}_\Phi(\mathcal{M}_1 \bullet \mathcal{M}_2)$. By Lemma 31, there exist $\mathcal{E}_1 \in \mathsf{decomps}_\Phi(\mathcal{M}_1)$, $\mathcal{E}_2 \in \mathsf{decomps}_\Phi(\mathcal{M}_2)$ such that $\mathcal{E}_1 \cup \mathcal{E}_2 \triangleright^* \mathcal{E}$. Hence, by definition of $\overline{\bullet}$, $\mathcal{E} \in \mathsf{profile}_\Phi(\mathcal{M}_1) \mathbin{\overline{\bullet}} \mathsf{profile}_\Phi(\mathcal{M}_2)$.

6.8 The Abstraction $\mathsf{profile}_\Phi$ is a Homomorphism

Having shown that $\mathsf{profile}_\Phi$ is a homormorphism w.r.t. renaming (Theorem 1), forgetting (Theorem 2) and composition (Theorem 3) in previous subsections, we immediately obtain the homomorphism result from the main paper.

Algorithm 2: The algorithm $\mathsf{abstractSID}(\Phi)$ computes a function f that maps each predicate $\mathsf{pred} \in \mathbf{Preds}(\Phi)$ to the set of profiles $\{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\}$.

```
$f_{prev} := \lambda\mathsf{pred} \,.\, \emptyset$;
repeat
    $f_{curr} := f_{prev}$;
    for $\mathsf{pred} \in \mathbf{Preds}(\Phi)$ do
        for $(\mathsf{pred} \Leftarrow \exists \mathbf{y} \,.\, x \to \mathbf{z_0} * \mathsf{pred}_1(\mathbf{z_1}) * \cdots * \mathsf{pred}_k(\mathbf{z_k})) \in \mathbf{Rules}(\Phi)$ do
            $\mathcal{P}_0 := \mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{z_0})$;
            for $\mathcal{F}_1 \in f_{prev}(\mathsf{pred}_1), \ldots, \mathcal{F}_k \in f_{prev}(\mathsf{pred}_k)$ do
                for $i \in \{1, \ldots, k\}$ do
                    $\mathcal{P}_i := \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_i),\mathbf{z_i}}(\mathcal{F}_i)$;
                $\mathcal{P} := \overline{\mathsf{forget}}_{\mathbf{y}}(\mathcal{P}_0 \mathbin{\overline{\bullet}} \mathcal{P}_1 \mathbin{\overline{\bullet}} \cdots \mathbin{\overline{\bullet}} \mathcal{P}_k)$;
                $f_{curr}(\mathsf{pred}) := f_{curr}(\mathsf{pred}) \cup \{\mathcal{P}\}$;
until $f_{curr} = f_{prev}$;
return $f_{curr}$
```

Theorem 4. *For all concrete heap graphs $\mathcal{M}$, $\mathcal{M}'$ and every SID Φ, we have*

$$\begin{aligned}
\overline{\mathsf{rename}}_{\mathbf{x},\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M})) &= \mathsf{profile}_{\Phi}(\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})) \\
\overline{\mathsf{forget}}_{\mathbf{x}}(\mathsf{profile}_{\Phi}(\mathcal{M})) &= \mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{x}}(\mathcal{M})) \\
\mathsf{profile}_{\Phi}(\mathcal{M}) \mathbin{\overline{\bullet}} \mathsf{profile}_{\Phi}(\mathcal{M}') &= \mathsf{profile}_{\Phi}(\mathcal{M} \bullet \mathcal{M}')
\end{aligned}$$

provided that $\mathsf{rename}_{\mathbf{x},\mathbf{y}}(\mathcal{M})$, $\mathsf{forget}_{\mathbf{x}}(\mathcal{M})$, *and* $\mathcal{M} \bullet \mathcal{M}'$ *are defined, respectively.*

7 Missing Proofs of Section 5 (An Effective Decision Procedure for Entailment)

Unless noted otherwise, we assume that all SIDs considered in this section satisfy all assumptions from Section 3 of the paper, i.e., we assume that they satisfy progress, connectivity, establishment, have only satisfiable unfoldings, and have pairwise different parameters. We restate the algorithm abstractSID as Algorithm 2 to make the present section more self contained.

7.1 Partial Correctness of abstractSID

Lemma 32. *Let $\mathcal{P} \in \mathsf{abstractSID}(\Phi)(\mathsf{pred})$. Then there exists a concrete heap graph $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ such that $\mathcal{P} = \mathsf{profile}_{\Phi}(\mathcal{M})$.*

Proof. We proceed by induction on the number of iterations of the `repeat` loop of abstractSID that are executed until $\mathcal{P}$ is added to f_{curr}.

- If $\mathcal{P}$ is discovered in the first iteration, there exists a rule $(\mathsf{pred} \Leftarrow x \to \mathbf{z}) \in \mathbf{Rules}(\Phi)$ such that $\mathcal{P}$ was computed when processing this rule. Let $\mathcal{M} \cong x \rightarrowtail \mathbf{z}$. Then $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ and $\mathcal{P} = \mathsf{profile}_{\Phi}(\mathcal{M})$.
- If $\mathcal{P}$ is discovered in a later iteration, there are a rule $(\mathsf{pred} \Leftarrow \exists \mathbf{y} \,.\, x \to \mathbf{z_0} * \mathsf{pred}_1(\mathbf{z_1}) * \cdots * \mathsf{pred}_k(\mathbf{z_k})) \in \mathbf{Rules}(\Phi)$ and profiles $\mathcal{P}_0$ and $\mathcal{F}_1, \ldots, \mathcal{F}_k$ such that (1) $\mathcal{P}_0 = \mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{z_0})$, (2) $\mathcal{F}_i \in f_{prev}(\mathsf{pred}_i)$ and (3) $\mathcal{P} = \overline{\mathsf{forget}}_{\mathbf{y}}(\mathcal{P}_0 \mathbin{\overline{\bullet}} \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z_1}}(\mathcal{F}_1) \mathbin{\overline{\bullet}} \cdots \mathbin{\overline{\bullet}} \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_k),\mathbf{z_k}}(\mathcal{F}_k))$ $(*)$.
  Since each $\mathcal{F}_i$, $1 \leq i \leq k$, was computed in a strictly earlier iteration of $\mathsf{abstractSID}$ (otherwise it would not be contained in f_{prev}), we obtain by induction that for each of these $\mathcal{F}_i$ there exists an $\mathcal{M}_i$ such that $\mathcal{M}_i \models_{\Phi} \mathsf{pred}_i(\mathsf{fv}(\mathsf{pred}_i))$ and $\mathcal{F}_i = \mathsf{profile}_{\Phi}(\mathcal{M}_i)$. Let $\mathcal{M}_0 \cong x \rightarrowtail \mathbf{z_0}$ and $\mathcal{M} = \mathsf{forget}_{\mathbf{y}}(\mathcal{M}_0 \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z_1}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_k),\mathbf{z_k}}(\mathcal{M}_k))$ $(\dagger)$.
  Applying that $\mathsf{profile}_{\Phi}$ is a homomorphism (Theorem 4) to $(*)$ and $(\dagger)$, we obtain that $\mathcal{P} = \mathsf{profile}_{\Phi}(\mathcal{M})$. By Lemma 14, $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$.

Lemma 33. *Let $\mathcal{M}$ be a concrete heap graph such that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$. Then $\mathsf{profile}_{\Phi}(\mathcal{M}) \in \mathsf{abstractSID}(\Phi)(\mathsf{pred})$.*

Proof. We proceed by induction on the number n of rules of the semantics used to derive that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$.

- Assume $n = 1$. In that case, there is a rule $(\mathsf{pred} \Leftarrow x \to \mathbf{z_0}) \in \mathbf{Rules}(\Phi)$ such that $\mathcal{M} \models_{\Phi} x \to \mathbf{z_0}$. In the first iteration of the `repeat` loop of $\mathsf{abstractSID}$, the rule $(\mathsf{pred} \Leftarrow x \to \mathbf{z_0})$ will be processed and $\mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{z_0}) = \mathsf{profile}_{\Phi}(\mathcal{M})$ will be added to $f_{curr}(\mathsf{pred})$. Since elements that have been added to f_{curr} will never be removed, $\mathsf{profile}_{\Phi}(\mathcal{M}) \in \mathsf{abstractSID}(\Phi)(\mathsf{pred})$.
- Assume $n > 1$. In that case there exist $k > 0$, $\mathsf{pred}_1, \ldots, \mathsf{pred}_k$, a variable x, and tuples of variables $\mathbf{y}, \mathbf{z_0}, \ldots, \mathbf{z_k}$ such that the rule $(\mathsf{pred} \Leftarrow \exists \mathbf{y} \,.\, x \to \mathbf{z_0} * \mathsf{pred}_1(\mathbf{z_1}) * \cdots * \mathsf{pred}_k(\mathbf{z_k})) \in \mathbf{Rules}(\Phi)$ was the first rule used to derive that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))$ holds. By Lemma 14, there exist $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$ such that

  $$\mathcal{M} = \mathsf{forget}_{\mathbf{y}}(\mathcal{M}_0 \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z_1}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_k),\mathbf{z_k}}(\mathcal{M}_k)). \quad (*)$$

  Note that for each $\mathcal{M}_i$, $1 \leq i \leq k$, strictly fewer rule applications than for $\mathcal{M}$ are needed to show that $\mathcal{M}_i \models_{\Phi} \mathsf{pred}_i(\mathsf{fv}(\mathsf{pred}_i))$ holds. By the induction hypothesis we thus have (for $1 \leq i \leq k$) that $\mathsf{profile}_{\Phi}(\mathcal{M}_i) \in \mathsf{abstractSID}(\Phi)(\mathsf{pred}_i)$.
  Since $\mathsf{abstractSID}$ processes all rules of Φ in each of its iterations, it will in particular process the above rule; and since it considers as profiles for pred_i all elements of $\mathsf{abstractSID}(\Phi)(\mathsf{pred}_i)$ found in previous iterations, there thus is an iteration of the third `for` loop in $\mathsf{abstractSID}$ in which each $\mathcal{P}_i$ is instantiated with $\mathsf{profile}_{\Phi}(\mathcal{M}_i)$, $1 \leq i \leq k$.
  For $\mathcal{P}_0 = \mathsf{profile}_{\Phi}(\mathcal{M}_0)$ and these choices of $\mathcal{P}_1, \ldots, \mathcal{P}_k$, $\mathsf{abstractSID}$ will thus add

  $$\begin{aligned}\overline{\mathsf{forget}}_{\mathbf{y}}(\mathsf{profile}_{\Phi}(\mathcal{M}_0) &\mathbin{\overline{\bullet}} \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z_1}}(\mathsf{profile}_{\Phi}(\mathcal{M}_1)) \\ &\mathbin{\overline{\bullet}} \cdots \mathbin{\overline{\bullet}} \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_k),\mathbf{z_k}}(\mathsf{profile}_{\Phi}(\mathcal{M}_k))) \quad (\dagger)\end{aligned}$$

to $f_{curr}(\mathsf{pred})$. Note that all these abstract operations are defined because (by assumption) the SID only has satisfiable unfoldings. Because $\mathsf{profile}_{\Phi}$ is a homomorphism (Theorem 4), (†) can be rewritten to

$$\mathsf{profile}_{\Phi}(\mathsf{forget}_{\mathbf{y}}(M_0 \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{z_1}}(\mathcal{M}_1) \bullet \cdots \bullet \mathsf{rename}_{\mathsf{fv}(\mathsf{pred}_k),\mathbf{z_k}}(\mathcal{M}_k))$$

By $(*)$, this is equal to $\mathsf{profile}_{\Phi}(\mathcal{M})$. Thus, $\mathsf{profile}_{\Phi}(\mathcal{M}) \in \mathsf{abstractSID}(\Phi)(\mathsf{pred})$.

Theorem 5. $\mathsf{abstractSID}(\Phi)(\mathsf{pred}) = \{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\}$.

Proof. Combine Lemmas 32 and 33.

7.2 Total Correctness of $\mathsf{abstractSID}(\Phi)$

We next show that $\mathsf{abstractSID}(\Phi)$ terminates, thus proving the total correctness of $\mathsf{abstractSID}$.

Lemma 34. *Let Φ be an SID. $\mathsf{abstractSID}(\Phi)$ terminates.*

Proof. That k be the maximum arity of the predicates $\mathbf{Preds}(\Phi)$. Assume w.l.o.g. that there exists a tuple $\mathbf{x} \in \mathbf{Var}^k$ such that for all $\mathsf{pred} \in \mathbf{Preds}(\Phi)$, $\mathsf{fv}(\mathsf{pred}) \subseteq \mathbf{x}$. By Theorem 5, $\mathsf{abstractSID}(\Phi)(\mathsf{pred}) = \{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\}$. Observe that $\{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\} \subseteq \mathbf{Profiles}^{\mathbf{x}}(\Phi)$. By Lemma 1, $\mathbf{Profiles}^{\mathbf{x}}(\Phi)$ is finite.

$\mathsf{abstractSID}(\Phi)$ terminates as soon as at the end of an iteration of the outermost `repeat` loop, $f_{curr}(\mathsf{pred}) = f_{prev}(\mathsf{pred})$ holds for all predicates. Thus $\mathsf{abstractSID}(\Phi)$ terminates after at most $|\mathbf{Profiles}^{\mathbf{x}}(\Phi)| \cdot |\mathbf{Preds}(\Phi)|$ iterations. As each iteration of the `repeat` loop in $\mathsf{abstractSID}$ terminates (because $\mathsf{profile}_{\Phi}(x \rightarrowtail \mathbf{z_0}$, $\overline{\mathsf{rename}}$, $\overline{\mathsf{forget}}$ and $\overline{\bullet}$ are decidable), $\mathsf{abstractSID}(\Phi)$ terminates.

7.3 Deciding Entailments

If we know the profile of a heap graph $\mathcal{M}$, we can decide for each predicate pred in the SID and each sequence of variables $\mathbf{x}$ whether $\mathcal{M}$ is a model of $\mathsf{pred}(\mathbf{x})$.

Lemma 35. *Let $\mathcal{M}$ be a concrete heap graph, $\mathsf{pred} \in \mathbf{Preds}(\Phi)$ and $\mathbf{x} \in \mathbf{Var}^*$. $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ holds iff $\{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle\} \in \mathsf{profile}_{\Phi}(\mathcal{M})$.*

Proof. $\Rightarrow$ Let $\mathcal{M}$ be a concrete heap graph such that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$. Because $\mathcal{M}$ is concrete, $\mathsf{calls}_{\mathcal{M}} = \emptyset$ and hence $\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle \in \mathsf{contexts}_{\Phi}(\mathcal{M})$. Trivially, $\{\mathcal{M}\} \in \mathbf{Decomp}_{\Phi}(\mathcal{M})$. Therefore $\mathcal{E} = \{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle\} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$ and thus $\mathcal{E} \in \mathsf{profile}_{\Phi}(\mathcal{M})$.

$\Leftarrow$ Let $\{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle\} \in \mathsf{profile}_{\Phi}(\mathcal{M})$. By definition of profiles, it immediately follows that $\{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle\} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$. By definition of context decompositions, there exists an $\mathcal{M}'$ with $\mathcal{C} = \langle \mathsf{FV}_{\mathcal{M}'}, \mathsf{pred}(\mathbf{x}), \emptyset\rangle \in \mathsf{contexts}_{\Phi}(\mathcal{M}')$ and $\{\mathcal{M}'\} \in \mathsf{decomps}_{\Phi}(\mathcal{M})$. Since $\mathcal{C} \in \mathsf{contexts}_{\Phi}(\mathcal{M}')$, $\mathcal{M}' \models_{\Phi} \mathsf{pred}(\mathbf{x})$. By definition of $\mathsf{decomps}_{\Phi}$, $\mathcal{M} = \mathcal{M}'$. By definition of contexts, we thus obtain that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$.

Algorithm 3: The decision procedure $\mathsf{entcheck}_{\Phi}(\mathsf{pred}_1(\mathbf{x_1}), \mathsf{pred}_2(\mathbf{x_2}))$ that returns whether $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$.

```
$f := \mathsf{abstractSID}(\Phi)$;
for $\mathcal{P} \in f(\mathsf{pred}_1)$ do
    $\mathcal{P}' := \overline{\mathsf{rename}}_{\mathsf{fv}(\mathsf{pred}_1),\mathbf{x_1}}(\mathcal{P})$;
    if $\neg\mathsf{accept}(\mathcal{P}', \mathsf{pred}_2(\mathbf{x_2}))$ then
        return $False$;
return $True$;
```

Theorem 6. *The entailment* $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$ *holds iff for all concrete heap graphs* $\mathcal{M}$ *with* $\mathcal{M} \models \mathsf{pred}_1(\mathbf{x_1})$, $\{\langle \mathsf{FV}_{\mathcal{M}}, \mathsf{pred}_2(\mathbf{x_2}), \emptyset \rangle\} \in \mathsf{profile}_{\Phi}(\mathcal{M})$.

Proof. Apply Lemma 35 for $\mathsf{pred}_2(\mathbf{x_2})$ to the models of $\mathsf{pred}_1(\mathbf{x_1})$.

As $\mathsf{abstractSID}$ can be used to compute the profiles of all models of the left-hand side of an entailment query (modulo renaming), we can decide the entailment query $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$ based on Theorem 6. To formalize this decision procedure, we define the following function.

$$\mathsf{accept}(\mathcal{P}, \mathsf{pred}(\mathbf{x})) := \begin{cases} True & \text{if } \exists \mathbf{y} \;.\; \{\langle \mathbf{y}, \mathsf{pred}(\mathbf{x}), \emptyset \rangle\} \in \mathcal{P} \\ False & \text{otherwise} \end{cases}$$

Observe that it immediately follows from Theorem 6 that $\mathcal{M} \models_{\Phi} \mathsf{pred}(\mathbf{x})$ iff $\mathsf{accept}(\mathsf{profile}_{\Phi}(\mathcal{M}), \mathsf{pred}(\mathbf{x})) = True$. We exploit this in the algorithm $\mathsf{entcheck}_{\Phi}$, Algorithm 3, which decides the entailment problem $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$ by using accept on top of $\mathsf{abstractSID}$.

Theorem 7. *The algorithm* $\mathsf{entcheck}_{\Phi}(\mathsf{pred}_1(\mathbf{x_1}), \mathsf{pred}_2(\mathbf{x_2}))$ *(Algorithm 3) always terminates and returns* $True$ *iff* $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$.

Proof. By Theorem 5 and Lemma 34, $\mathsf{abstractSID}$ always terminates with the result $\mathsf{abstractSID}(\Phi)(\mathsf{pred}) = \{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}(\mathsf{fv}(\mathsf{pred}))\}$ (for every $\mathsf{pred} \in \mathbf{Preds}(\Phi)$). As $\mathsf{entcheck}_{\Phi}$ calls $\overline{\mathsf{rename}}$ and accept a finite number of times: $\mathbf{Profiles}^{\mathsf{fv}(\mathsf{pred}_1)}(\Phi)$ is finite by Lemma 1 and accept and $\overline{\mathsf{rename}}$ are called at most $\left|\mathbf{Profiles}^{\mathsf{fv}(\mathsf{pred}_1)}(\Phi)\right|$ many times. Furthermore, $\overline{\mathsf{rename}}$ and accept terminate. Thus $\mathsf{entcheck}_{\Phi}$ terminates.

Since the variable $\mathcal{P}$ in $\mathsf{entcheck}_{\Phi}$ ranges over the set

$$\{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}_1(\mathsf{fv}(\mathsf{pred}_1))\}\,,$$

by Theorem 1 we obtain that $\mathcal{P}'$ ranges over the set

$$\{\mathsf{profile}_{\Phi}(\mathcal{M}) \mid \mathcal{M} \models_{\Phi} \mathsf{pred}_1(\mathbf{x_1})\}\,.$$

By Theorem 6, $\mathsf{accept}(\mathsf{profile}_{\Phi}(\mathcal{M}), \mathsf{pred}_2(\mathbf{x_2})) = True$ iff $\mathcal{M} \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$. The algorithm $\mathsf{entcheck}_{\Phi}$ thus returns $True$ iff for all $\mathcal{M} \models_{\Phi} \mathsf{pred}_1(\mathbf{x_1})$ also $\mathcal{M} \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$, i.e., iff $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$.

The decidability of the entailment problem immediately follows from Theorem 7.

Corollary 2. *It is decidable whether the entailment* $\mathsf{pred}_1(\mathbf{x}_1) \models_{\Phi} \mathsf{pred}_2(\mathbf{x}_2)$ *holds.*

8 Deciding Entailment Between Established Symbolic Heaps

In this section we will show how to extend our entailment checker from queries of the form $\mathsf{pred}_1(\mathbf{x_1}) \models_{\Phi} \mathsf{pred}_2(\mathbf{x_2})$ to entailment queries $\varphi \models_{\Phi} \psi$ for symbolic heaps φ, ψ. As in the main paper, we simplify the presentation by assuming that φ and ψ do not contain pure formulas. We make the following three additional assumptions. First, we assume that the left-hand side of the query, φ, is quantifier free.[3] Note that this is w.l.o.g.: We can simply drop the quantifier prefix in a trivial Skolemization step. Second, we assume that the right-hand side of the query, ψ, is established. Third, we assume (w.l.o.g. as argued in Section 5) that φ and ψ are "all-satisfiable", i.e., only have satisfiable unfoldings.

To check $\varphi \models_{\Phi} \psi$, we generalize $\mathsf{entcheck}_{\Phi}$ (Algorithm 3) as follows.

1. Since φ and ψ are established and Φ satisfies connectivity and progress (and thus all models of Φ are rooted, cf. Lemma 7), we can decompose both φ and ψ into subformulas such that the subformulas correspond to the *strongly connected components* of the models of φ and ψ, respectively. Formally, we decompose φ into $\varphi_1, \ldots, \varphi_n$ such that
   1. $\varphi_1 * \varphi_2 * \cdots * \varphi_n$ is syntactically identical to φ modulo reordering of atomic formulas,
   2. for each φ_i, all models $\mathcal{M} \models_{\Phi} \varphi_i$ are rooted, and
   3. for each $\varphi_i \neq \varphi_j$, there is a model $\mathcal{M} \models_{\Phi} \varphi_i * \varphi_j$ that is *not* rooted.

   Analogously, we decompose ψ into $\exists \mathbf{y} \,.\, \psi_1 \bullet \cdots \bullet \psi_m$ with the same properties. (Note the quantifier prefix $\mathbf{y}$, which cannot be avoided for the right-hand side).
2. If $m \neq n$, we can return immediately that the entailment does *not* hold.
3. Otherwise we compute for each φ_i (ψ_j) that is *not* a single predicate call an SID that satisfies establishment, progress and connectivity and that has a predicate $\mathsf{pred}_{\varphi_i}$ (pred_{ψ_j}) such that for all concrete heap graphs $\mathcal{M}$, $\mathcal{M} \models_{\Phi} \varphi_i$ iff $\mathcal{M} \models_{\Phi} \mathsf{pred}_{\varphi_i}(\mathsf{fv}(\varphi_i))$ (and $\mathcal{M} \models_{\Phi} \psi_j$ iff $\mathcal{M} \models_{\Phi} \mathsf{pred}_{\psi_j}(\mathsf{fv}(\psi_j))$).
   This is possible because each of these subformulas is established and connected and progress can always be ensured by rewriting a rule with multiple points-to assertions into a sequence of recursive calls to rules with single points-to assertions (possibly increasing the arity of the predicates in the process).
4. We set φ'_i (ψ'_j) equal to φ_i (ψ_j) if it is a single predicate call and equal to an appropriate call to $\mathsf{pred}_{\varphi_i}$ (pred_{ψ_j}) otherwise. Furthermore, we let Φ' be the SID obtained by taking the (w.l.o.g. disjoint) union of Φ and the newly generated SIDs.

[3] And thus necessarily established because Φ is established.

5. This way we obtain an equivalent entailment query $\varphi'_1 * \ldots * \varphi'_n \models_{\Phi'} \exists \mathbf{y} . \psi'_1 * \cdots * \psi'_n$ where each subformula is a call to a predicate of the SID Φ'. Note that Φ' satisfies establishment, connectivity and progress.
6. We decide this query as follows.
   1. We use $\mathsf{abstractSID}(\Phi')$ (Algorithm 2) to compute the set of profiles for all predicates of Φ'
   2. We iterate over all tuples $\langle \mathcal{P}_1, \ldots, \mathcal{P}_n \rangle$ such that (for $1 \leq i \leq k$), $\mathcal{P}_i \in \mathsf{abstractSID}(\Phi')(\varphi'_i)$.
   3. For each of these tuples we compute $\mathcal{P} := \mathcal{P}_1 \mathbin{\bar{\bullet}} \cdots \mathbin{\bar{\bullet}} \mathcal{P}_n$.
   4. We now check whether there is a variable sequence $\mathbf{x} \in \mathsf{fv}(\mathcal{P})^*$ such that for $\psi'_i := \psi_i[\mathbf{y}/\mathbf{x}]$, $1 \leq i \leq n$, it holds that

      $$\{\langle \mathsf{fv}(\psi'_1), \psi'_1, \emptyset \rangle, \ldots, \langle \mathsf{fv}(\psi'_n), \psi'_n, \emptyset \rangle\} \in \mathcal{P}.$$

      By a straightforward generalization of Theorem 6, the entailment holds iff such a sequence $\mathbf{x}$ exists.

The above algorithm is used in Harrsh to discharge entailments between symbolic heaps.

9 SID Definitions for the Experiments

The SIDs used in Table 1 in the main paper are presented below.

1. Check entailment $\texttt{sll}(x_1, x_2) \models \texttt{odd}(x_1, x_2)$ w.r.t.

$$\begin{aligned}
\texttt{odd} &\Longleftarrow x_1 \to (x_2)\\
\texttt{odd} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{even}(y, x_2)\\
\texttt{sll} &\Longleftarrow x_1 \to (x_2)\\
\texttt{sll} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{sll}(y, x_2)\\
\texttt{even} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{odd}(y, x_2)
\end{aligned}$$

2. Check entailment $\texttt{even}(x_1, x_2) \models \texttt{sll}(x_1, x_2)$ w.r.t.

$$\begin{aligned}
\texttt{odd} &\Longleftarrow x_1 \to (x_2)\\
\texttt{odd} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{even}(y, x_2)\\
\texttt{sll} &\Longleftarrow x_1 \to (x_2)\\
\texttt{sll} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{sll}(y, x_2)\\
\texttt{even} &\Longleftarrow \exists y\colon\ x_1 \to (y) * \texttt{odd}(y, x_2)
\end{aligned}$$

3. Check entailment $\texttt{rtree}(x_2, x_3, x_1) \models \texttt{ltree}(x_1, x_2, x_3)$ w.r.t.

$$\begin{aligned}
\texttt{parent} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{lroot} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1)\\
\texttt{tree2} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{rtree} &\Longleftarrow \exists r\colon\ x_1 \to (x_3, r, x_2) * \texttt{parent}(x_3, x_1) * \texttt{tree2}(r, x_1)\\
\texttt{lltree} &\Longleftarrow \exists p \exists r\colon\ x_1 \to (x_2, r, p) * \texttt{tree}(r, x_1) * \texttt{lltree}(p, x_1, x_3, x_4)\\
\texttt{lltree} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1) * \texttt{lroot}(x_3, x_1, x_4)\\
\texttt{tree} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{tree} &\Longleftarrow \exists y \exists z\colon\ x_1 \to (y, z, x_2) * \texttt{tree}(y, x_1) * \texttt{tree}(z, x_1)\\
\texttt{ltree} &\Longleftarrow \exists p\colon\ x_1 \to (\mathbf{null}, \mathbf{null}, p) * \texttt{lltree}(p, x_2, x_3)
\end{aligned}$$

4. Check entailment $\texttt{ltree}(x_1, x_2, x_3) \models \texttt{grtree}(x_2, x_3, x_1)$ w.r.t.

$$\begin{aligned}
\texttt{parent} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{lroot} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1)\\
\texttt{rtree} &\Longleftarrow \exists r\colon\ x_1 \to (x_3, r, x_2) * \texttt{parent}(x_3, x_1) * \texttt{tree}(r, x_1)\\
\texttt{rtree} &\Longleftarrow \exists l \exists r\colon\ x_1 \to (l, r, x_2) * \texttt{rtree}(l, x_1, x_3) * \texttt{tree}(r, x_1)\\
\texttt{lltree} &\Longleftarrow \exists p \exists r\colon\ x_1 \to (x_2, r, p) * \texttt{tree}(r, x_1) * \texttt{lltree}(p, x_1, x_3, x_4)\\
\texttt{lltree} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1) * \texttt{lroot}(x_3, x_1, x_4)\\
\texttt{tree} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{tree} &\Longleftarrow \exists y \exists z\colon\ x_1 \to (y, z, x_2) * \texttt{tree}(y, x_1) * \texttt{tree}(z, x_1)\\
\texttt{grtree} &\Longleftarrow \exists l \exists r\colon\ x_1 \to (l, r, x_2) * \texttt{rtree}(l, x_1, x_3) * \texttt{tree}(r, x_1)\\
\texttt{ltree} &\Longleftarrow \exists p\colon\ x_1 \to (\mathbf{null}, \mathbf{null}, p) * \texttt{lltree}(p, x_1, x_2, x_3)
\end{aligned}$$

5. Check entailment $\texttt{grtree}(x_2, x_3, x_1) \models \texttt{ltree}(x_1, x_2, x_3)$ w.r.t.

$$\begin{aligned}
\texttt{parent} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{lroot} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1)\\
\texttt{rtree} &\Longleftarrow \exists r\colon\ x_1 \to (x_3, r, x_2) * \texttt{parent}(x_3, x_1) * \texttt{tree}(r, x_1)\\
\texttt{rtree} &\Longleftarrow \exists l \exists r\colon\ x_1 \to (l, r, x_2) * \texttt{rtree}(l, x_1, x_3) * \texttt{tree}(r, x_1)\\
\texttt{lltree} &\Longleftarrow \exists p \exists r\colon\ x_1 \to (x_2, r, p) * \texttt{tree}(r, x_1) * \texttt{lltree}(p, x_1, x_3, x_4)\\
\texttt{lltree} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, x_3) * \texttt{tree}(r, x_1) * \texttt{lroot}(x_3, x_1, x_4)\\
\texttt{tree} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)\\
\texttt{tree} &\Longleftarrow \exists y \exists z\colon\ x_1 \to (y, z, x_2) * \texttt{tree}(y, x_1) * \texttt{tree}(z, x_1)\\
\texttt{grtree} &\Longleftarrow \exists l \exists r\colon\ x_1 \to (l, r, x_2) * \texttt{rtree}(l, x_1, x_3) * \texttt{tree}(r, x_1)\\
\texttt{ltree} &\Longleftarrow \exists p\colon\ x_1 \to (\mathbf{null}, \mathbf{null}, p) * \texttt{lltree}(p, x_1, x_2, x_3)
\end{aligned}$$

6. Check entailment $\texttt{atll}(x_1, x_2, x_3) \models \texttt{tll}(x_1, x_2, x_3)$ w.r.t.

$$\begin{aligned}
\texttt{tll} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_3) : \{x_1 = x_2\}\\
\texttt{tll} &\Longleftarrow \exists l \exists m \exists r\colon\ x_1 \to (l, r, \mathbf{null}) * \texttt{tll}(l, x_2, m) * \texttt{tll}(r, m, x_3)\\
\texttt{atll} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_3) : \{x_1 = x_2, x_1 \neq x_3\}\\
\texttt{atll} &\Longleftarrow \exists l \exists m \exists r\colon\ x_1 \to (l, r, \mathbf{null}) * \texttt{atll}(l, x_2, m) * \texttt{atll}(r, m, x_3) : \{x_1 \neq x_3\}
\end{aligned}$$

7. Check entailment $\texttt{tll}(x_1, x_2, x_3) \models \texttt{atll}(x_1, x_2, x_3)$ w.r.t.

$$\begin{aligned}
\texttt{tll} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_3) : \{x_1 = x_2\}\\
\texttt{tll} &\Longleftarrow \exists l \exists m \exists r\colon\ x_1 \to (l, r, \mathbf{null}) * \texttt{tll}(l, x_2, m) * \texttt{tll}(r, m, x_3)\\
\texttt{atll} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_3) : \{x_1 = x_2, x_1 \neq x_3\}\\
\texttt{atll} &\Longleftarrow \exists l \exists m \exists r\colon\ x_1 \to (l, r, \mathbf{null}) * \texttt{atll}(l, x_2, m) * \texttt{atll}(r, m, x_3) : \{x_1 \neq x_3\}
\end{aligned}$$

8. Check entailment $\texttt{tll}^{lin}(x_1, x_2, x_3) \models \texttt{tll}(x_1, x_2, x_3)$ w.r.t.

$$\begin{aligned}
\texttt{tll} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_3) : \{x_1 = x_2\}\\
\texttt{tll} &\Longleftarrow \exists l \exists m \exists r\colon\ x_1 \to (l, r, \mathbf{null}) * \texttt{tll}(l, x_2, m) * \texttt{tll}(r, m, x_3)\\
\texttt{twoleafptrs} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2) * \texttt{oneptr}(x_2, x_3)\\
\texttt{tll}^{lin} &\Longleftarrow \exists r\colon\ x_1 \to (x_2, r, \mathbf{null}) * \texttt{twoleafptrs}(x_2, r, x_3)\\
\texttt{oneptr} &\Longleftarrow x_1 \to (\mathbf{null}, \mathbf{null}, x_2)
\end{aligned}$$

9. Check entailment $\texttt{dllHT}(x_1, x_2, x_3, x_4) \models \texttt{dllTH}(x_3, x_4, x_1, x_2)$ w.r.t.

$$\begin{aligned}
\texttt{dllTH} &\Longleftarrow x_1 \to (x_2, x_3) * \texttt{pto}(x_3, x_1, x_4)\\
\texttt{dllTH} &\Longleftarrow \exists y\colon\ x_1 \to (x_2, y) * \texttt{dllTH}(y, x_1, x_3, x_4)\\
\texttt{pto} &\Longleftarrow x_1 \to (x_2, x_3)\\
\texttt{dllHT} &\Longleftarrow x_1 \to (x_3, x_2) * \texttt{pto}(x_3, x_4, x_1)\\
\texttt{dllHT} &\Longleftarrow \exists y\colon\ x_1 \to (y, x_2) * \texttt{dllHT}(y, x_1, x_3, x_4)
\end{aligned}$$

10. Check entailment $\texttt{dllTH}(x_1, x_2, x_3, x_4) \models \texttt{dllHT}(x_3, x_4, x_1, x_2)$ w.r.t.

$$\begin{aligned}
\texttt{dllTH} &\Longleftarrow x_1 \to (x_2, x_3) * \texttt{pto}(x_3, x_1, x_4)\\
\texttt{dllTH} &\Longleftarrow \exists y\colon\ x_1 \to (x_2, y) * \texttt{dllTH}(y, x_1, x_3, x_4)\\
\texttt{pto} &\Longleftarrow x_1 \to (x_2, x_3)\\
\texttt{dllHT} &\Longleftarrow x_1 \to (x_3, x_2) * \texttt{pto}(x_3, x_4, x_1)\\
\texttt{dllHT} &\Longleftarrow \exists y\colon\ x_1 \to (y, x_2) * \texttt{dllHT}(y, x_1, x_3, x_4)
\end{aligned}$$

11. Check entailment $\texttt{dlgridR}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \models \texttt{dlgridL}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ w.r.t.

$$\begin{aligned}
\texttt{dlgridRR} &\Longleftarrow x_1 \to (x_5, x_2, x_3, \mathbf{null}) * \texttt{bot}(x_2, x_6, x_4, x_1)\\
\texttt{dlgridL} &\Longleftarrow \exists b \exists l\colon\ x_5 \to (x_7, x_6, l, \mathbf{null}) * \texttt{dlgridL}(x_1, x_2, x_3, x_4, l, b, x_5, x_8) * \texttt{bot}(x_6, x_8, b, x_5)\\
\texttt{dlgridL} &\Longleftarrow x_5 \to (x_7, x_6, x_1, \mathbf{null}) * \texttt{dlgridLL}(x_1, x_2, x_5, x_6, x_3, x_4) * \texttt{bot}(x_6, x_8, x_2, x_5)\\
\texttt{dlgridR} &\Longleftarrow \exists b \exists r\colon\ x_1 \to (r, x_2, x_3, \mathbf{null}) * \texttt{dlgridR}(r, b, x_1, x_2, x_5, x_6, x_7, x_8) * \texttt{bot}(x_2, b, x_4, x_1)\\
\texttt{dlgridR} &\Longleftarrow x_1 \to (x_5, x_2, x_3, \mathbf{null}) * \texttt{dlgridRR}(x_5, x_6, x_1, x_2, x_7, x_8) * \texttt{bot}(x_2, x_6, x_4, x_1)\\
\texttt{dlgridLL} &\Longleftarrow x_1 \to (x_3, x_2, x_5, \mathbf{null}) * \texttt{bot}(x_2, x_4, x_6, x_1)\\
\texttt{bot} &\Longleftarrow x_1 \to (x_2, \mathbf{null}, x_3, x_4)
\end{aligned}$$

10 Full Experimental Results

Table 2 on this and the following pages contains our full experimental results. We discuss these results below the table. Note that the Harrsh, Songbird and Slide input files for all benchmarks can be found at [1].

Benchmark		Time (ms)			Profiles		
File	Status	HRS	SB	SLD	#P	#D	#C
external_acyclicity	true	1	386	TO	2	3	3
external_contradicting_disequality	false	1	29	TO	2	1	1
external_contradicting_null	false	0	29	TO	2	2	2
external_equality	true	1	31	(X)	2	2	2
external_equality_missing	false	1	29	(X)	2	2	2
external_null	true	1	30	(X)	2	3	3
external_null_missing	false	0	29	(X)	2	2	2
sll-extra-rule_sll	true	2	433	(X)	1	2	2
sll-no-progress_sll	true	1	66	(X)	1	2	2
sll_sll-no-progress	true	0	33	(X)	1	1	1
sll_sll-no-progress2	false	0	(U)	(X)	1	1	1

sll_sll-no-progress3	true	2	28	(X)	2	5	6
sll_sll-no-progress4	false	0	(U)	(X)	2	2	2
2-dl-grid	true	102	30 (*)	36	5	12	14
2-grid	true	17	60	36	5	7	7
dlgrid-left-right	true	7810	TO	(X)	5	87	208
dlgrid	true	7705	195	36	5	67	156
dlgrid2-dlgrid	true	11784	244	36	7	95	218
acyc-dll_dll	true	15	202	TO	3	10	14
dll_acyc-dll	false	3	127	TO	2	2	2
dll_backward_forward	true	18	41	58	3	27	45
dll_dll	true	12	39	36	3	10	14
dll_forward_backward	true	18	41	62	3	27	45
acyclic-sll_sll	true	1	98	TO	1	2	2
even-sll_sll	true	3	29	44	2	4	4
mixedarity_ls	false	1	–	44	2	2	2
odd-or-even-sll_sll	true	5	45	44	3	6	6
odd-sll_sll	true	2	23	44	2	4	4
oneptr_ls	true	0	21	43	1	2	2
ptr_sll	true	0	10	43	1	2	2
ptrs_sll	true	2	12	44	3	6	6
reverse-ptr_reverse-sll-call	true	0	21	43	1	2	2
reverse-sll_reverse-sll-call	true	1	35	44	1	2	2
sll-min-length_sll	true	4	31	44	4	8	8
sll_acyclic-sll	false	1	42	TO	2	2	2
sll_even-sll	false	4	10	40	2	6	6
sll_mixedarity	true	1	(X)	48	1	2	2
sll_odd-or-even-sll-wo-basecase	false	14	(U)	44	3	17	17
sll_odd-or-even-sll	true	11	(U)	46	2	12	12
sll_odd-sll	false	4	12	46	2	6	6
sll_ptr	false	0	10	40	2	1	1
sll_reverse-sll-call	false	1	(U)	90	1	2	2
sll_sll-min-length	false	21	11	41	4	26	26
sll_sll	true	1	29	36	1	2	2
sll_twoptr	false	1	21	40	3	3	3
twoptr_ls	true	1	23	43	2	4	4
twosll_sll	true	6	331	44	2	4	4
wrongarity2_ls	false	0	–	44	1	0	0
wrongarity_sll	false	0	–	44	1	0	0
existential-sll_sll	false	2	(U)	(X)	2	3	3
existential-sll_sll2	true	16	31	(X)	3	13	16
parameter_reordering	true	2	35	44	2	4	4
ptr-to-null_sll	true	0	22	45	1	2	2
sll-to-null_sll	true	4	34	44	2	5	6
sll_existential-sll	true	8	29	(X)	2	10	12
sll_sll-to-null	true	9	31	(X)	3	10	10

sll_sll-to-null2	true	2	34	44	3	4	4
neq_not_eq	false	0	5	(X)	n/a	n/a	n/a
pure1	true	0	5	(X)	n/a	n/a	n/a
pure2	true	0	5	(X)	n/a	n/a	n/a
pure3	false	0	5	(X)	n/a	n/a	n/a
pure4	true	0	6	(X)	n/a	n/a	n/a
singleptr1	true	0	5	35	1	1	1
singleptr10	false	0	(U)	(X)	1	1	1
singleptr2	false	0	(U)	42	1	1	1
singleptr3	false	0	5	(X)	1	1	1
singleptr4	true	0	5	(X)	1	1	1
singleptr5	true	0	6	37	1	1	1
singleptr6	false	0	6	36 (*)	1	1	1
singleptr7	true	0	6	(X)	1	1	1
singleptr8	true	0	5	(X)	1	1	1
singleptr9	true	0	6	(X)	1	1	1
singleptr_twoptrs	false	0	6	41	1	2	2
singleptr_twoptrs2	false	0	6	(X)	1	2	2
twoptrs_singleptr	false	0	6	42	2	1	1
almost-linear-treep_treep	true	4	95	48	2	2	2
greater-ptree_leaf-tree	true	870	1297	55	9	57	87
leaf-tree_greater-ptree	true	580	TO	57	7	70	116
leaf-tree_ptree	true	203	TO	57	6	39	63
ptree_leaf-tree	false	900	(U)	55	11	75	117
small-ptree_leaf-tree	false	33	5581	55	3	14	21
treep_almost-linear-treep	false	17	37	50	3	3	3
treep_treep	true	3	723	36	1	1	1
almost-linear-tree_tree	true	1	123	44	2	2	2
ptr-with-external-null2_tree	true	2	32	(X)	2	3	3
ptr-with-external-null_tree	true	9	32	(X)	2	3	3
ptr-with-nullalias_tree	true	0	25	(X)	1	2	2
ptrwonull_tree	false	0	(U)	(X)	1	2	2
tree-depthtwo_tree	true	1	31	43	2	2	2
tree_almost-linear-tree	false	9	34	44	3	3	3
tree_tree-depthtwo	false	4	22	40	3	2	2
tree_tree	true	1	1585	36	1	1	1
treefragment-plus-tree_tree	true	15	5327	(X)	3	3	3
acyc-tll_tll	true	66	8940	TO	2	2	2
oneptr_tll	true	1	24	59 (*)	1	1	1
tll-classes_simple-tll	true	39	37	68 (*)	3	3	4
tll-classes_tll	true	15	34	59 (*)	3	3	4
tll-parent_tll-parent	true	764	6033	36	2	2	2
tll_acyc-tll	false	15	120	TO	2	1	1
tll_tll	true	52	5964	36	2	2	2
list-segments-different-order	true	17	–	(X)	3	23	37

Table 2: Full experimental reults.

How to read the table.

- We report on the performance of Harrsh (HRS), Songbird (SB) and Slide (SLD) on a variety of SIDs. The file names match those in the archive available at [1].
- Beside the run times, the table contains the size of the abstraction computed by Harrsh. More specifically, we report (1) the total number of profiles in the fixed point of abstractSID (#P), (2) the total number of context decompositions across all profiles (#D), and (3) the total number of contexts across all decompositions of all profiles (#C).
- We use the following abbreviations in the table:
  - (TO) means that the tool did not finish within the timeout of 180s.
  - (X) means that the program crashed on the input.
  - (U) means that the program terminated within the time limit with result "unknown" (as opposed to valid or invalid).
  - (*) denotes a wrong result returned by a tool (valid instead of invalid or vice-versa).
  - Our separation logic is not typed. The benchmarks mixedarity_ls.hrs, wrongarity2_ls.hrs and wrongarity_sll.hrs did not type check in Songbird. The corresponding cells are marked – in the table.
  - n/a: No profiles/contexts were computed.
- Why Slide crashed (according to the error messages produced by Slide):
  - Equalities used in a way not supported by Slide: external_equality.hrs, external_equality_missing.hrs, external_null.hrs, existential-sll_sll2.hrs, sll_sll-to-null.hrs, ptr-with-external-null2_tree.hrs, ptr-with-external-null_tree.hrs, ptr-with-nullalias_tree.hrs, treefragment-plus-tree_tree.hrs
  - Possible dangling pointers: external_null_missing.hrs, sll_sll-no-progress3.hrs, sll_sll-no-progress4.hrs, existential-sll_sll.hrs, sll_existential-sll.hrs. (Note that this is a false alarm in several cases.)
    Progress violated: sll-extra-rule_sll.hrs, sll-no-progress_sll.hrs, sll_sll-no-progress.hrs, sll_sll-no-progress2.hrs
  - Other/unclear: dlgrid-left-right.hrs

References

1. Supplementary material. The webpage below provides access to proofs, our tool, its source code, and our benchmarks, https://github.com/katelaan/harrsh
2. Dodds, M., Plump, D.: From hyperedge replacement to separation logic and back. ECEASST 16 (2008)
3. Habel, A.: Hyperedge Replacement: Grammars and Languages, LNCS, vol. 643. Springer (1992)
4. Jansen, C., Göbe, F., Noll, T.: Generating inductive predicates for symbolic execution of pointer-manipulating programs. In: Graph Transformation - 7th International Conference, ICGT 2014, Held as Part of STAF 2014, York, UK, July 22-24, 2014. Proceedings. pp. 65–80 (2014)

5. Jansen, C., Katelaan, J., Matheja, C., Noll, T., Zuleger, F.: Unified reasoning about robustness properties of symbolic-heap separation logic. In: Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings. pp. 611–638 (2017)