Deciding Memory Safety for Single-Pass Heap Manipulating Programs

POPL 2020

Umang Mathur et al.

August 11, 2021

Problems Considered

Automatic verification of Heap-manipulating programs:

- Assertion checking: deacidable reasoning.
- Memory safety: no access outside the allocated locations.

$$
\begin{aligned}
\langle stmt \rangle ::= {} & \mathbf{skip} \mid x := y \mid x := y{\cdot}p \mid y{\cdot}p := x \mid a := y{\cdot}d \mid y{\cdot}d := a \mid \mathbf{alloc}(x) \mid \mathbf{free}(x) \\
& \mid a := b \mid a := f(\mathbf{c}) \mid \mathbf{assume}\,(\langle cond \rangle) \mid \langle stmt \rangle \,;\, \langle stmt \rangle \\
& \mid \mathbf{if}\,(\langle cond \rangle)\,\mathbf{then}\,\langle stmt \rangle\,\mathbf{else}\,\langle stmt \rangle \mid \mathbf{while}\,(\langle cond \rangle)\,\langle stmt \rangle \\
\langle cond \rangle ::= {} & x = y \mid a = b \mid \langle cond \rangle \vee \langle cond \rangle \mid \neg \langle cond \rangle
\end{aligned}
$$

Previous Work

[1] Umang Mathur et al. Decidable Verification of Uninterpreted Programs. POPL 2019
Coherent Program: Admit decidable verification.
For program without heap manipulating statements.

Contributions

- Assertion checking:
  - Coherence is not enough.
  - A Class of heap-manipulating program: alias-awareness.
  - Alias-awareness + Coherence: decidable and PSPACE-complete.
- Memory safety:
  - A class of heap structure.
  - Forest data-structures are alias-aware.
  - Initial forest data-structures + streaming coherence: decidable.

Uninterpreted Programs

- Programs with constants, functions and predicates that are uninterpreted.
- Signature $\Sigma = (\mathcal{C}_{\mathsf{Data}}, \mathcal{F}_{\mathsf{Data} \to \mathsf{Data}}, \mathcal{R})$
- Model $M = \langle A, \mathcal{I} \rangle$
- Interpretations are given by the models.

  Syntax:

$$
\begin{aligned}
\langle stmt \rangle ::= &\ \mathbf{skip} \mid x := c \mid x := y \mid x := f(\mathbf{z}) \mid \mathbf{assume}\,(\langle cond \rangle) \mid \langle stmt \rangle ; \langle stmt \rangle \\
&\mid \mathbf{if}\,(\langle cond \rangle)\,\mathbf{then}\,\langle stmt \rangle\,\mathbf{else}\,\langle stmt \rangle \mid \mathbf{while}\,(\langle cond \rangle)\,\langle stmt \rangle
\end{aligned}
$$

$$
\langle cond \rangle ::= x = y \mid x = c \mid c = d \mid R(\mathbf{z}) \mid \langle cond \rangle \vee \langle cond \rangle \mid \neg \langle cond \rangle
$$

Verification Problem for Uninterpreted Programs

$$P \models \phi$$

iff

1. for every model M, and
2. for every execution ρ feasible of P under the model M, ϕ holds in M at the end of ρ.

Executions and Terms

- Execution: Alphabet Π is the set of all elementary statements. The set of executions can be recursively defined by regular expression.
- Terms: Recursively defined, variables, constants, functions..

Example

```
assume(x != y);
x := f(x);
y := f(y);
x := f(x);
y := f(y);
assume(x = y);
```

Terms symbolically represent values across different models.

Coherence

Definition (Coherence)

A Execution is coherent if

- Memoizing: term computed along the execution must always be stored in some variable.
- Early assumption: making equality assumptions before forgetting superterm.

A program is coherent if all its executions are coherent.

Example

$$\pi_7 \triangleq \mathsf{u} := \mathsf{f}(\mathsf{w}) \cdot \underbrace{\mathsf{u} := \mathsf{f}(\mathsf{u}) \cdots \mathsf{u} := \mathsf{f}(\mathsf{u})}_{n} \cdot \mathsf{v} := \mathsf{f}(\mathsf{w}) \cdot \underbrace{\mathsf{v} := \mathsf{f}(\mathsf{v}) \cdots \mathsf{v} := \mathsf{f}(\mathsf{v})}_{n} \cdot \mathbf{assume}(\mathsf{u} \neq \mathsf{v})$$

Decidability of Verification Problem of Uninterpreted Program

In [1] the decidability of coherent programs is proved.

Heap Manipulating Programs

- Programs with constants, functions, pointers that are uninterpreted.
- Signature $\Sigma = (\mathcal{C}_{\text{Data}}, \mathcal{C}_{\text{Loc}}, \mathcal{F}_{\text{Loc}\to\text{Data}}, \mathcal{F}_{\text{Loc}}, \mathcal{F}_{\text{Data}})$
- Model $M = \langle U_{\text{Data}}, U_{Loc}, \mathcal{I} \rangle$
- Interpretations are given by the models.

  Syntax:

$$
\begin{aligned}
\langle \mathit{stmt} \rangle ::= {} & \textbf{skip} \mid x := y \mid x := y{\cdot}p \mid y{\cdot}p := x \mid a := y{\cdot}d \mid y{\cdot}d := a \mid \textbf{alloc}(x) \mid \textbf{free}(x) \\
& \mid a := b \mid a := f(\mathbf{c}) \mid \textbf{assume}\,(\langle \mathit{cond} \rangle) \mid \langle \mathit{stmt} \rangle\,;\,\langle \mathit{stmt} \rangle \\
& \mid \textbf{if}\,(\langle \mathit{cond} \rangle)\,\textbf{then}\,\langle \mathit{stmt} \rangle\,\textbf{else}\,\langle \mathit{stmt} \rangle \mid \textbf{while}\,(\langle \mathit{cond} \rangle)\,\langle \mathit{stmt} \rangle \\
\langle \mathit{cond} \rangle ::= {} & x = y \mid a = b \mid \langle \mathit{cond} \rangle \vee \langle \mathit{cond} \rangle \mid \neg \langle \mathit{cond} \rangle
\end{aligned}
$$

Example

```
while(x ≠ NIL){
    if(x·key = k1) then {
        alloc(z);
        z·key ← k2;
        z·next ← x·next;
        x·next ← z;
        x ← z;
    }
    x ← x·next;
}
```

key(x)

next(x)

- Pointer fields are similar to uninterpreted functions + updatable.
- Coherene and decidability?

Coherence is not enough

Counterexample:

Example

```
z1 := x.next;
assume(z1 != z2);
y.next := z2;
z3 := x.next;
assume(z2 = z3);
```

Consider `Term(z3)` under `x = y` and `x != y`.
The result of the verification depends on the aliasing information.

Alias-awareness

Definition (Alias-aware)

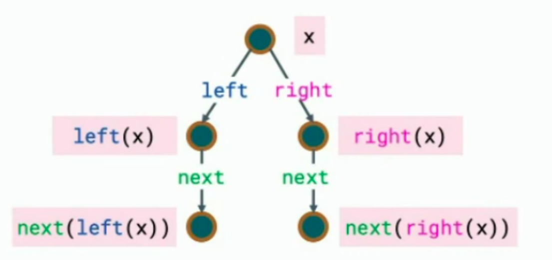
An execution ρ is called alias-aware if for every prefix of ρ that is of the form $\sigma \cdot x.h := u$, the aliasing relationship between x and all other location variables are known.

Alias-awareness + coherence: checking assertion of heap manipulating programs are decidable

Memory Safety for Heap Manipulating Programs

Alias-awareness + Coherence

Forest data-structure



Intuition: Distinct traversal starting from same/different locations are distinct.

Forest Data Structure

Definition 7 (Forest Data-structures). A heap structure $\mathcal{M} = (U_{\text{Loc}}, U_{\text{Data}}, \mathcal{I})$ over a signature Σ is said to be a forest data-structure with respect to a reachability specification $\varphi = \{\varphi_k\}_{k=1}^{n}$ if

(1) for every $1 \leq i \leq n$, each set of stopping locations is a singleton set of the form $\mathsf{Stop}_i = \{\mathsf{stop}_i\}$,

(2) for every $c \in \bigcup_{k=1}^{n} \mathsf{Stop}_k$, and for every $f \in \mathcal{F}_{\text{Loc}}$, we have that $\mathcal{I}(f(c)) = \mathcal{I}(c)$, and

(3) for every $t_i \in \mathsf{Terms}(\mathsf{Start}_i, \mathsf{Pointers}_i) \cup \mathsf{Stop}_i$ and $t_j \in \mathsf{Terms}(\mathsf{Start}_j, \mathsf{Pointers}_j) \cup \mathsf{Stop}_j$ we have, if $\mathcal{I}(t_i) = \mathcal{I}(t_j)$, then either $t_i = t_j \in \mathsf{Start}_i \cap \mathsf{Start}_j$, or $\mathcal{I}(t_i) = \mathcal{I}(t_j) = \mathcal{I}(\mathsf{stop}_i) = \mathcal{I}(\mathsf{stop}_j)$.

MemSafety: Basic Idea of Decision Procedure

- With the result in [1], we can prove execution of coherent programs are regular.
- Basic Idea: construct an automaton $\mathcal{A}_{MS}$ that its languange is exactly all the coherent executions that are memory safe.
- Check Exec$(P) \subseteq L(\mathcal{A}_{MS})$.
- which can be reduced to checking intersection of regular language.