Progess Report 3

Presentor: Xie Li

May 11, 2021

Overview

- Paper reading:
  1. Beyond Reachability: Shape Abstraction in the Presence of Pointer Arithmetic (SAS'06)
  2. Symbolic Execution with Separation Logic (APLAS'05)
- Program considered and semantic.
- Current problems and plans.

Contribution of 1

Overview: Devised an shape analysis algorithm based on the abstract interpretation framework and separation logic.

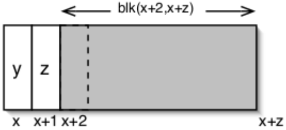
- Defined a shape analysis for programs that mutate link data structure with pointer arithmetic. (Do abstraction based on separation logic)
- A widening operator to accelerate the analysis.

Basic Ideas: Node and Multiword-linklist

$$s, h \models t_1 \mapsto t_2 \text{ iff } \exists n \in \mathbb{N}. s(t_1) = n, \mathsf{dom}(h) = \{n\} \text{ and } h(n) = s(t_2)$$

$$s, h \models \mathtt{blk}(t_1, t_2) \text{ iff}$$

Basis of abstract domain: `blk`,`nd` and `mls`.



$$\mathsf{nd}(x, y, z) \stackrel{\text{def}}{=} (x \mapsto y) * (x+1 \mapsto z) * \mathsf{blk}(x+2, x+z)$$

$$\mathsf{mls}(x, y) \stackrel{\text{def}}{=} (\exists z'.\, \mathsf{nd}(x, y, z')) \vee (\exists y', z'. \mathsf{nd}(x, y', z') * \mathsf{mls}(y', y))$$

No pointer arithmetic outside `nd`.

Abstraction: An Example

Definition of `nd`:

$$\mathsf{nd}(x, y, z) \stackrel{\text{def}}{=} (x \mapsto y) * (x+1 \mapsto z) * \mathsf{blk}(x+2, x+z)$$

Example:

$$(x \mapsto y) * (x+1 \mapsto z+b) * \mathsf{blk}(x+2, x+z)$$
$$* (x+z \mapsto a) * (x+z+1 \mapsto b) * \mathsf{blk}(x+z+2, x+z+b)$$

By the definition of `nd`:

$$(x \mapsto y) * (x+1 \mapsto z+b) * \mathsf{blk}(x+2, x+z) * \mathsf{nd}(x+z, a, b)$$

A true implication:

$$(x \mapsto y) * (x+1 \mapsto z+b) * \mathsf{blk}(x+2, x+z) * \mathsf{nd}(x+z, a, b) \Longrightarrow \mathsf{nd}(x, y, z+b)$$

Difficulty: information lost.

Program Considered

$$
\begin{aligned}
e &::= n \mid x \mid e{+}e \mid e{-}e \\
B &::= e{=}e \mid e{\neq}e \mid e{\leq}e \\
S &::= x{:=}e \mid x{:=}[e] \mid [e]{:=}e \mid x{:=}\mathsf{sbrk}(e) \\
C &::= S \mid C\,;C \mid \mathtt{if}(B)\ \{C\}\ \mathtt{else}\ \{C\} \mid \mathtt{while}(B)\{C\} \mid \mathtt{local}\ x\,;C
\end{aligned}
$$

Concrete states:

$\mathsf{States} \stackrel{\text{def}}{=} \mathsf{Stacks} \times \mathsf{Heaps} \qquad \mathsf{Stacks} \stackrel{\text{def}}{=} \mathsf{Vars} \rightarrow \mathsf{Ints} \qquad \mathsf{Heaps} \stackrel{\text{def}}{=} \mathsf{Nats}^{+} \rightharpoonup_{\text{fin}} \mathsf{Ints}$

Table 1. Symbolic Heaps

$E, F ::= n \mid x \mid x' \mid E{+}E \mid E{-}E$	$H ::= E{\mapsto}E \mid \mathsf{blk}(E,E) \mid \mathsf{nd}(E,E,E)$
$P ::= E{=}E \mid E{\neq}E \mid E{\leq}E \mid \mathsf{true}$	$\mid \mathsf{mls}(E,E) \mid \mathsf{true} \mid \mathsf{emp}$
$\Pi ::= P \mid \Pi \wedge \Pi$	$\Sigma ::= H \mid \Sigma * \Sigma$
	$Q ::= \Pi \wedge \Sigma$

Basic Settings

Symbolic heap: Q, which contains primed variables.
Let SH be the set of all symbolic heap.
Abstract domain $\mathcal{D}$:
$\mathcal{S} \in \mathcal{D} \stackrel{\text{def}}{=} \mathcal{P}_{\text{fin}}(\mathsf{SH}) \cup \{\top\}$

Concretization:
The concretization $\gamma(Q)$ of Q is the set of concrete states satisfying $\exists \vec{y'}.Q$.
The concretization of $\mathcal{S}$:

$$\gamma(\mathcal{S}) \stackrel{\text{def}}{=} \textbf{if } (\mathcal{S} \neq \top) \textbf{ then } \left(\bigcup_{Q \in \mathcal{S}} \gamma(Q)\right) \textbf{ else } (\mathsf{States} \cup \{\text{fault}\})$$

Elements in $\mathcal{D}$ are ordered by subset relation:

$$\mathcal{S} \sqsubseteq \mathcal{S}' \iff (\mathcal{S}' = \top \vee (\mathcal{S} \in \mathcal{P}(\mathsf{SH}) \wedge \mathcal{S}' \in \mathcal{P}(\mathsf{SH}) \wedge \mathcal{S} \subseteq \mathcal{S}'))$$

Abstraction Function

The abstraction function $\texttt{Abs} : \mathcal{D} \to \mathcal{D}$ contains five phases:

- Synthesize node from RAM configurations.
- Simplify arithmetic expressions from explosion of arithmetic constraints.
- Abstract size field.
- Reason about multiword list.
- Filter out inconsistent symbolic heaps.

Abstraction Rules: Node Synthesis

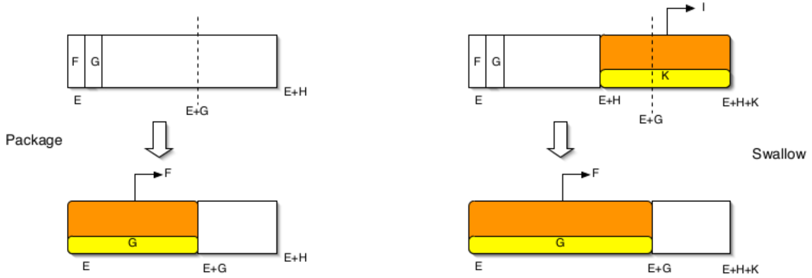
Table 2 Node Synthesis Rules

Package Rule	Swallow Rule
Precondition: $2 \le G \le H$	Precondition: $H+1 \le G \le H+K$
$Q * (E \mapsto F, G) * \mathsf{blk}(E+2, E+H)$ $\Rightarrow Q * \mathsf{nd}(E, F, G) * \mathsf{blk}(E+G, E+H)$	$Q * (E \mapsto F, G) * \mathsf{blk}(E+2, E+H) * \mathsf{nd}(E+H, I, K)$ $\Rightarrow Q * \mathsf{nd}(E, F, G) * \mathsf{blk}(E+G, E+H+K)$

Package2 Rule

Precondition: $2 \le G \le H$ with x' fresh

$$Q * \mathsf{blk}(E, E+1) * (E+1 \mapsto G) * \mathsf{blk}(E+2, E+H) \Rightarrow Q * \mathsf{nd}(E, x', G) * \mathsf{blk}(E+G, E+H)$$



Abstraction Rules: Arithmetic Simplification

A symbolic heap $Q \equiv \Pi \wedge \Sigma$ is in *n-simple form* iff:

1. Q contains only n-simple expressions.
2. $\mathit{\Pi}$ does not contain any primed variables.
3. $\mathit{\Pi} \equiv x_1{=}N_1 \wedge \ldots \wedge x_k{=}N_k \wedge 0{\leq}M_1 \wedge \ldots \wedge 0{\leq}M_l$ where all x_i's are distinct variables that occur in Q only in the left of equation $x_i{=}N_i$.

where

$$N, M ::= x_1{+}\cdots{+}x_k{-}y_1{-}\cdots{-}y_l{+}m$$

Abstraction Rules: Arithmetic Simplification

Table 3 Rules for Transforming Symbolic Heaps to n-Simple Form

Substitution1 Rule	Substitution2 Rule
$x{=}E \wedge Q \;\Rightarrow\; x{=}E \wedge (Q[E/x])$	$x'{=}E \wedge Q \;\Rightarrow\; Q[E/x']$
(if $x{=}E$ is n-simple and $x{\in}\mathsf{fv}(Q)$)	(if $x'{=}E$ is n-simple and $x'{\in}\mathsf{fv}(Q)$)
Merge Rule	Simplify Rule
$E{\neq}0 \wedge 0{\leq}E \wedge Q \;\Rightarrow\; 0 \leq E{-}1 \wedge Q$	$Q[E/y'] \;\Rightarrow\; Q[x'/y']$
	(if E is not n-simple, $y'{\in}\mathsf{fv}(Q)$ and $x'{\notin}\mathsf{fv}(Q, E)$)
Drop Rule	
$P \wedge Q \;\Rightarrow\; Q$ (if atomic predicate P is not n-simple, or it contains some primed x')	

Abstraction Rules: Abstract Size Field

Size Rule

$$Q * \mathsf{nd}(E, F, x') \;\Rightarrow\; Q * \mathsf{nd}(E, F, y') \quad (\text{if } x' \in \mathsf{fv}(Q, E, F) \text{ but } y' \notin \mathsf{fv}(Q, E, F))$$

Abstraction Rules: Multilist Rules

Table 4 Rules for Multiword-List Abstraction

Notation: $L(E,F) ::= \mathsf{mls}(E,F) \mid \mathsf{nd}(E,F,H)$ $\quad U(E,F) ::= \mathsf{blk}(E,F) \mid E \mapsto F$

Append Rule

$$Q * L_0(E,x') * L_1(x',G) \;\Rightarrow\; Q * \mathsf{mls}(E,G)$$

$$(\text{if } x' \notin \mathsf{fv}(Q,G) \text{ and } (L_0 \equiv \mathsf{nd}(E,x',F) \;\Rightarrow\; E \text{ or } F \text{ is a primed variable}))$$

Forget1 Rule

$$Q * \mathsf{blk}(E,E) \;\Rightarrow\; Q * \mathsf{emp}$$

Forget2 Rule

$$Q * L(x',E) \;\Rightarrow\; Q * \mathsf{true}$$

$$(\text{if } x' \notin \mathsf{fv}(Q))$$

Forget3 Rule

$$Q * U(E,F) \;\Rightarrow\; Q * \mathsf{true}$$

Abstraction Rules: Filter Inconsistency

$$\mathcal{S}' \stackrel{\text{def}}{=} \textbf{if } (\mathcal{S} = \top) \textbf{ then } \top \textbf{ else } \{Q \in \mathcal{S} \mid Q \nvdash \mathsf{false}\}.$$

n-Canonical Form

Definition 1 (n-Canonical Form). *A symbolic heap Q is n-canonical iff*

1. *it is n-simple and $Q \not\vdash \mathsf{false}$,*
2. *it contains neither* blk *nor* $\mapsto$*,*
3. *if x' occurs left in Q, it is either shared or directly pointed to, and*
4. *if x' occurs as size of a node predicate in Q, it occurs only once in Q.*

Use $\mathcal{C}_n$ as the set of n-canonical form in $\mathcal{D}$.

Widening Operator

The function `rep`

$$(\forall Q, Q' \in \mathsf{rep}(\mathcal{S}).\, Q \vdash Q' \;\Rightarrow\; Q = Q') \quad \wedge \quad (\forall Q \in \mathcal{S}.\, \exists Q' \in \mathsf{rep}(\mathcal{S}).\, Q \vdash Q')$$

and the widening operator is given by:

$$\mathcal{S} \nabla \mathcal{S}' = \begin{cases} \mathcal{S} \cup \{Q' \in \mathsf{rep}(\mathcal{S}') \mid \neg(\exists Q \in \mathcal{S}.\, Q' \vdash Q)\} & \text{if } \mathcal{S} \neq \top \text{ and } \mathcal{S}' \neq \top \\ \top & \text{otherwise} \end{cases}$$

Proposition 5. *The ∇ operator satisfies the following two axioms:*

1. *For all $\mathcal{S}, \mathcal{S}' \in \mathcal{C}_n$, we have that $\gamma(\mathcal{S}) \cup \gamma(\mathcal{S}') \subseteq \gamma(\mathcal{S} \nabla \mathcal{S}')$.*
2. *For every infinite sequence $\{\mathcal{S}'_i\}_{i \geq 0}$ in $\mathcal{C}_n$, the widened sequence $\mathcal{S}_0 = \mathcal{S}'_0$ and $\mathcal{S}_{i+1} = \mathcal{S}_i \nabla \mathcal{S}'_{i+1}$ converges.*

Abstract Semantic

Table 5 Abstract Semantics

Let $A[e]$ and A be syntactic subclasses of atomic commands defined by:

$$A[e] ::= [e]{:=}\,e \mid x{:=}\,[e] \qquad A ::= x{:=}\,e \mid x{:=}\,\mathsf{sbrk}(e).$$

The abstract semantics $[\![C]\!]\colon \mathcal{D} \to \mathcal{D}$ is defined as follows:

$$
\begin{aligned}
[\![C_0\,;C_1]\!]\mathcal{S} &= ([\![C_1]\!] \circ [\![C_0]\!])\mathcal{S}\\
[\![\mathtt{if}(B)\ \{C_0\}\ \mathtt{else}\ \{C_1\}]\!]\mathcal{S} &= ([\![C_0]\!] \circ \mathsf{filter}(B))\mathcal{S} \sqcup ([\![C_1]\!] \circ \mathsf{filter}(\neg B))\mathcal{S}\\
[\![\mathtt{local}\ x\,; C]\!]\mathcal{S} &= \textbf{if}\,([\![C]\!](\mathcal{S}[y'/x]) = \top)\ \textbf{then}\ \top\ \textbf{else}\ ([\![C]\!](\mathcal{S}[y'/x]))[x'/x]\\
[\![\mathtt{while}(B)\{C\}]\!]\mathcal{S} &= (\mathsf{filter}(\neg B) \circ \mathit{wfix})(\mathcal{S}_0, F)\\
&\qquad (\text{where } \mathcal{S}_0 = \mathsf{Abs}(\mathcal{S}) \text{ and } F = \mathsf{Abs} \circ [\![C]\!] \circ \mathsf{filter}(B))\\
[\![A[e]]\!]\mathcal{S} &= \textbf{if}\ (\mathcal{S} = \top\ \vee\ \exists Q \in \mathcal{S}.\, Q \rightsquigarrow_e^* \mathrm{fault})\ \ \textbf{then}\ \top\\
&\qquad \textbf{else}\ \{Q_1 \mid Q \in \mathcal{S}\ \wedge\ Q \rightsquigarrow_e^* Q_0\ \wedge\ (Q_0, A[e] \Longrightarrow Q_1)\}\\
[\![A]\!]\mathcal{S} &= \textbf{if}\ (\mathcal{S} = \top)\ \textbf{then}\ \top\ \textbf{else}\ \{Q_0 \mid Q \in \mathcal{S}\ \wedge\ (Q, A \Longrightarrow Q_0)\}
\end{aligned}
$$

where primed variables are assumed fresh, and $\mathsf{filter}\colon \mathcal{D} \to \mathcal{D}$ and $-\colon \mathcal{C}_n \times \mathcal{C}_n \to \mathcal{C}_n$ and $\mathit{wfix}\colon \mathcal{C}_n \times [\mathcal{C}_n \to \mathcal{C}_n] \to \mathcal{C}_n$ are functions defined below:

$$\mathsf{filter}(B)(\mathcal{S}) = \textbf{if}\ (\mathcal{S}{=}\top)\ \textbf{then}\ \top\ \textbf{else}\ \{B \wedge Q \mid Q \in \mathcal{S}\ \text{ and }\ (B \wedge Q \nvdash \mathsf{false})\}.$$

$$\mathcal{S}_0 - \mathcal{S}_1 = \textbf{if}\ (\mathcal{S}_0{\neq}\top \wedge \mathcal{S}_1{\neq}\top)\ \textbf{then}\ (\mathcal{S}_0 - \mathcal{S}_1)\ \textbf{else}\ \Big(\textbf{if}\ (\mathcal{S}_1{=}\top)\ \textbf{then}\ \emptyset\ \textbf{else}\ \top\Big)$$

$\mathit{wfix}(\mathcal{S}, F)$ is the first stabilizing element $\mathcal{S}_k$ of the below sequence $\{\mathcal{S}_i\}_{i\geq 0}$:

$$\mathcal{S}_0 = \mathcal{S} \qquad \mathcal{S}_1 = \mathcal{S}_0 \nabla F(\mathcal{S}) \qquad \mathcal{S}_{i+2} = \mathcal{S}_{i+1} \nabla (F(\mathcal{S}_{i+1} - \mathcal{S}_i)).$$

Soundness

Proposition 6. *Suppose that* $[\![C]\!]\mathcal{S} = \mathcal{S}'$. *If both* $\mathcal{S}$ *and* $\mathcal{S}'$ *are non-*$\top$ *abstract values, then there is a proof of a Hoare triple* $\{\mathcal{S}\}C\{\mathcal{S}'\}$ *in separation logic.*