A Local Shape Analysis based on Separation Logic

Authors: Dino Distefano, Peter W. O'Heran and Hongseok Yang
Presentor: Xie Li

TACAS 2006

Introduction

A **Local Shape Analysis** based on **Separation Logic**

- What is **Shape Analysis**?
- What is **Separation Logic**?
- How the analysis works?
- What does **Local** means?

Shape Analysis

- Questions in heap content: NULL-pointers, May-Alias, Must-Alias, Reachability, Disjointness, **Shape**.
- **Shapes** characterize data structures: singly linked list, linked list with cycle, doubly linked list, a binary tree...
- According to [1], shape analysis computes for each point in the program:
  A finite, conservative representation of the heap-allocated data structures that could arise when a path to this program point executed.

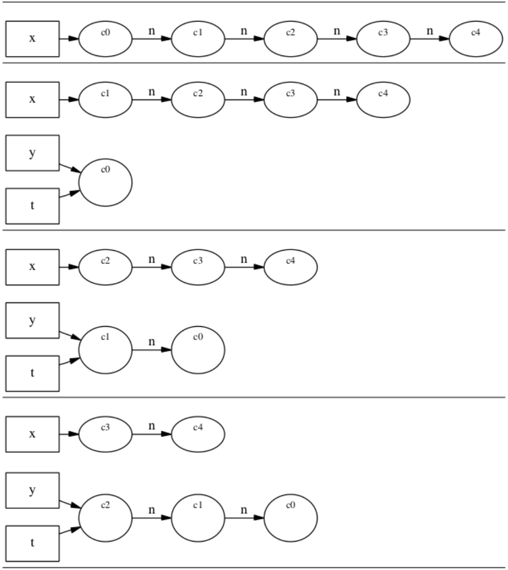
[1] Reinhard Wilheim et al. Shape Analysis. CC 2000.

Shape Analysis: Example

Example

```
List reverse(List x){
    List y, t;
    y = NULL;
    while(x != NULL){
        t = y;
        y = x;
        x = x->n;
        y->n = NULL;
        y->n = t;
    }
    return y;
}
```

Execution state: cells, connectivity and values of pointer variables.



Idea of the Paper

- Problem of classic shape analysis: updating of a abstract location may affect properties for other cells.
- Separation logic formula is also capable to express the configuration of memory.
- Utilize existing symbolic execution method for separation logic[2].

[2] Josh Berdine, Cristiano Calcagno and Peter W. O'Hearn. Symbolic execution with separation logic.

Program Considered

Syntax:

$$b ::= E{=}E \mid E{\neq}E$$
$$p ::= x{:=}\,E \mid x{:=}\,[E] \mid [E]{:=}\,F \mid \mathbf{new}(x) \mid \mathbf{dispose}(E)$$
$$c ::= p \mid c\,;c \mid \mathbf{while}\ b\ \mathbf{do}\ c \mid \mathbf{if}\ b\ \mathbf{then}\ c\ \mathbf{else}\ c$$

Example

$$\begin{array}{l}\mathbf{while}(c \neq 0)\mathbf{do}\{\\ t := c;\\ c := c \rightarrow tl;\\ \mathbf{dispose}(t);\\ \}\end{array}$$

Concrete State Semantic

$$\mathsf{Values} = \mathsf{Locations} \cup \{\mathsf{nil}\} \qquad \mathsf{Heaps} = \mathsf{Locations} \rightharpoonup_f \mathsf{Values}$$
$$\mathsf{Stacks} = (\mathsf{Vars} \cup \mathsf{Vars}') \to \mathsf{Values} \qquad \mathsf{States} = \mathsf{Stacks} \times \mathsf{Heaps}$$

Graphically speaking,

We use S to denote the set `States`.

Concrete Execution Semantic

Primitive commands: $x := E, x := [E], [E] := F, \mathbf{new}(x), \mathbf{dispose}(E)$.
Assume the locations and values are all non-negative integers.

$$(s, h), p \Longrightarrow (s', h')$$

```
int main(){

    x = 1;

    new(y);


    [y]:=x;


    dispose(y);

}
```

```
int main(){

    new(x)

    if(x = y){

        z := a;

    } else {
        z := b;
    }
    dispose(x);
}
```

$$\mathcal{C}[\![c]\!] : \mathcal{P}(S) \rightarrow \mathcal{P}(S)$$

Separation Logic: Symbolic Heap

Pure Spatial

$$\Pi \mid \Sigma$$

$$\exists x'_1 x'_2 \ldots x'_n . \left(\bigwedge_{P \in \Pi} P\right) \wedge \left(\bigstar_{Q \in \Sigma} Q\right)$$

$\mathcal{SH}$

Separation Logic: Semantic of Symbolic Heaps

Example

- $s, h \models E \mapsto F$:
  $s_0(x) = 1, s_0(y) = 10$ and $h_0(1) = 10$.
  Then $s_0, h_0 \models x \mapsto y$.

- $s, h \models \texttt{ls}(E, F)$: intuitively, this means we have a path from E to F.
  - $s_0(x) = 1, s_0(y) = 10$ and $h_0(1) = 10$. Then $s_0, h_0 \models \texttt{ls}(x, y)$
  - $s_1(x) = 1, s_1(y) = 2, s_1(z) = 3, s_1(w) = 4$ and $h_1(1) = 2, h_1(2) = 3, h_1(3) = 4$.
    Then $s_1, h_1 \models \texttt{ls}(x, w)$. Or $s_1, h_1 \models x \mapsto y * \texttt{ls}(y, w)$.

- $s, h \models \texttt{junk}$ iff $h \neq \emptyset$

Symbolic Execution Semantic

Primitive commands: $x := E, x := [E], [E] := F, \mathbf{new}(x), \mathbf{dispose}(E)$.

Assume the locations and values are all non-negative integers.

$$\Pi \mid \Sigma, p \Longrightarrow \Pi' \mid \Sigma'$$

```
int main(){
    // true|emp
    x = 1;
    new(y);
    [y]:=x;
    dispose(y);
}
```

```
int main(){
    new(x)
    if(x = y){
        z := a;
    } else {
        z := b;
    }
    dispose(x);
}
```

$$\mathcal{I}[\![c]\!] : \mathcal{P}(\mathcal{SH}) \to \mathcal{P}(\mathcal{SH})$$

Concretization

The link between concrete semantic and symbolic semantic:

$$\gamma : \mathcal{P}(\mathcal{SH}) \rightarrow \mathcal{P}(\texttt{States})$$

Theorem

The symbolic semantics is a sound overapproximation of the concrete semantics:

$$\forall X \in \mathcal{P}(\mathcal{SH}).\mathcal{C}[\![c]\!](\gamma(X)) \subseteq \gamma(\mathcal{I}[\![c]\!]X)$$

General Semantic Setting

Working with complete lattice: D.
D is constructed from $\mathcal{P}(S')$, where $S' = S \cup \{\top\}$. $\top$ is a special element corresponds to memory fault.
semantic of a command $[\![c]\!] : D \to D$.
Semantic for key commands:

$$[\![c\,;c']\!] = [\![c]\!]\;;[\![c']\!] \qquad [\![\textbf{if } b \textbf{ then } c \textbf{ else } c']\!] = (\mathsf{filter}(b)\;;[\![c]\!]) \sqcup (\mathsf{filter}(\neg b)\;;[\![c']\!])$$

$$[\![\textbf{while } b \textbf{ do } c]\!] = \lambda d.\ \mathsf{filter}(\neg b)\left(\mathit{fix}\ \lambda d'.\ d \sqcup (\mathsf{filter}(b)\;;[\![c]\!])(d')\right)$$

where $\texttt{filter}(b) : D \to D$.
Execution semantic:
$p \Longrightarrow \subseteq S \times (S \cup \{\top\})$
The execution semantic on the powerset of S' is a function $\bar{p} : \mathcal{P}(S') \to \mathcal{P}(S')$:

$$\bar{p}X = \{\sigma' \mid \exists \sigma \in X.(\sigma, p \Longrightarrow \sigma') \vee (\sigma = \sigma' = \top)\}$$

Problem Encountered

$\mathcal{SH}$ is an infinite set.

Define an abstract domain for the fix-point convergence and abstraction rules for the conversion.

- Expression replacement for primed variables.
- Garbage collection rules.
- List abstraction rules.

The conversion is given by $\rightsquigarrow$. The abstract domain after the abstraction is $\mathcal{CSH}$ and corresponding abstract semantic function $\mathcal{A}[\![c]\!] : \mathcal{P}(\mathcal{CSH}) \rightarrow \mathcal{P}(\mathcal{CSH})$.

The Analysis

Theorem

$\mathcal{CSH}$ is finite.

Theorem

The abstract semantic is a sound overapproximation of the concrete semantic.

$$\forall X \in \mathcal{P}(\mathcal{SH}).\mathcal{C}[\![c]\!](\gamma(X)) \subseteq \gamma(\mathcal{A}[\![c]\!]X)$$

The Analysis

Example

A program to dispose a list.

$\texttt{Program}$: **while** $(c \neq 0)$ **do** $(t := c; c := c \rightarrow tl; \mathbf{dispose(t)})$

$\texttt{Pre:}$ $\{\} \mid \{\texttt{ls}(c, 0)\}$

$\texttt{Post:}$ $\{c = 0\} \mid \{\}$

$\texttt{Inv:}$ $\{c = 0\} \mid \{\} \vee \{\} \mid \{\texttt{ls}(c, 0)\}$

Locality

Theorem (Frame Rule)

For all $X, Y \in \mathcal{P}(\mathcal{CSH})$, *if* $\mathtt{Vars}(Y) \cap Mod(X) = \emptyset$ *then* $\gamma(\mathcal{A}[\![c]\!](X * Y)) \subseteq \gamma(\mathcal{A}[\![c]\!]X) * Y$

Conclusion

- Concrete semantic.
- Symbolic semantic. (Symbolic heap)
- Abstract semantic. (Canonical symbolic heap)
- Locality.

What can be done with the sound over-approximation?