SNU 4541.664A Program Analysis
Spring 2005
Note 13

Prof. Kwangkeun Yi

요약해석으로 디자인 한 실제 분석기의 사례

Airac: C 프로그램의 배열 인덱스 오류 분석기

C 프로그램의 배열 인덱스 오류

```
int *c = (int *)malloc(sizeof(int)*10);
c[i] = 1; c[i+f()] = 1; c[*k + (*g)()] = 1;
x = c; x[1] = 1;
y = c+f(); y[i] = 1;
z->a = c; (z->a)[i] = 1;
foo(c+2); int foo(int *d) { … d[i] = 1; …}
```

의미구조: 상태 전이

$$\begin{array}{rcl} \mathit{Pointer} & = & \mathit{BaseAddr} \times \mathit{Size} \times \mathit{Offset} \\ \mathit{Machine} & = & \mathit{Stack} \times \mathit{Env} \times \mathit{Mem} \times \mathit{Cmd} \times \mathit{Dump} \end{array}$$

For program "$dec^+\ e$", its semantics is $\mathit{lfp} F$

$$\begin{array}{l} F : 2^{(\mathit{Machine}^\omega)} \rightarrow 2^{(\mathit{Machine}^\omega)} \\ F(X) \quad = \quad \{\langle \emptyset, \emptyset, \emptyset, dec^+\ e, \emptyset \rangle\} \\ \qquad\qquad\quad \cup \{ s_0 s_1 \ldots s_{n+1} | s_0 s_1 \ldots s_n \in X, s_n \rightarrow s_{n+1} \} \end{array}$$

The transition relation $\rightarrow$ is defined for each C construct.

요약된 의미구조: 요약 상태 전이

$$
\begin{array}{rcl}
2^{Pointer} & \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} & 2^{\hat{Pointer}} \\
\hat{Pointer} & = & AllocCite \times \hat{\mathbb{Z}} \times \hat{\mathbb{Z}} \\
\hat{\mathbb{Z}} & = & \{\bot\} \cup \{[a,b] \mid a, b \in \mathbb{Z} \cup \{-\infty, \infty\}, a \leq b\} \\
\alpha P & = & \{\alpha' p \mid p \in P\} \\
\alpha' \langle a, s, o \rangle & = & \langle \ell, [s,s], [o,o] \rangle \quad a \in \textsf{allocated-at}(\ell)
\end{array}
$$

$$\hat{Machine} = \hat{Stack} \times \hat{Mem} \times Cmd \times \hat{Dump}$$

For program $dec^+\ e$, its abstract semantics is $lfp\,\hat{F}$:

$$\begin{array}{l} \hat{F} : 2^{(\hat{Machine}^{\omega})} \to 2^{(\hat{Machine}^{\omega})} \\ \begin{array}{rcl} \hat{F}(X) & = & \{\langle\perp, \perp, \perp, dec^+\ e, \perp\rangle\} \\ & & \cup\{s_0 s_1 \ldots s_{n+1} | s_0 s_1 \ldots s_n \in X, s_n \to^{\#} s_{n+1}\} \end{array} \end{array}$$

The abstract transition relation $\to^{\#}$ is defined for each C construct.

고정점 알고리즘: 요약 상태 전이의 계산

프로그램의 요약 의미:

$$
\begin{aligned}
&\{\langle l_0, X_0\rangle \to^{\#} \cdots \to^{\#} \langle l_n, X_n\rangle \to^{\#} \langle l, Y\rangle \to^{\#} \cdots\ ,\\
&\ \langle l_0, X_0\rangle \to^{\#} \cdots \to^{\#} \langle l_n, X_n\rangle \to^{\#} \langle l', Y'\rangle \to^{\#} \cdots\ ,\\
&\ \cdots\}
\end{aligned}
$$

- The equations that we solve are about the abstract program states $T(l \to l')$ at each flow edge $l \to l'$.
- A flow edge $l \to l'$ is between two program points l and l' that are linked by the evaluation:

$$\langle l, X\rangle \to^{\#} \langle l', X'\rangle.$$

- Suppose there are two edges $l_1 \to l$ and $l_2 \to l$ flowing into l. The equation for edge $l \to l'$ is

$$T(l \to l') = X \quad \textit{where} \quad \langle l, T(l_1 \to l) \sqcup T(l_2 \to l)\rangle \to^{\#} \langle l', X\rangle.$$

- The fixpoint algorithm is a working set algorithm.
  - The working set consists of equations whose right-hand-side we have to re-evaluate.
  - On behalf of the equation for $T(l \to l')$, we only use the program point l for the working set element.
  - When a computed machine state for $T(l \to l')$ is moved, we add the next program point l' to the working set.
- The fixpoint algorithm consists of two parts: widening iterations followed by narrowing iterations.

분석 정확도 향상을 위해 적용된 기술들

- Unique renaming: variable names are used for abstract locations
- Narrowing after widening
- Context pruning (backward analysis): precise information is extracted from conditional expressions of branch expression.
- Polyvariant analysis: function-inlining effect by labeling function-body expressions uniquely to each call-site.
- Static loop unrolling: loop-unrolling effect by labeling loop-body expressions uniquely to each iteration.

분석의 속도 향상을 위해 적용된 기술들

- Selective join: the abstract machine join (or the partial order operation) consider only those parts that have been moved.
- Stack obviation: the abstract machine's stack component is not used when joining abstract machines.
- Wait-at-join: a way of controlling the order of selecting things to do from the worklist.