Automatic Complexity Analysis of Programs

Florian Frohn
Supervisor: Jürgen Giesl

Term Rewrite System (TRS) $\mathcal{R}$

\[
\begin{align*}
    \text{plus}(s(x), y) & \rightarrow \text{plus}(x, s(y)) \\
    \text{plus}(0, y) & \rightarrow y
\end{align*}
\]

Integer Transition System (ITS) $\mathcal{I}$

\[
\begin{align*}
    \text{plus}(x, y) & \rightarrow \text{plus}(x - 1, y + 1) \mid x > 0 \\
    \text{plus}(x, y) & \rightarrow y \mid x = 0
\end{align*}
\]

Java Program $\mathcal{J}$

```java
int plus(int x, int y) {
    while (x > 0) {
        x--; y++;
    }
    return y;
}
```

Java $\rightarrow$ ITS

- execute program symbolically
- symbolically evaluation graph
- translate graph to ITS
- apply techniques for ITSs
- translates $\mathcal{J}$ to $\mathcal{I}$

$\mathcal{O}(x)$

iFM '17

Recursive ITS $\rightarrow$ ITS

- analyze non-recursive sub-ITSs independently
- approximate runtime and size of result
- replace calls to analyzed program parts with obtained bounds
- currently restricted to $N$

FroCoS '17

Strategy Switching

- goal: analyze complexity w.r.t. unrestricted evaluation strategy
- prove that eager evaluation is worst
- apply techniques for eager evaluation
- eager evaluation is worst for plus

$\mathcal{O}(x)$

FroCoS '17

Loop Detection

- syntactic criterion for linear & exponential lower bounds
- $\text{plus}(s(x), y) \rightarrow \text{plus}(x, s(y))$ is a decreasing loop

$\mathcal{O}(x)$

JAR '17

Automated Induction Technique

- automatically "guess" conjecture
- $\text{plus}(\varepsilon(0), \varepsilon(0)) \rightarrow \varepsilon^{x+1}(0)$
- prove it via induction
- infer lower bound from proof

$\mathcal{O}(n)$

RTA '15

Applications

- DARPA STAC program [4]
- used to find or prove the absence of DoS vulnerabilities in Java programs

Space/Time Analysis for Cybersecurity (STAC) – https://www.darpa.mil/program/space-time-analysis-for-cybersecurity