AProVE (KoAT + LoAT)

Nils Lommen, Florian Frohn, and Jürgen Giesl

Motivation

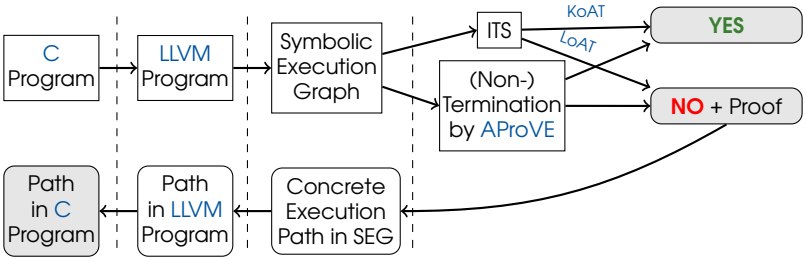
Goal: Prove or disprove termination of C programs

```
while (x3 > 0) do
  [x1; x2] ← [x4; x5^2]
  while (x1 < x2 ∧ x1 > 0) do
    [x1; x2] ← [3 · x1; 2 · x2]
  end
  [x3] ← [x3 − 1]
end
```

- Does this program terminate?

General Architecture

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- Programs are transformed into *Integer Transition Systems (ITSs)*
- ITSs are analyzed by our tools KoAT and LoAT



- KoAT – Ranking Functions
- KoAT – TWN-Loops

KoAT – Proving Termination via Ranking Functions

- Loops: `while (`φ`) do` η `end`
- Linear Ranking Functions
  - Search $\mathbf{a} \in \mathbb{Z}^{d+1}$ such that $f(\mathbf{a}, \mathbf{x}) = a_0 + a_1 x_1 + \cdots + a_d x_d$ yields well-founded order on $\mathbb{N}$
  - Decreasing:
    $$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f(\mathbf{a}, \mathbf{x}) \geq f(\mathbf{a}, \eta(\mathbf{x})) + 1$$
  - Boundedness:
    $$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f(\mathbf{a}, \mathbf{x}) > 0$$

```
while (x_1 > 0)
  [x_1] ← [x_1 − 1]
end
```

KoAT – Proving Termination via Ranking Functions

- Loops: `while` (φ) `do` η `end`
- Multiphase-Linear Ranking Functions (Ben-Amram, Genaim) & (RH '22)
  - Search $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{Z}^{d+1}$ such that $f_1(\mathbf{a}_1, \mathbf{x})$, $\ldots, f_k(\mathbf{a}_k, \mathbf{x})$ yields well-founded order on $\mathbb{N}$
  - Decreasing:

$$\forall \mathbf{x} \in \mathbb{Z}^d. \qquad \varphi \to f_1(\mathbf{a}_1, \mathbf{x}) \geq f_1(\mathbf{a}_1, \eta(\mathbf{x})) + 1$$

$$\forall \mathbf{x} \in \mathbb{Z}^d.\ \forall i \in \{2, \ldots, k\}.\ \varphi \to f_i(\mathbf{a}_i, \mathbf{x}) + f_{i-1}(\mathbf{a}_{i-1}, \mathbf{x}) \geq f_i(\mathbf{a}_i, \eta(\mathbf{x})) + 1$$

  - Boundedness:

$$\forall \mathbf{x} \in \mathbb{Z}^d.\ \varphi \to f_k(\mathbf{a}_k, \mathbf{x}) > 0$$

```
while (x1 > 0)
  [x1]   [x1 + x2]
  [x2] ← [x2 - 1 ]
end
```

KoAT – Termination of TWN-Loops

Does the loop terminate?

$$\texttt{while } (x_1 < x_2 \wedge x_1 > 0) \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 3 \cdot x_1 \\ 2 \cdot x_2 \end{bmatrix} \quad \texttt{end}$$

- Yes!
- Value of x_1 eventually *outgrows* value of x_2
- At some point we always have

$$3^n \cdot e_1 - 2^n \cdot e_2 \geq 0.$$

- Reduce Termination to an existential formula over $\mathbb{Z}$ (SAS '20)
  - linear arithmetic: co-NP-complete
  - non-linear arithmetic: non-termination is semi-decidable

Results of SV-Comp 2025

Termination:

- Proton: 3685 points
- UAutomizer: 3334 points
- AProVE (KoAT + LoAT): 2219 points

Observations:

- AProVE (KoAT + LoAT) lags behind in categories *bit-vectors* and *other*
- witness missing (false(termination))
- floating points cannot be handled by AProVE (KoAT + LoAT)
- investigate results in comparison to AProVE 2022

Challenges & Open Problems

- Challenges we came across:
  - familiarize with AProVE
  - familiarize with rules of SV-Comp
  - many different repositories must be adapted
  - many different deadlines

- Improvements of AProVE (KoAT + LoAT)
  - use newer clang and LLVM version
  - improve certificate generation for non-termination
  - handle floating points
  - many more ...

Conclusion

- AProVE (KoAT + LoAT) is a framework to analyze termination of C Programs
- ITSs are analyzed by our tools KoAT and LoAT
  - KoAT – Ranking Functions
  - KoAT – TWN-Loops
- **Many** possible ways to improve AProVE (KoAT + LoAT)

Check out our *poster* or *tool demo session* (Wednesday 2pm)

`https://koat.verify.rwth-aachen.de/svcomp25`

Thank You!