Fifteen Years of VIPER

Marco Eilers
Malte Schwerhoff
Alexander J. Summers
Peter Müller

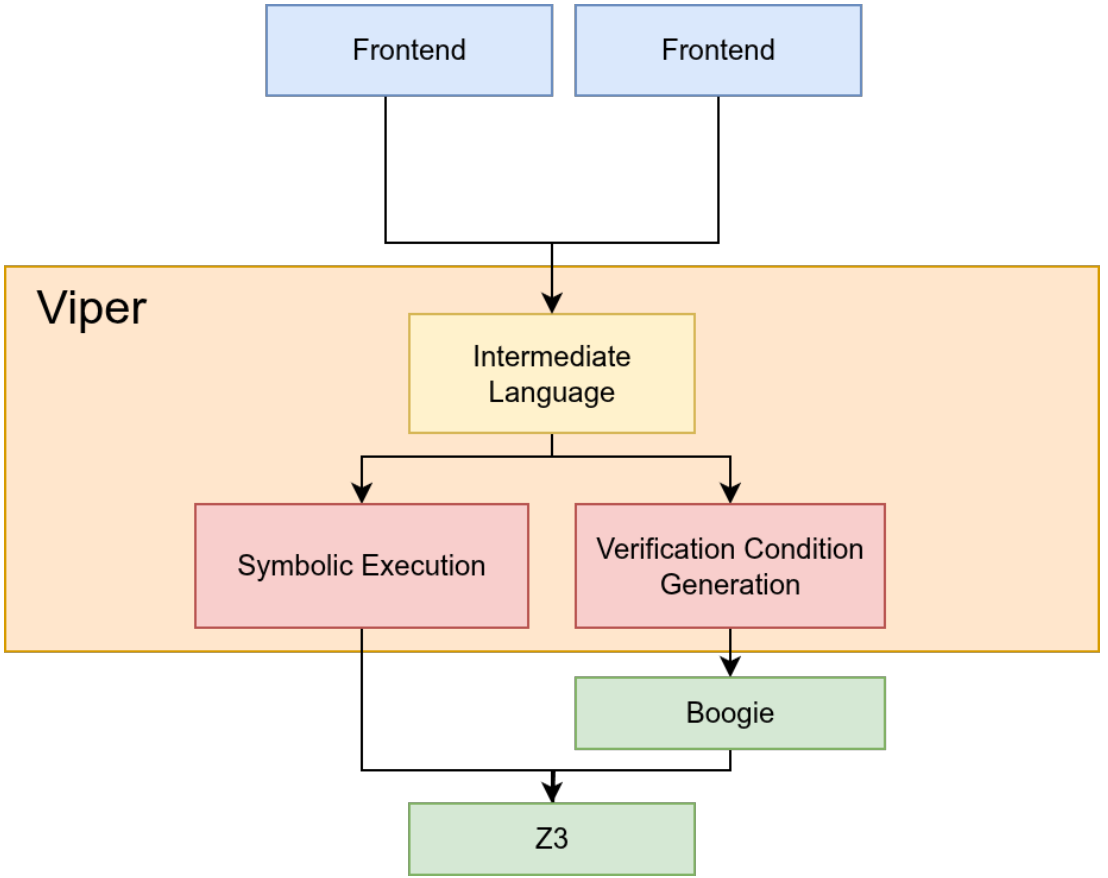
ETH zürich

THE UNIVERSITY OF BRITISH COLUMBIA

Verification
Infrastructure for
PERmission-Based Reasoning

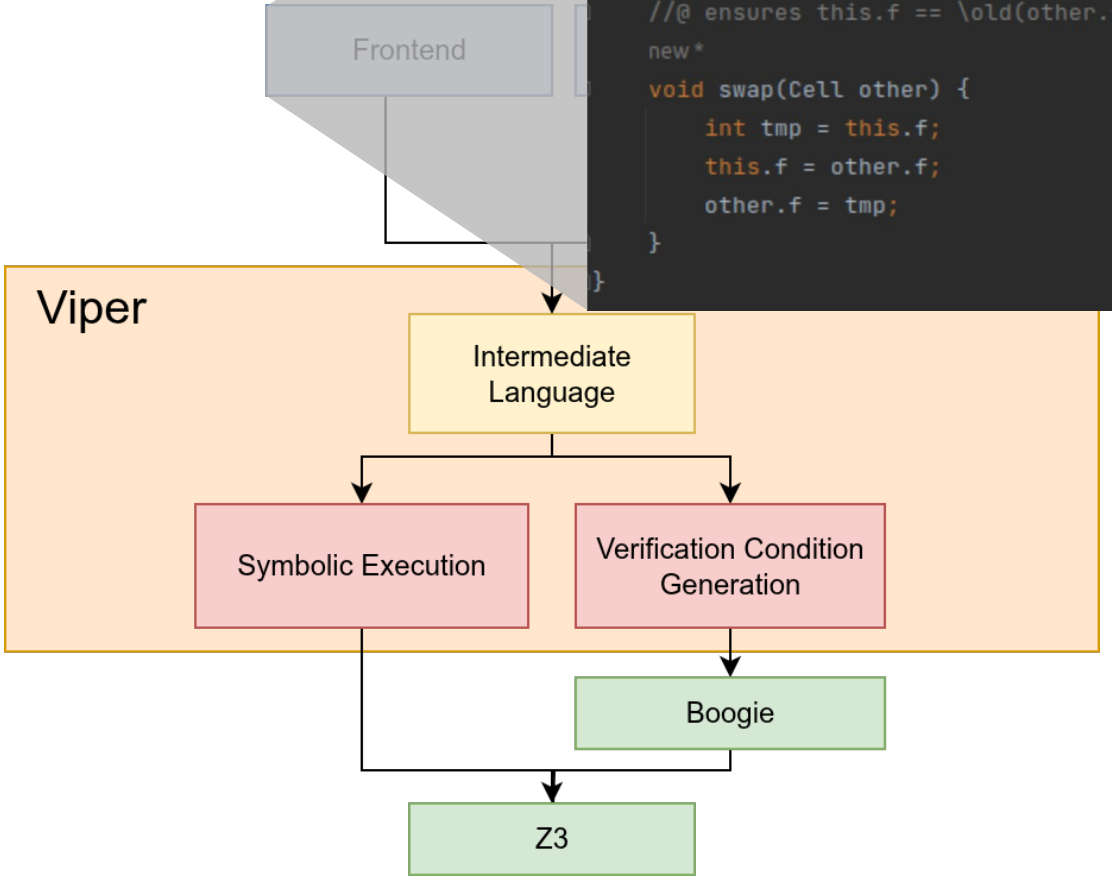
Verification
Infrastructure for
PERmission-Based Reasoning
(based on Separation Logic)

The Viper Verification Infrastructure

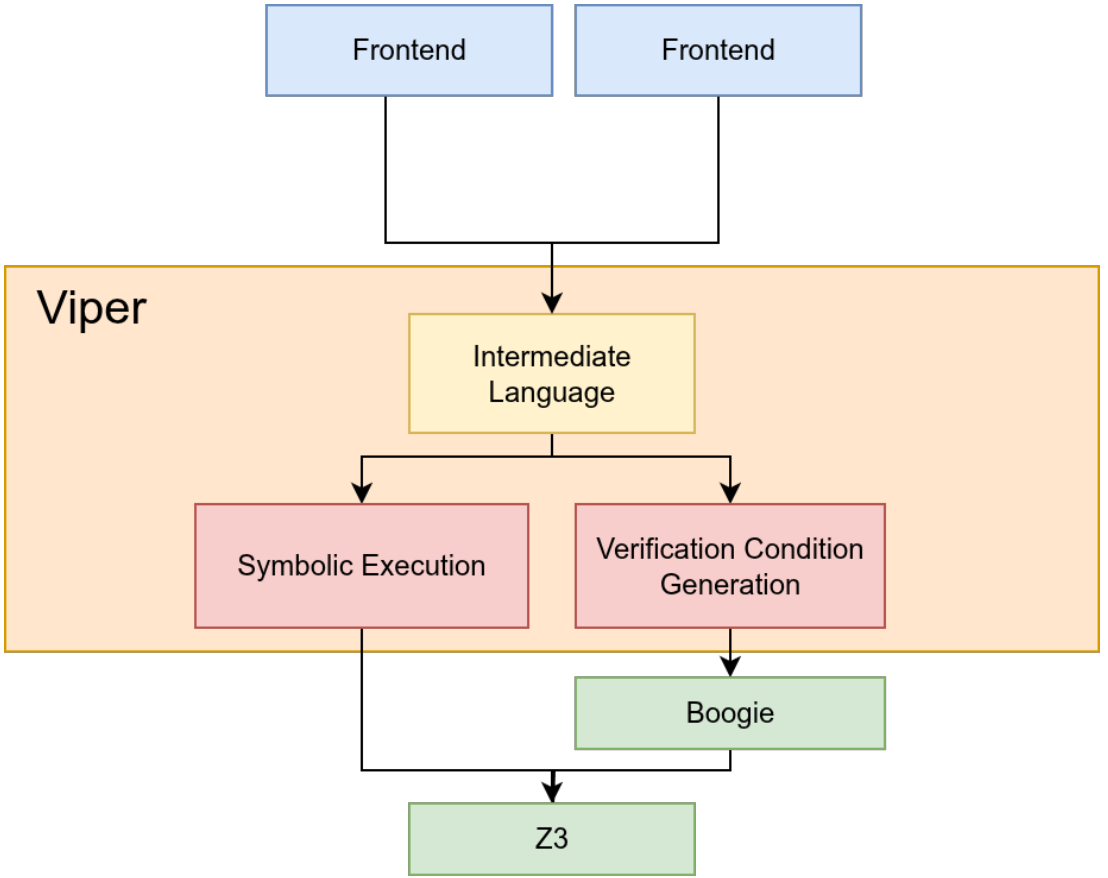


The Viper Verification Infra

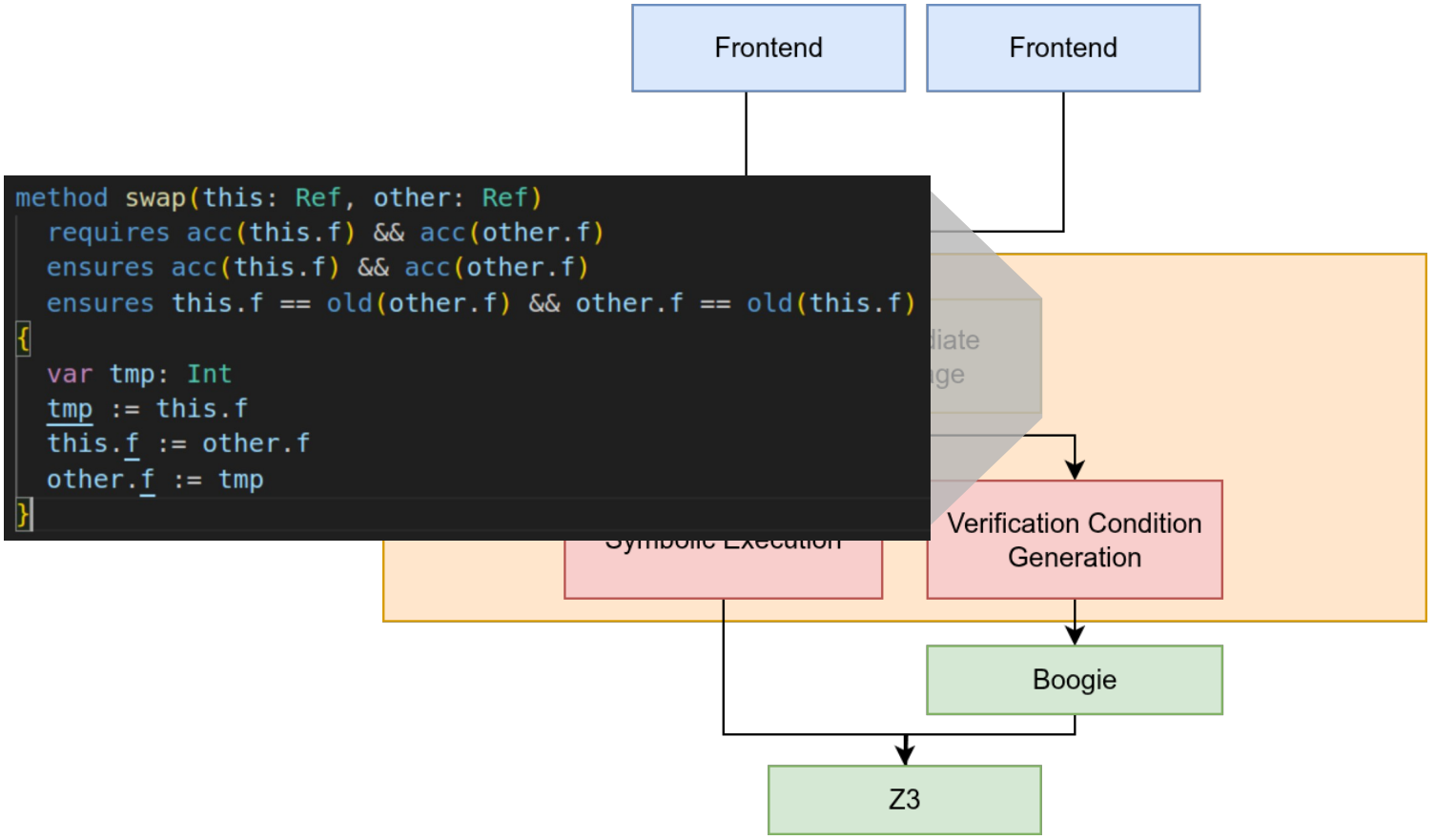
```
class Cell {
    4 usages
    private int f;

    //@ context Perm(f, write) ** Perm(other.f, write);
    //@ ensures this.f == \old(other.f) ** other.f == \old(this.f);
    new*
    void swap(Cell other) {
        int tmp = this.f;
        this.f = other.f;
        other.f = tmp;
    }
}
```



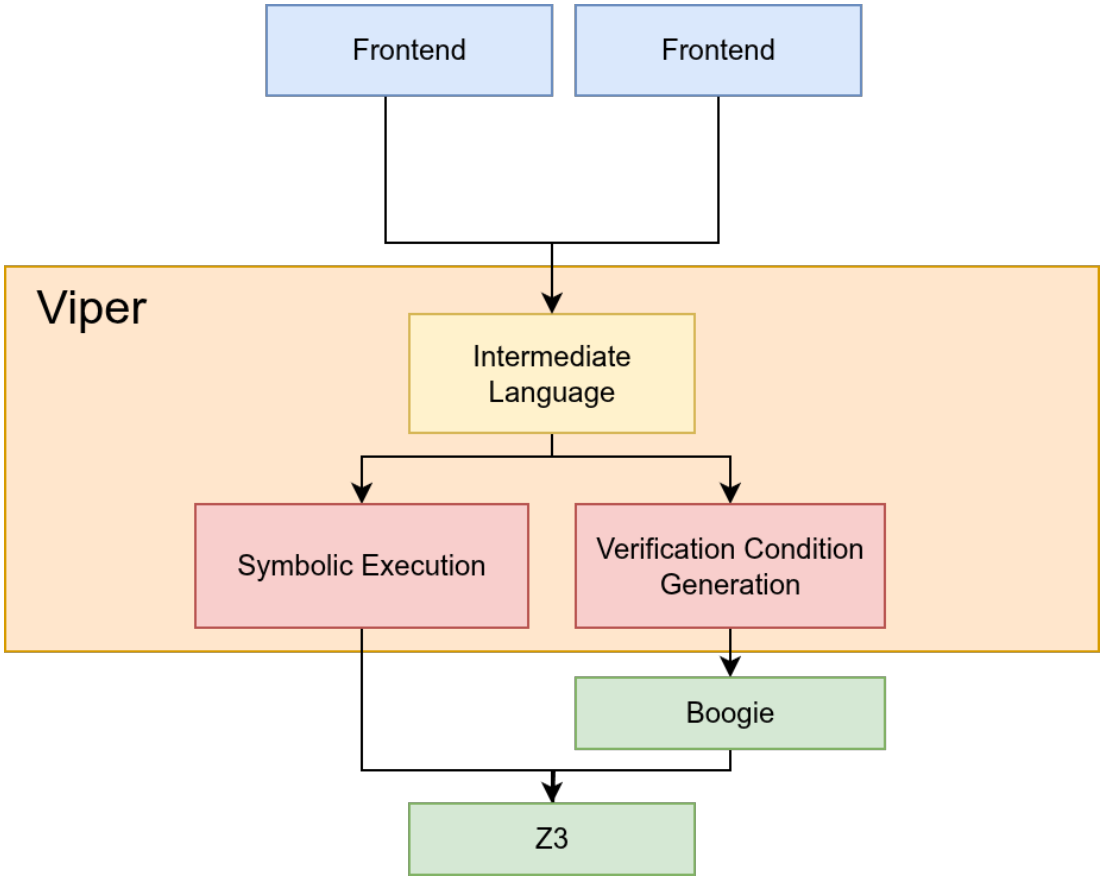
The Viper Verification Infrastructure



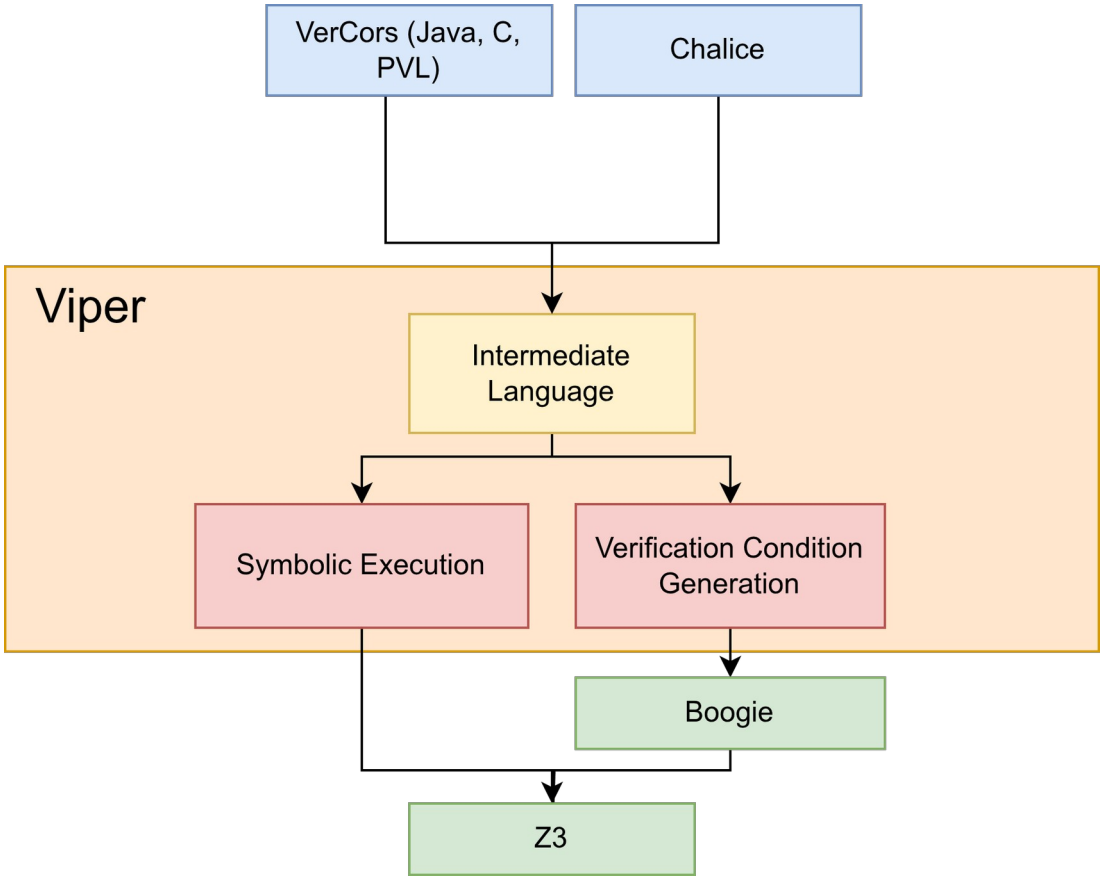
The Viper Verification Infrastructure



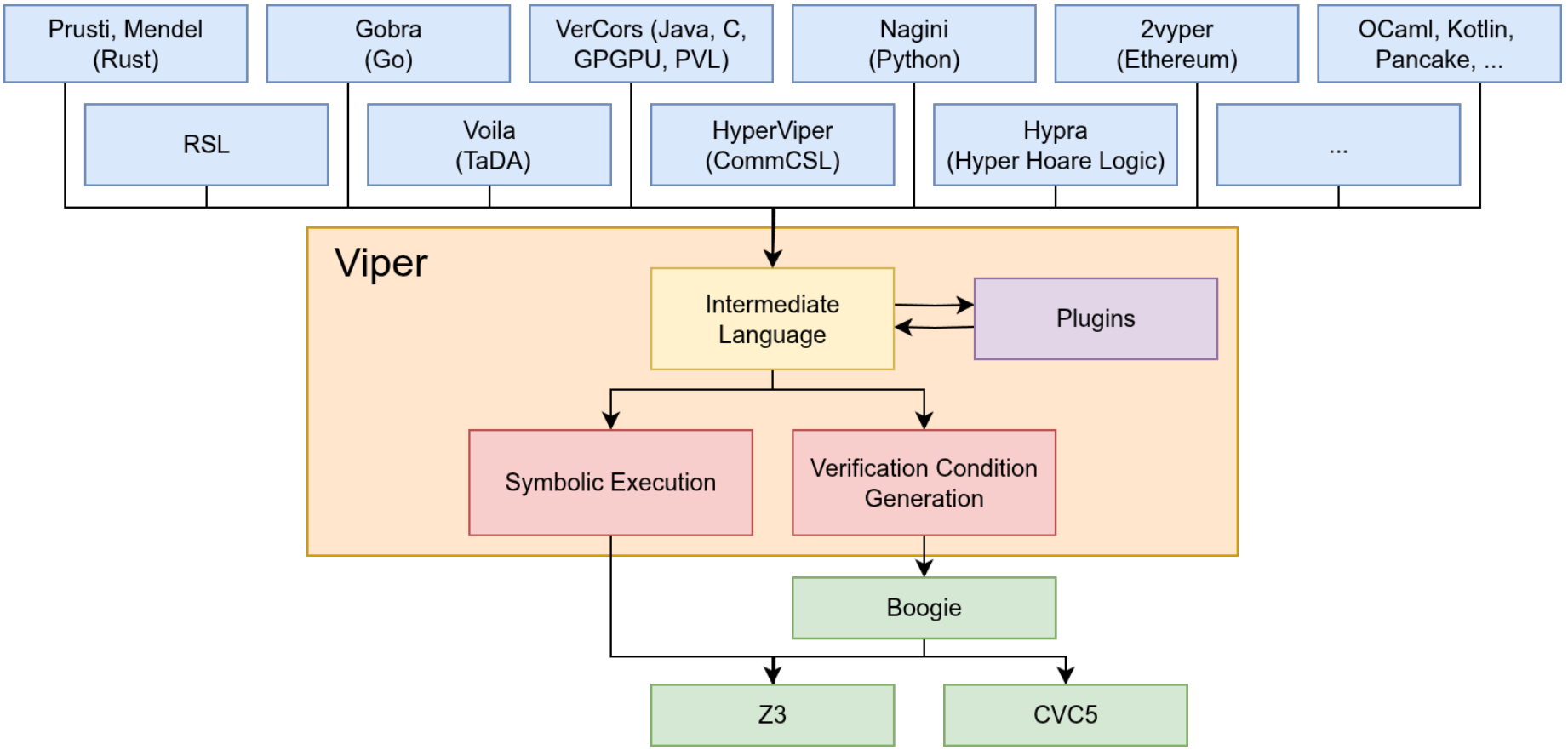
The Viper Verification Infrastructure



The Viper Verification Infrastructure (2016)



The Viper Verification Infrastructure (today)

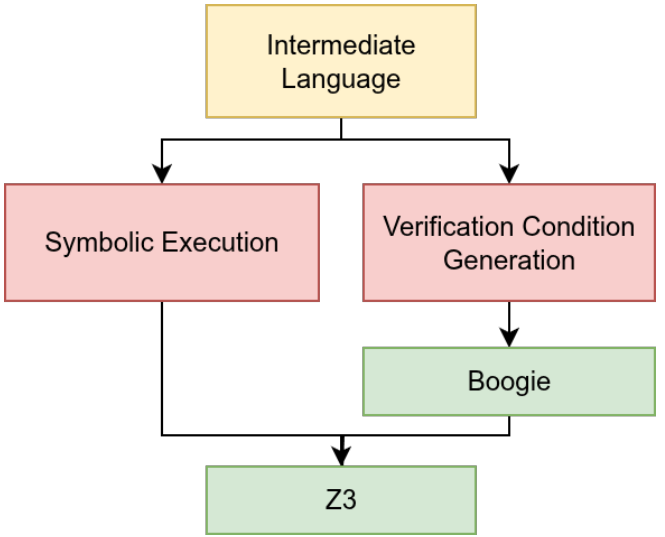


Looking back:

What were our design principles?
What tradeoffs did we make?
How did they work out?
How did Viper evolve?

Architecture: Multiple backends

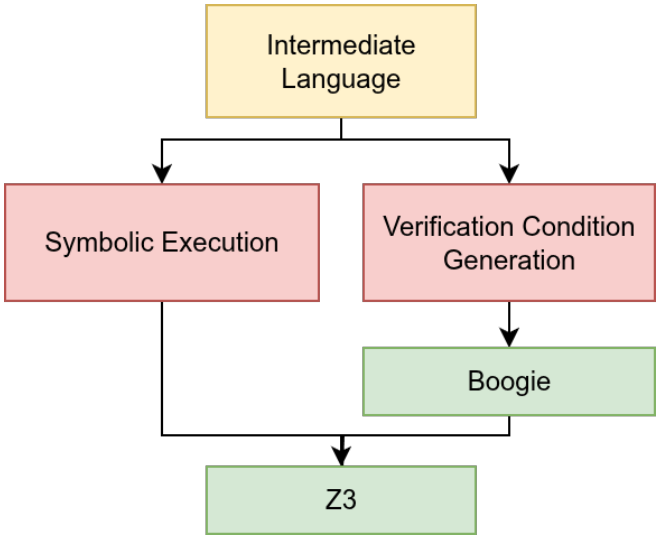
Unique feature for IVLs: **Multiple backends**



Architecture: Multiple backends

Unique feature for IVLs: **Multiple backends**

- Supporting same language, logic
- Goal: transparent to users



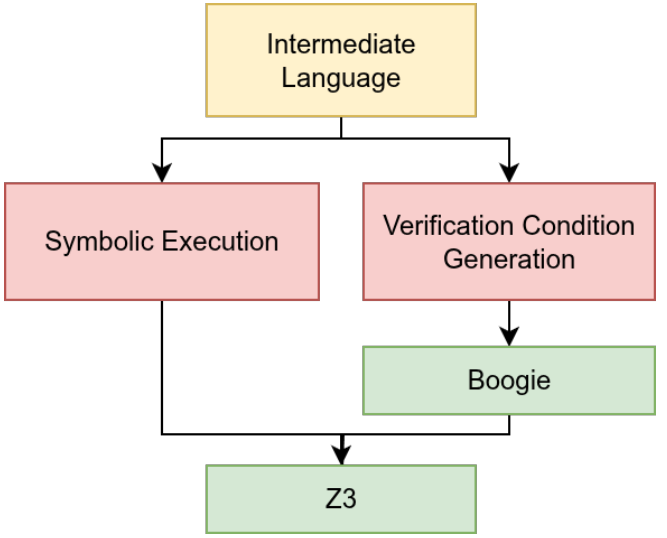
Architecture: Multiple backends

Unique feature for IVLs: **Multiple backends**

- Supporting same language, logic
- Goal: transparent to users

Pros:

- Different performance/completeness characteristics
  - **Crucial** to enable several important case studies
  - Going further: **Multiple algorithms** per backend (CAV 2024)
- Differential testing



Architecture: Multiple backends

Unique feature for IVLs: **Multiple backends**

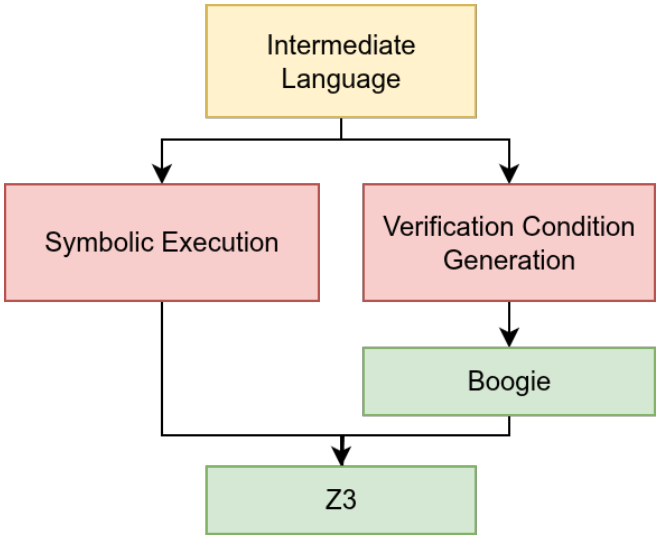
- Supporting same language, logic
- Goal: transparent to users

Pros:

- Different performance/completeness characteristics
  - **Crucial** to enable several important case studies
  - Going further: **Multiple algorithms** per backend (CAV 2024)
- Differential testing

Cons:

- Maintenance effort
  - Especially problematic for a research group



Architecture: Feature Reuse

Lots of features are needed in multiple frontends

- Termination checking
- Data type encoding
- Hyperproperty checking
- …

Architecture: Feature Reuse

Lots of features are needed in multiple frontends

- Termination checking
- Data type encoding
- Hyperproperty checking
- …

These can be **encoded** into the core language

- No need to include in Viper language
- Would blow up Viper language unnecessarily

Architecture: Feature Reuse

Lots of features are needed in multiple frontends

- Termination checking
- Data type encoding
- Hyperproperty checking
- …

These can be **encoded** into the core language

- No need to include in Viper language
- Would blow up Viper language unnecessarily

We want to avoid re-implementing the encoding in every frontend.

Architecture: Feature Reuse

Lots of features are needed in multiple frontends

- Termination checking
- Data type encoding
- Hyperproperty checking
- …

These can be **encoded** into the core language

- No need to include in Viper language
- Would blow up Viper language unnecessarily

We want to avoid re-implementing the encoding in every frontend.

→ We introduced a **plugin-system** to enable feature reuse.

Architecture: Feature Reuse

Lots of features are needed in multiple frontends

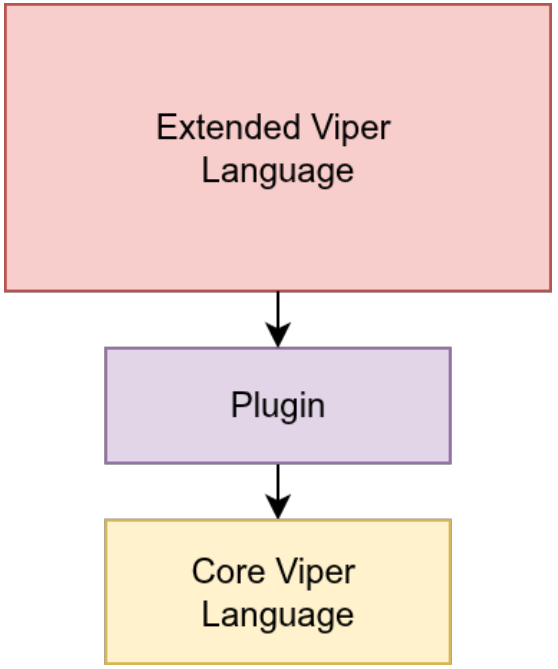
- Termination checking
- Data type encoding
- Hyperproperty checking
- …

These can be **encoded** into the core language

- No need to include in Viper language
- Would blow up Viper language unnecessarily

We want to avoid re-implementing the encoding in every frontend.

→ We introduced a **plugin-system** to enable feature reuse.

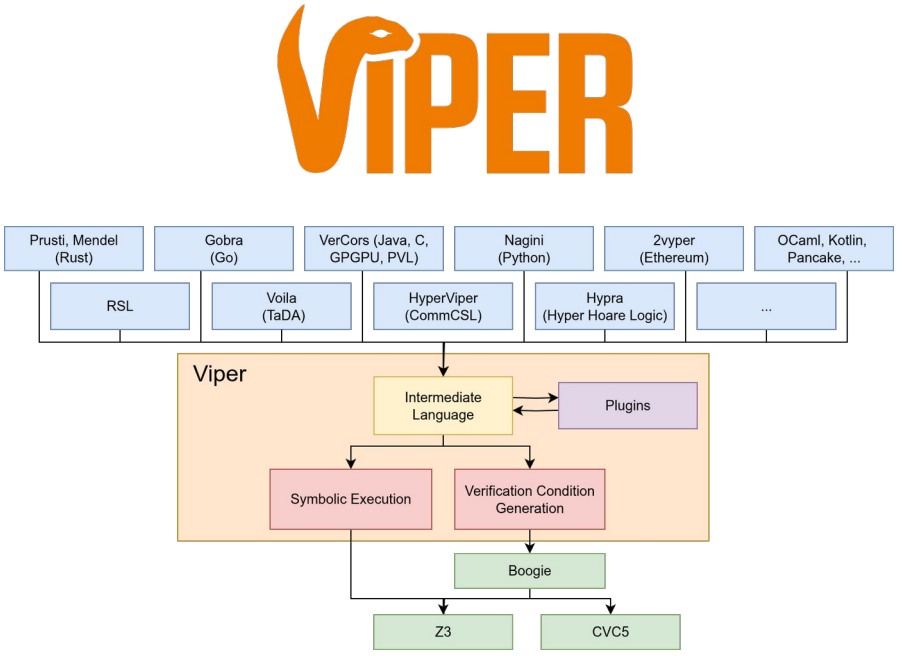


Other Topics in the Paper

- Language design:
  - Error reporting
  - Human-readability
- Logic:
  - Degree of automation
  - Formalization
  - Choice of a non-standard logic

Other Topics in the Paper

- Language design:
  - Error reporting
  - Human-readability
- Logic:
  - Degree of automation
  - Formalization
  - Choice of a non-standard logic



viper.ethz.ch