



ACM SIGSOFT International Symposium on Software Testing and Analysis

Synthesize Solving Strategy for Symbolic Execution

Zhenbang Chen
(zbchen@nudt.edu.cn)

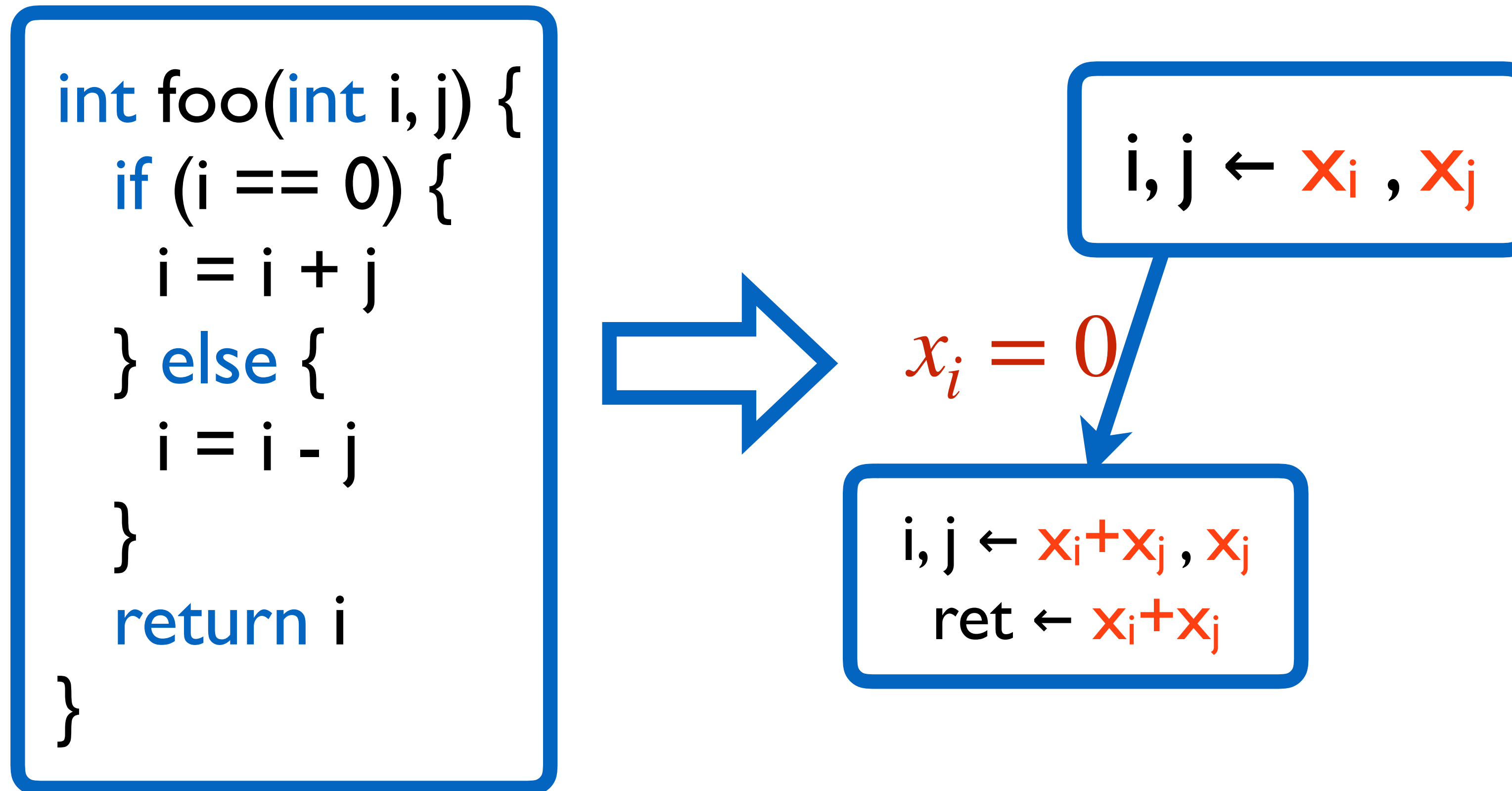
*Joint work with Zehua Chen, Ziqi Shuai, Guofeng Zhang, Weiyu Pan,
Yufeng Zhang, and Ji Wang*



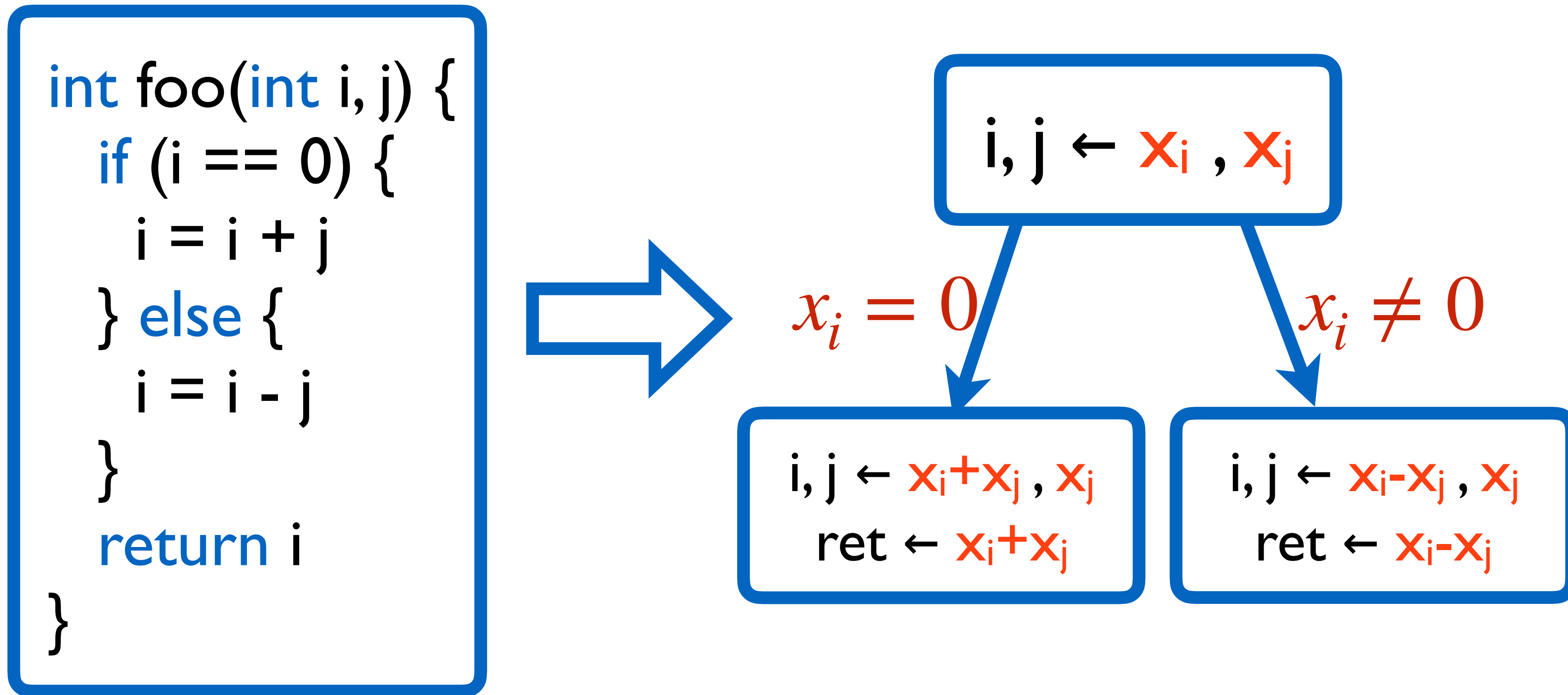
Symbolic Execution

```
int foo(int i, j) {  
    if (i == 0) {  
        i = i + j  
    } else {  
        i = i - j  
    }  
    return i  
}
```

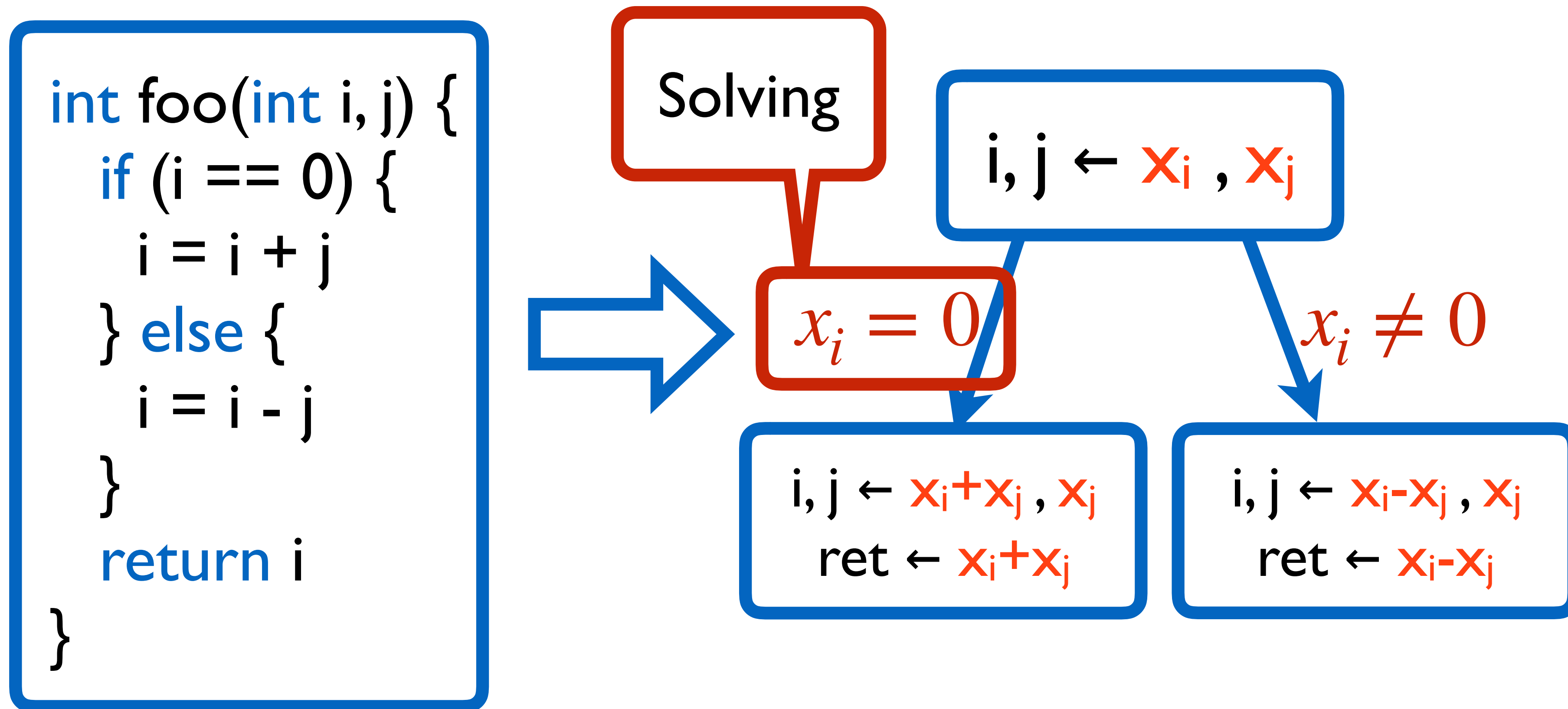
Symbolic Execution



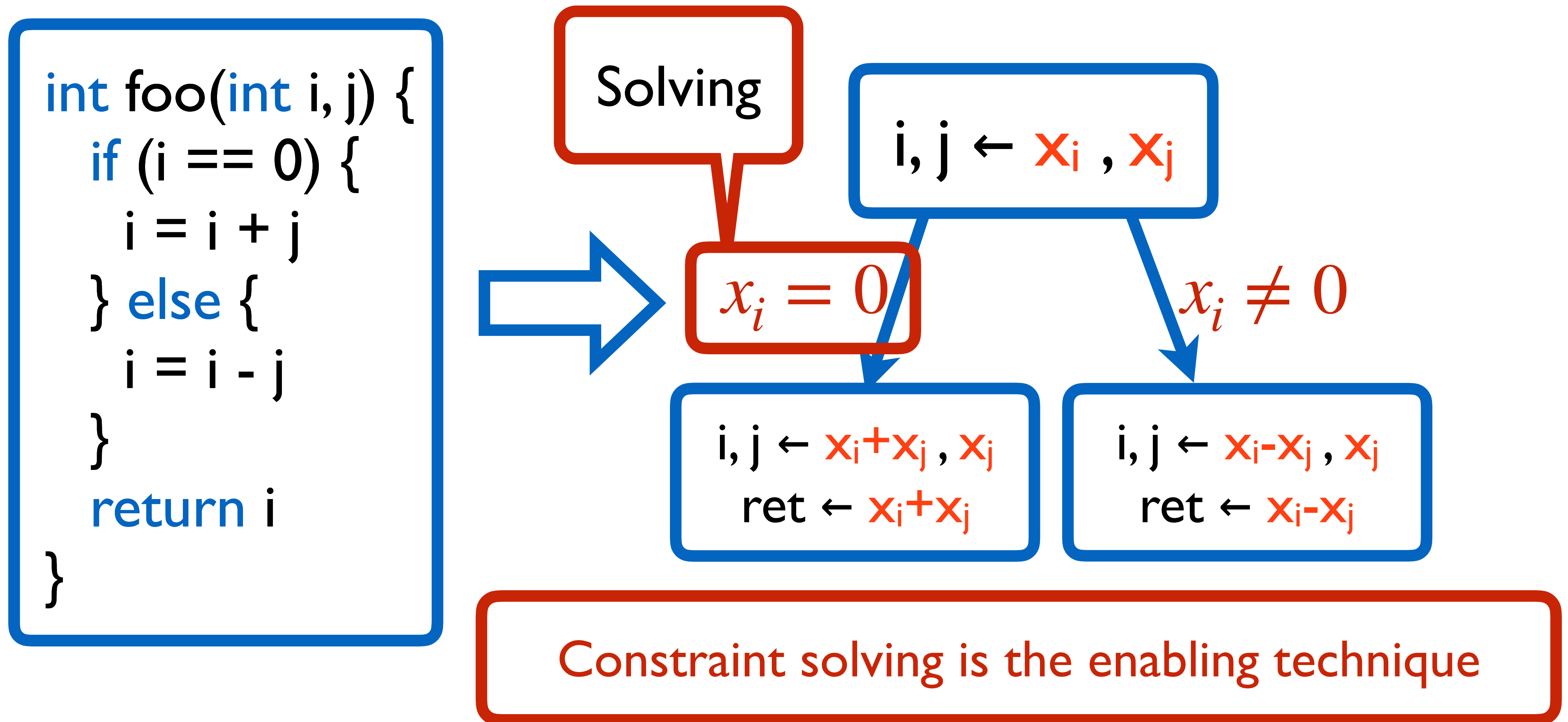
Symbolic Execution



Symbolic Execution

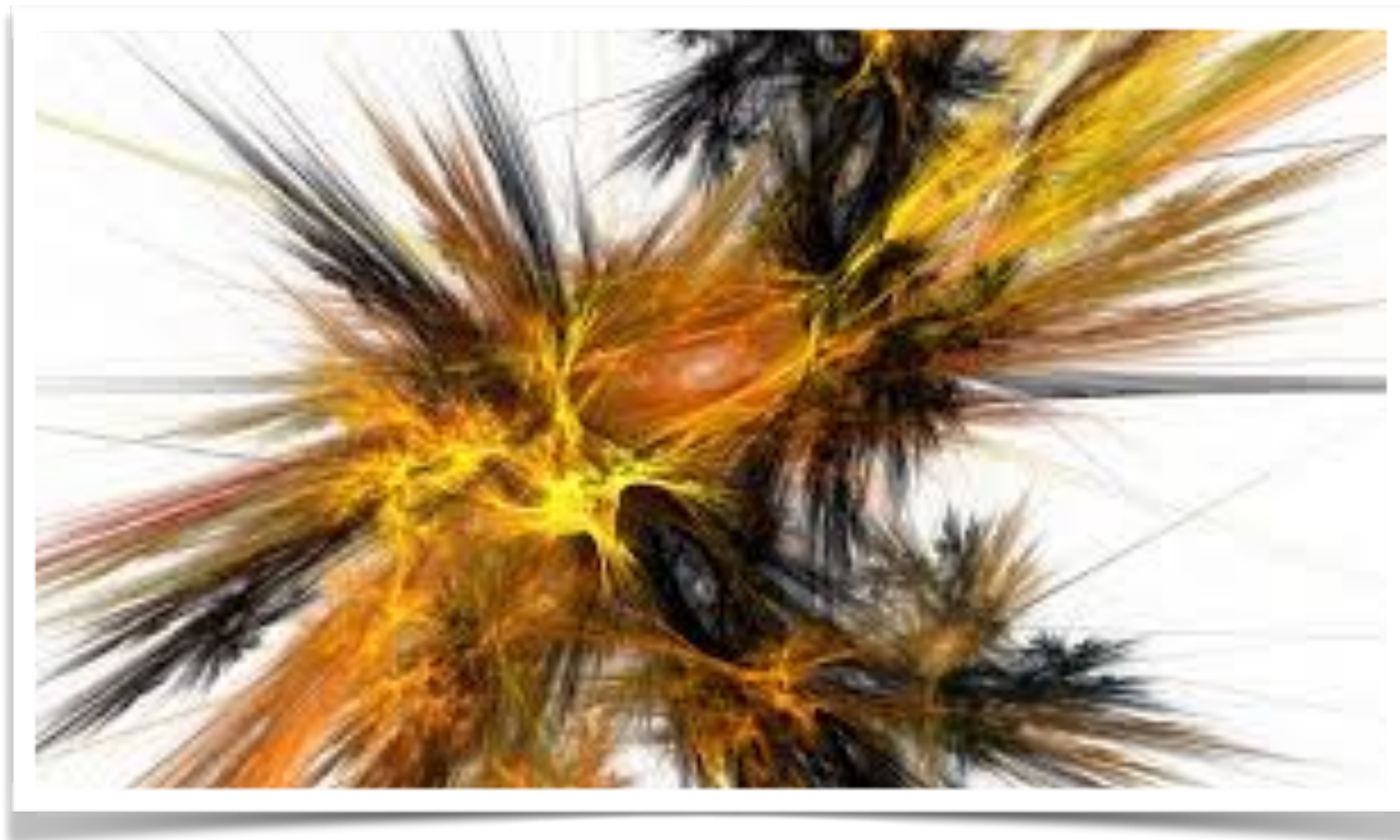


Symbolic Execution



Challenges of Symbolic Execution

Path explosion



Challenges of Symbolic Execution

Path explosion



```
int foo(int a[], int j) {  
    int c = 0;  
    while (j-- > -1) {  
        if (a[j] > 0)  
            c++;  
        else  
            c-- ;  
    }  
    return c;  
}
```


Challenges of Symbolic Execution

Path explosion

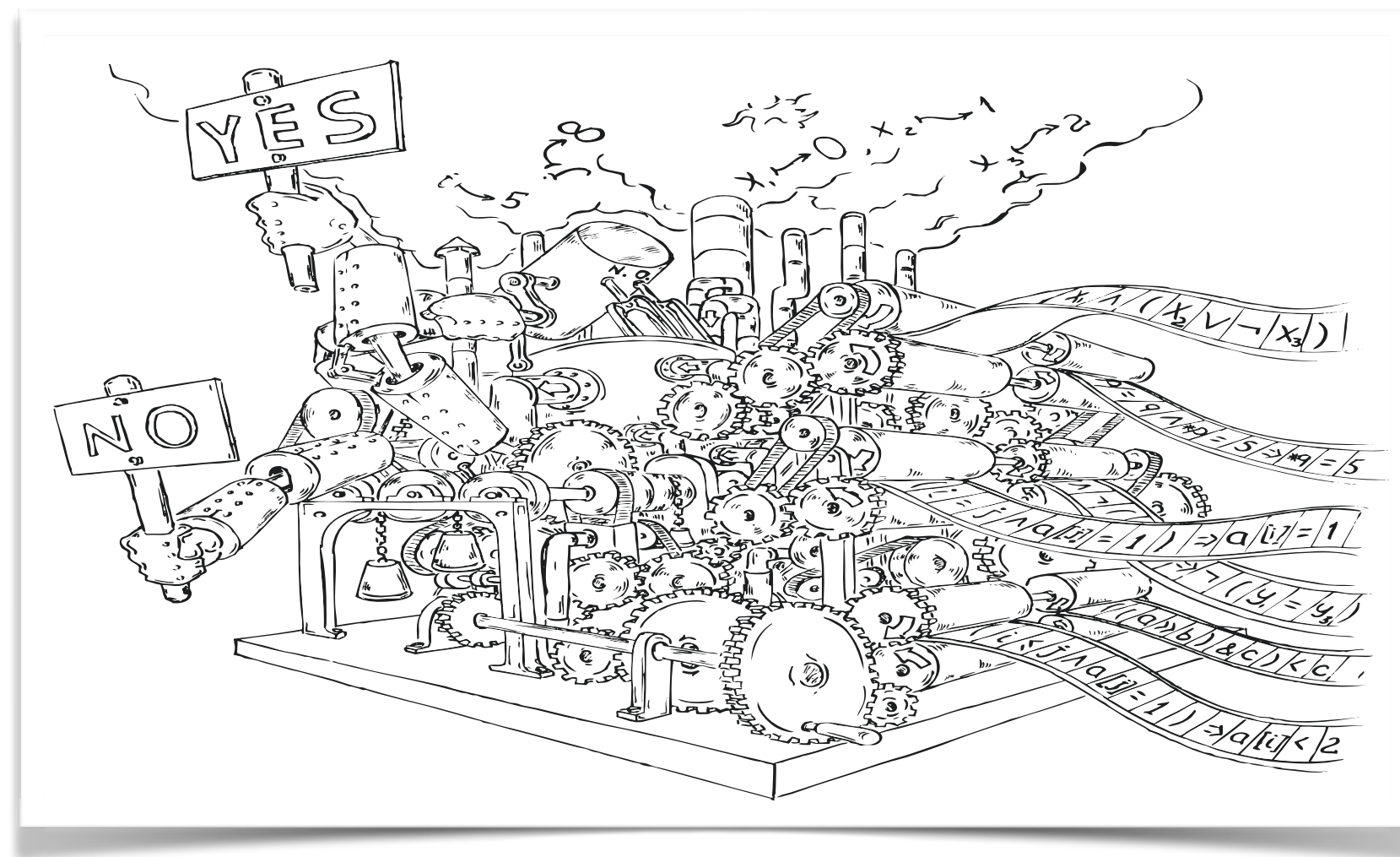


```
int foo(int a[], int j) {  
    int c = 0;  
    while (j-- > -1) {  
        if (a[j] > 0)  
            c++;  
        else  
            c-- ;  
    }  
    return c;  
}
```

$$2^j$$

Challenges of Symbolic Execution

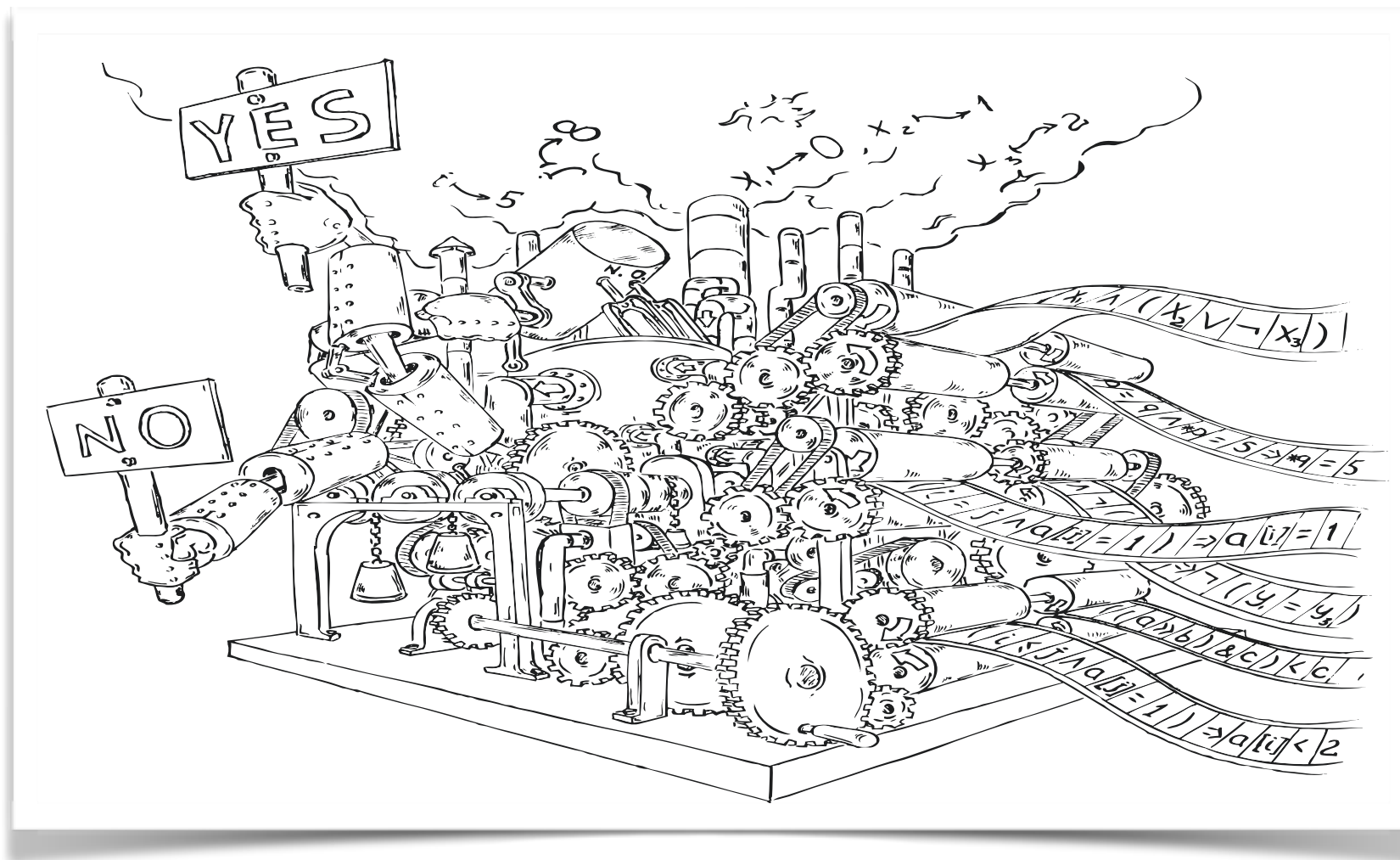
Constraint Solving



Decision Procedures An Algorithmic Point of View, Second Edition, 2016

Challenges of Symbolic Execution

Constraint Solving



Decision Procedures An Algorithmic Point of View, Second Edition, 2016



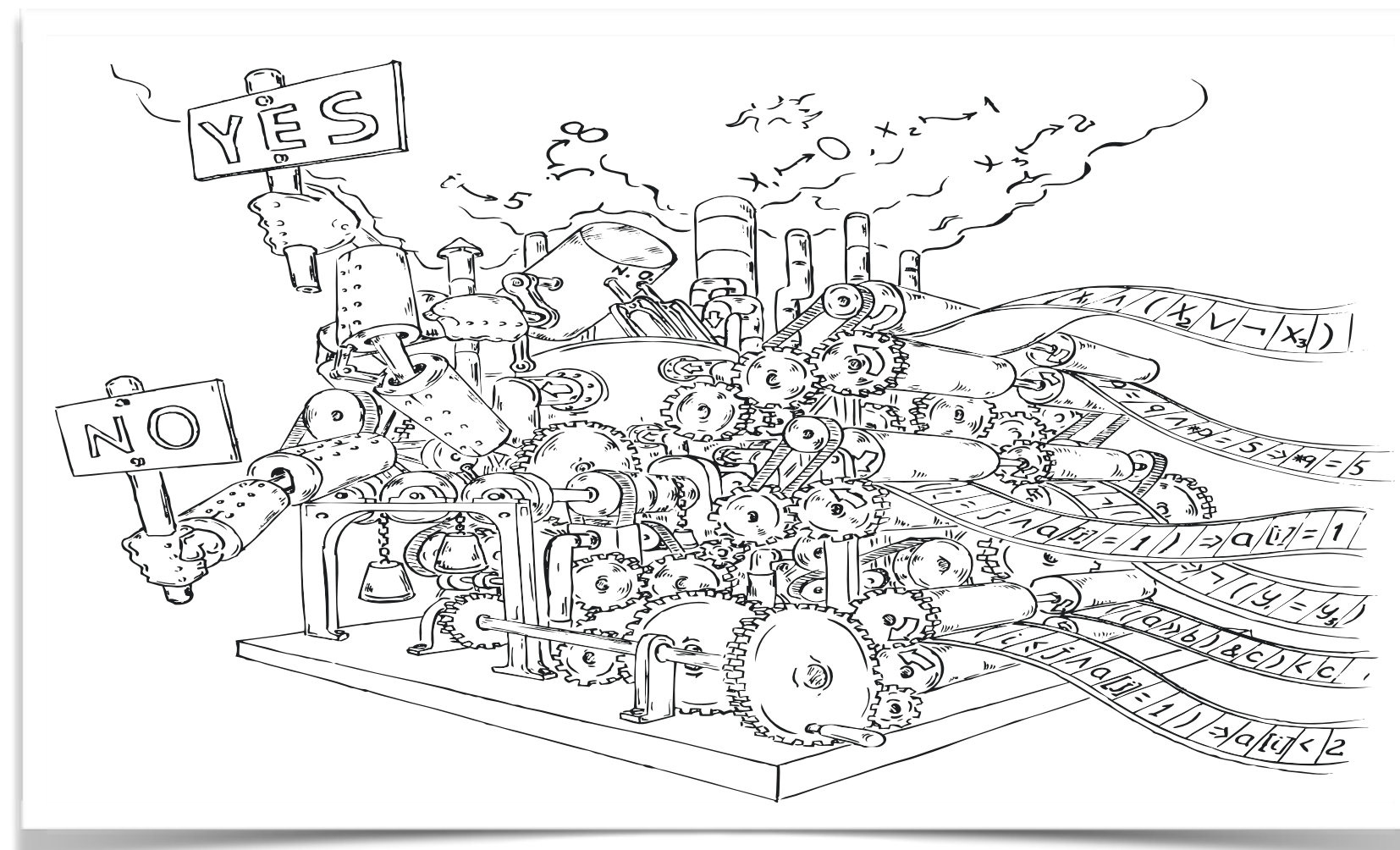
- Undecidable in general
- High complexity in computation

Challenges of Symbolic Execution

Path explosion



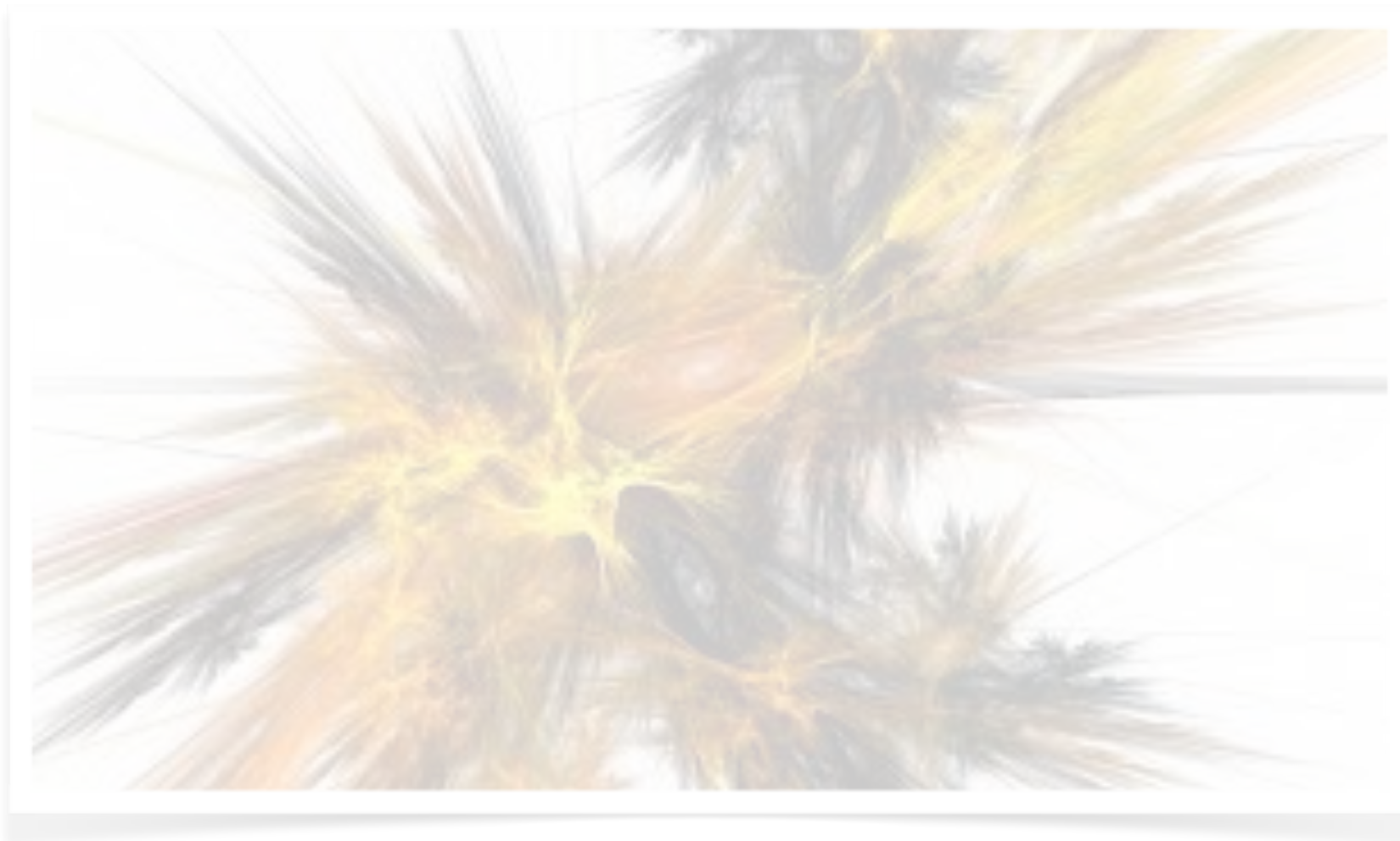
Constraint Solving



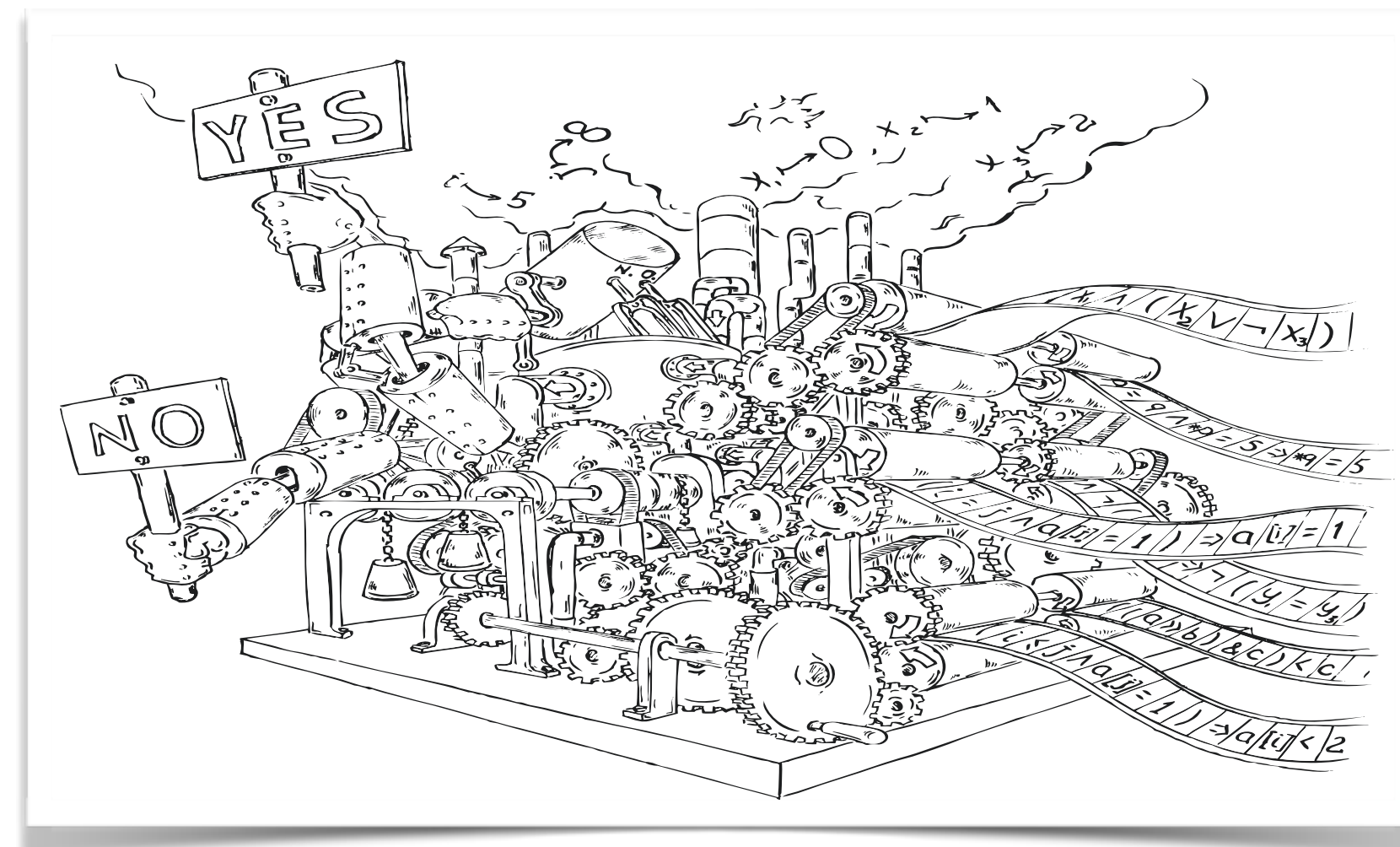
Decision Procedures An Algorithmic Point of View, Second Edition, 2016

Our Work's Target

Path explosion



Constraint Solving



Decision Procedures An Algorithmic Point of View, Second Edition, 2016

Existing Work of Optimizing Constraint Solving in SE

Existing Work of Optimizing Constraint Solving in SE

- Query cache (partial) and simplification
 - KLEE[OSDI'08], KLEE-Array[ISSTA'17]

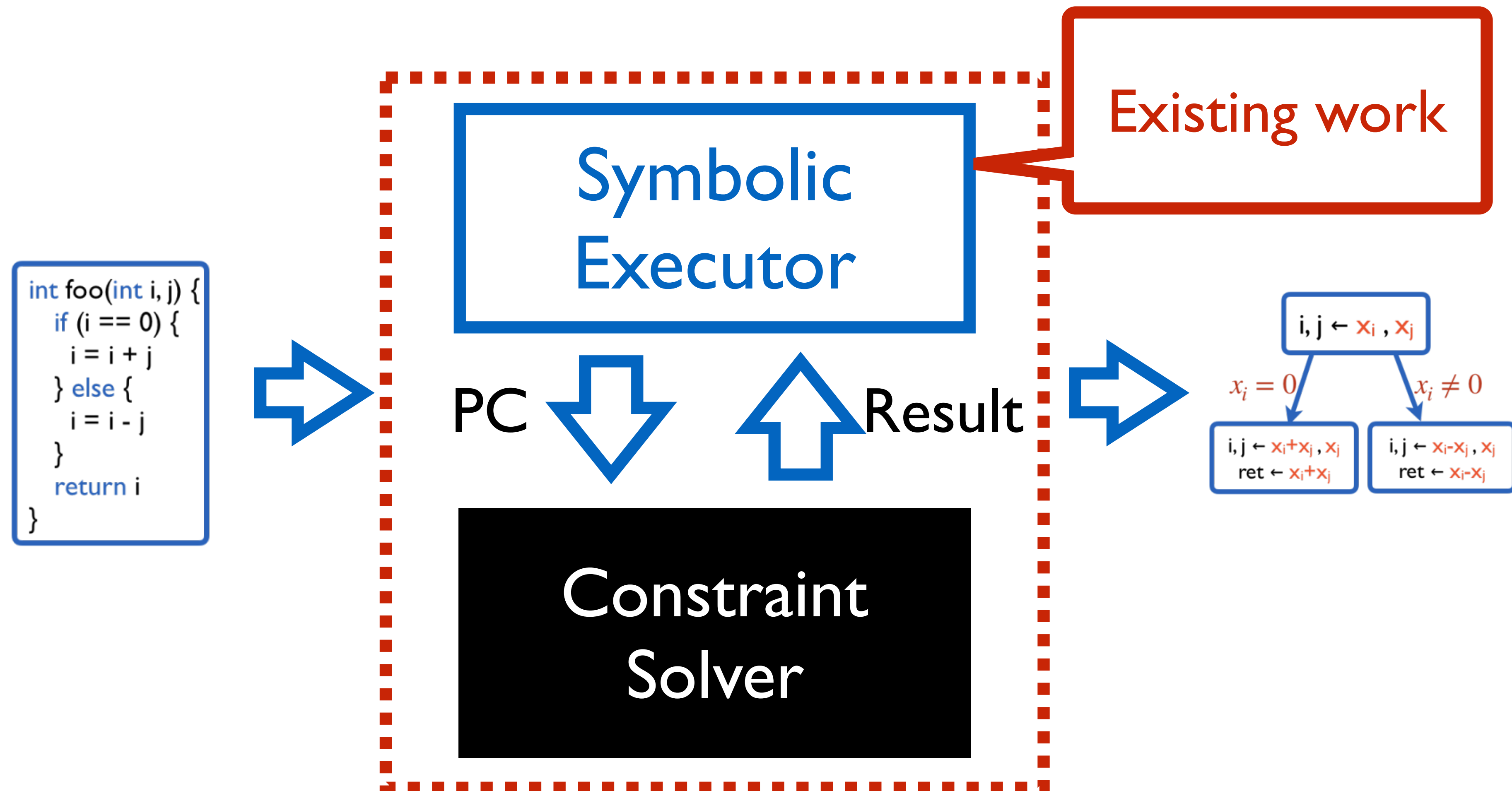
Existing Work of Optimizing Constraint Solving in SE

- Query cache (partial) and simplification
 - KLEE[OSDI'08], KLEE-Array[ISSTA'17]
- Query reduction
 - SSE[ISSRE'12], Cloud9[PLDI'12]

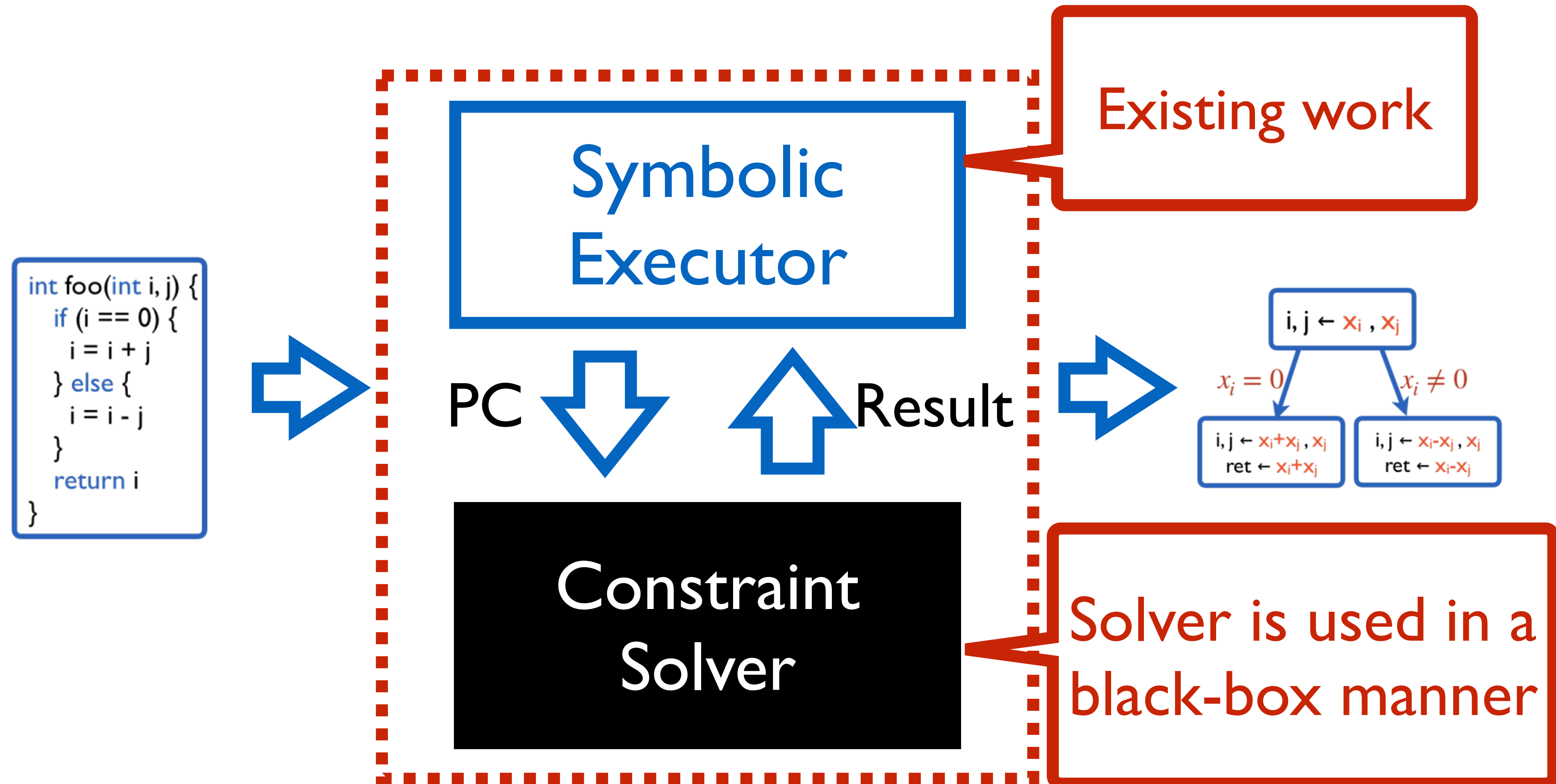
Existing Work of Optimizing Constraint Solving in SE

- Query cache (partial) and simplification
 - KLEE[OSDI'08], KLEE-Array[ISSTA'17]
- Query reduction
 - SSE[ISSRE'12], Cloud9[PLDI'12]
- Query reuse
 - Green[FSE'12], GreenTrie[ISSTA'15]

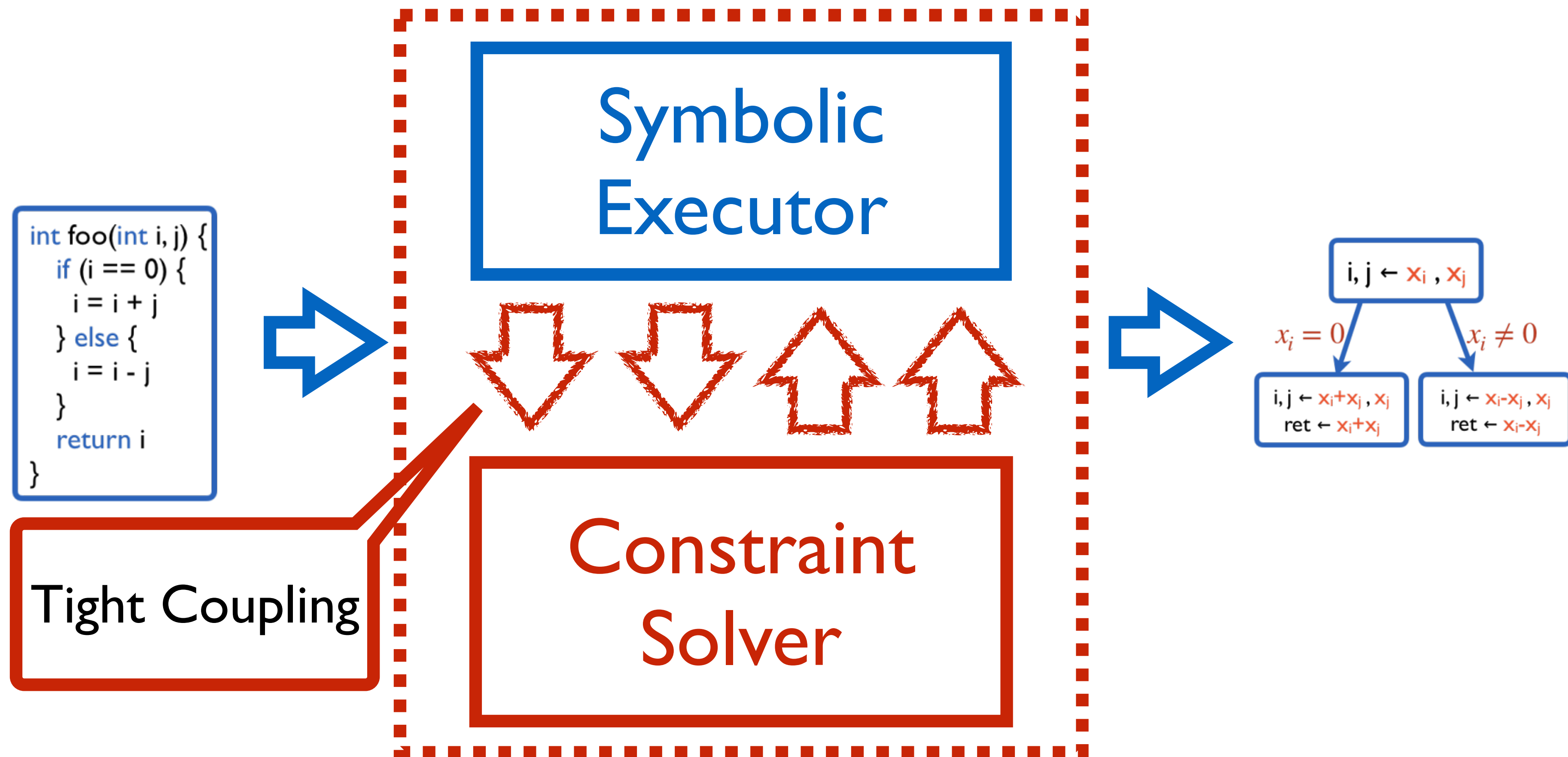
Our Observation



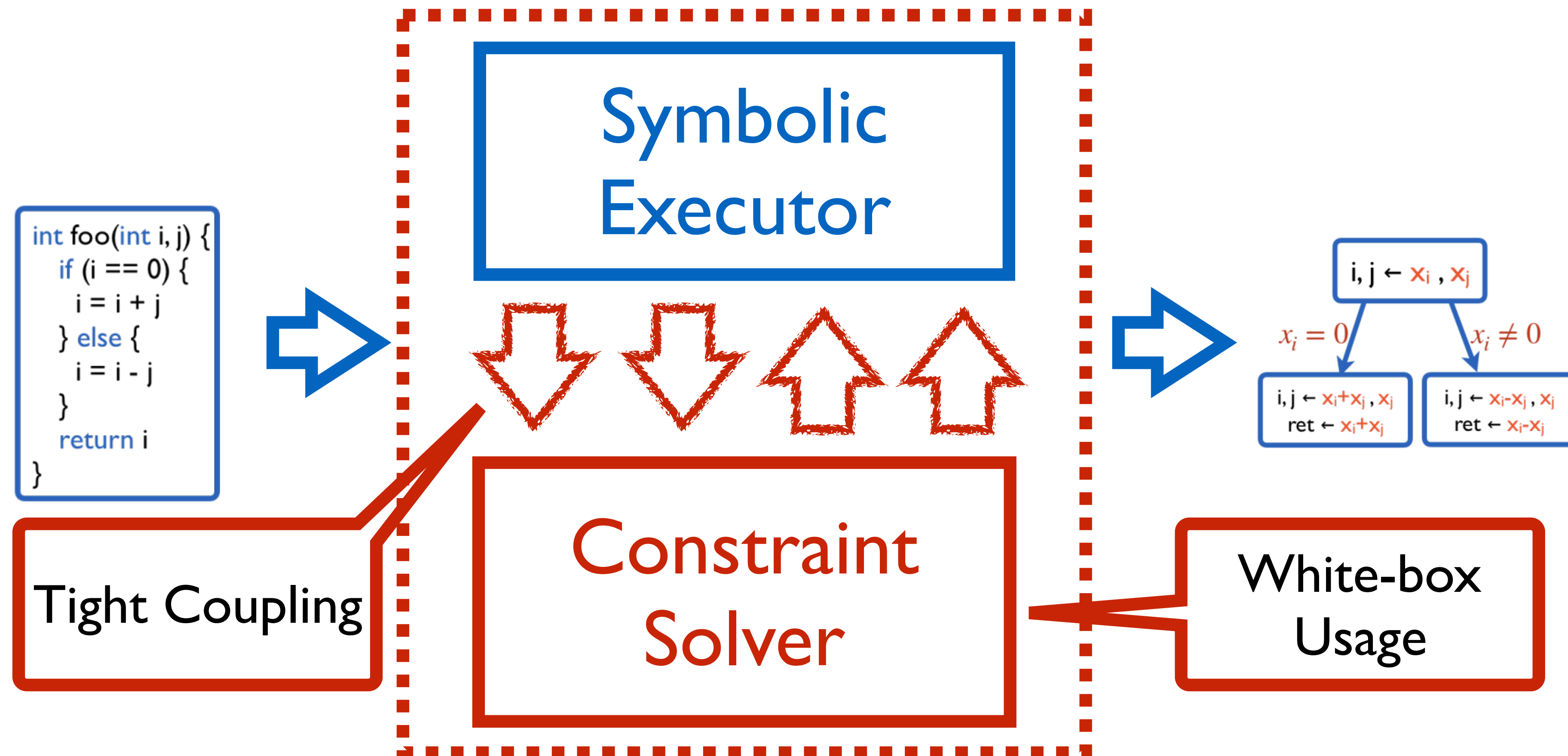
Our Observation



Our Argument



Our Argument

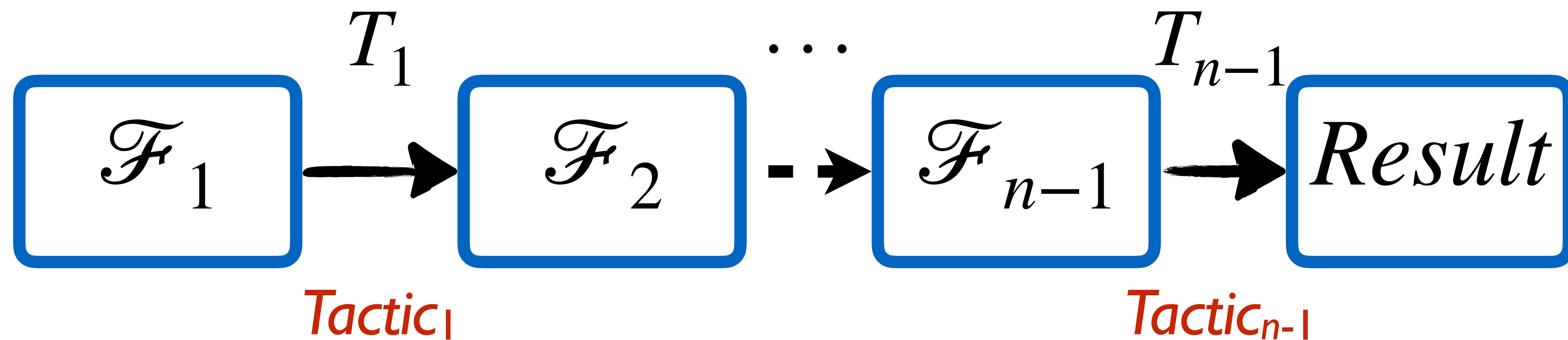


SMT Solving Strategy

- Customize the solving procedure

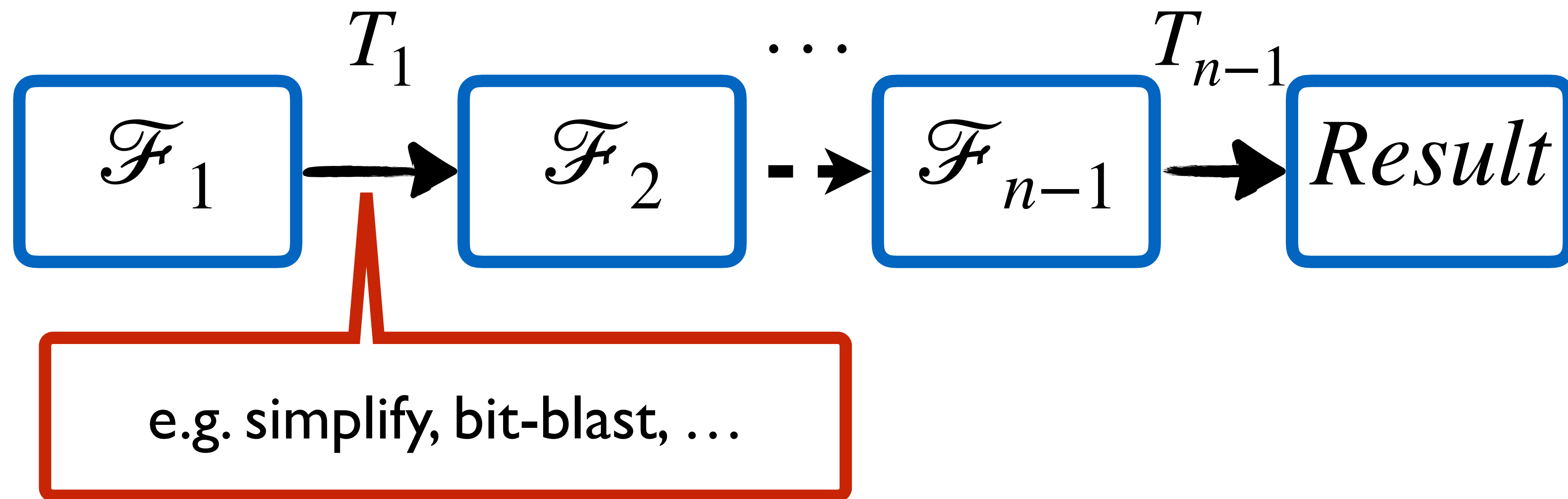
SMT Solving Strategy

- Customize the solving procedure



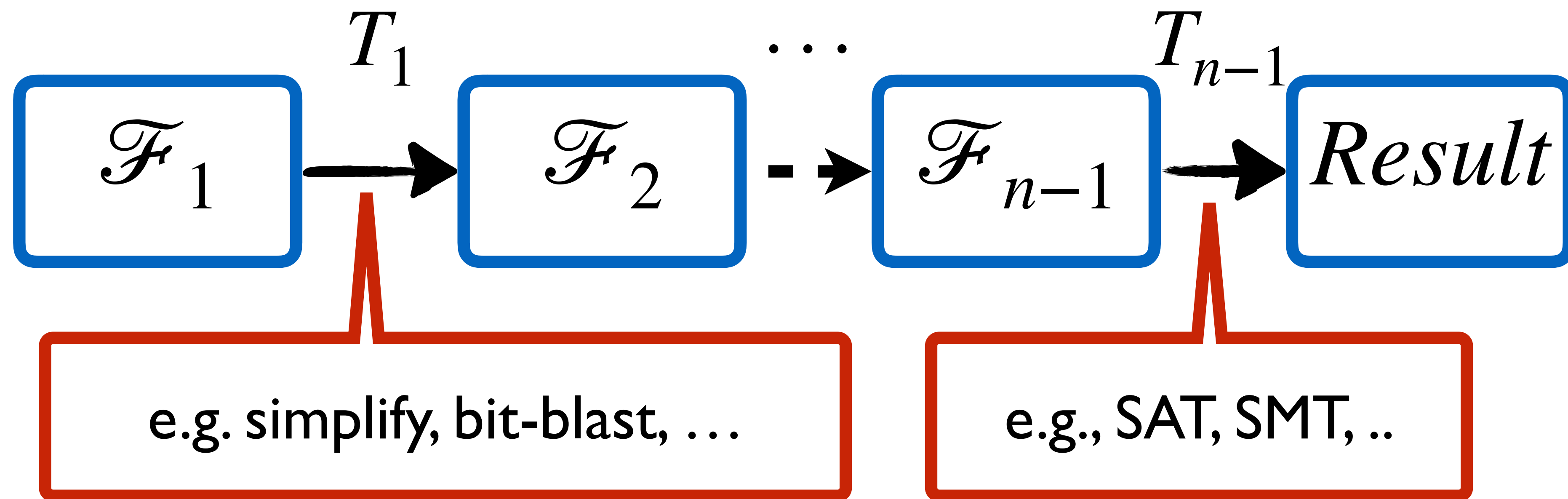
SMT Solving Strategy

- Customize the solving procedure

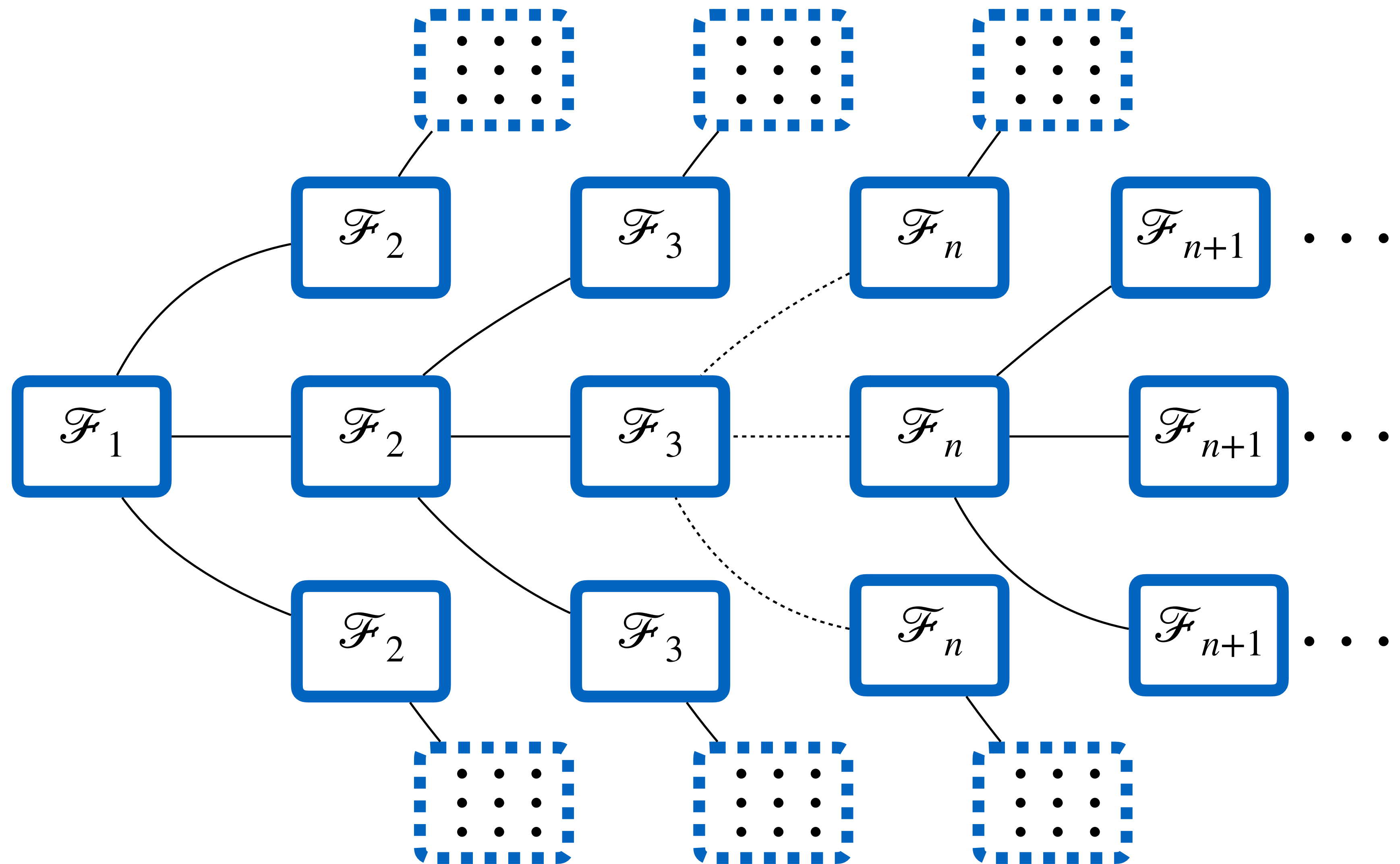


SMT Solving Strategy

- Customize the solving procedure



SMT Solving Strategy



SMT Solving Strategy

- Solving strategy has a **great** influence to solving performance

SMT Solving Strategy

- Solving strategy has a **great** influence to solving performance

$$x^3 = 8.0$$

QF_BVFP formula, x is
a **double** variable

SMT Solving Strategy

- Solving strategy has a **great** influence to solving performance

$$x^3 = 8.0$$

QF_BVFP formula, x is
a **double** variable

56 seconds by using Z3's
default strategy

SMT Solving Strategy

- Solving strategy has a **great** influence to solving performance

$$x^3 = 8.0$$

QF_BVFP formula, x is
a **double** variable

56 seconds by using Z3's
default strategy

22 seconds by using the
following customized one
⟨simplify, SMT⟩

Our Key Insight

- A program's symbolic execution is a **specific** constraint solving problem

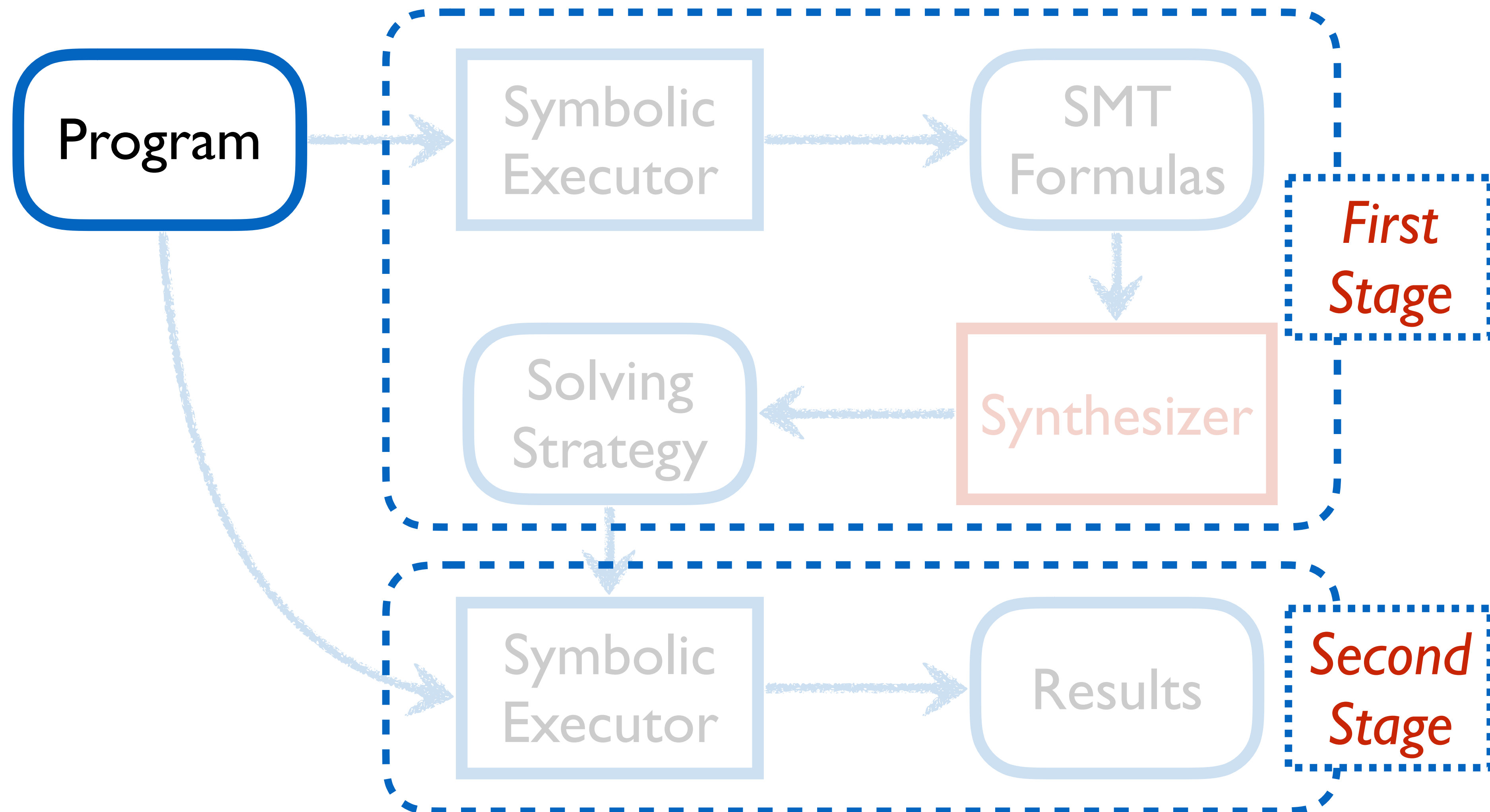
Our Key Insight

- A program's symbolic execution is a **specific** constraint solving problem
- We can use solving strategy to **customize the solver for the program** to solve the program's path constraints efficiently

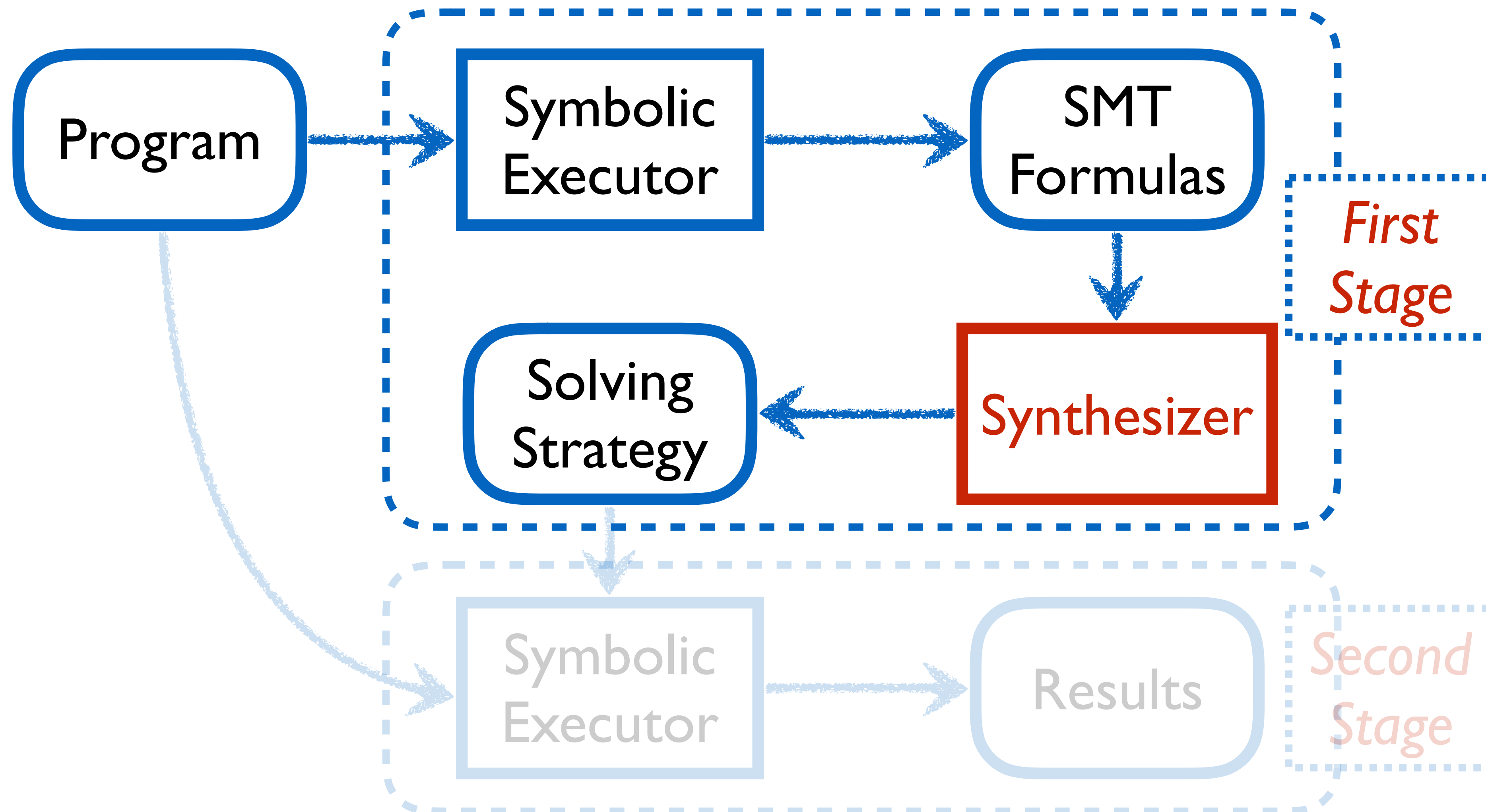
Our Key Idea

- **Online synthesize** a solving strategy for the program under symbolic execution
- The synthesized solving strategy can **improve the efficiency of solving the program's path conditions** in symbolic execution

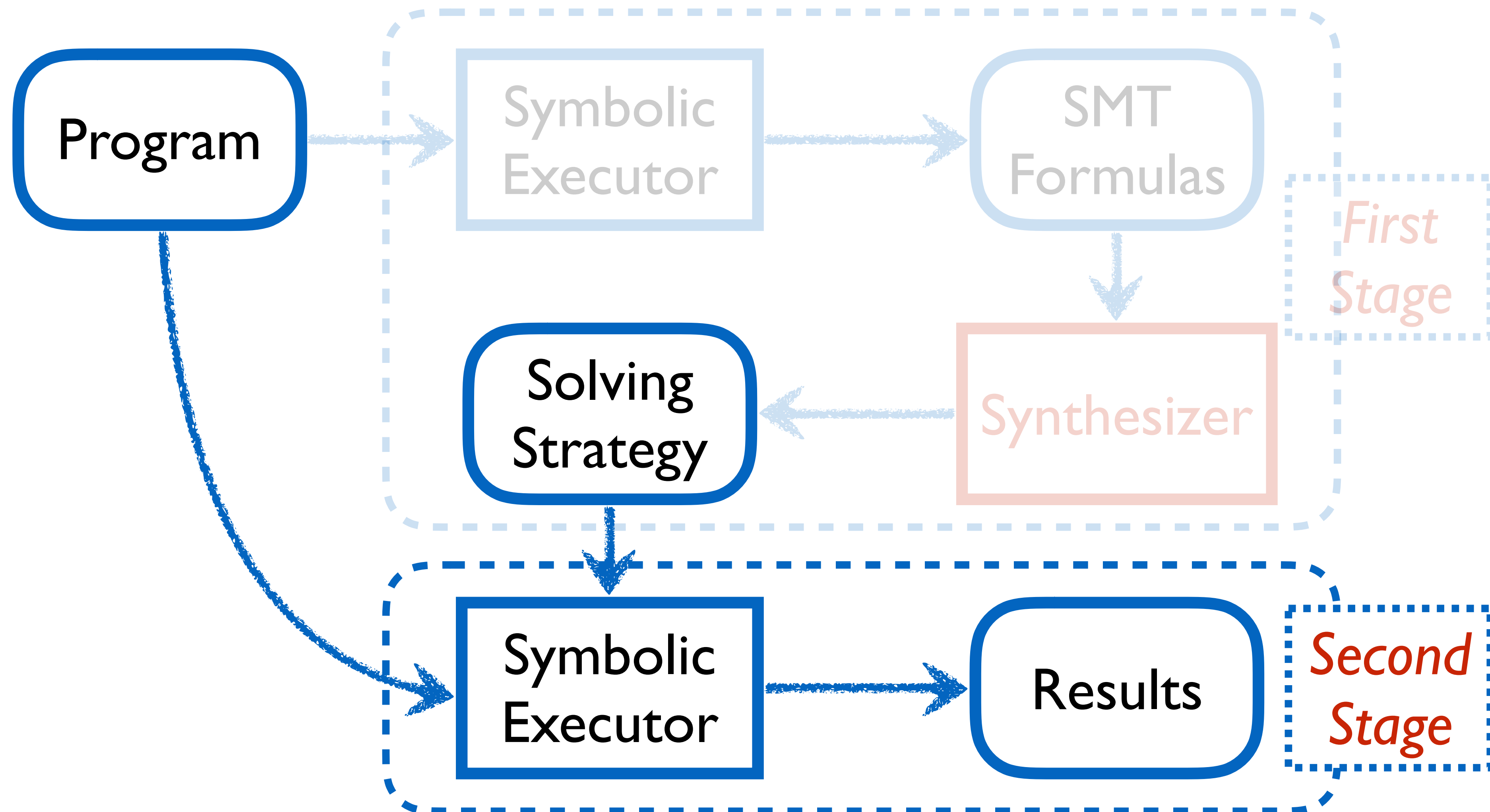
High-Level Framework



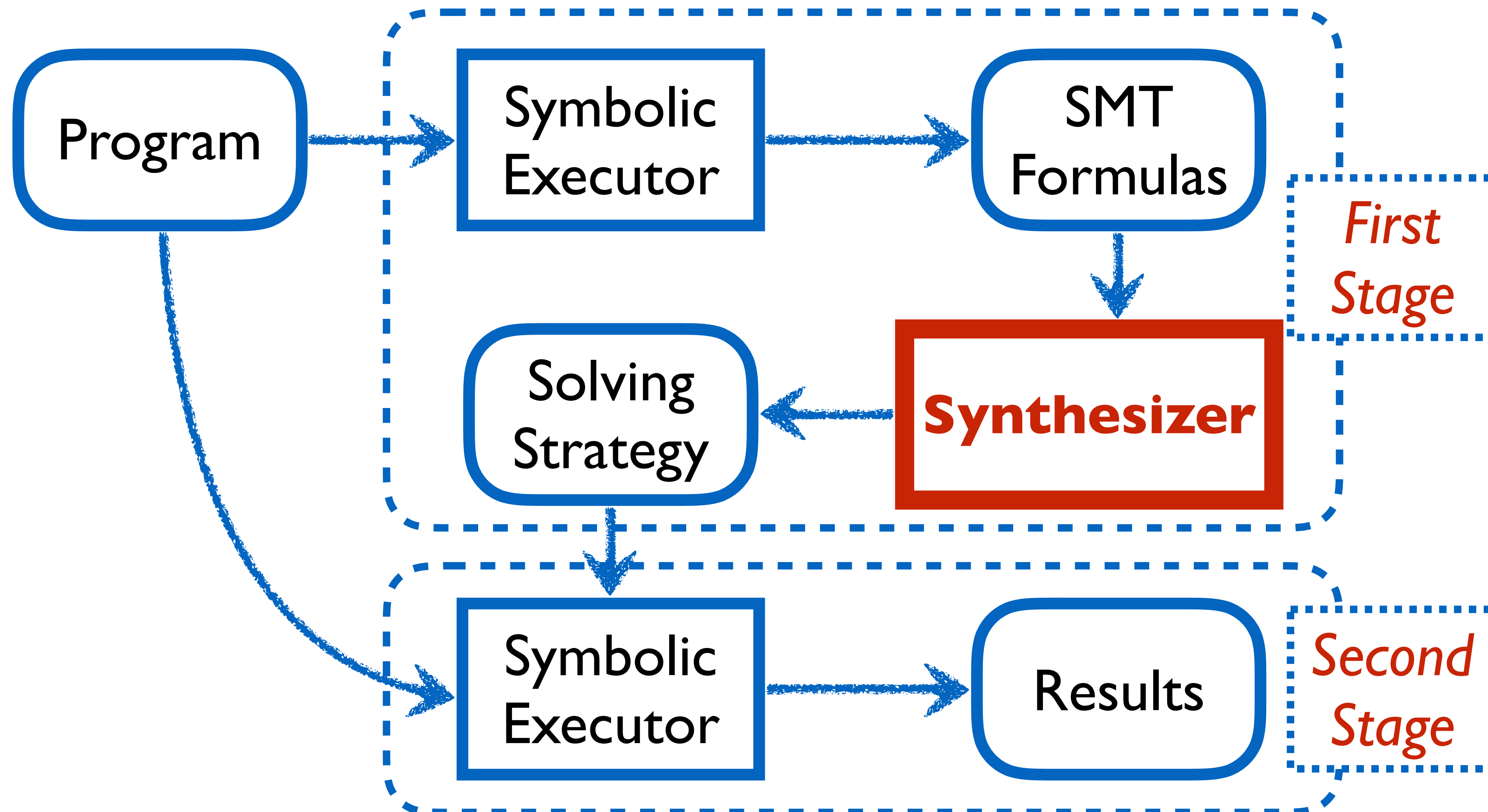
High-Level Framework



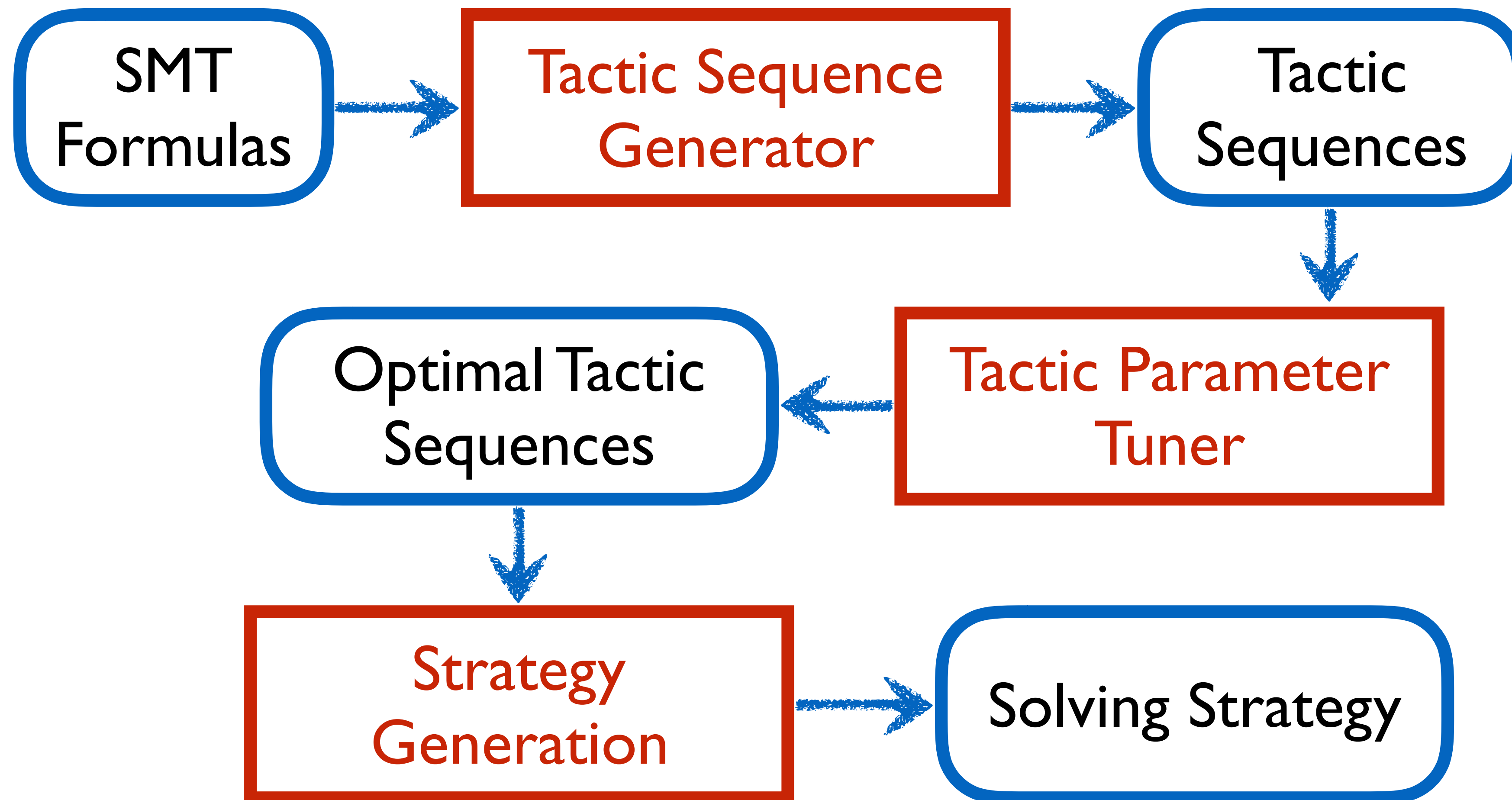
High-Level Framework



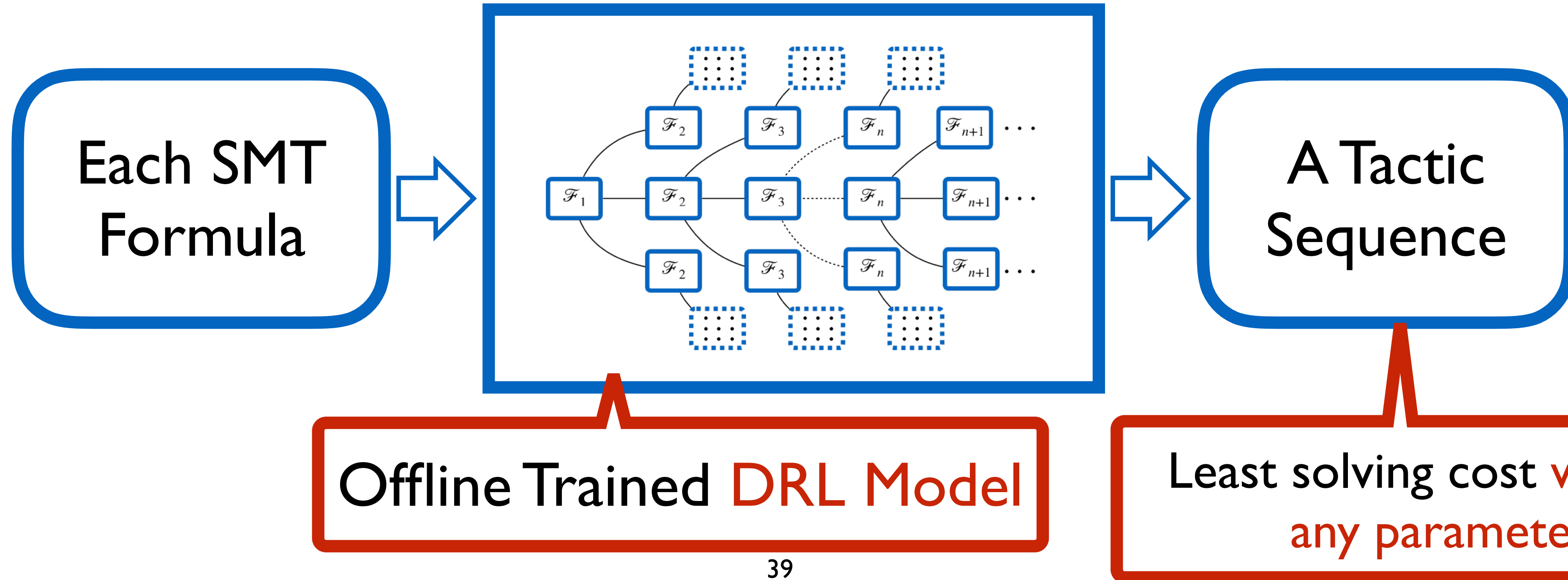
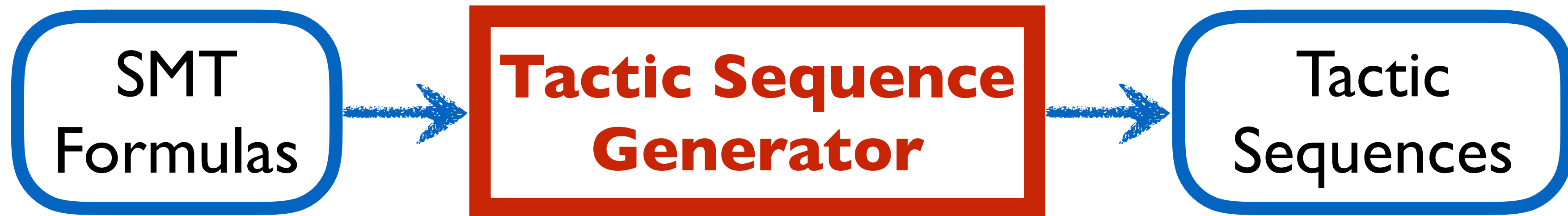
High-Level Framework

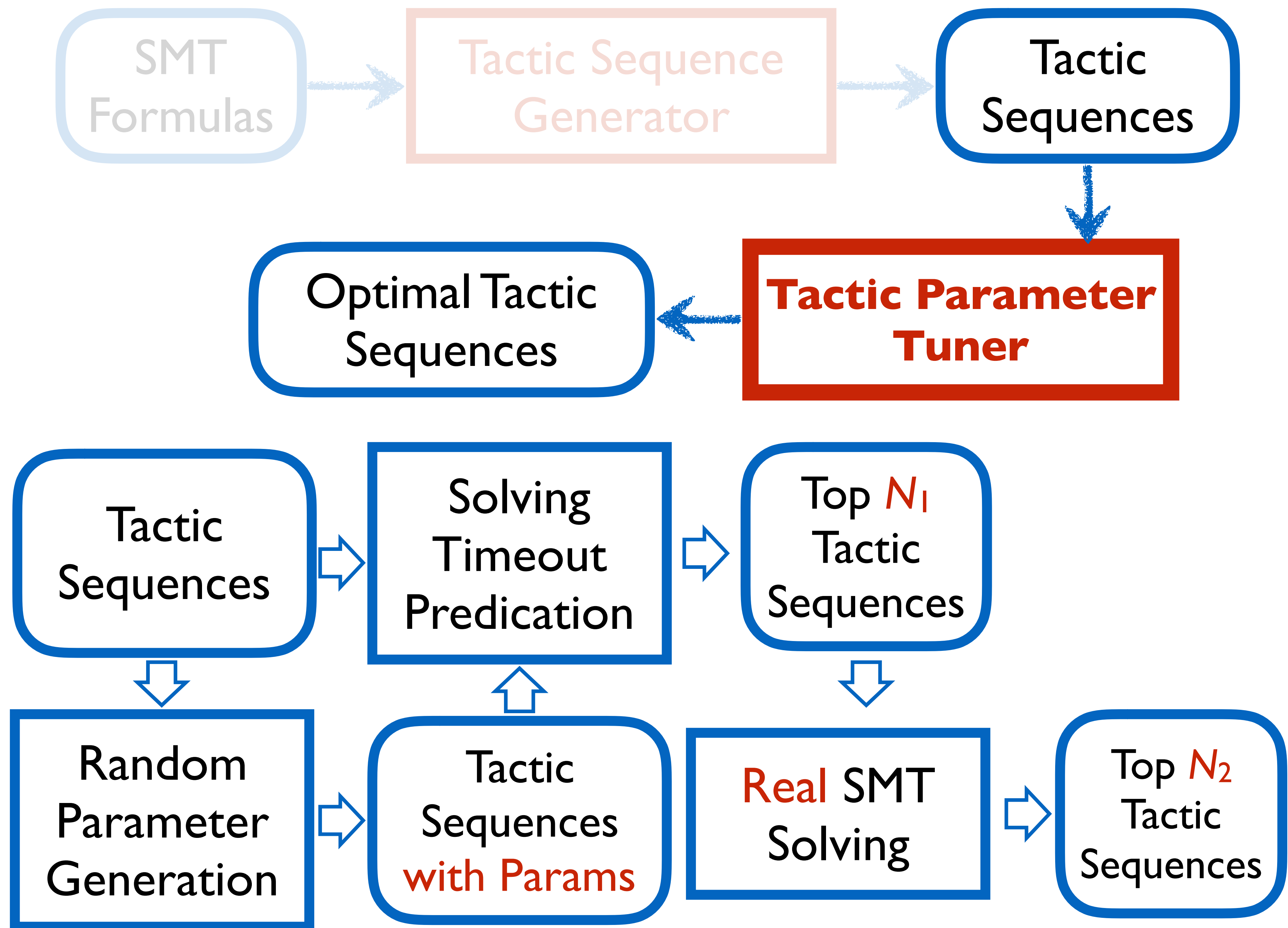


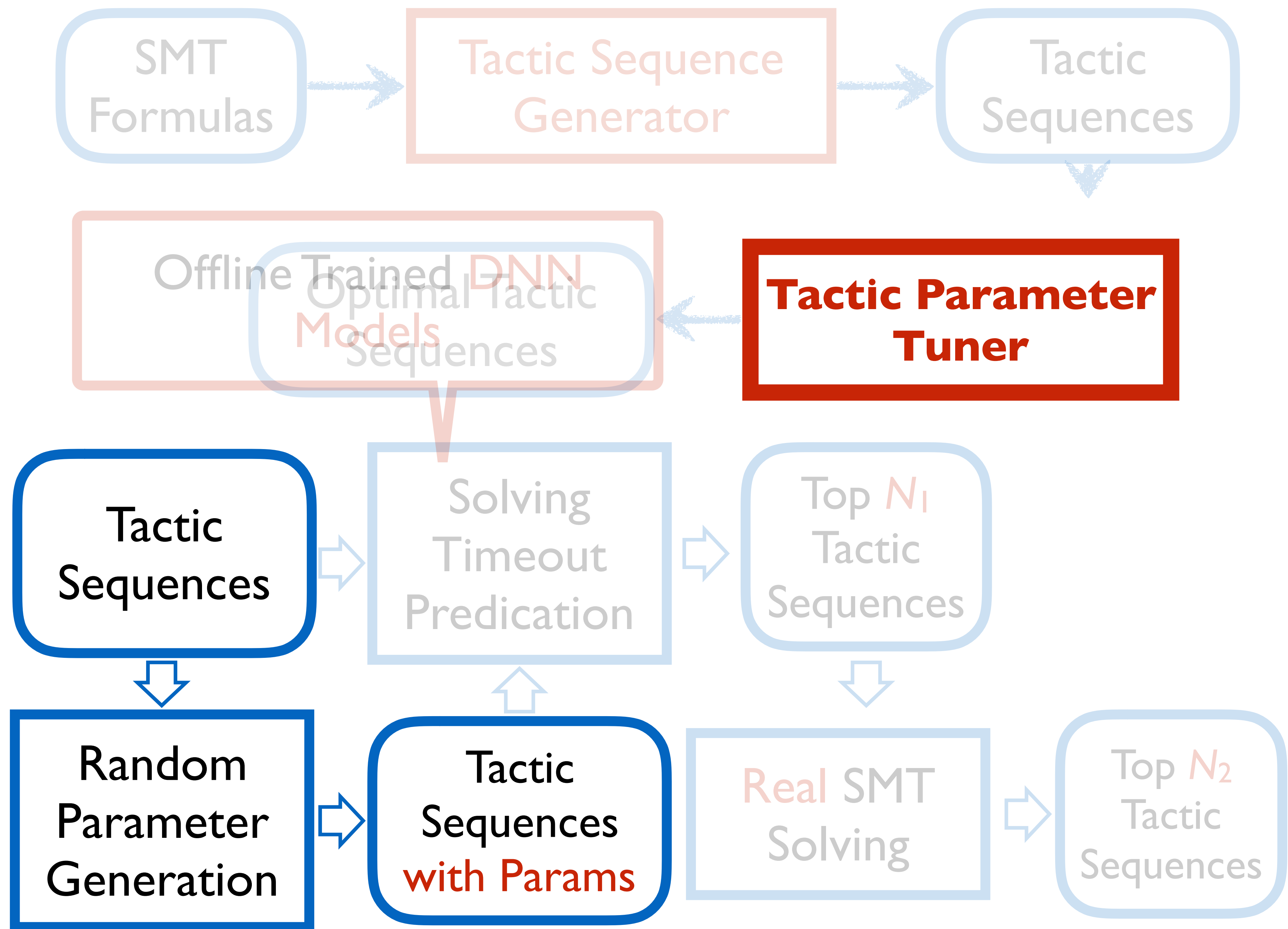
Synthesizer's Details

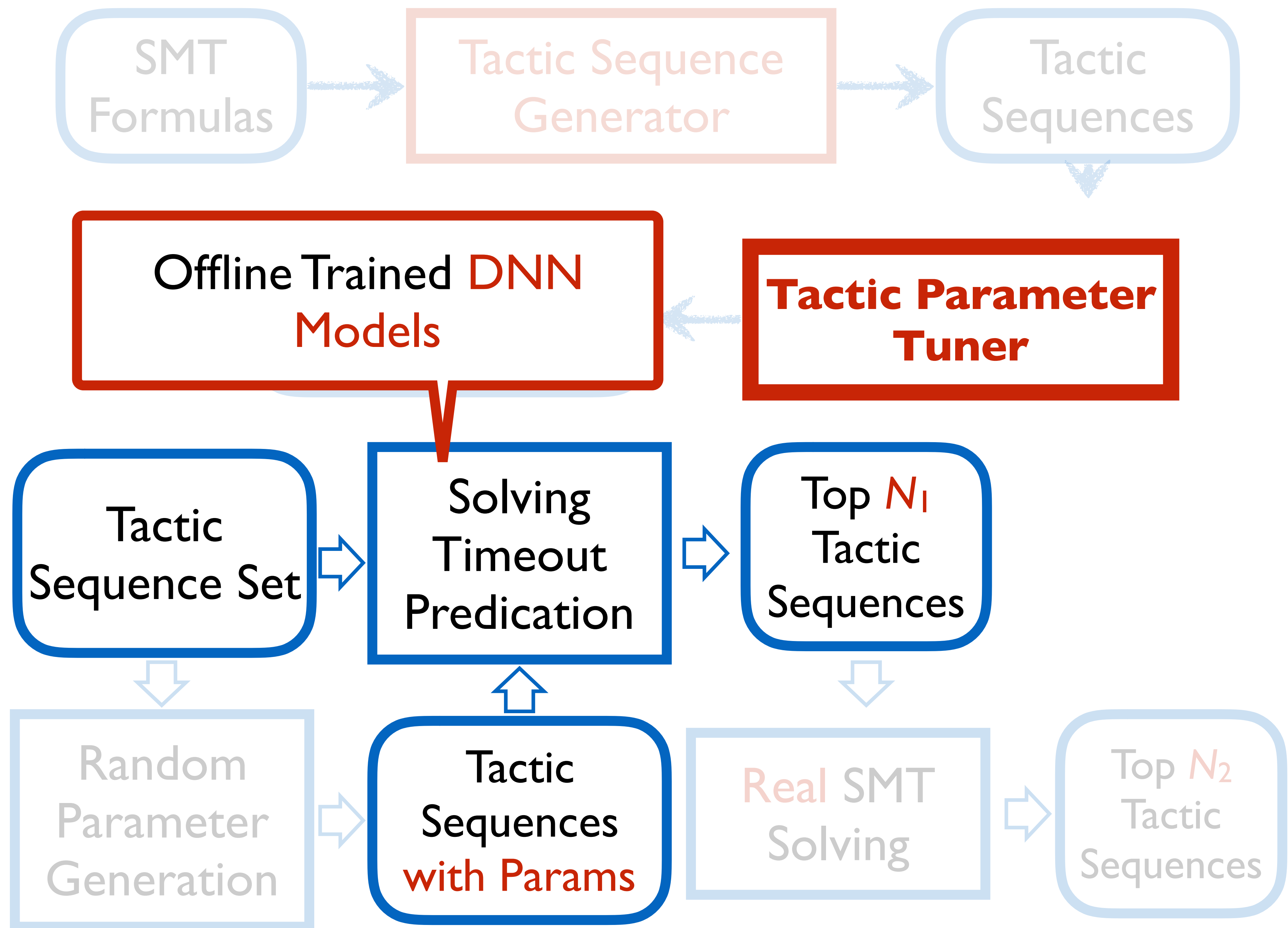


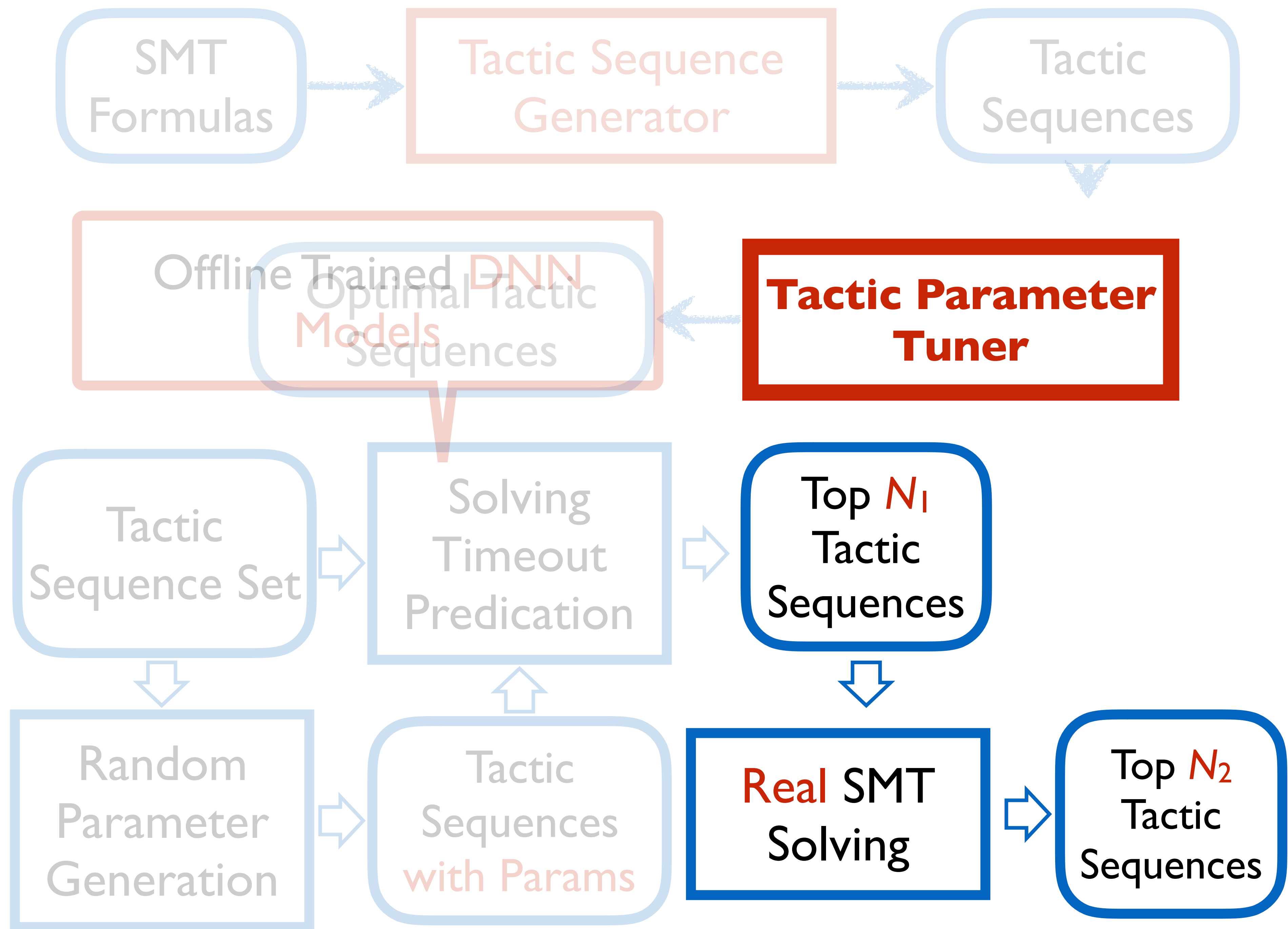
Synthesizer's Details











Optimal Tactic Sequences

Tactic Parameter Tuner

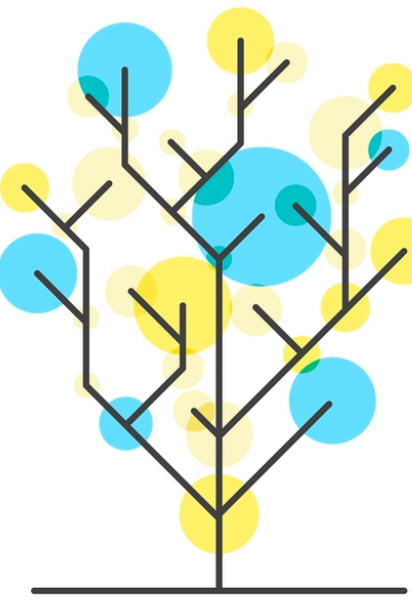
Strategy Generation

Solving Strategy

Optimal Tactic Sequences

Prob-based Predicates

Validation SMT formulas

 **Decision-Tree Based Learning**

Entropy-based solving cost estimation:

$$\begin{aligned} &(|S'_v \downarrow c|/|S'_v|) \times \mathcal{H}_{TS}(S'_v \downarrow c) \\ &+ \\ &(|S'_v \downarrow \neg c|/|S'_v|) \times \mathcal{H}_{TS}(S'_v \downarrow \neg c) \end{aligned}$$

A Composite Solving Strategy

Evaluation

- Research questions
 - Effectiveness
 - Queries (solved formulas)
 - Paths
 - Generalization ability

Evaluation

- Implementation
 - KLEE with Z3
 - Pytorch for ML models
 - JPF-based concolic engine

Evaluation

- Implementation
 - KLEE with Z3
 - Pytorch for ML models
 - JPF-based concolic engine

The deep learning models trained for C programs are directly used for Java programs

ML Models Training

- DRL model for generating tactic sequences
 - 14 **randomly** selected Coreutils programs
 - 300 SMT formulas **randomly** from each
 - 4200 in total

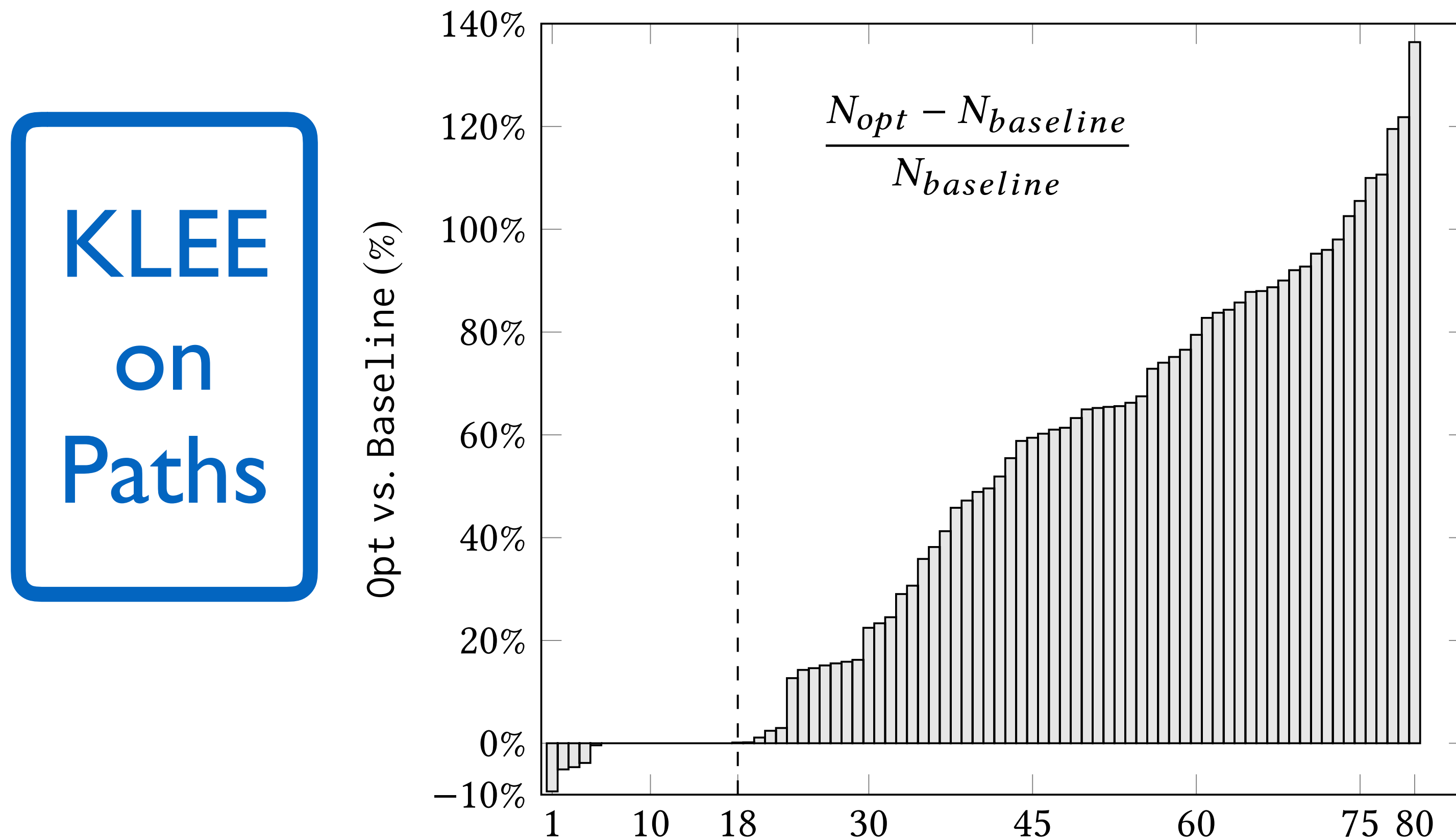
ML Model Training

- DNN models for predicating timeout
 - SAT, SMT, QFNRA-NLSAT, QFNRA
 - 8 randomly selected Coreutils programs
 - QF_BV, QF_ABV, QF_ABVFP, QF_BVFP SMT-LIB2 benchmarks
 - Timeout threshold: 30 seconds

Evaluation

- Benchmark for KLEE
 - 80 Coreutils programs
- Benchmark for Java concolic engine
 - 34 open-source Java programs
 - 327506 SLOCs in total

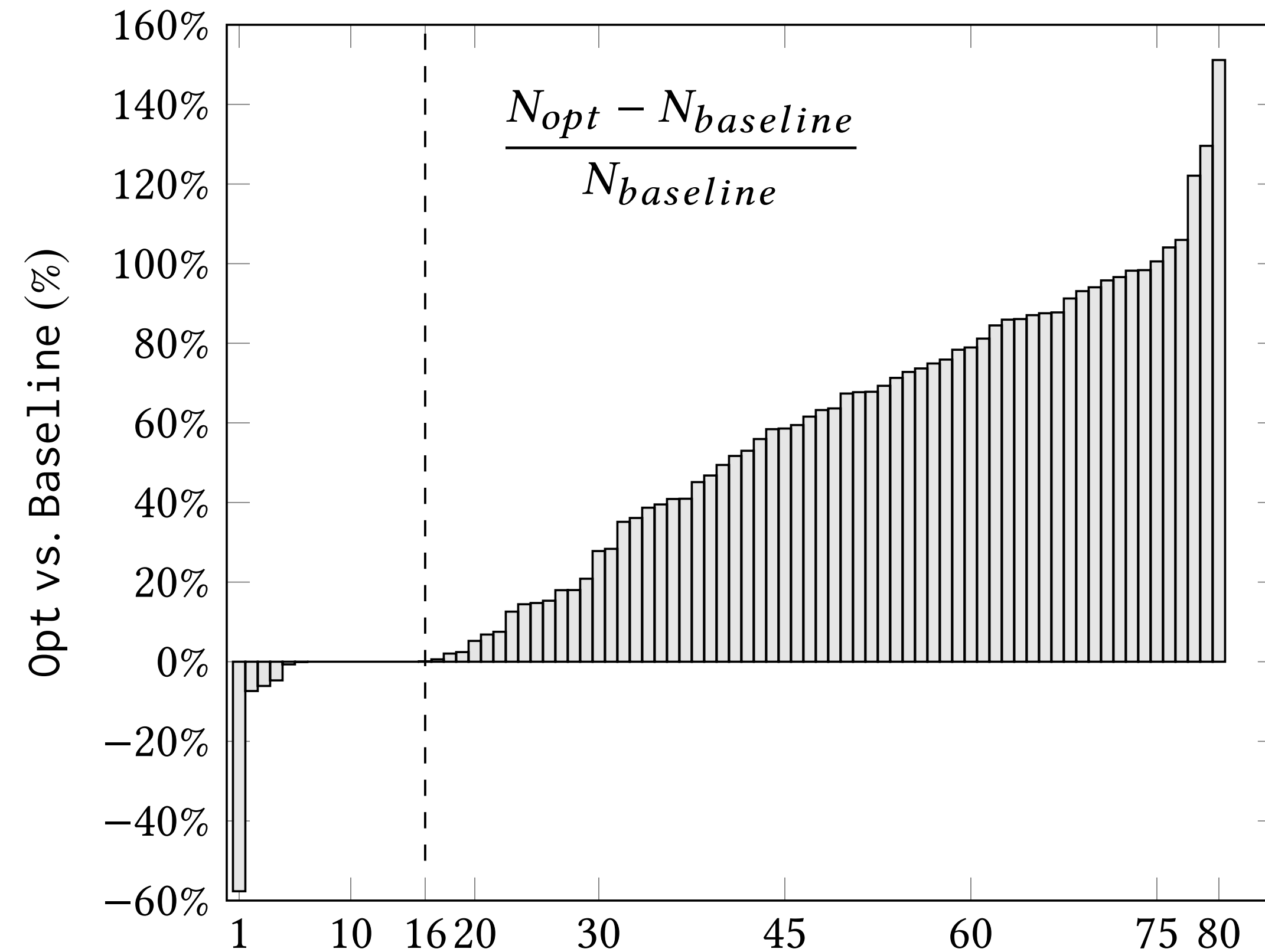
Results of Effectiveness



Improve the explored paths for 63 programs, 66.11% on average

Results of Effectiveness

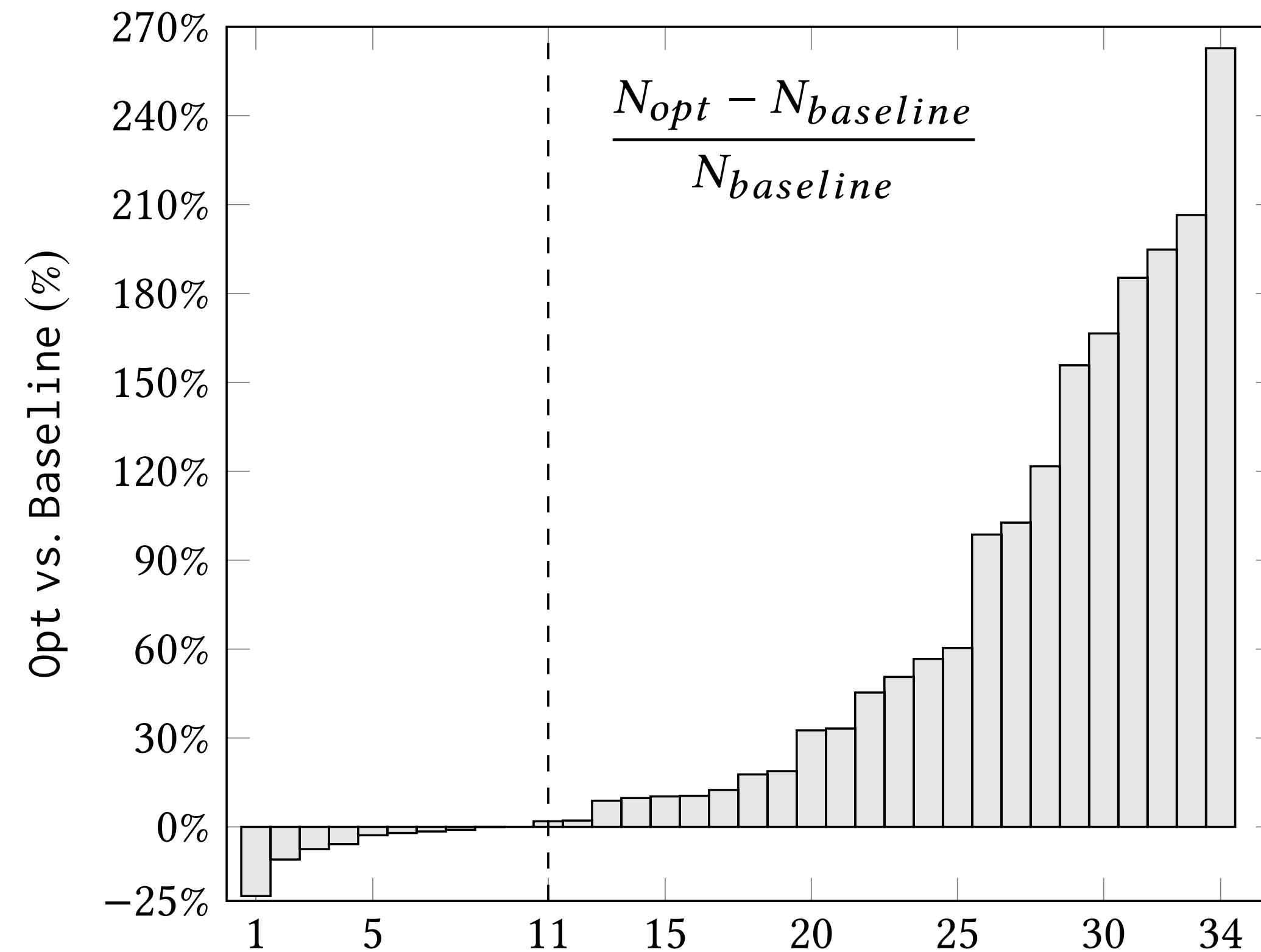
KLEE on
Queries



Improve the queries for 65 programs, 58.76% on average

Results of Generalization Ability

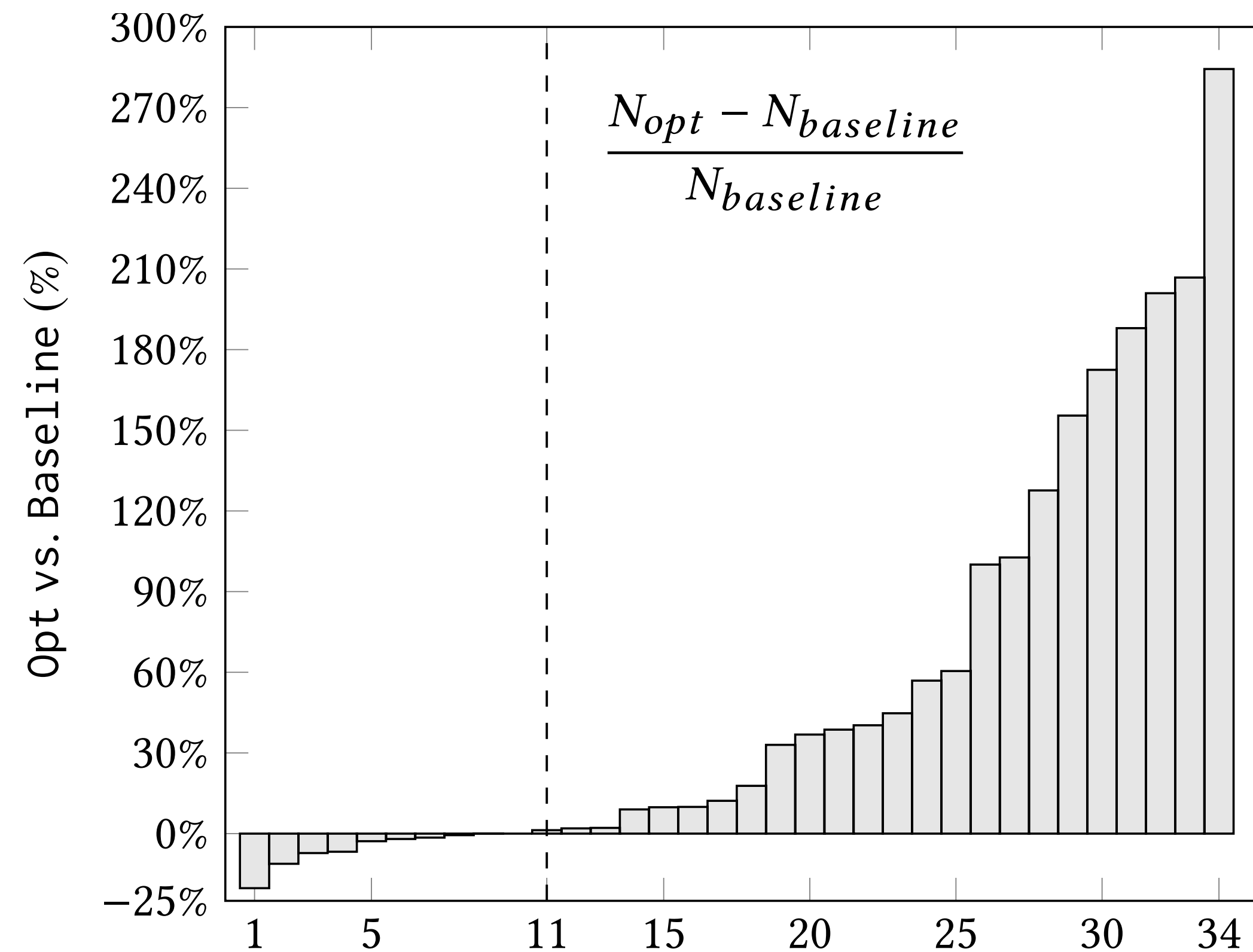
Paths on
Java
Programs



Improve the explored paths for 24 programs, 102.6% on average

Results of Generalization Ability

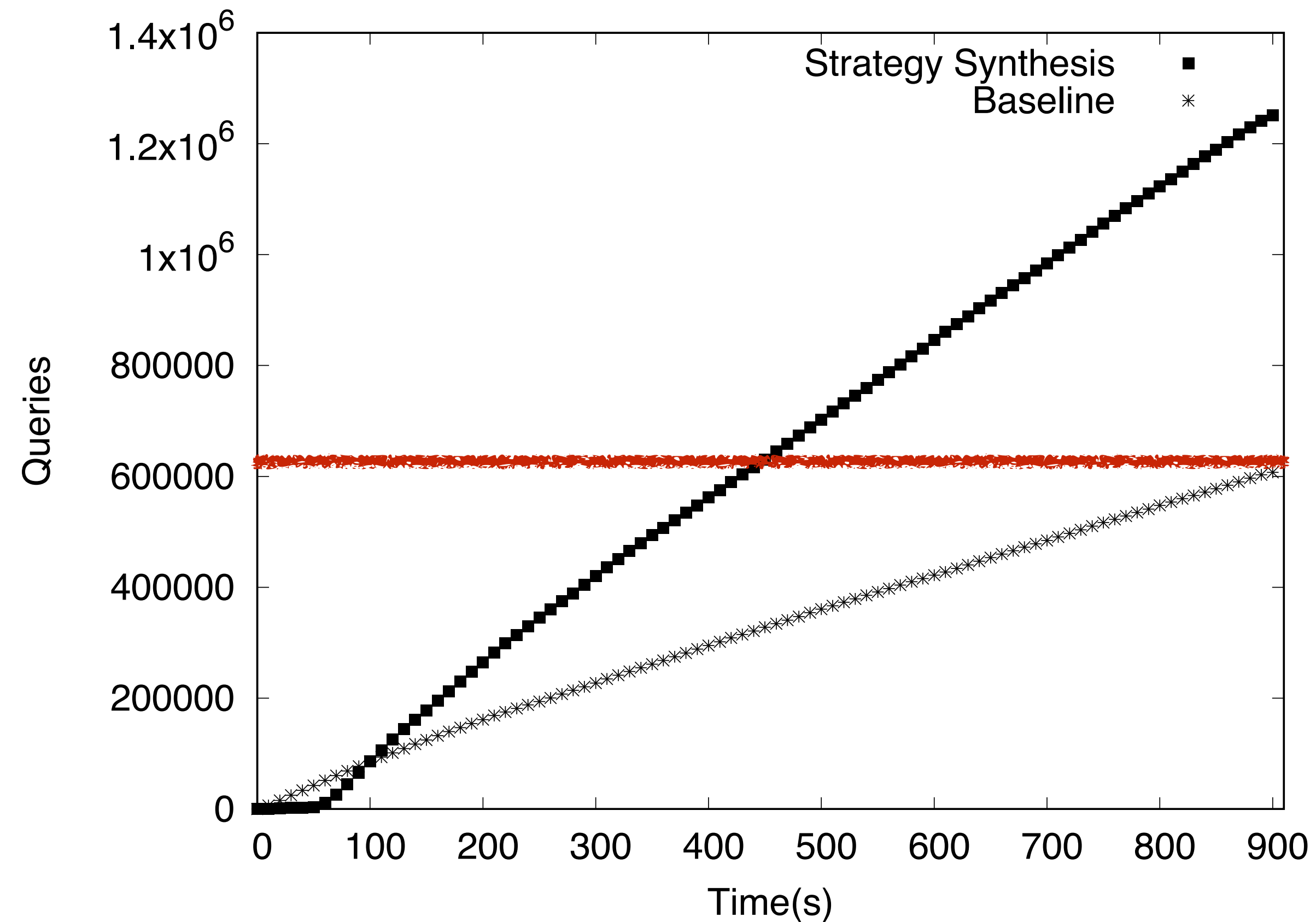
Queries
on Java
Programs



Improve the queries for 24 programs, 100.24% on average

Results of Generalization Ability

Trend of
Queries
on Java
Programs

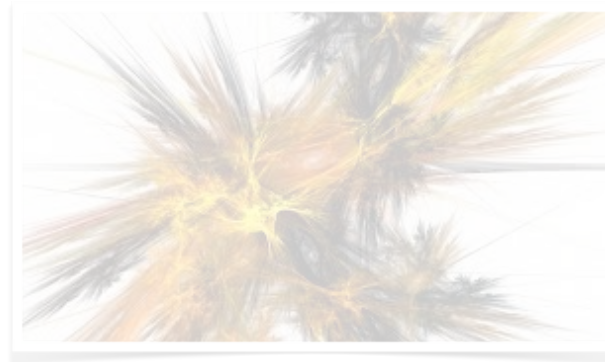


2.07x speedup for solving the same amount of queries

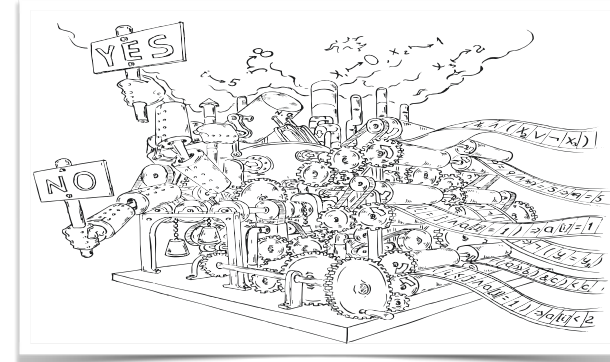
Conclusion

Our Work's Target

Path explosion



Constraint Solving

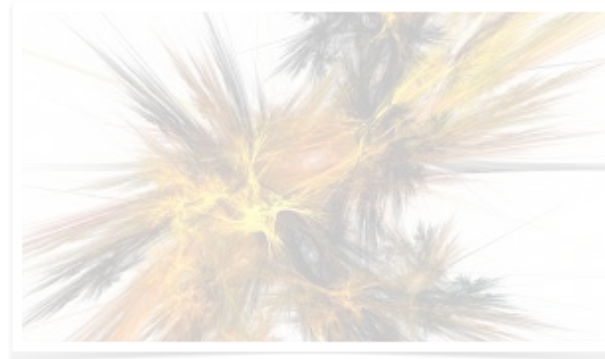


Decision Procedures An Algorithmic Point of View, Second Edition, 2016

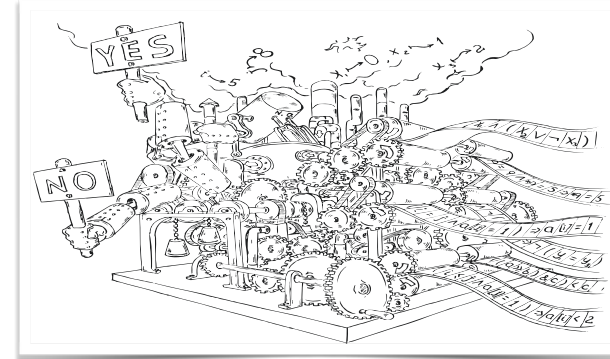
Conclusion

Our Work's Target

Path explosion

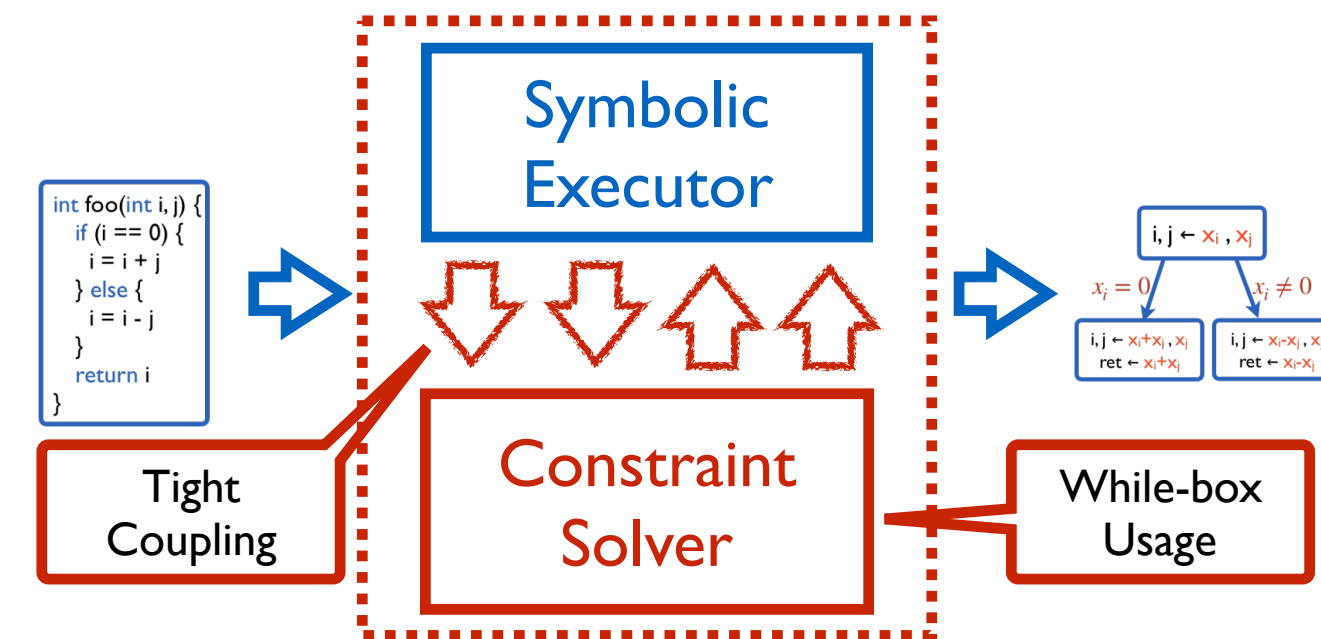


Constraint Solving



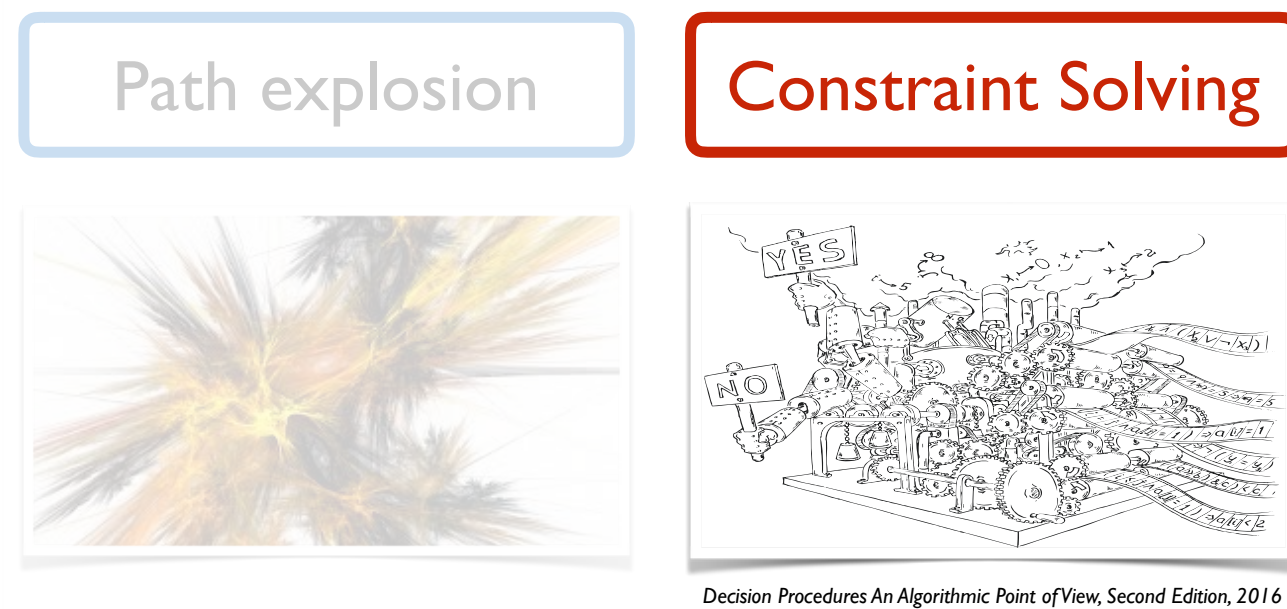
Decision Procedures An Algorithmic Point of View, Second Edition, 2016

Our Argument

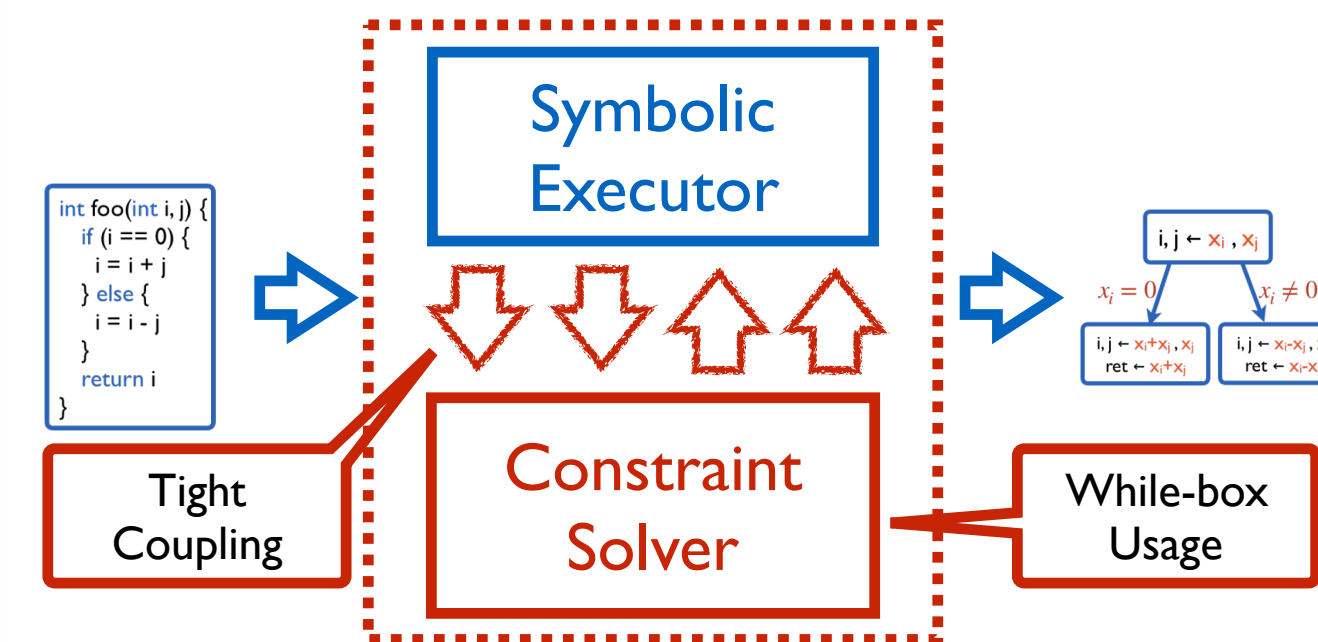


Conclusion

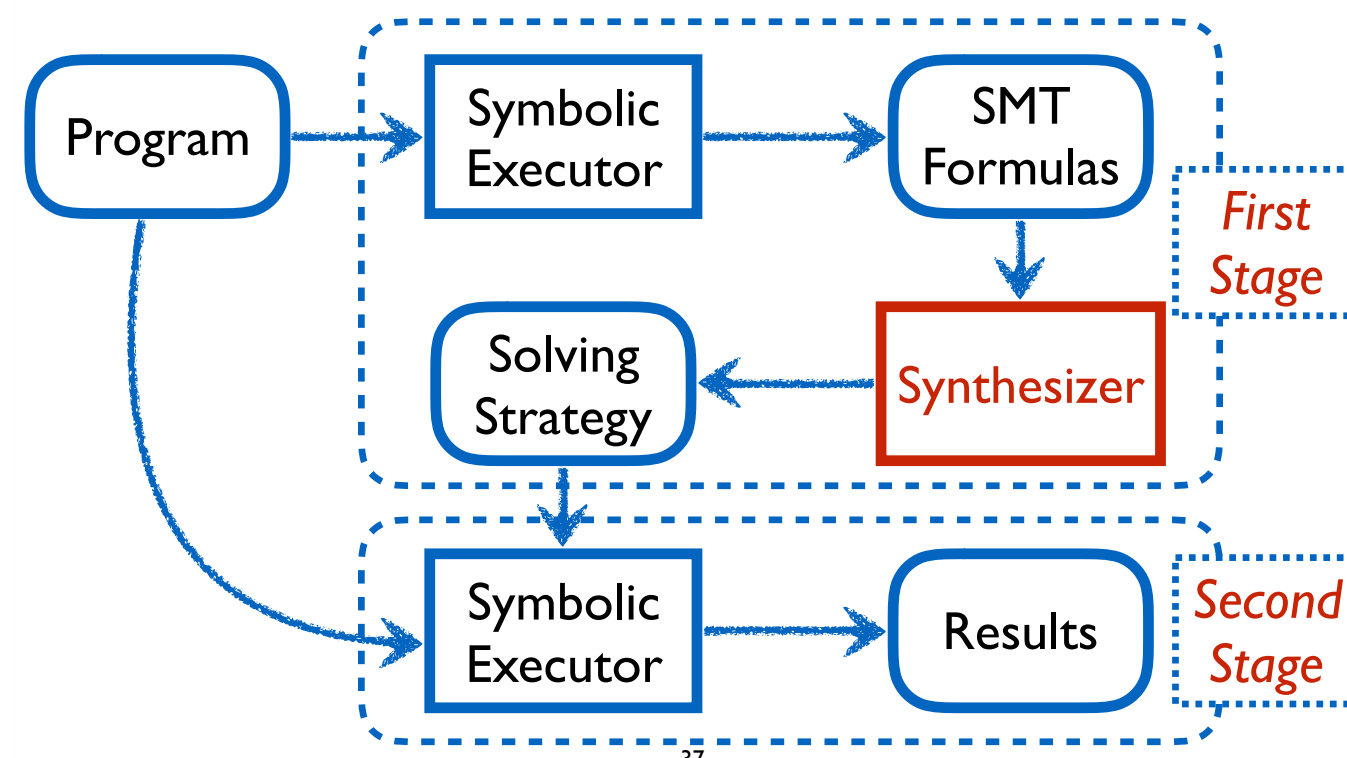
Our Work's Target



Our Argument

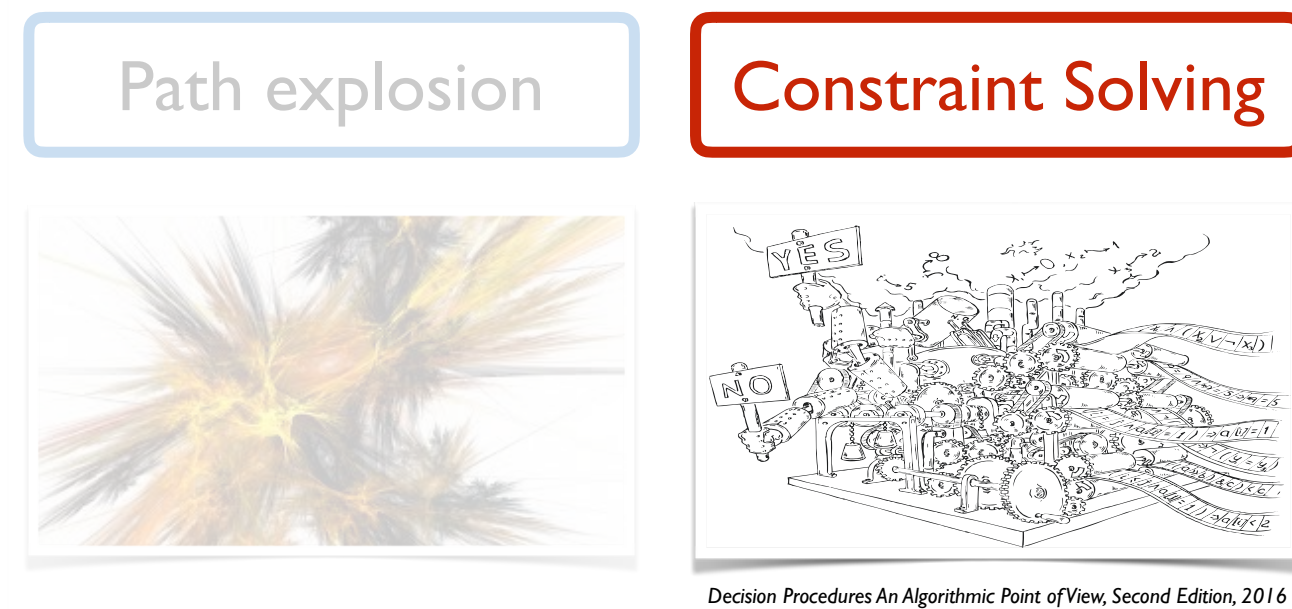


High-Level Framework

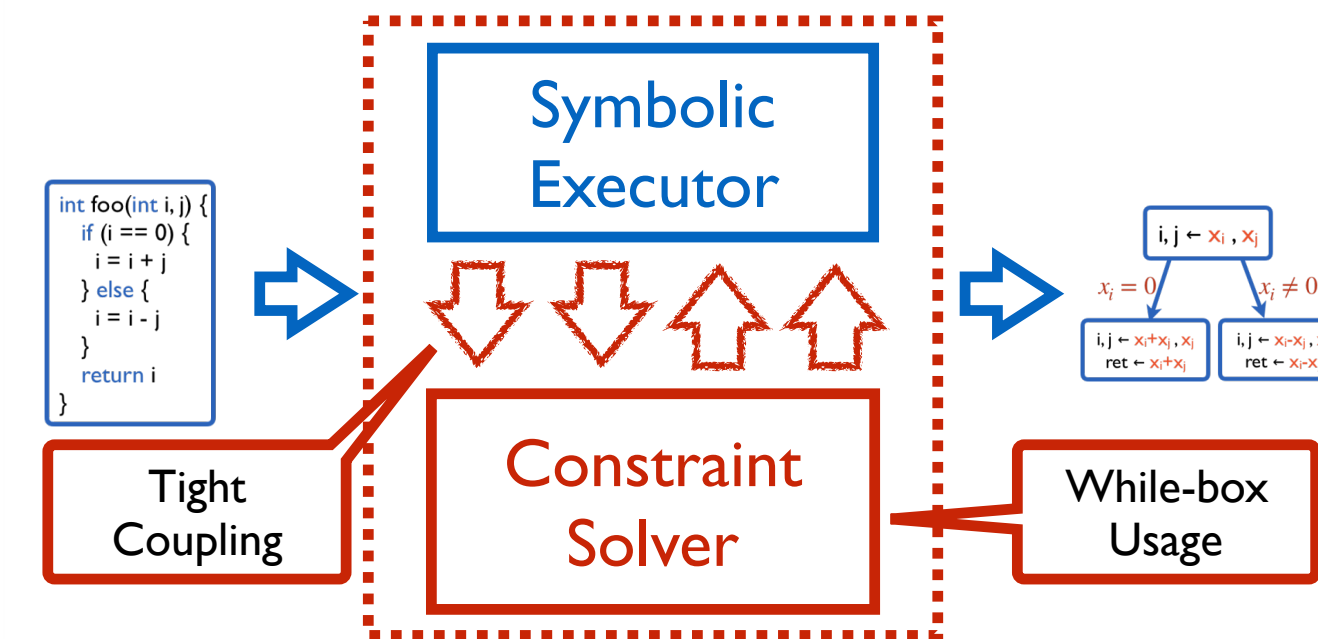


Conclusion

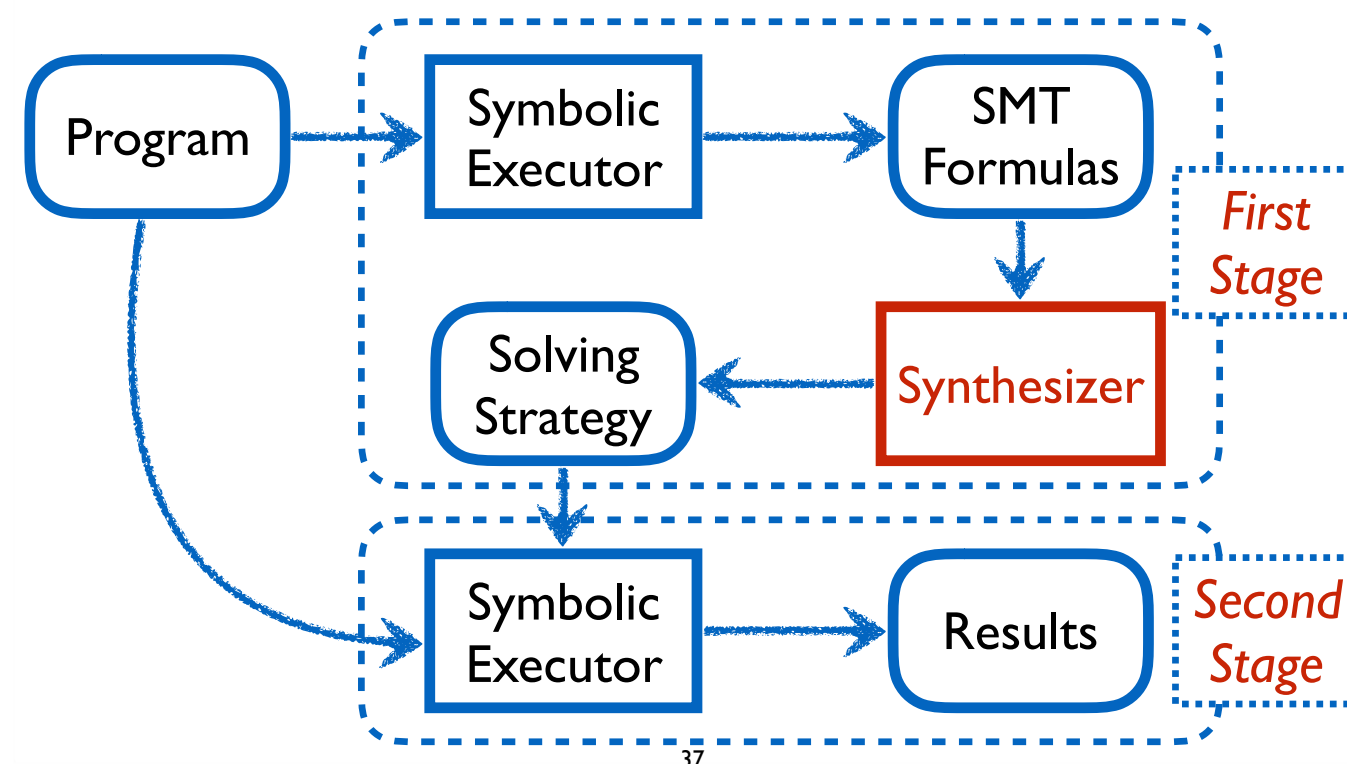
Our Work's Target



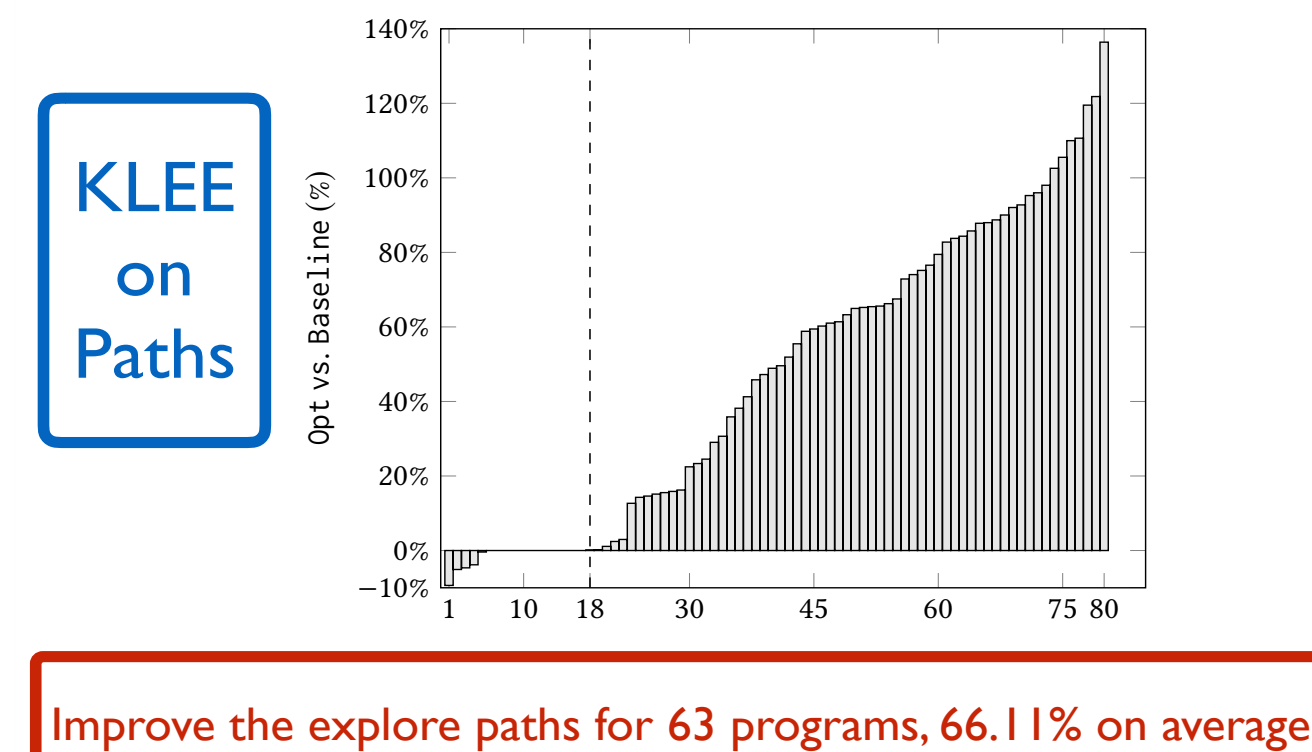
Our Argument



High-Level Framework



Results of Effectiveness





Thank you!
Q&A