# Effective Test Execution for SPLs

Sabrina Souto

(Supervisor: Marcelo d’Amorim)

Federal University of Pernambuco - UFPE

## General Assumption

- We assume that tests are not designed for one specific product
  - A test can run against several products!

## General Problem

- Which products to select for running a particular SPL test against?
  - One can miss fault-revealing products in this set ⇨ **Soundness**
  - One can add irrelevant products in this set ⇨ **Efficiency**

## Previous selection strategies

- **Black-box testing**

  (e.g., CIT, sampling, exhaustive…)
  [Cohen et. al., ROSETEA 2006]; [Yilmaz et. al., TSE 2006] ;
  [Perrouin et. al., ICST 2010]; [Cabral et. al., SPLC 2010] …

- **White-box testing**

  [Reisner et. al., ICSE 2010], [Kim et. al., ISSRE 2012];
  [Kim et. al., AOSD 2011] …

- **Gray-box testing**

  [Garvin et. al., ISSRE 2011]; [Shi et. al., FASE 2012];
  [Song et. al., ICSE 2012] …

## Hypothesis

Few valid configurations are reachable from individual tests

Substantiated from results in literature:
[Reisner *et. al*., ICSE 2010] and [Kim *et. al*., AOSD 2011]

## Questions (revisited in the end)

- Is efficiency important in this context?
- Is combinatorial explosion (scalability) relatively less important for testing?
- Is soundness important for testing?

# Questions (revisited in the end)

- Is efficiency important in this context?
- Is combinatorial explosion (scalability) relatively less important for testing?
- Is soundness important for testing?

## Questions (revisited in the end)

- Is efficiency important in this context?
- Is combinatorial explosion (scalability) relatively less important for testing?
- Is soundness important for testing?

## Questions (revisited in the end)

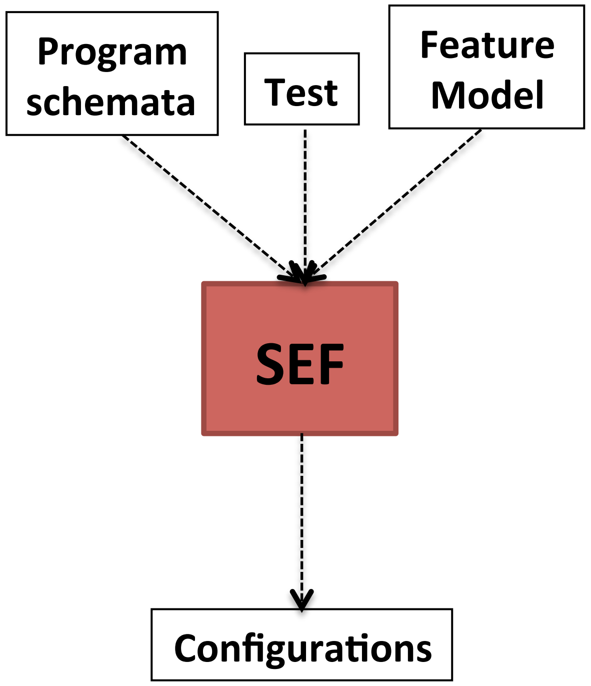
- Is efficiency important in this context?
- Is combinatorial explosion (scalability) relatively less important for testing?
- Is soundness important for testing?

**Our answer is YES to all questions!**

# Symbolic Execution of Features – SEF

Symbolic Execution has been proposed in the 70's

Reisner *et al*. proposed SEF in 2010 (ICSE)



## Program Schemata

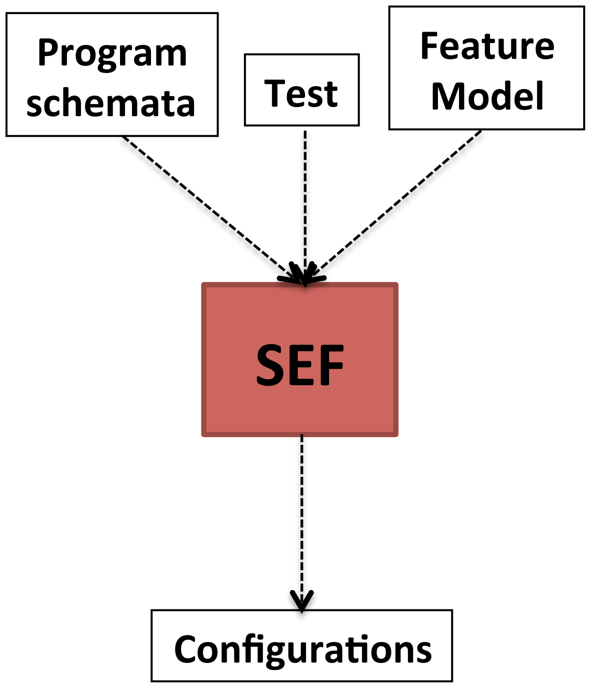
```
#IFDEF FEATURE_1
    x = 10;
#ELIF
    x = 20;
#ENDIF
```

```
static boolean feature1;
if(feature1){x = 10;} else { x = 20;}
```

# Symbolic Execution of Features – SEF

Symbolic Execution has been proposed in the 70's

Reisner *et al*. proposed SEF in 2010 (ICSE)



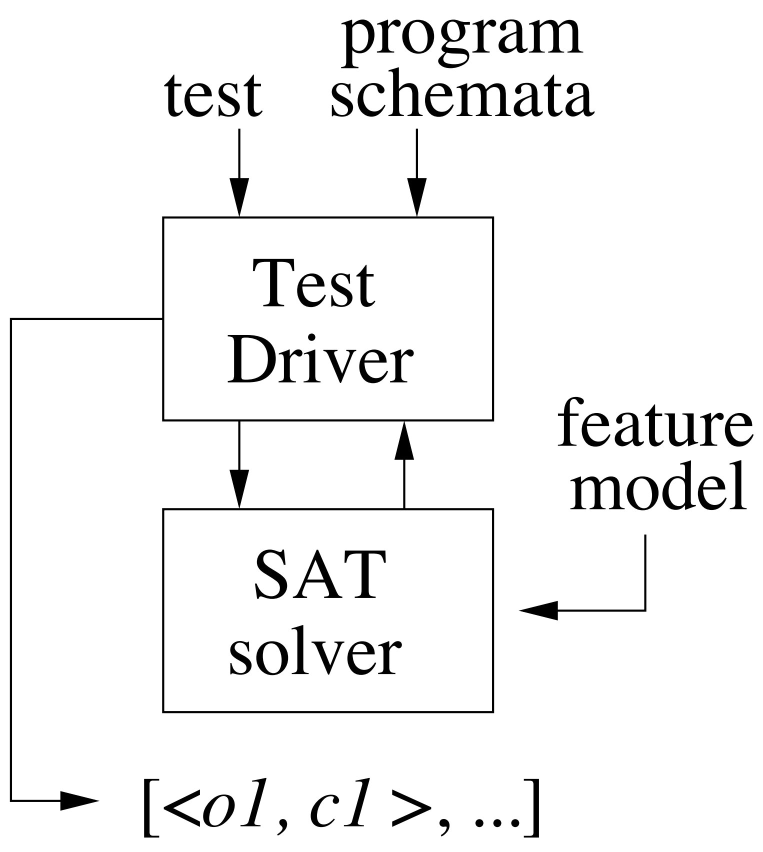
# Symbolic Execution of Features – SEF

Symbolic Execution has been proposed in the 70's

Reisner *et al.* proposed SEF in 2010 (ICSE)

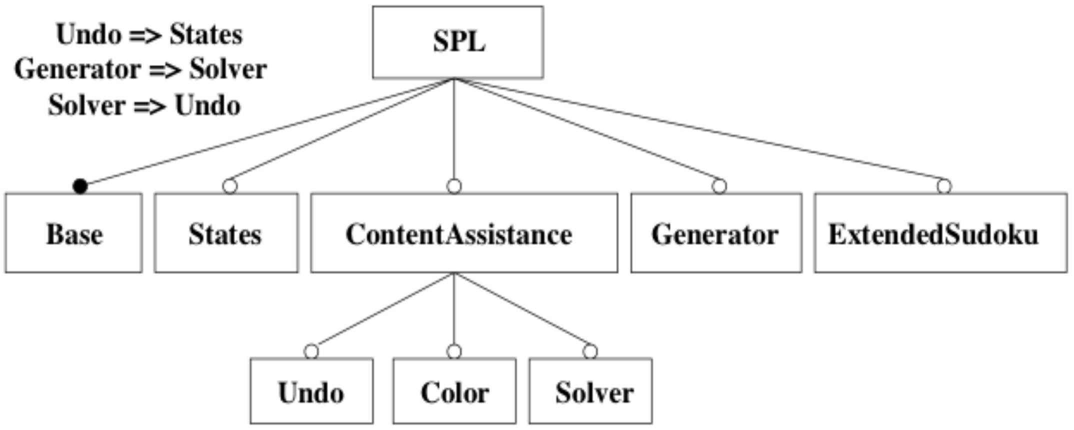
Our goal is to apply SEF to reach sound and efficient testing for SPLs

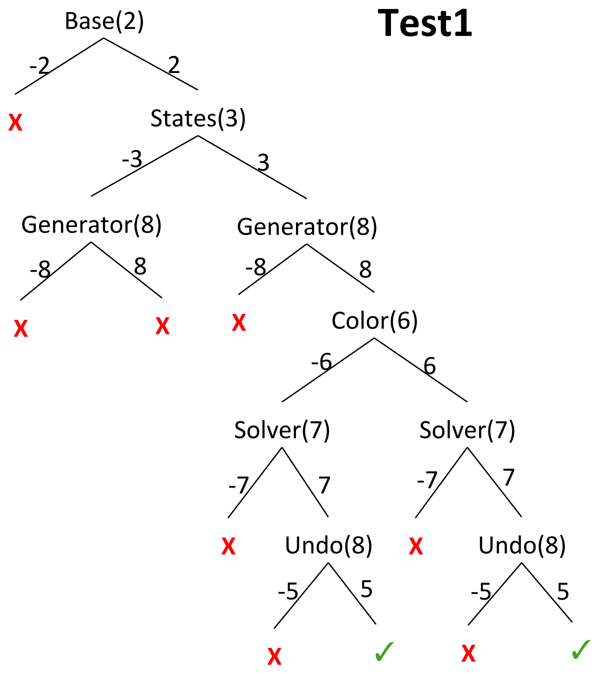
Insight for efficient implementation is to partition state in two: one concrete and one symbolic.



# Example

Sudoku Feature Model





**Valid paths:**

[2 3 8 6 7 5]

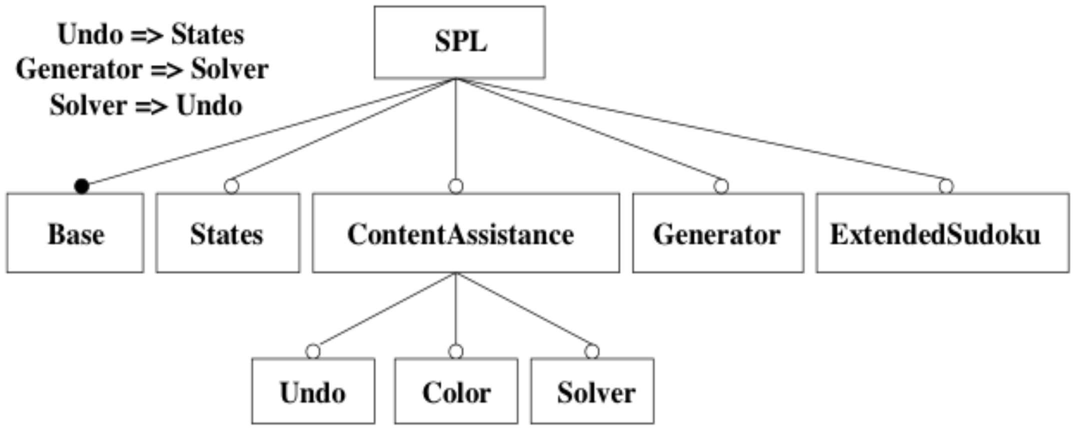
[2 3 8 -6 7 5]

**Invalid paths:**

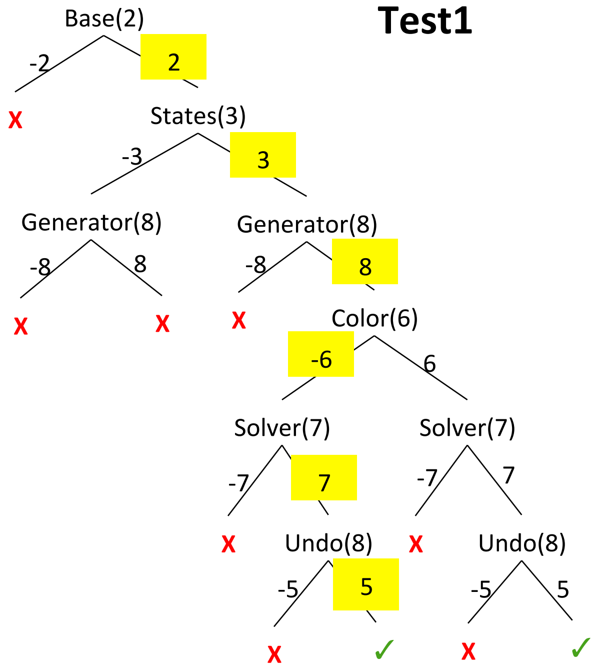
| | |
|---|---|
| [-2] | [2 3 8 -6 -7] |
| [2 -3 8] | [2 3 8 6 -7] |
| [2 -3 -8] | [2 3 8 -6 7 -5] |
| [2 3 -8] | [2 3 8 6 7 -5] |

# Example

## Sudoku Feature Model



**Test1**



**Valid paths:**

[2 3 8 6 7 5]

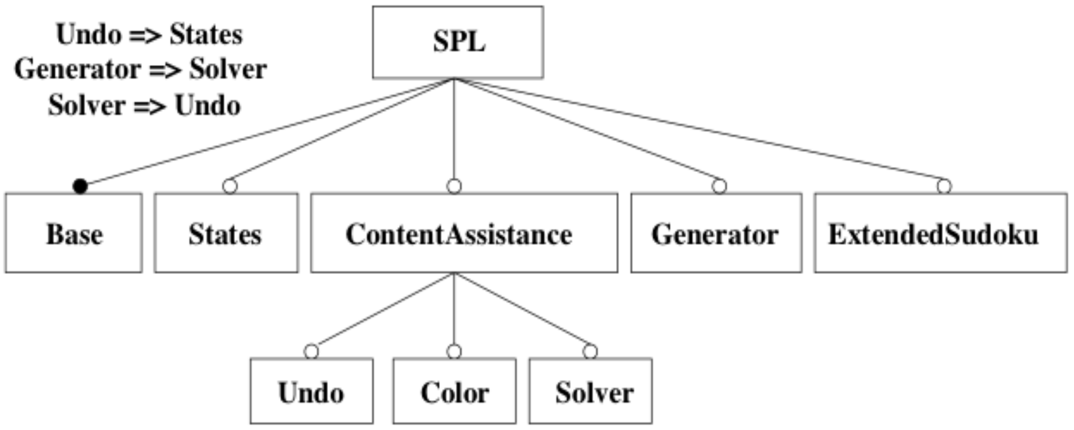
[2 3 8 -6 7 5]

**Invalid paths:**

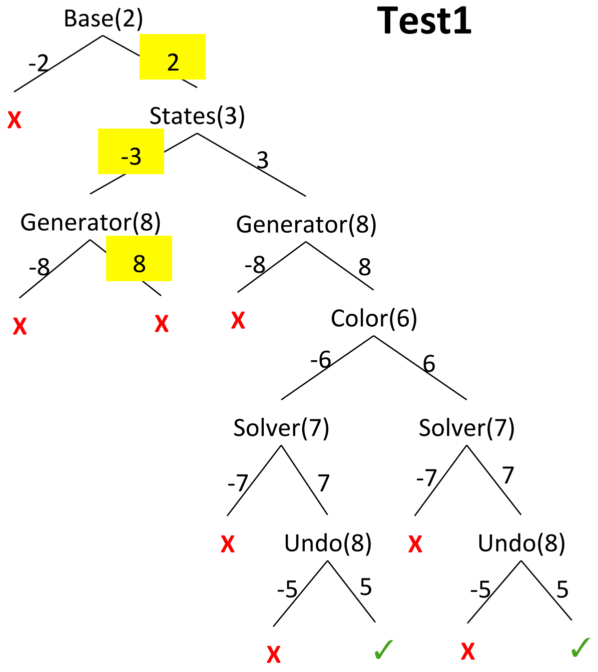
| | |
|---|---|
| [-2] | [2 3 8 -6 -7] |
| [2 -3 8] | [2 3 8 6 -7] |
| [2 -3 -8] | [2 3 8 -6 7 -5] |
| [2  3 -8] | [2 3 8  6 7 -5] |

# Example

## Sudoku Feature Model



**Test1**



**Valid paths:**

[2 3 8 6 7 5]

[2 3 8 -6 7 5]

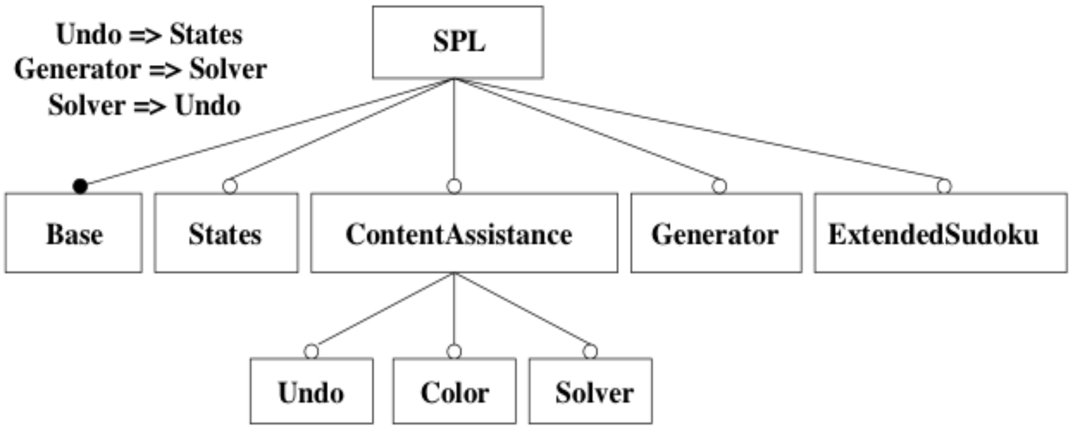
Feature Model

**Invalid paths:**

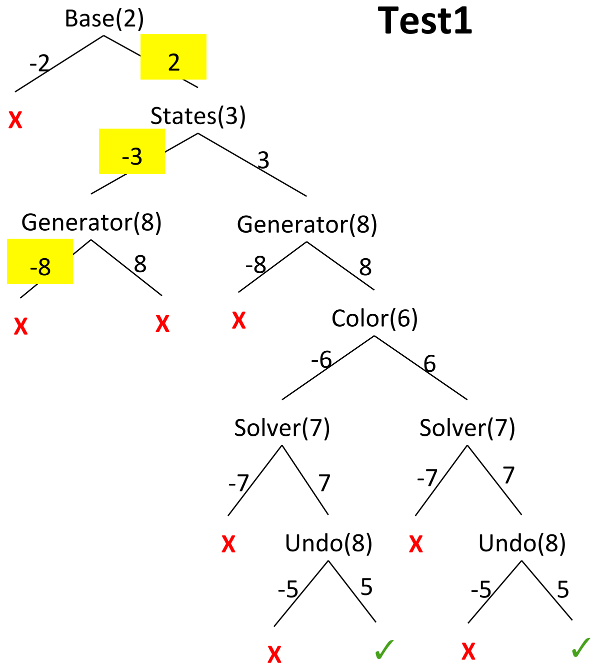
| | |
|---|---|
| [-2] | [2 3 8 -6 -7] |
| [2 -3 8] | [2 3 8 6 -7] |
| [2 -3 -8] | [2 3 8 -6 7 -5] |
| [2 3 -8] | [2 3 8 6 7 -5] |

# Example

## Sudoku Feature Model



**Test1**



**Valid paths:**

[2 3 8 6 7 5]

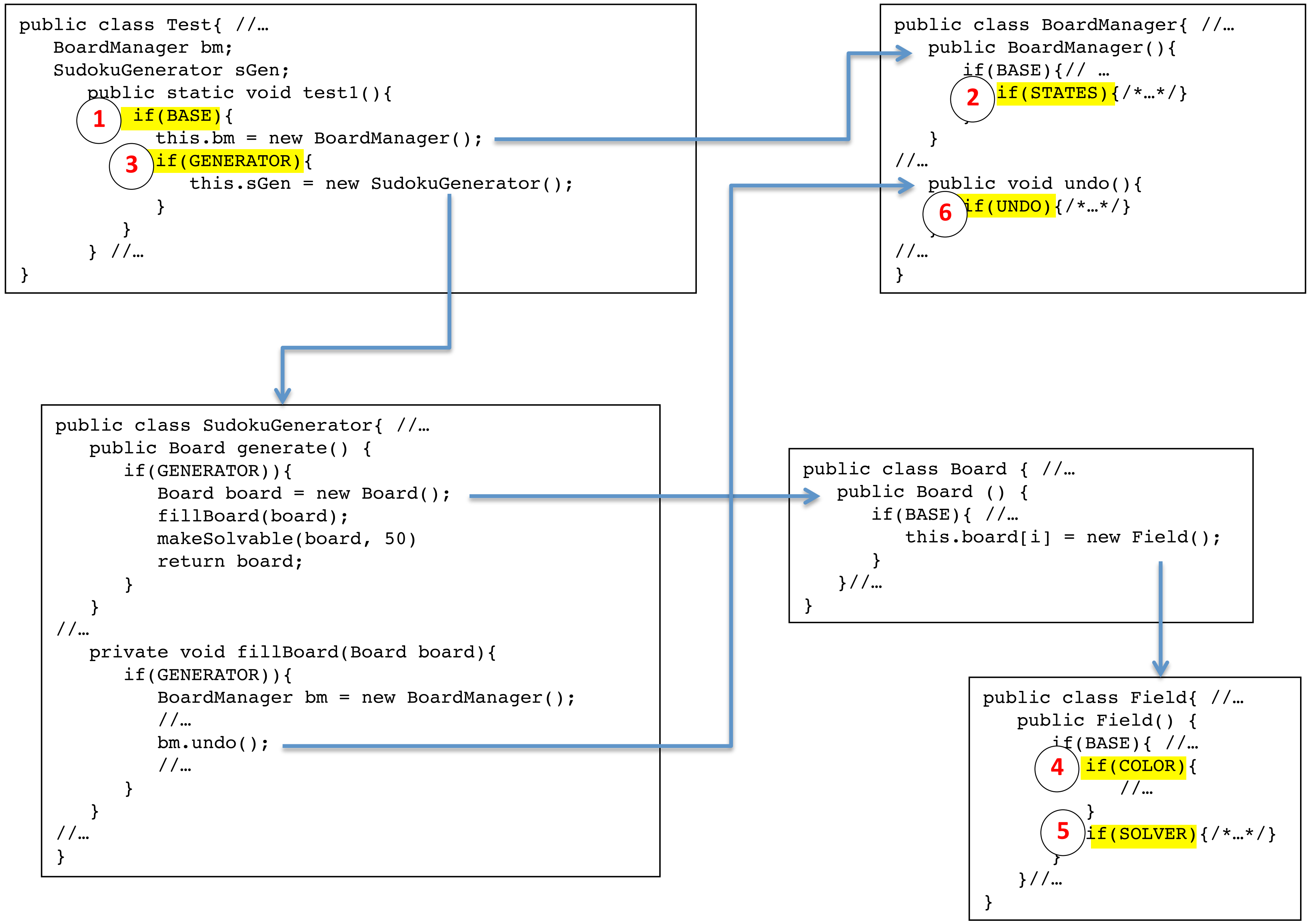
[2 3 8 -6 7 5]

Feature Model

Test

**Invalid paths:**

| | |
|---|---|
| [-2] | [2 3 8 -6 -7] |
| [2 -3 8] | [2 3 8 6 -7] |
| [2 -3 -8] | [2 3 8 -6 7 -5] |
| [2 3 -8] | [2 3 8 6 7 -5] |



```
public class Test{ //…
   BoardManager bm;
   SudokuGenerator sGen;
      public static void test1(){
          if(BASE){
            this.bm = new BoardManager();
            if(GENERATOR){
               this.sGen = new SudokuGenerator();
            }
         }
      } //…
}
```

```
public class BoardManager{ //…
   public BoardManager(){
      if(BASE){//  …
         if(STATES){/*…*/}
      }
   }
//…
   public void undo(){
      if(UNDO){/*…*/}
   }
//…
}
```

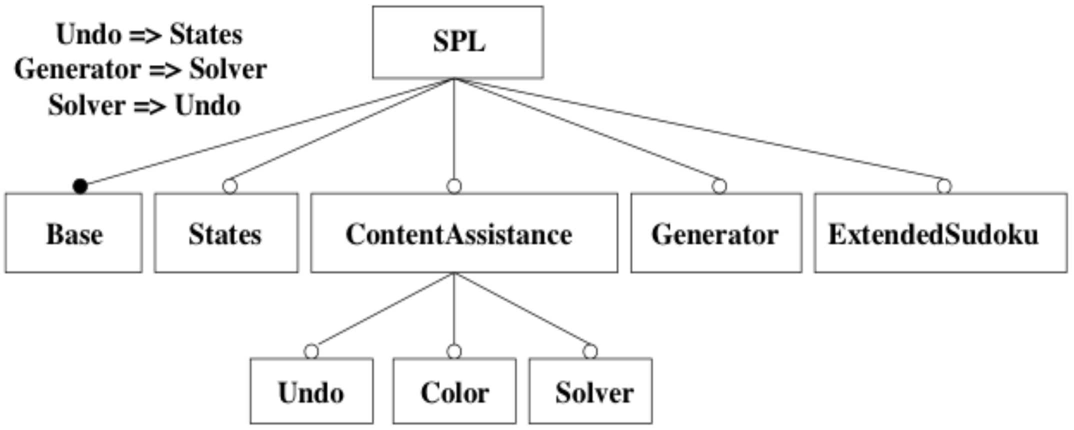
```
public class SudokuGenerator{ //…
   public Board generate() {
      if(GENERATOR)){
         Board board = new Board();
         fillBoard(board);
         makeSolvable(board, 50)
         return board;
      }
   }
//…
   private void fillBoard(Board board){
      if(GENERATOR)){
         BoardManager bm = new BoardManager();
         //…
         bm.undo();
         //…
      }
   }
//…
}
```

```
public class Board { //…
   public Board () {
      if(BASE){ //…
         this.board[i] = new Field();
      }
   }//…
}
```

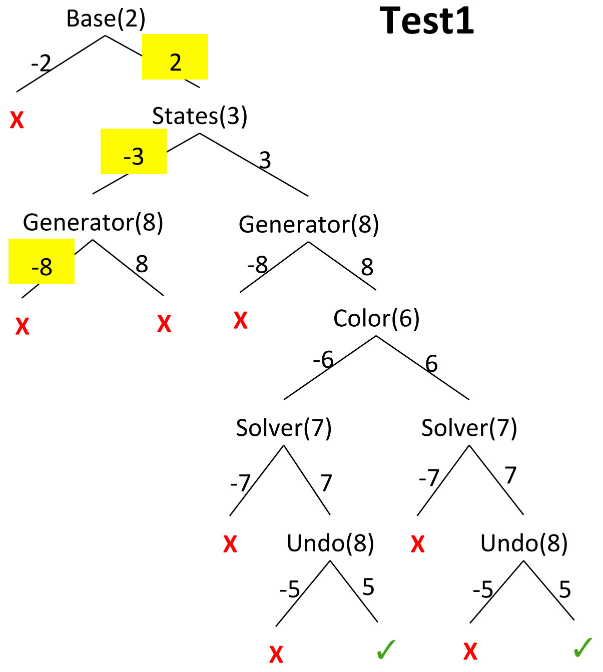
```
public class Field{ //…
   public Field() {
      if(BASE){ //…
         if(COLOR){
            //…
         }
         if(SOLVER){/*…*/}
      }
   }//…
}
```

# Example

## Sudoku Feature Model



**Test1**



**Valid paths:**

[2 3 8 6 7 5]

[2 3 8 -6 7 5]

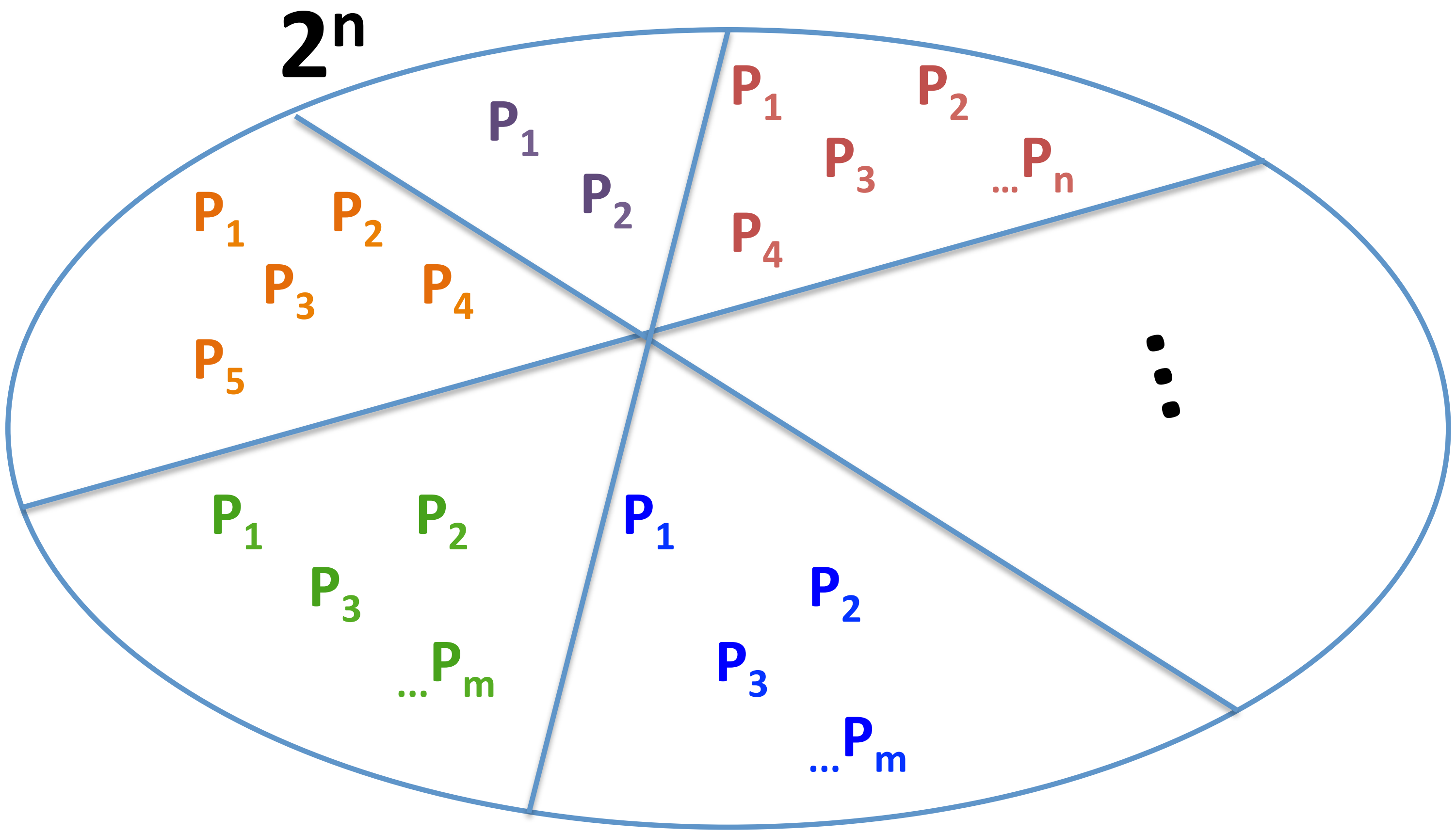
Feature Model

Test

**Invalid paths:**

| | |
|---|---|
| [-2] | [2 3 8 -6 -7] |
| [2 -3 8] | [2 3 8 6 -7] |
| [2 -3 -8] | [2 3 8 -6 7 -5] |
| [2 3 -8] | [2 3 8 6 7 -5] |

# Partition of configuration space

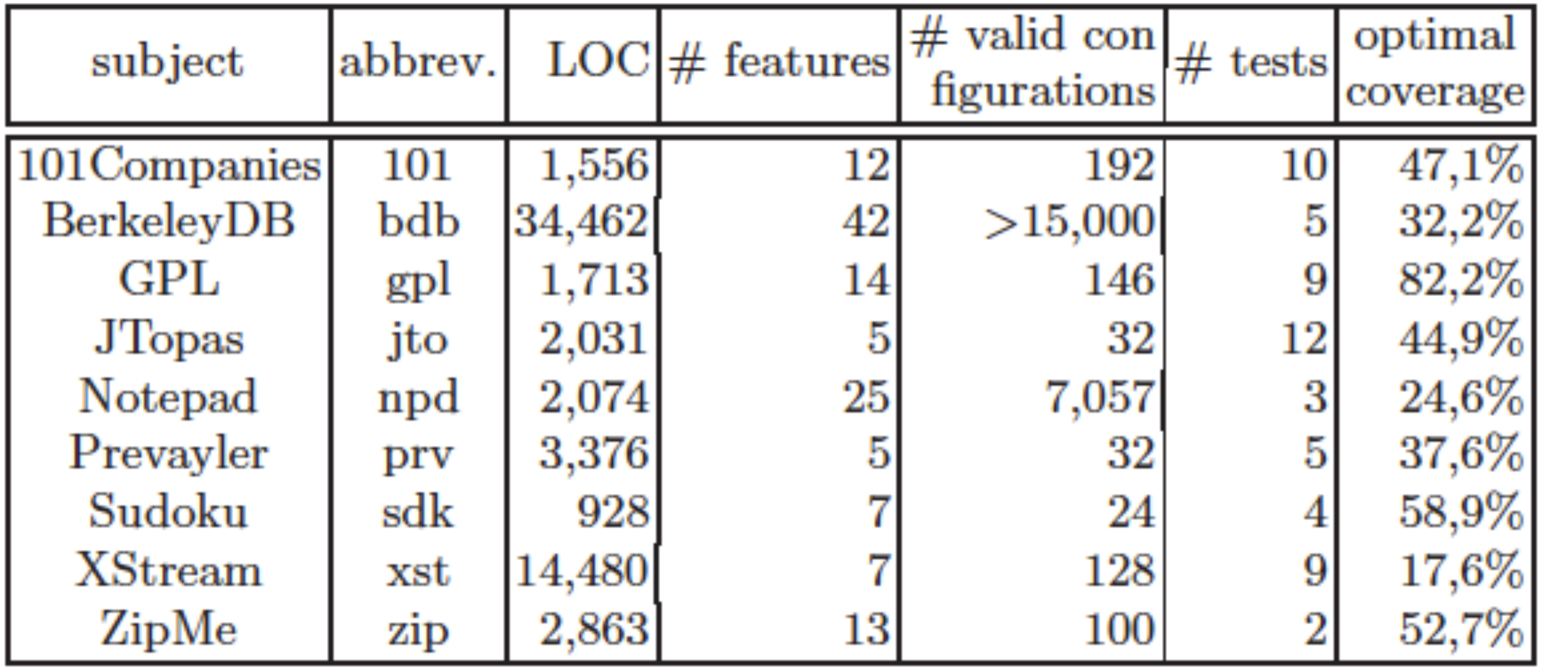

$2^n$
$P_1$ $P_2$
$P_1$ $P_2$ $P_3$ $\ldots P_n$ $P_4$
$P_1$ $P_2$ $P_3$ $P_4$ $P_5$
$P_1$ $P_2$ $P_3$ $\ldots P_m$
$P_1$ $P_2$ $P_3$ $\ldots P_m$
⋮

There is no distinction amongst the configurations from the same partition with respect to the **states** they elaborate during the test execution.

## Research Questions

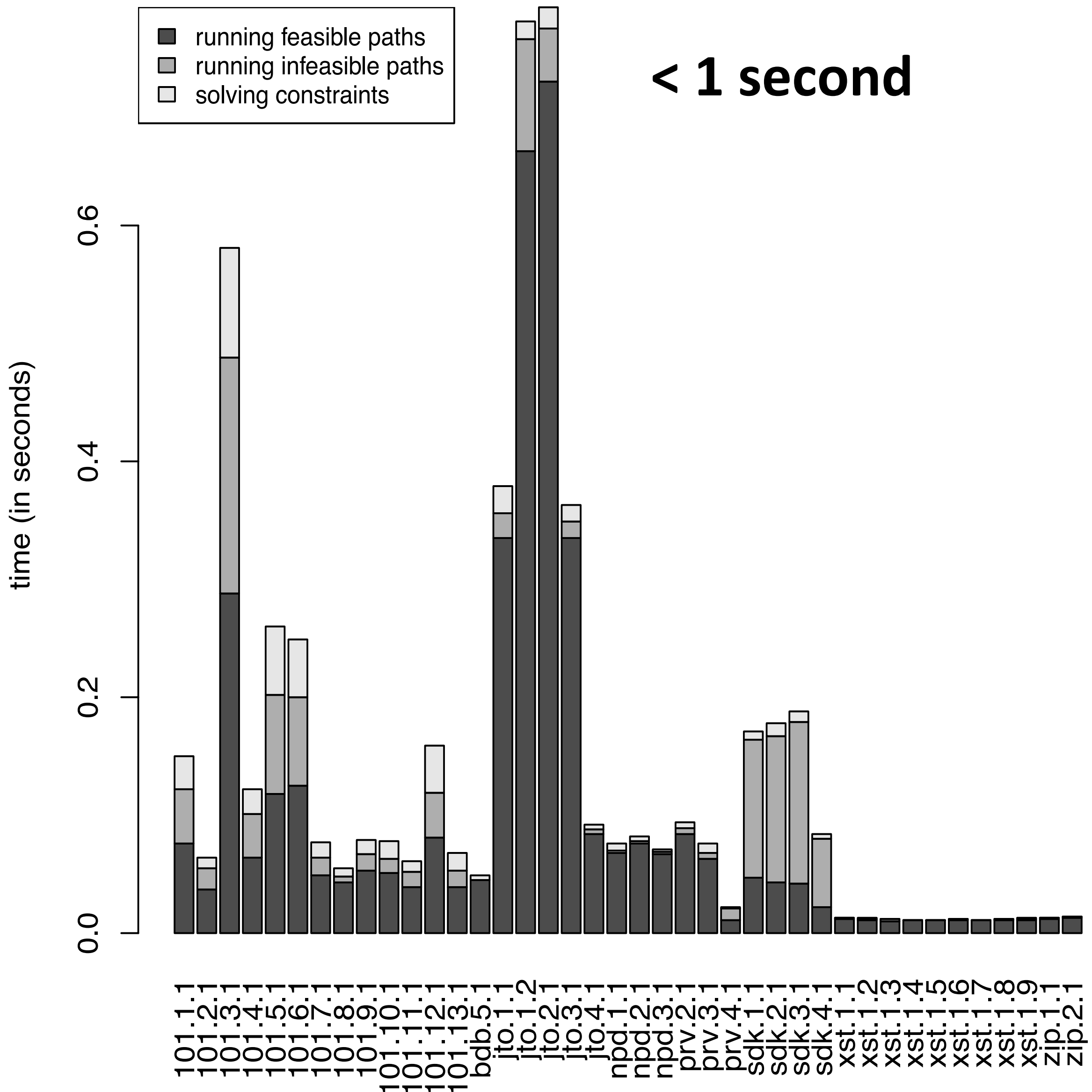
- **RQ1**: How SEF compares to an ideal sampler?
- **RQ2**: What is the impact of further constraining the model for test execution?

# Subjects of Analysis



| subject | abbrev. | LOC | # features | # valid con figurations | # tests | optimal coverage |
|---|---|---|---|---|---|---|
| 101Companies | 101 | 1,556 | 12 | 192 | 10 | 47,1% |
| BerkeleyDB | bdb | 34,462 | 42 | >15,000 | 5 | 32,2% |
| GPL | gpl | 1,713 | 14 | 146 | 9 | 82,2% |
| JTopas | jto | 2,031 | 5 | 32 | 12 | 44,9% |
| Notepad | npd | 2,074 | 25 | 7,057 | 3 | 24,6% |
| Prevayler | prv | 3,376 | 5 | 32 | 5 | 37,6% |
| Sudoku | sdk | 928 | 7 | 24 | 4 | 58,9% |
| XStream | xst | 14,480 | 7 | 128 | 9 | 17,6% |
| ZipMe | zip | 2,863 | 13 | 100 | 2 | 52,7% |

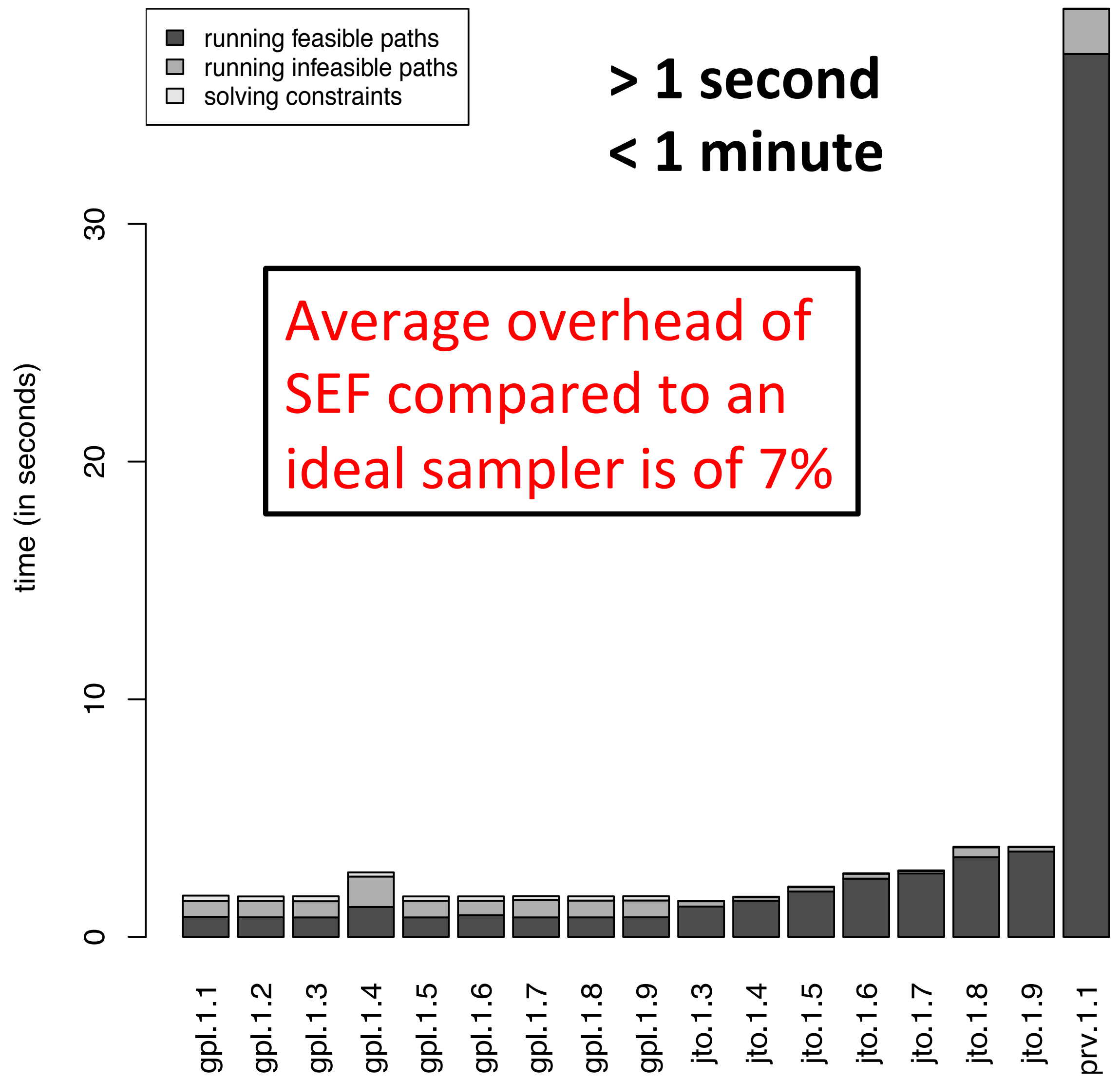
# RQ1
How SEF compares to an ideal sampler?



**< 1 second**

# RQ1

## How SEF compares to an ideal sampler?



**> 1 second**

**< 1 minute**

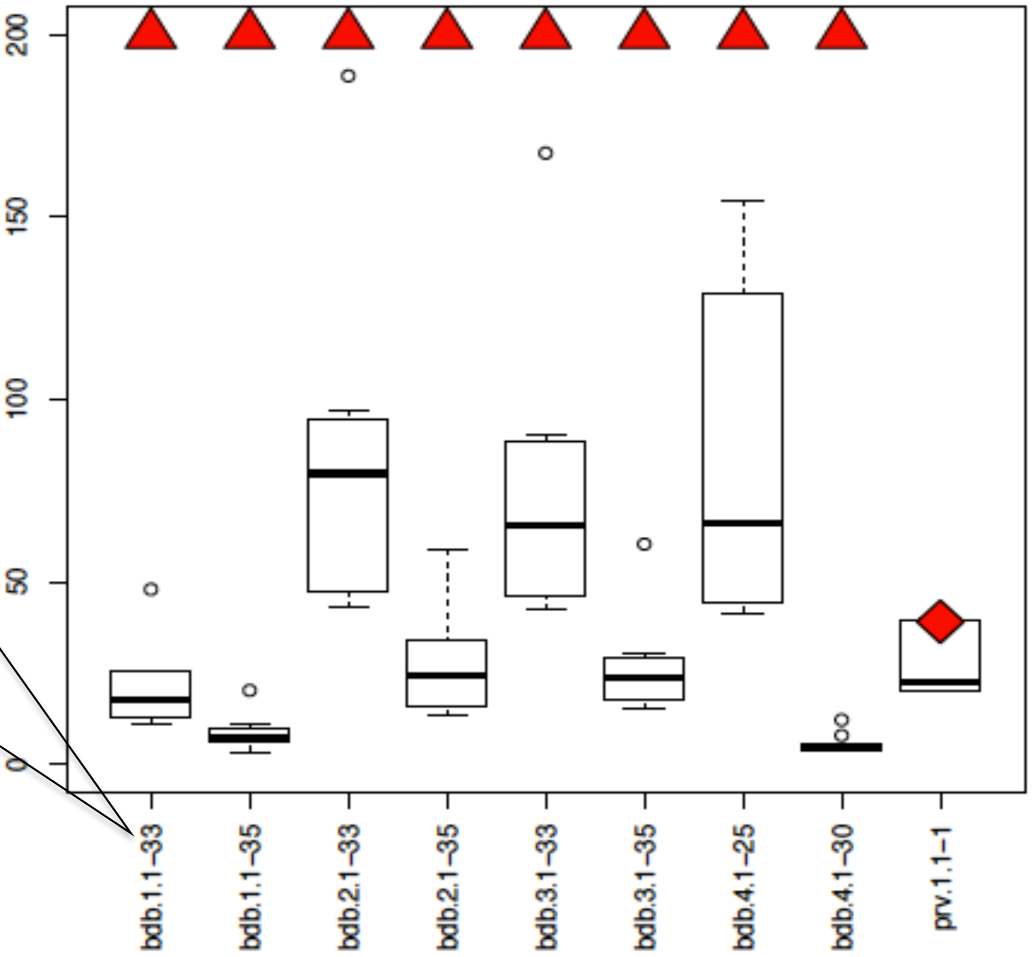
Average overhead of SEF compared to an ideal sampler is of 7%

## Observations

- We considered 9 previously-used SPLs with existing test cases (We did not generate new)
- For most SEF runs it took less than 1 second
- BerkeleyDB took much longer to run!
  – System tests covering most features in each path

## RQ2

## What is the impact of further constraining the model on SEF?



Can one identify groups of related features from changes and one test?

## Possible Future Work

(I need your help here)

- Better understand scalability for SPL testing
  - Evaluate the importance of scalability for testing on a much larger set of SPLs
- Address scalability (if indeed a big problem)
  - Possible Solutions:
    - Use compositional symbolic execution
    - Collapse paths with similar states

## Possible Future Work

(I need your help here)

- Improve automation
  - Program schemata generation can be impractical
  - Possible Solution:
    - Generate products on demand and use incremental compilation
    - Use Concolic Execution of Features (CEF)
    - We can also contribute to find type-errors

## Questions Revisited

- Is efficiency important?
- Is combinatorial explosion (scalability) relatively less important for testing?
- Is soundness important for testing?

Our answer was YES to all!
What is your opinion?