LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

Benedikt Damböck

Verification of Java Programs with Exceptions with CPAchecker

- **Final presentation of the master thesis**
- **Mentor: Dr. Philipp Wendler**
- **Supervised by: Prof. Dr. Dirk Beyer**
- **Date of presentation: 06.12.2023**

State of Java Exceptions in CPAchecker

- Exception control flow is not represented in CFA
- Adds code inside the `try`, `catch`, and `finally` blocks to CFA
- Analysis results are arbitrary

State of Java Exceptions in CPAchecker


```
public class Main {

  private static void foo(){
   throw new NullPointerException();
  }
  private static void bar(){}
  public static void main(String[] args){
    boolean entered = false;
    try {
      foo();
      bar();
    } catch(NullPointerException n){
      entered = true;
    } catch(RuntimeException r){
      entered = false;
    } finally {
      System.out.println("Hello World");
    }
    assert entered;
  }
}
```

Motivation

- Many different approaches in related work

- 2 approaches compatible with CPAchecker:
  - Implementation in CFA
  - Implementation in analysis

- Implementation in CFA
  - Advantage: Implementation in one point of the program
  - Disadvantage: Path explosion problem

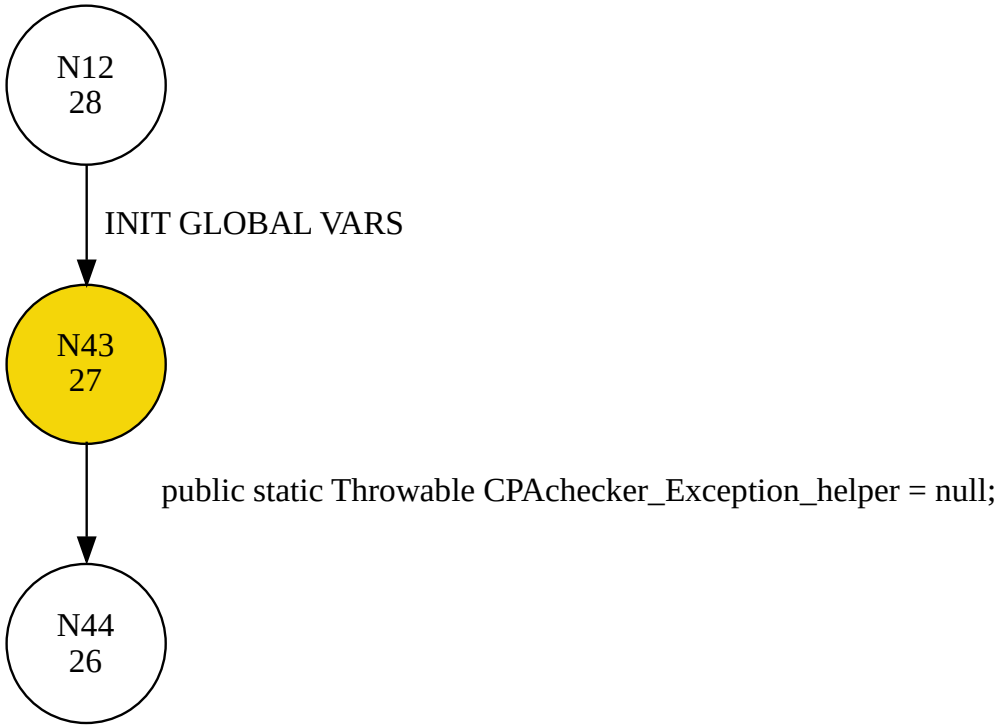
General Approach

- Representing exception control flow in a CFA using non-exception Java control flow.
- Track active exception with a global helper variable
- Conditional statements used to handle an exception

Global Helper Variable

- Add to program model :

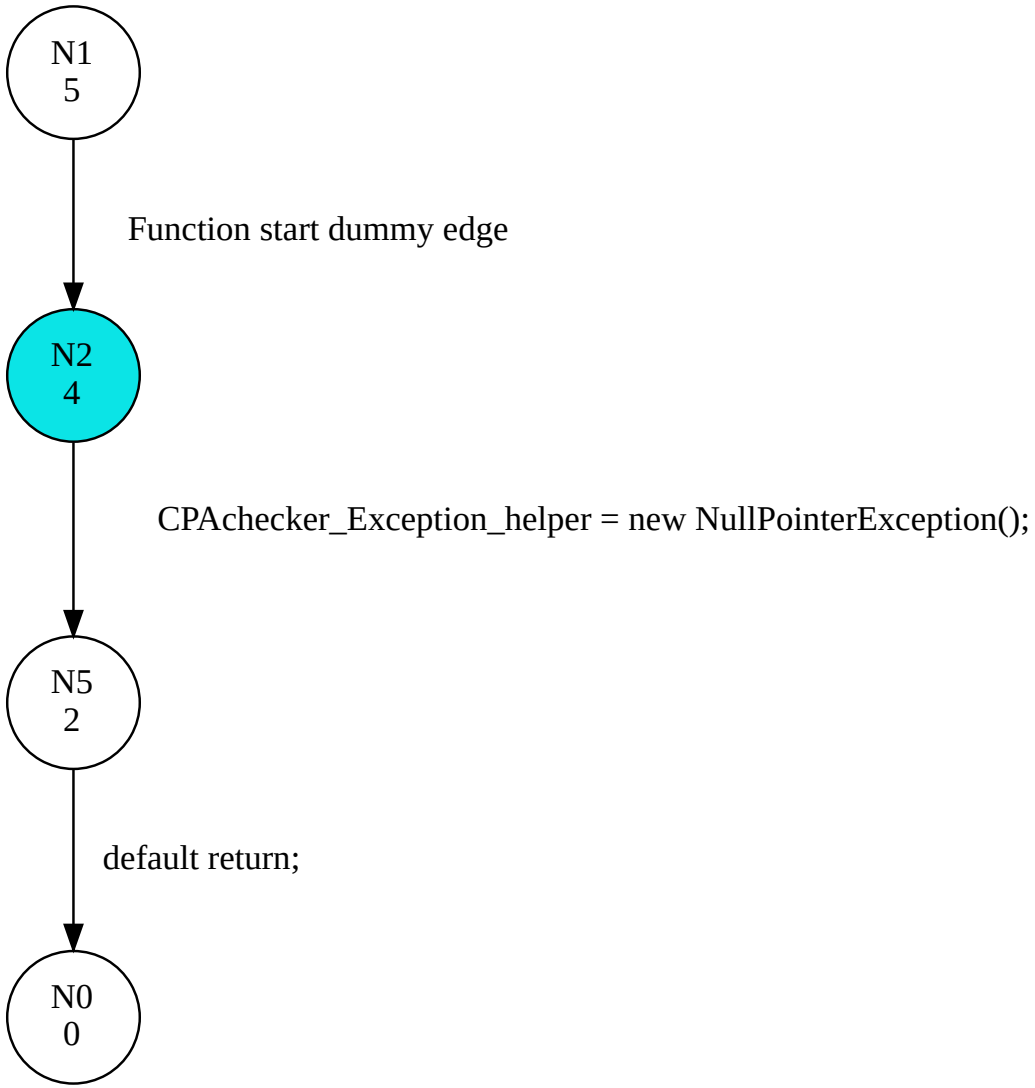
```
public static Throwable helperVariable =
null;
```



Throw

- Throw statement: `throw` expression
- Throw goes to the next `try`-`catch`(-`finally`) that can handle it
- Translation:

```
helperVariable = expression;
```



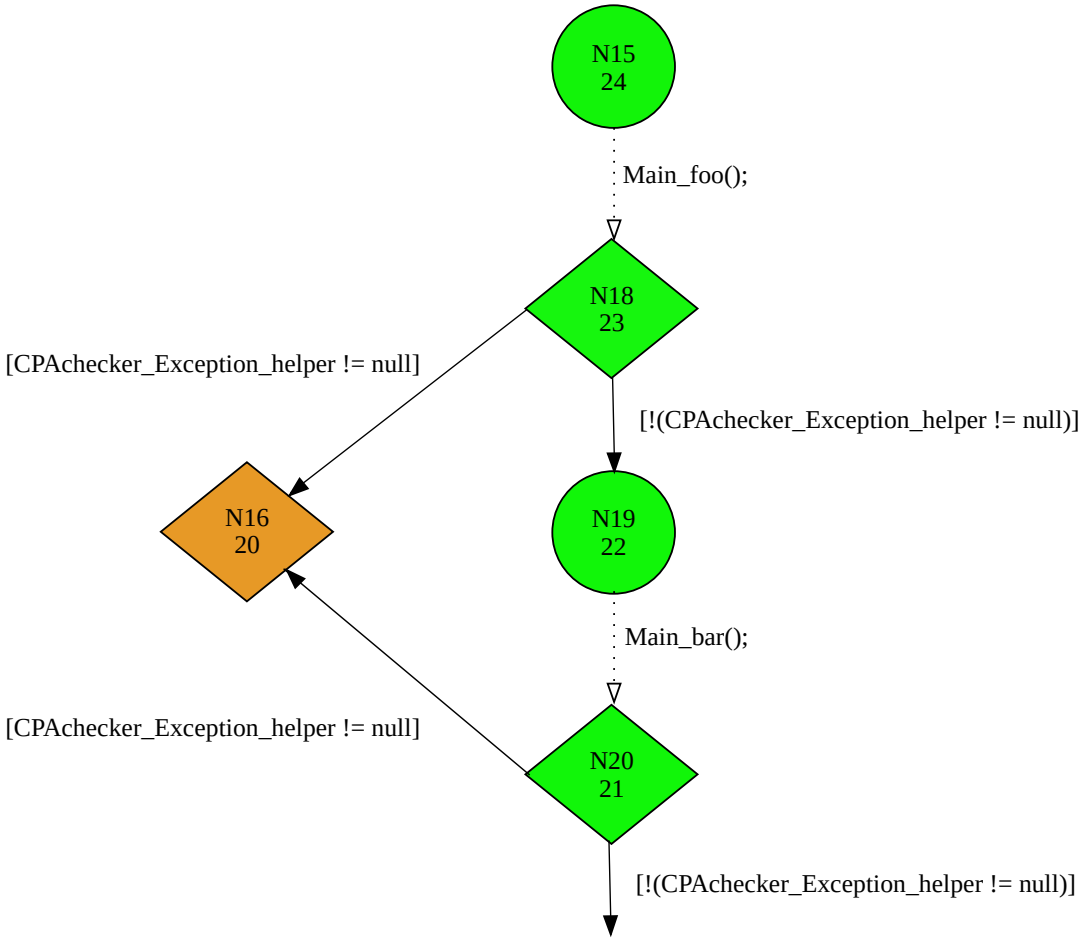
Handling an Exception

- Two-step process:

  Step 1: Check if an exception is actively impacting the program

  Step 2: Check if the exception can be handled

Step 1: Checking for Exception



Step 2: Catching Exception

- Normal catch syntax:

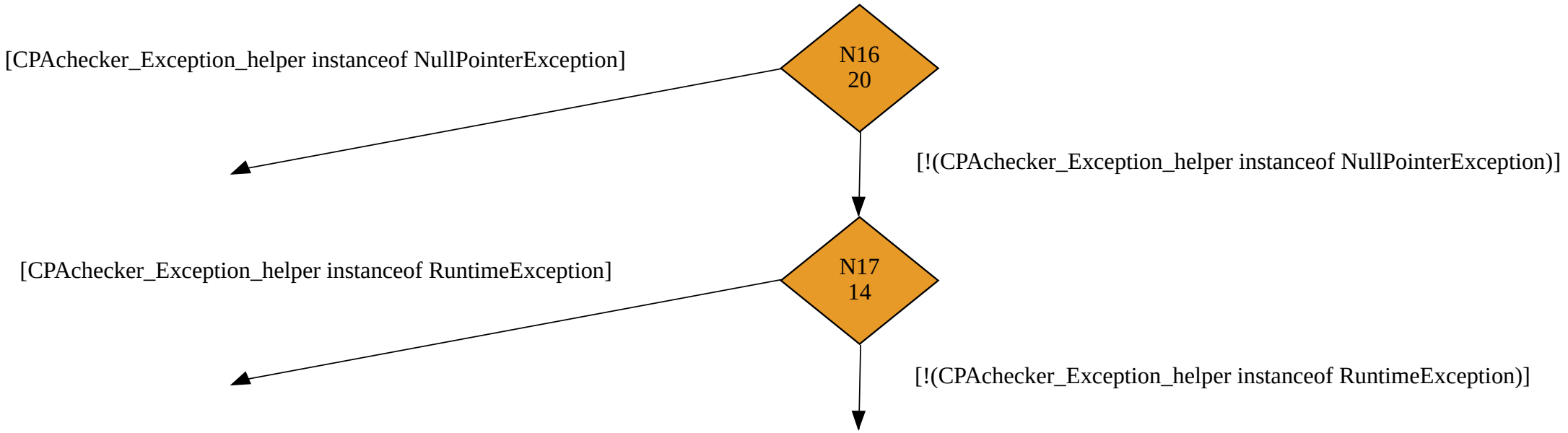
```
catch (CatchFormalParameter)

CatchFormalParameter =
  {VariableModifier} CatchType VariableDeclaratorId
```

- Example:

```
} catch(NullPointerException n){
  \\catchBlock
} catch(RuntimeException r){
  \\catchBlock
}
```

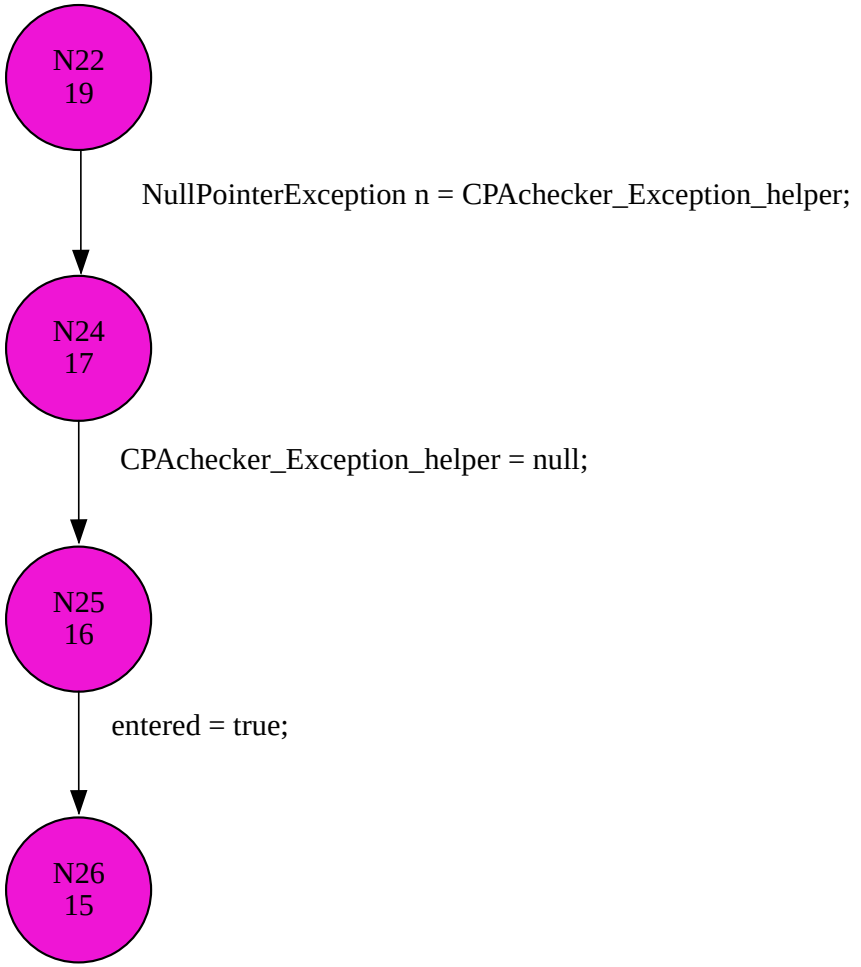
Step 2: Catching Exception



Step 2: Adding Content

- Adding a variable for `CatchFormalParameter`
- Set helper variable to `null`
- Add content of `catch` block:

```
entered = true;
```



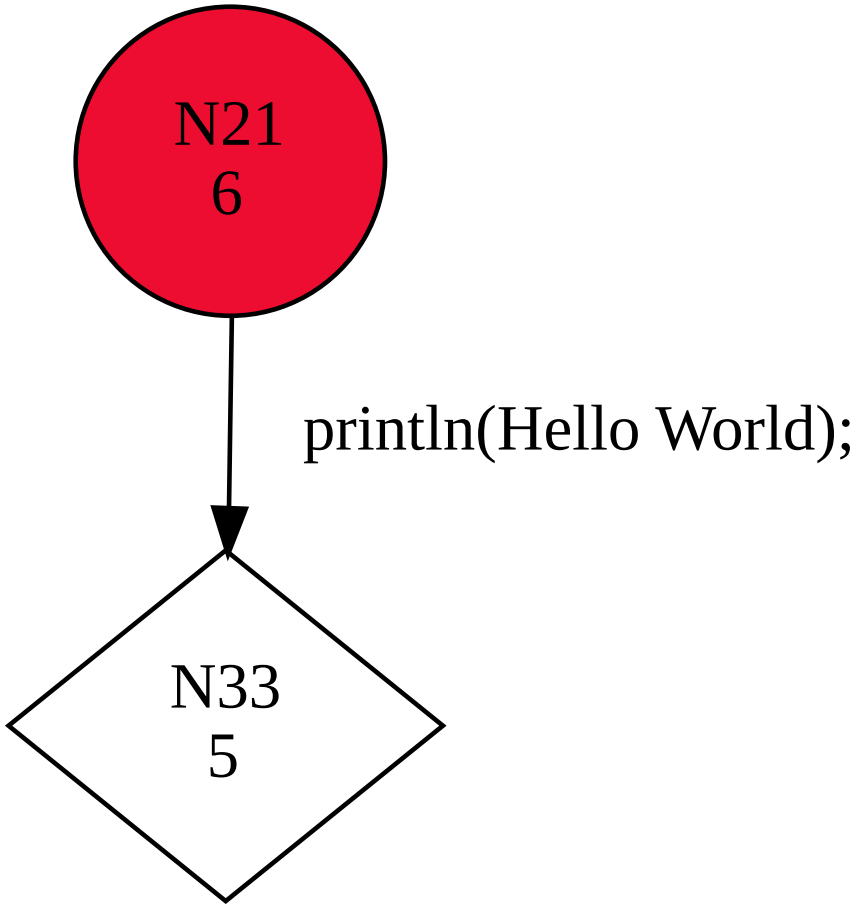
Finally

- Finally clauses always executed

- Two different approaches discussed:
  - Add `finally` block to all eligible paths
  - Map control flow after `finally` with local boolean variable

- `Finally` Block:

```
System.out.println("Hello World");
```



Abnormal Execution

- Every abnormal execution condition is unique
- Separate implementation of parts in each scenario
- Handle exception control flow with the previously discussed approach

Evaluation - CPAchecker

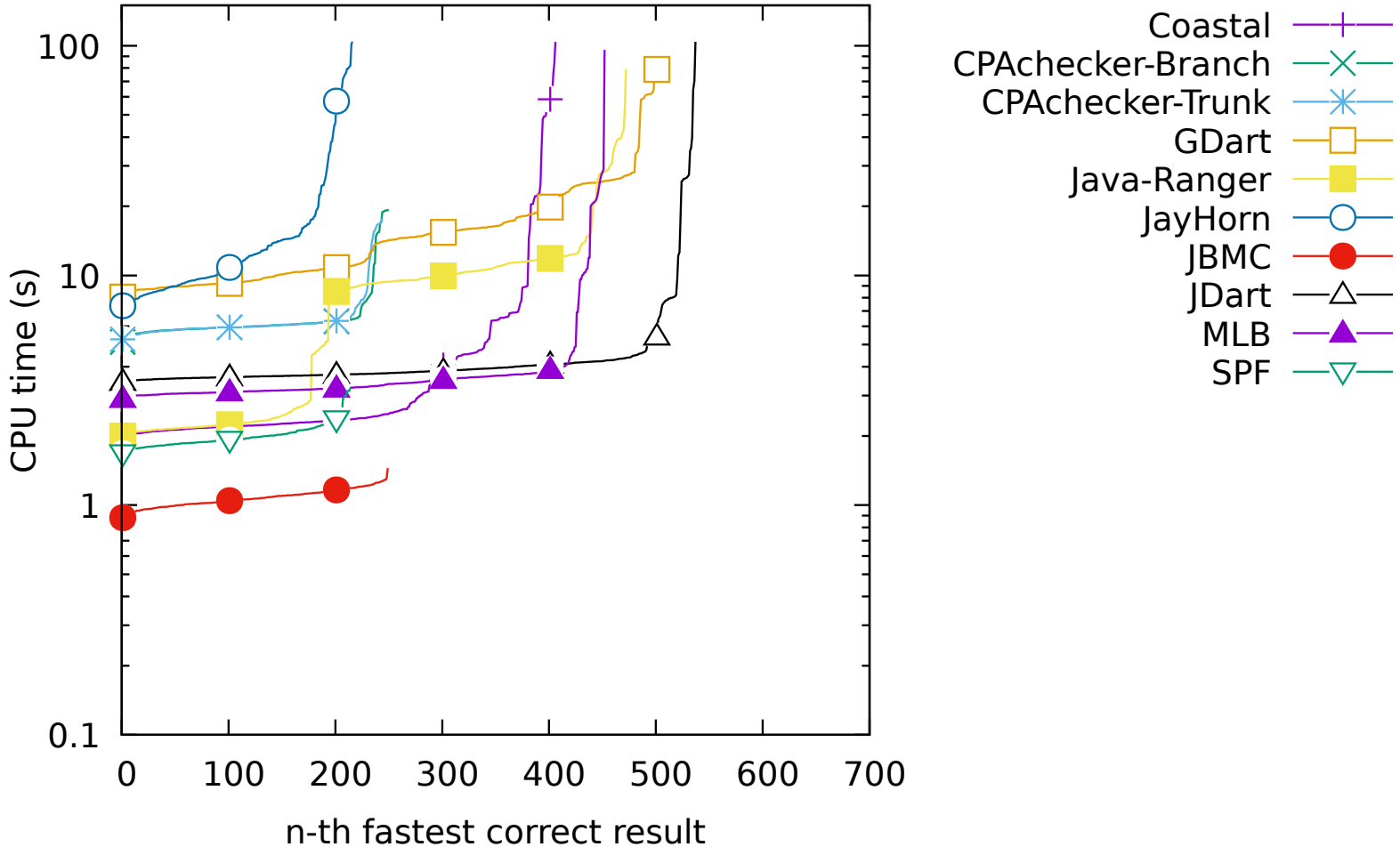
- Composition of value analysis and runtime type analysis
- Correctly analyzed all but one program with developer thrown exceptions
- Programs with abnormal executions not analyzed correctly
- Programs with library method calls in exception constructs not analyzed correctly
- Performance of implementation was not worse on this dataset

More paths didn’t lead to performance loss

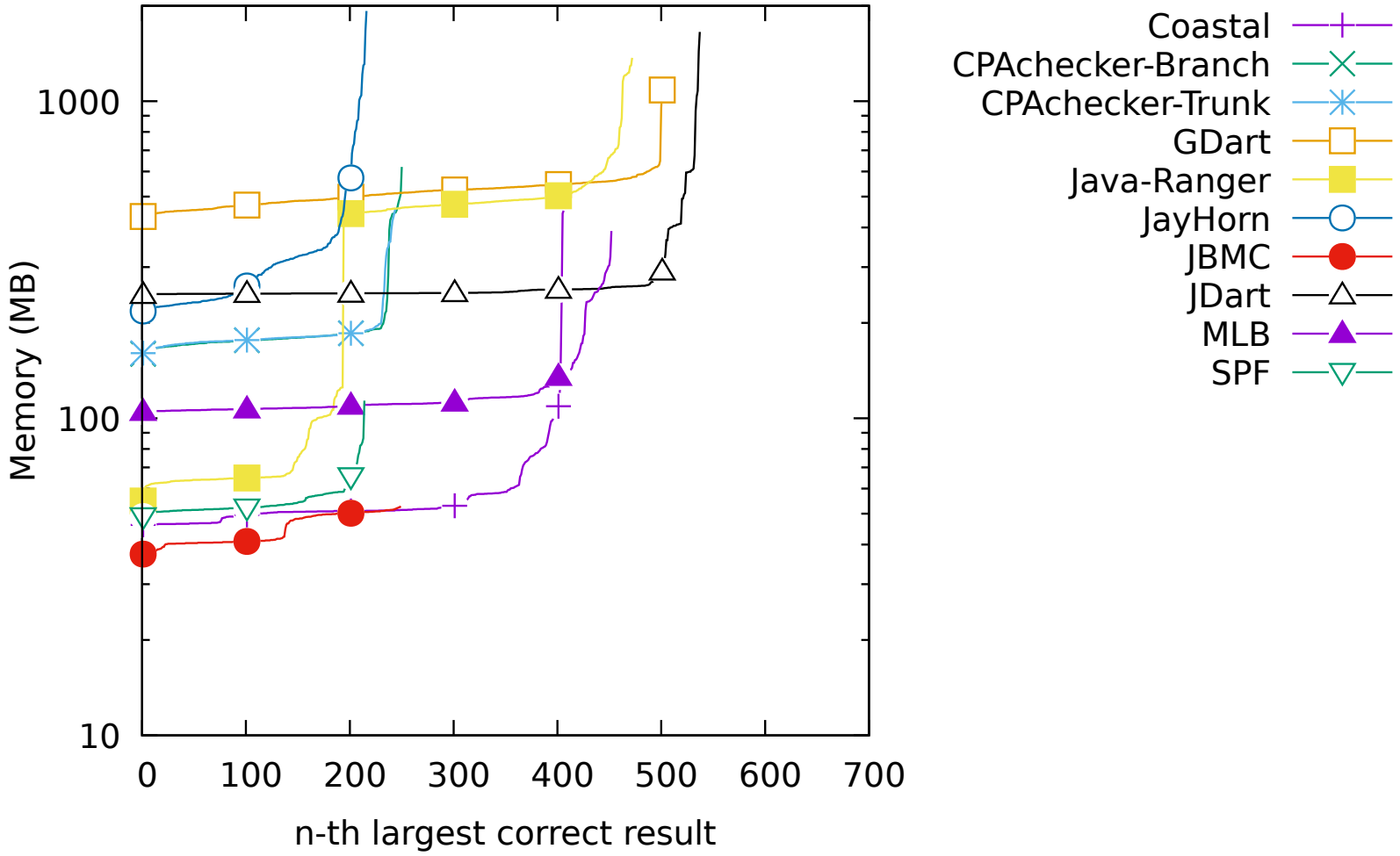
Evaluation – State-of-the-Art Tools

- Performance: CPAchecker in the middle of the pack
- Low number of correct results
- Large number of wrong proofs, wrong alarms and errors

Evaluation – State-of-the-Art Tools



Evaluation – State-of-the-Art Tools



Study: CPAchecker Problems with Java Programs

- Anonymous classes
- Increment operator at array index position
- Bug in `ErrorPathShrinker` class
- No variable for main method parameter
- Small number of analyses for Java programs

Conclusion

- CPAchecker currently does not represent exception control flow in CFA
- Introduced approach to handle exceptional control flow with standard Java control flow in CFAs
- Improved accuracy of CPAchecker
- Implementation possible in CFA construction of CPAchecker
- More paths didn’t lead to performance loss
- Unable to handle abnormal execution and library method calls
- Interesting topic: Performance comparison between exception handling in CFA vs exception handling in analysis

Abnormal Execution

- Handling on a case-by-case basis
- Example Division by zero
- Declare a temporary integer variable
- Conditional statement that checks if variable in divisor is zero
- If zero
  - Assign new ArithmeticException object to helper variable
  - Handle exceptional control flow, as discussed earlier
- Otherwise
  - Assign operation to temporary variable
- Replace original statement with temporary variable

Method call in operation

- Example: Method used in the operation: bar() in foo(bar())
- Declare a temporary variable with the return type of the method used within the operation
- Assign the method call to this variable.
- Use value of operation instead of method call
- Apply exception handling step after

Nesting

- Nesting in try: exception path leads to next exception handling if catch block exist in outer try
- Nesting in catch: exception path leads to finally block, end of method if not also nested
- Nesting in finally:
  - Exception path leads to end of method if not also nested
  - No execution of rest of finally block

Finally with Variable