TECHNISCHE UNIVERSITÄT DARMSTADT

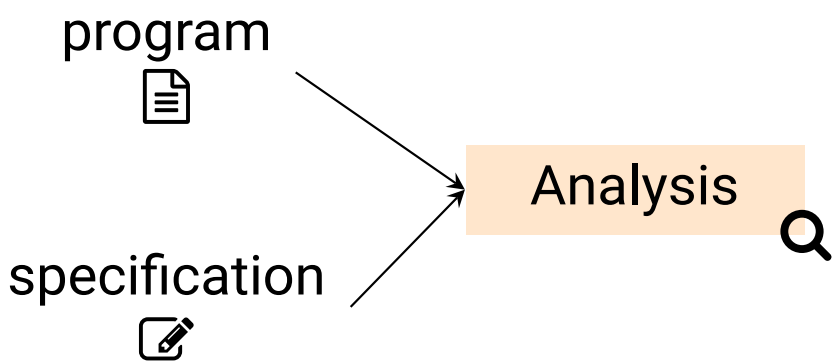
Static Analysis + Runtime Verification Combination Patterns from the Literature

Marie-Christine Jakobs
joint work with Heiko Mantel

SF 4.0
Software-Factory 4.0

Static Analysis

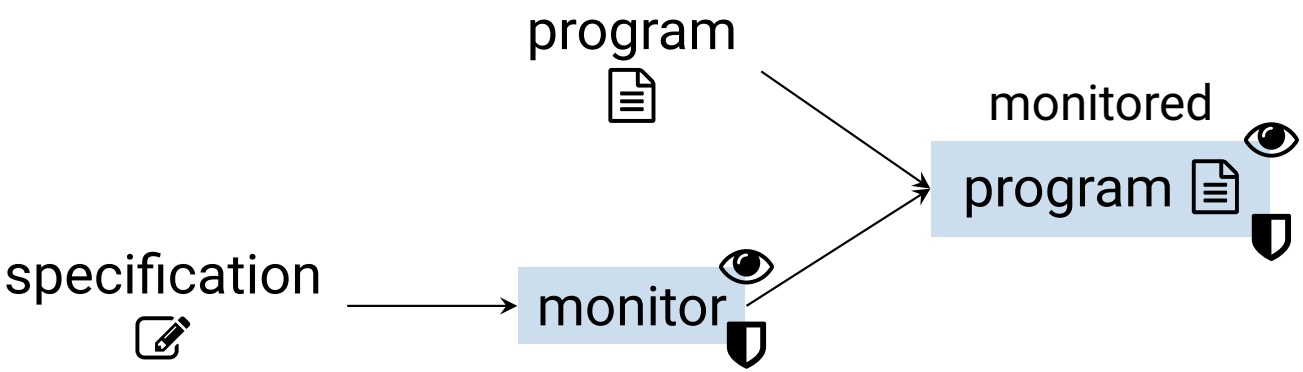
TECHNISCHE UNIVERSITÄT DARMSTADT



- Analysis of **all** executions (**before** execution)
- Derive knowledge about program
- **Report** specification violation

Runtime Verification

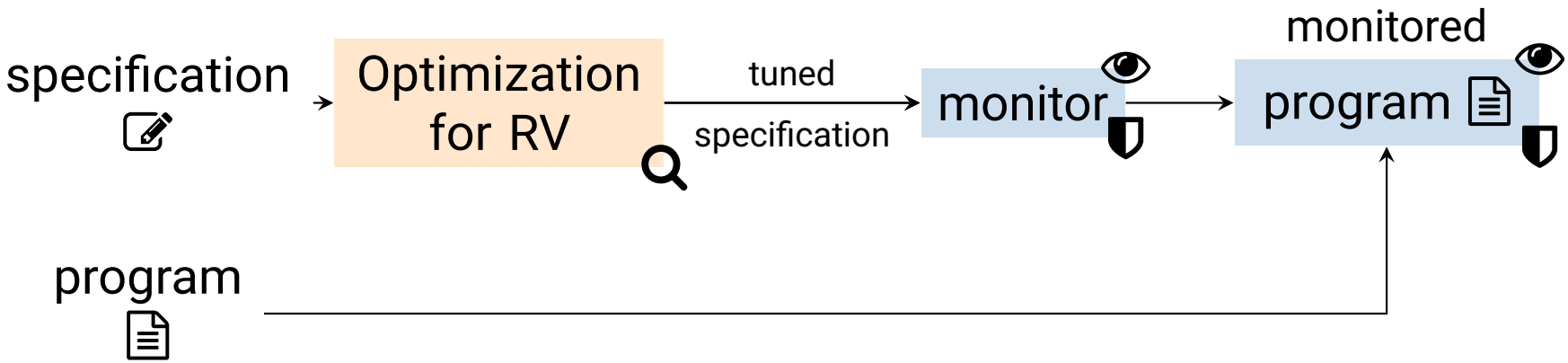
TECHNISCHE UNIVERSITÄT DARMSTADT



- Analysis of **single** execution (**during**, after execution)
- **React** to specification violation
  (reporting, stop, suppression, recover, etc.)
- Monitoring causes runtime overhead

Pattern 1a: Program-independent Tuning of the Specification

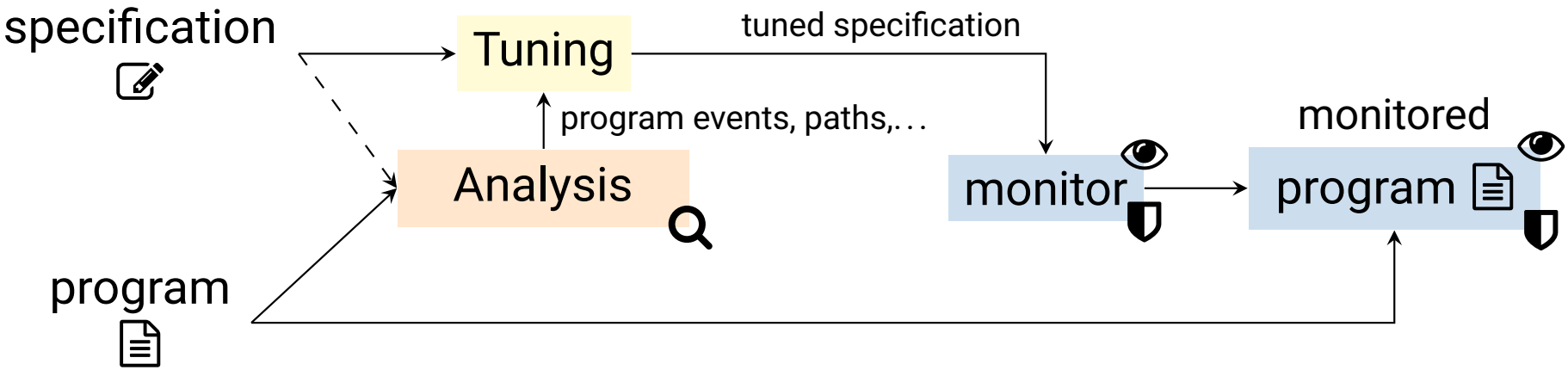
TECHNISCHE UNIVERSITÄT DARMSTADT



- Logic, automata
- Manual, graph algorithm
- Reporting, user-defined code

Pattern 1b:
Program-based Tuning of the Specification

TECHNISCHE UNIVERSITÄT DARMSTADT

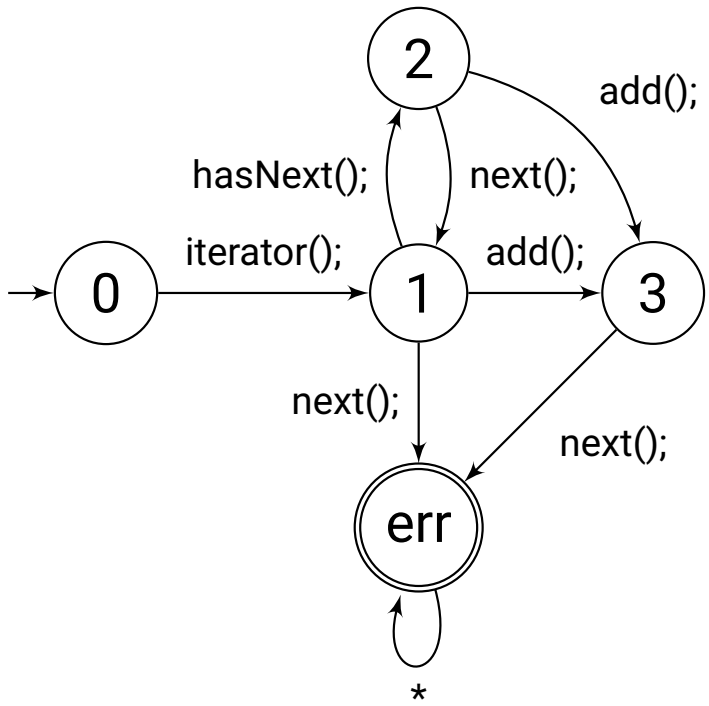


- Automata
- Control-flow analysis, etc.
- Reporting, stop

Example Absent Event Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

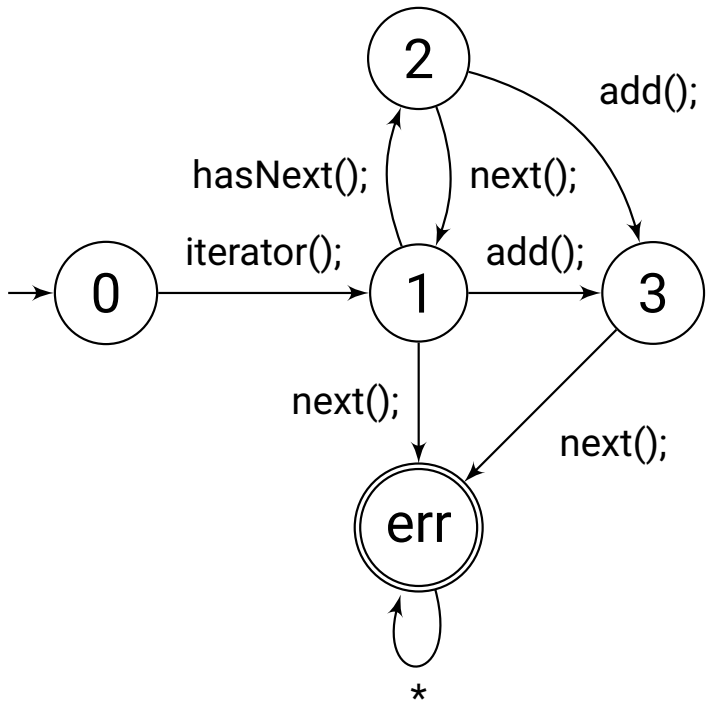
```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



Example Absent Event Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



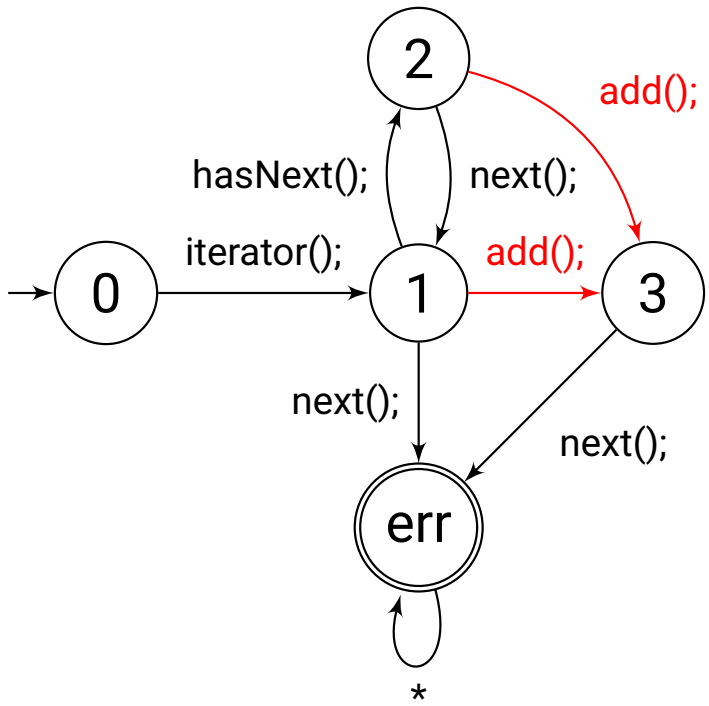
Events

- iterator();
- hasNext();
- next();

Example Absent Event Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



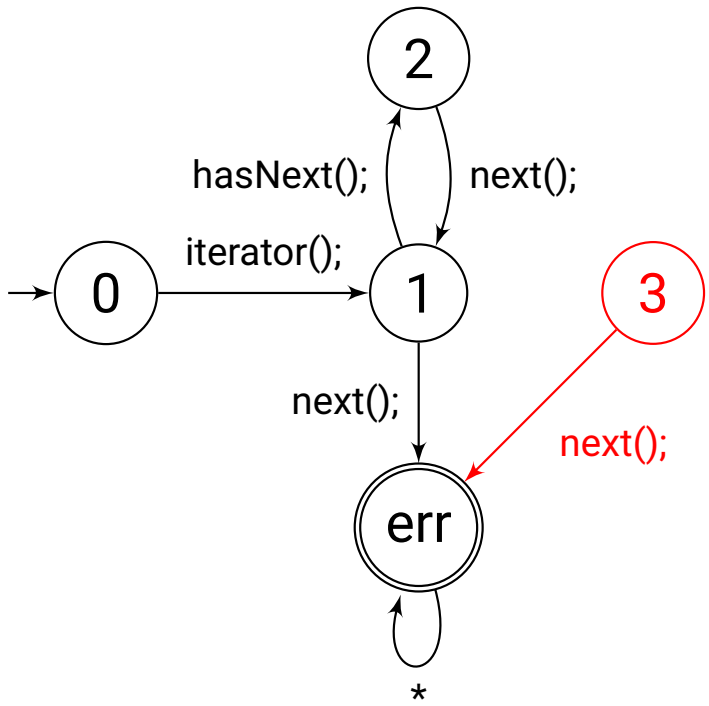
Events

- iterator();
- hasNext();
- next();

Example Absent Event Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



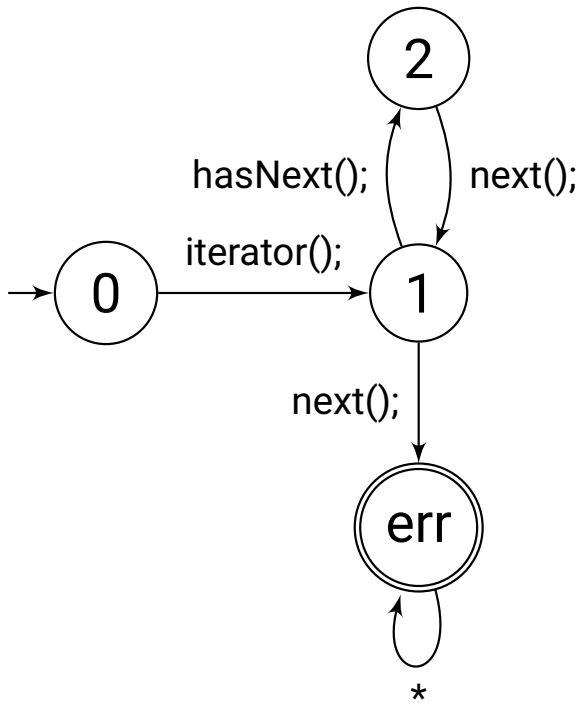
Events

- iterator();
- hasNext();
- next();

Example Absent Event Pruning

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



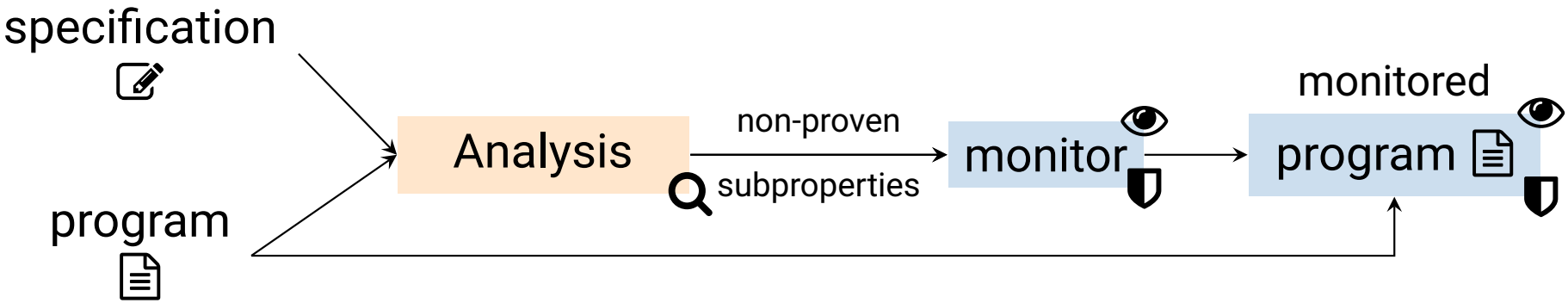
Events

- iterator();
- hasNext();
- next();

Pattern 2a:
Monitor (Sub)Properties That You Cannot Verify!

TECHNISCHE UNIVERSITÄT DARMSTADT

Requires decomposable specification



- Memory safety, information or data flow, automata
- Dependence analysis, type analysis, dataflow analysis, BMC, deductive verification, etc.
- Reporting, stop

Example

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
   boolean nonEmpty = false;
   for(int i=0;i<args.length;i++) {
      nonEmpty = true;
      assert(0<=i && i<args.length);
      System.out.println(args[i]);
   }
   if (nonEmpty) {
      assert(0<args.length);
      System.out.println(args[0]!= null);
   }
}
```

Example

TECHNISCHE UNIVERSITÄT DARMSTADT

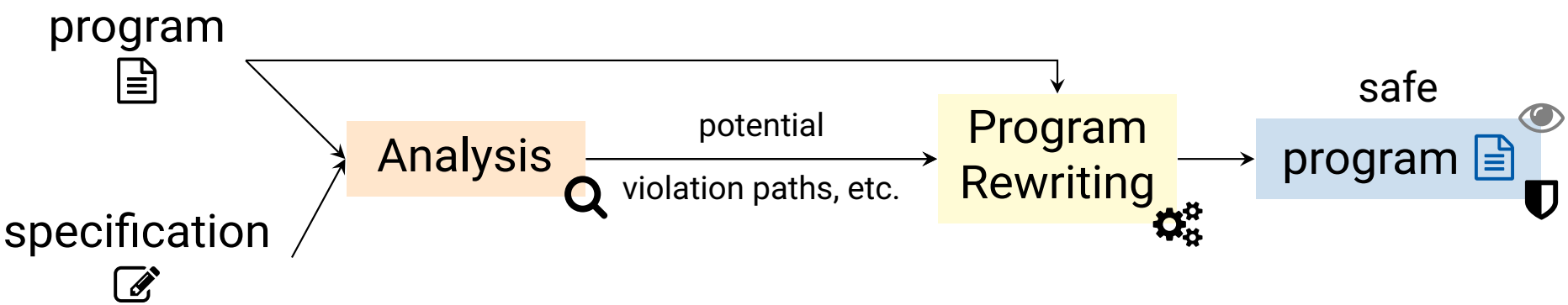
```
public static void main(String[] args) {
   boolean nonEmpty = false;
   for(int i=0;i<args.length;i++) {
      nonEmpty = true;
      assert(0<=i && i<args.length);   \\proven, do not monitor
      System.out.println(args[i]);
   }
   if (nonEmpty) {
      assert(0<args.length);   \\not proven, monitor
      System.out.println(args[0]!= null );
   }
}
```

Pattern 2b:

Rewrite Program and Enforce What You Cannot Verify!

TECHNISCHE UNIVERSITÄT DARMSTADT

Idea Use analysis to exclude (proven or spurious) paths from monitoring, enforce property on non-proven paths



- Automata
- Predicate model checking (bounded CEGAR loops)
- Stop, surpression

Example Zero Overhead Runtime Monitoring

TECHNISCHE UNIVERSITÄT DARMSTADT

Original program

```
public static void main(String[] args) {
   boolean nonEmpty = false;
    if (args.length>0)
      nonEmpty = true;
    if (nonEmpty) {
      assert(0<args.length);
      System.out.println(args [0]);
   }
}
```

Example Zero Overhead Runtime Monitoring

TECHNISCHE UNIVERSITÄT DARMSTADT

Original program

```
public static void main(String[] args) {
  boolean nonEmpty = false;
  if (args.length>0)
    nonEmpty = true;
  if (nonEmpty) {
    assert(0<args.length);
    System.out.println(args [0]);
  }
}
```

Spurious counterexample

nonEmpty=false;!(args.length>0);nonEmpty;!(0<args.length);

Safe path

nonEmpty=false;(args.length>0);nonEmpty=true;nonEmpty;0<args.length;System.out.println(args[0]);

Violating path

nonEmpty=false;(args.length>0);nonEmpty=true;nonEmpty;!(0<args.length);System.out.println(args[0]);

Example Zero Overhead Runtime Monitoring

TECHNISCHE UNIVERSITÄT DARMSTADT

Original program

```
public static void main(String[] args) {
  boolean nonEmpty = false;
  if (args.length>0)
    nonEmpty = true;
  if (nonEmpty) {
    assert(0<args.length);
    System.out.println(args [0]);
  }
}
```

Rewritten program

```
public static void main(String[] args) {
  boolean nonEmpty = false;
  if (args.length>0)
    nonEmpty = true;
  if (nonEmpty) {
    if (!(0< args.length))
      System.exit(0);
    System.out.println(args [0]);
  }
}
```

Spurious counterexample

nonEmpty=false;!(args.length>0);nonEmpty;!(0<args.length);

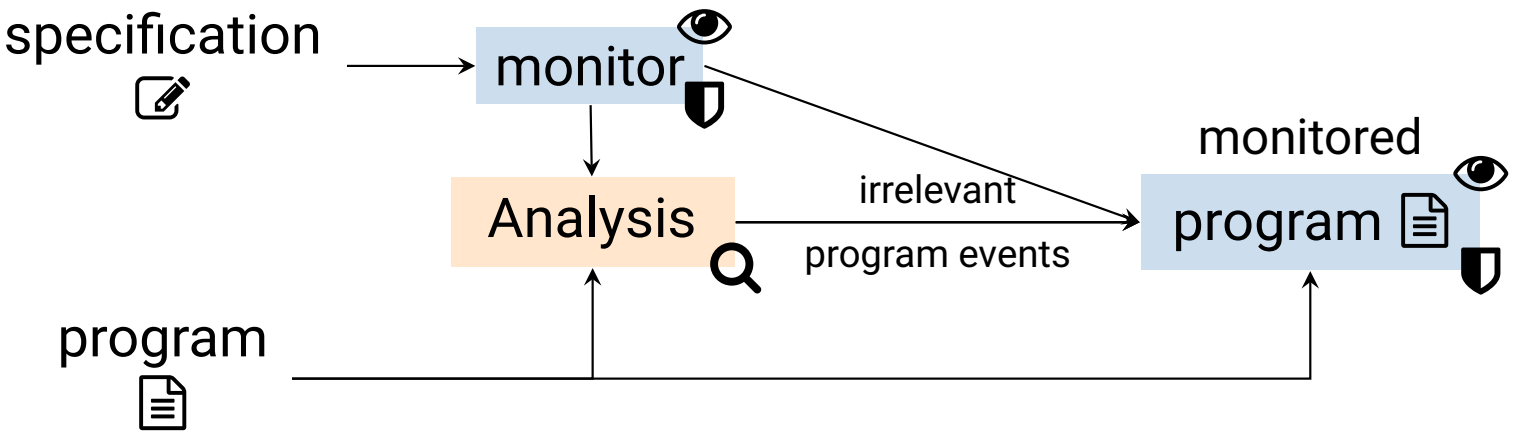
Safe path

nonEmpty=false;(args.length>0);nonEmpty=true;nonEmpty;0<args.length;System.out.println(args[0]);

Violating path

nonEmpty=false;(args.length>0);nonEmpty=true;nonEmpty;!(0<args.length);System.out.println(args[0]);

Pattern 3a: Skip Irrelevant Events

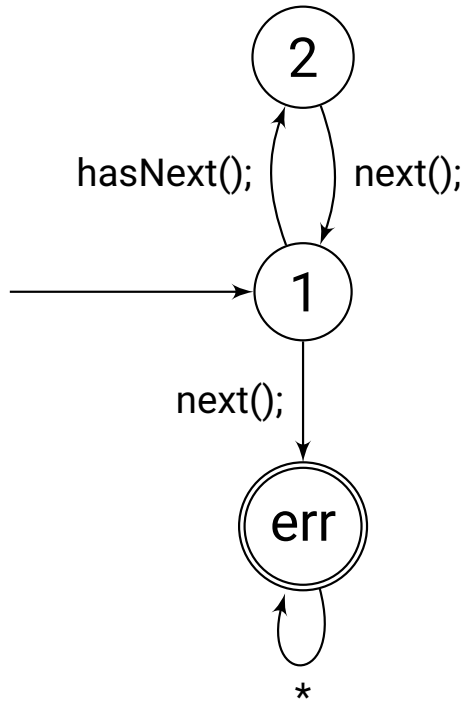


- Automata
- Dataflow analysis, path analysis, etc.
- Report, stop, user-defined code

Example NopShadow Analysis

TECHNISCHE UNIVERSITÄT DARMSTADT

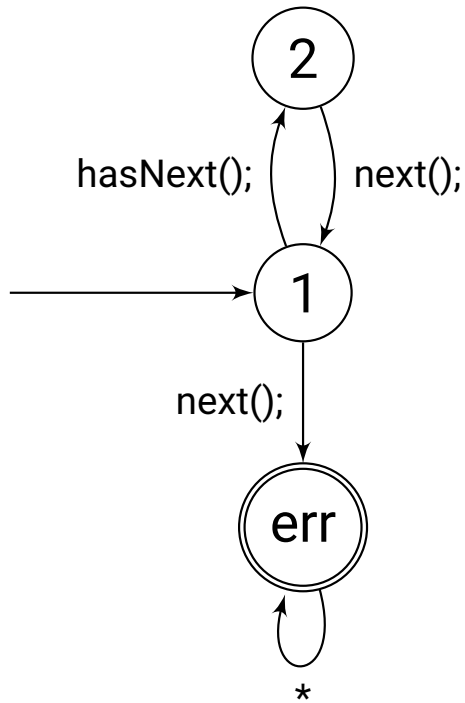
```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator();
   while(it.hasNext())
      System.out.print(it.next());
}
```



Example NopShadow Analysis

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
  // f-1 b-{err}
   Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
         iterator ();
  // f-1 b-{err}
  while(it.hasNext())
    // f-2 b-{err, 1}
    System.out.print(it.next());
    // f-1 b-{err}
  // f-1
}
```

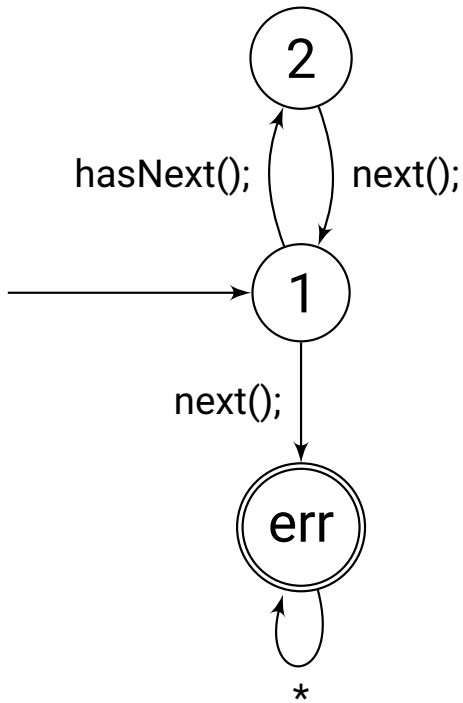


Compute automaton states reachable forward and backward

Example NopShadow Analysis

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
  // f-1 b-{err}
   Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
         iterator ();
  // f-1 b-{err}
  while(it.hasNext())
    // f-2 b-{err, 1}
    System.out.print(it.next ());
    // f-1 b-{err}
  // f-1
}
```

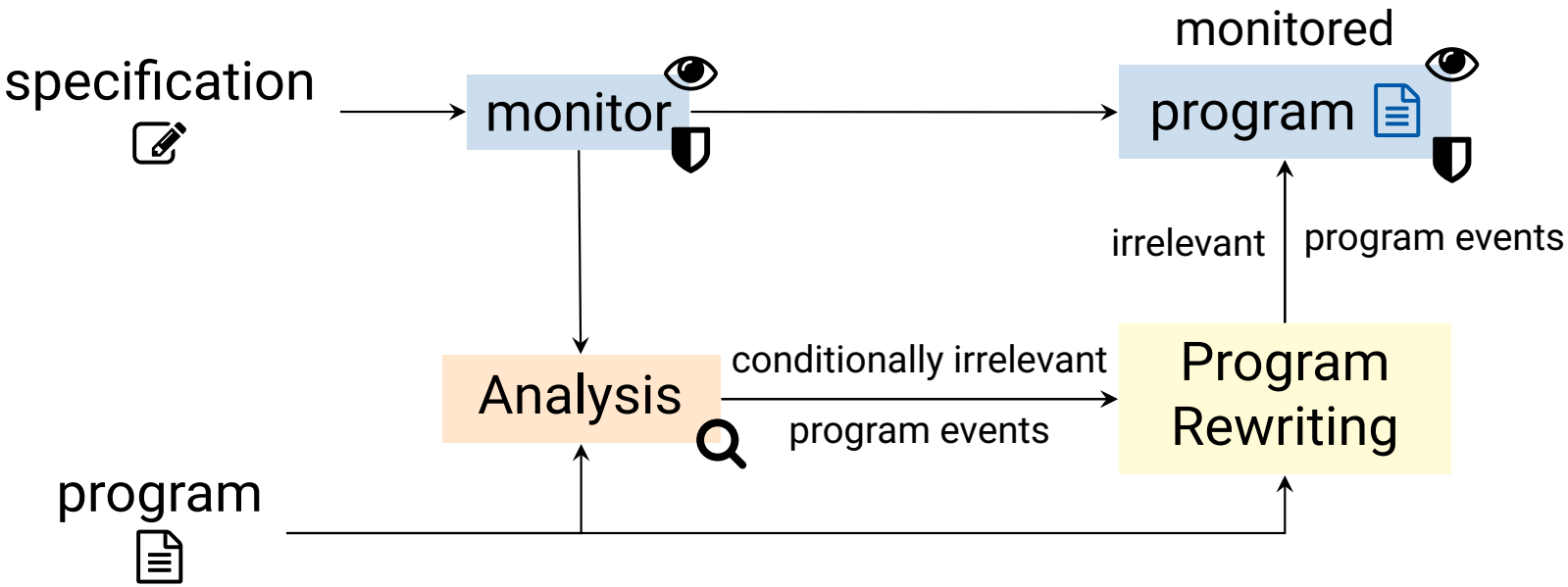


Compute automaton states reachable forward and backward

- $1 \in \{err\} \Leftrightarrow 2 \in \{err\} \wedge 2 \neq err \implies$ next() event irrelevant
- $1 \in \{err, 1\} \not\Leftrightarrow 2 \in \{err, 1\} \wedge 2 \neq err \implies$ keep hasNext() event

Pattern 3b:
Rewrite Program to Skip Irrelevant Events

TECHNISCHE UNIVERSITÄT DARMSTADT

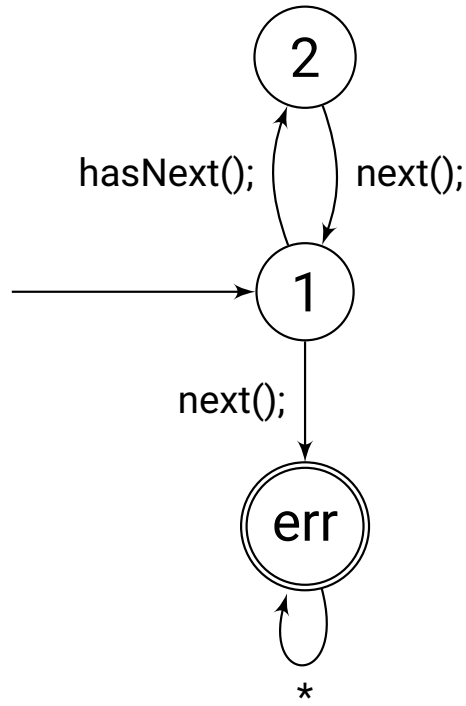


- Automata, memory safety
- Automata analysis, weakest precondition
- Report, stop

Example Stutter Equivalent Loop Transformation

TECHNISCHE UNIVERSITÄT DARMSTADT

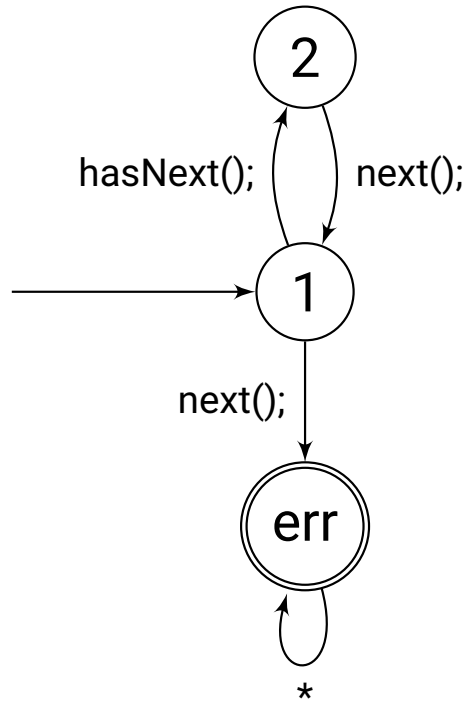
```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator ();
   while(it.hasNext())
      System.out.print(it.next ());
}
```



Example Stutter Equivalent Loop Transformation

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator ();
   while(it.hasNext())
      System.out.print(it.next ());
}
```



Monitor behavior does not change after first loop iteration

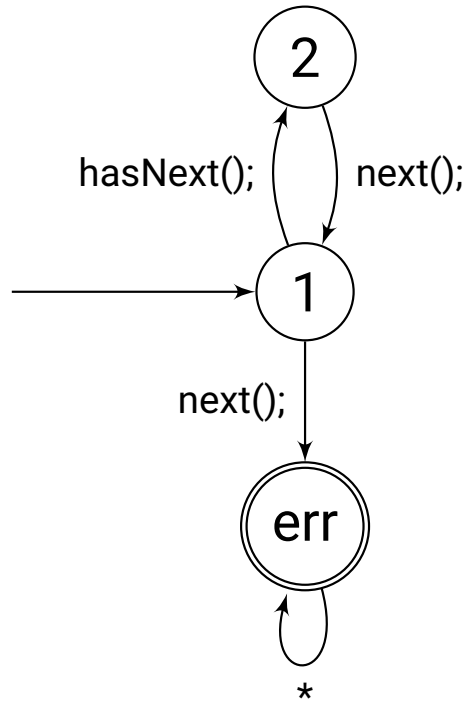
Example Stutter Equivalent Loop Transformation

TECHNISCHE UNIVERSITÄT DARMSTADT

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator ();
   while(it.hasNext())
      System.out.print(it.next ());
}
```

```
public static void main(String[] args) {
    Iterator<String> it = Arrays.
        asList("Hello", "World", "!").
        iterator ();
   if (it.hasNext())
      System.out.print(it.next ());

   while(it.hasNext())
      System.out.print(it.next ());
}
```



Monitor behavior does not change after first loop iteration

Future Cooperation Opportunities and Future Research Directions

TECHNISCHE UNIVERSITÄT DARMSTADT

- Specify allowed instead of forbidden behavior
- Go beyond fixed or automata-based specifications
- Go beyond reporting, stop

- Tool-independent formalization, soundness proofs
- Combination of approaches