Model-based Test Case Generation for (Dynamic) Software Product Lines

TECHNISCHE UNIVERSITÄT DARMSTADT

The 6th Meeting on Feature-oriented Software Development
2014, May 04-07 – Schloss Dagstuhl, Germany

IMOTEP

DFG Priority Programme 1593
Design for Future – Managed Software Evolution

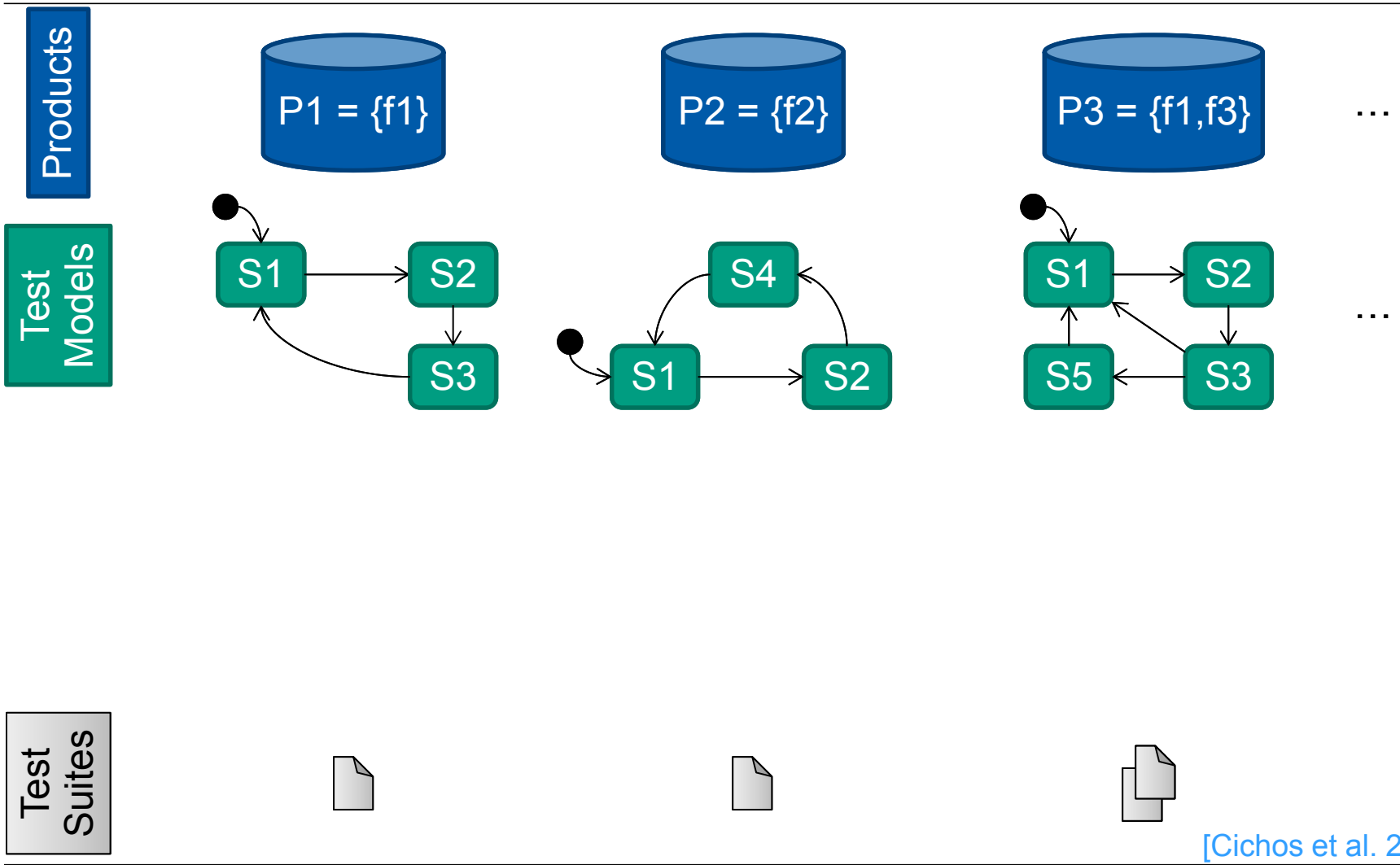
Johannes Bürdek

johannes.buerdek@es.tu-darmstadt.de
Tel.+49 6151 16 76089

ES Real-Time Systems Lab
Prof. Dr. rer. nat. Andy Schürr
Dept. of Electrical Engineering and Information Technology
Dept. of Computer Science (adjunct Professor)

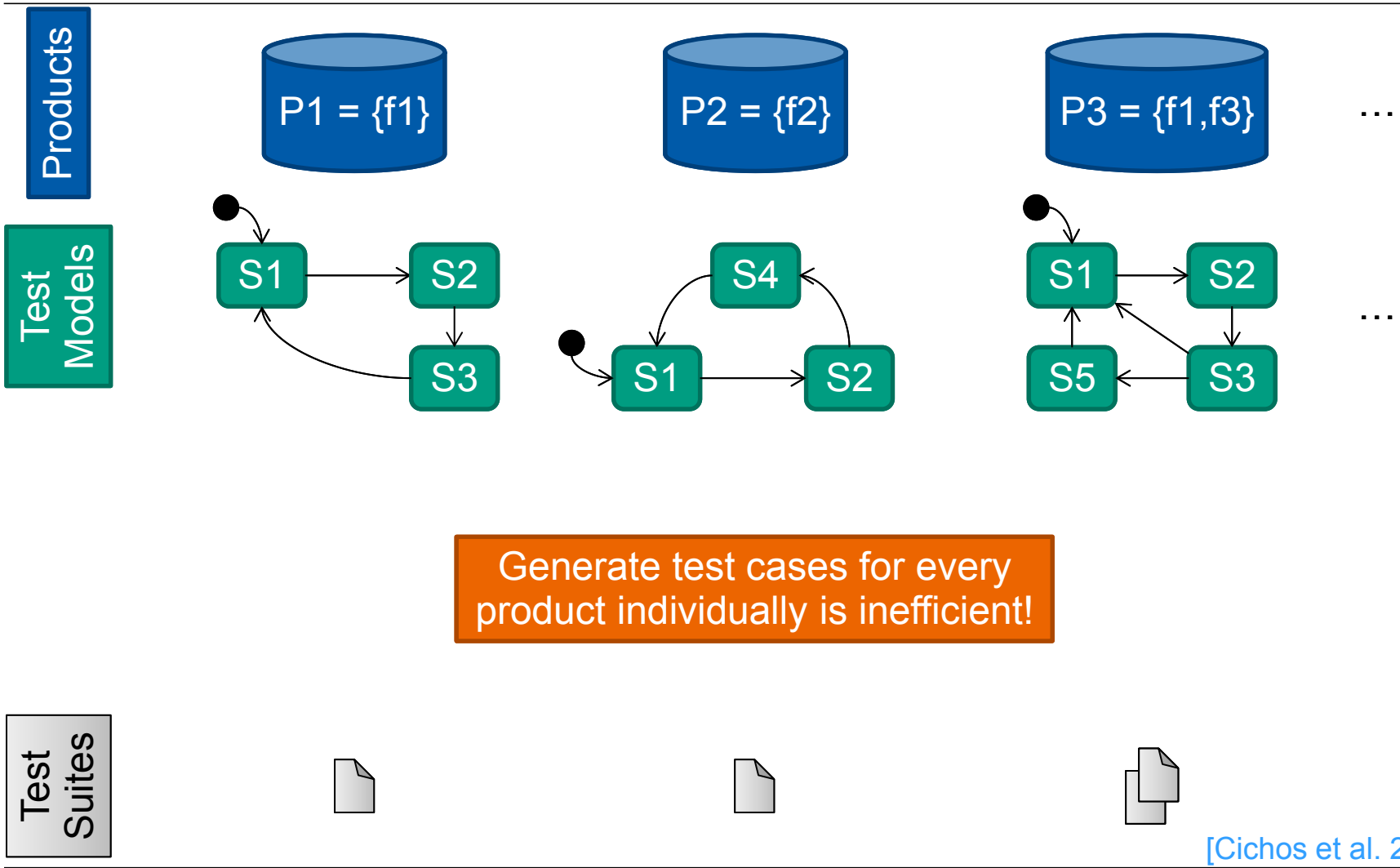
www.es.tu-darmstadt.de

Testing Software Product Lines



[Cichos et al. 2011]

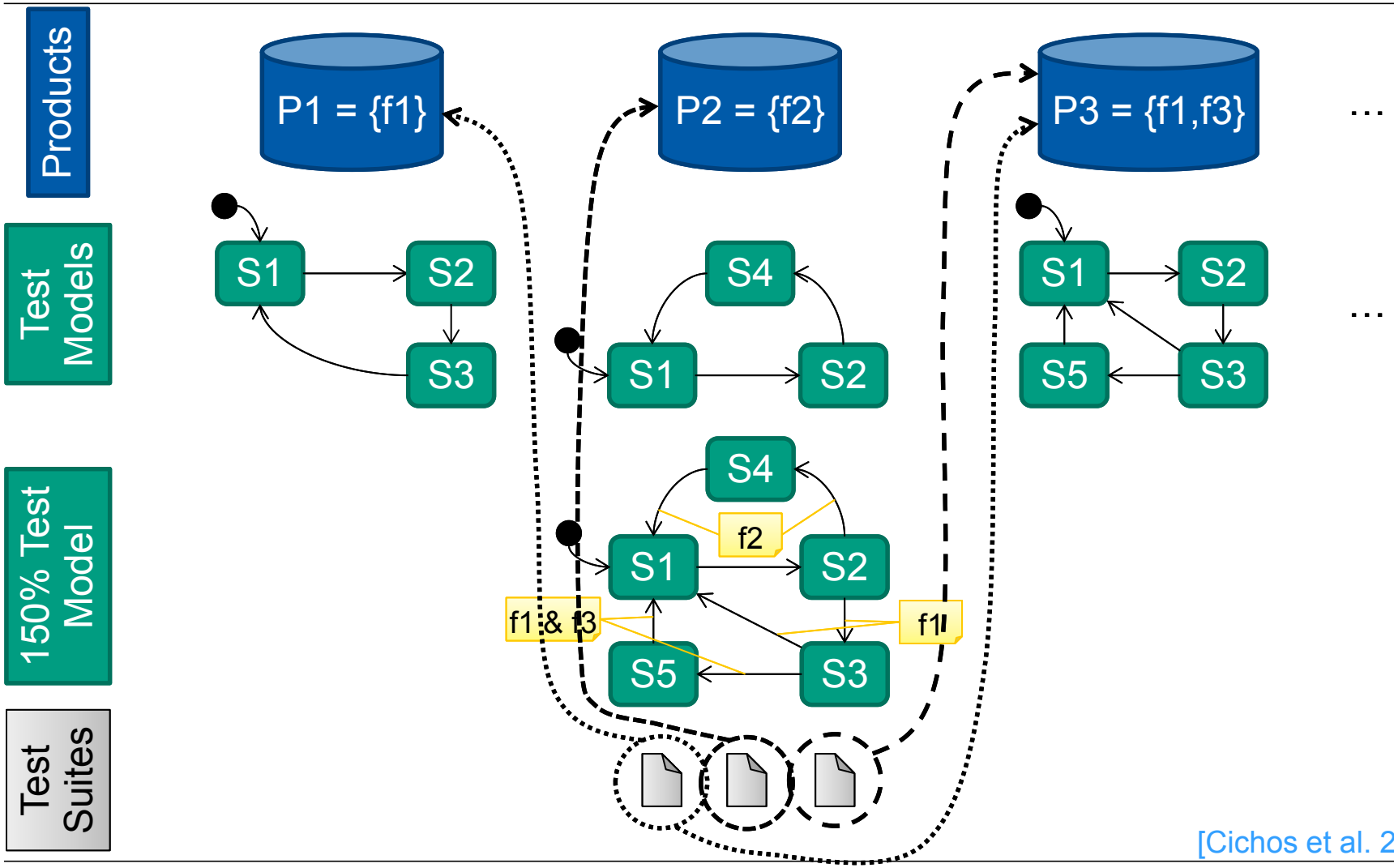
Testing Software Product Lines



Generate test cases for every product individually is inefficient!

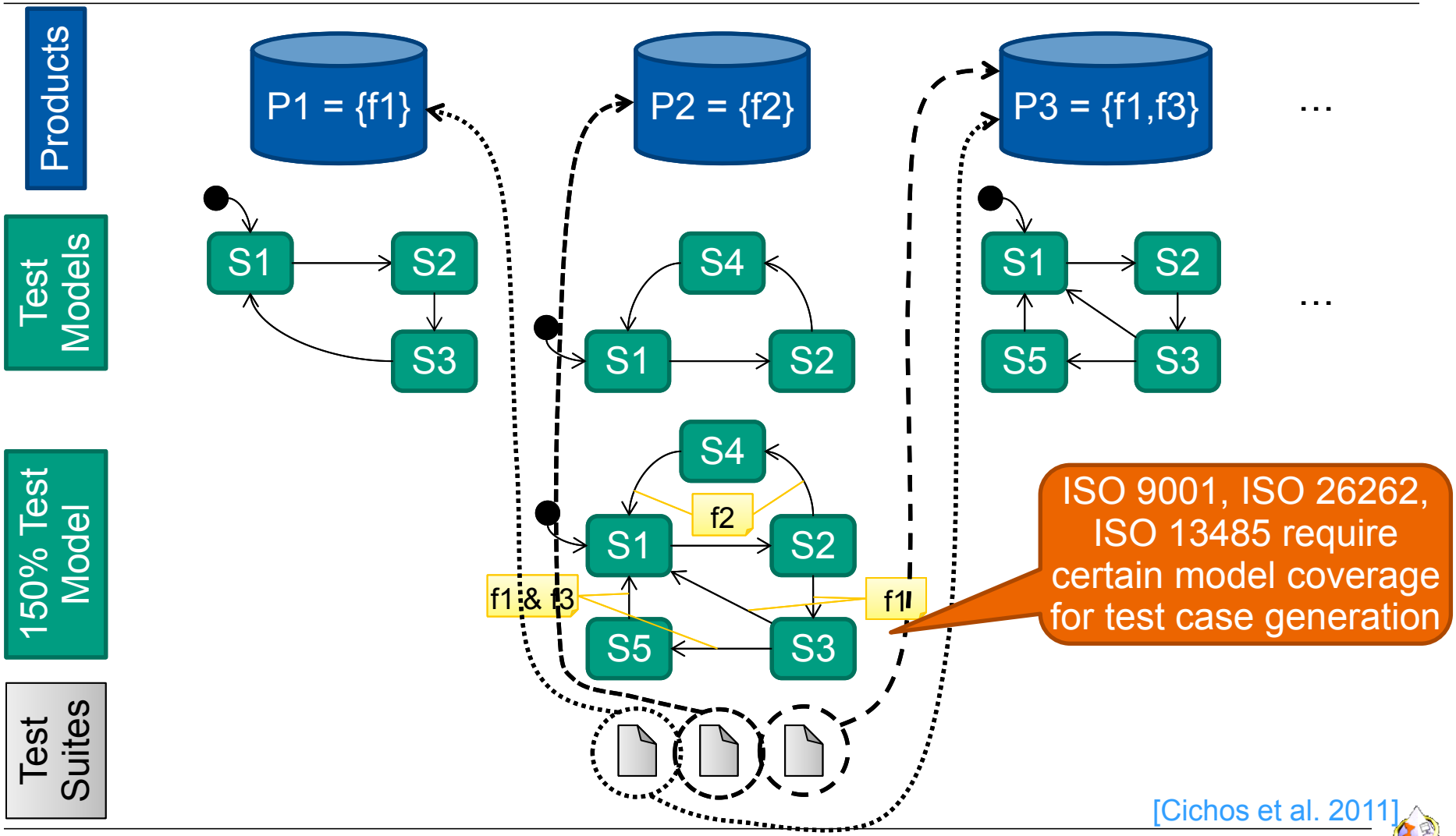
[Cichos et al. 2011]

Testing Software Product Lines



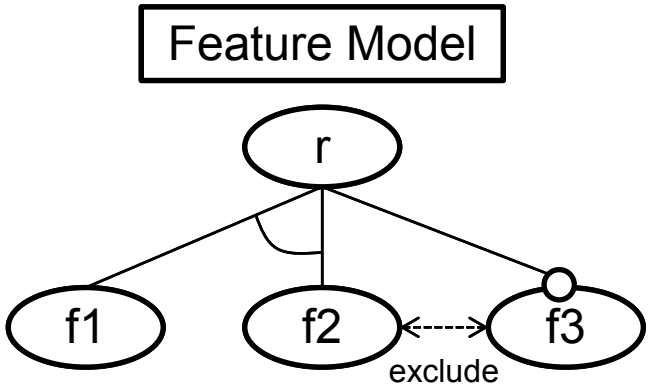
[Cichos et al. 2011]

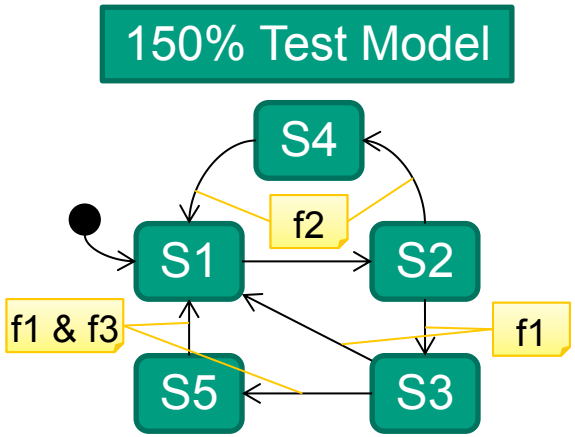
Testing Software Product Lines



[Cichos et al. 2011]

SPL Test Suite Generation



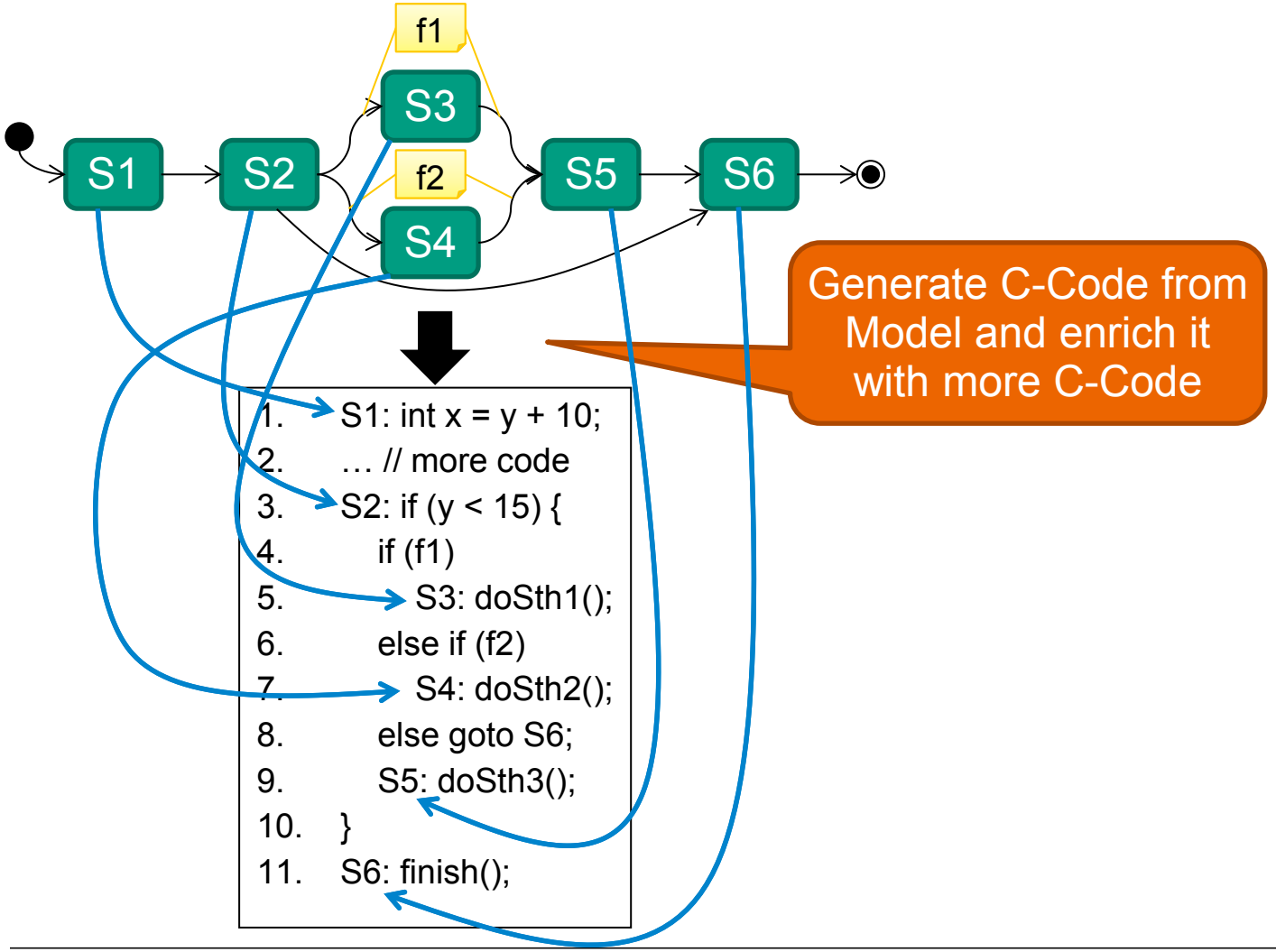


SPL Test Suite (State Coverage)

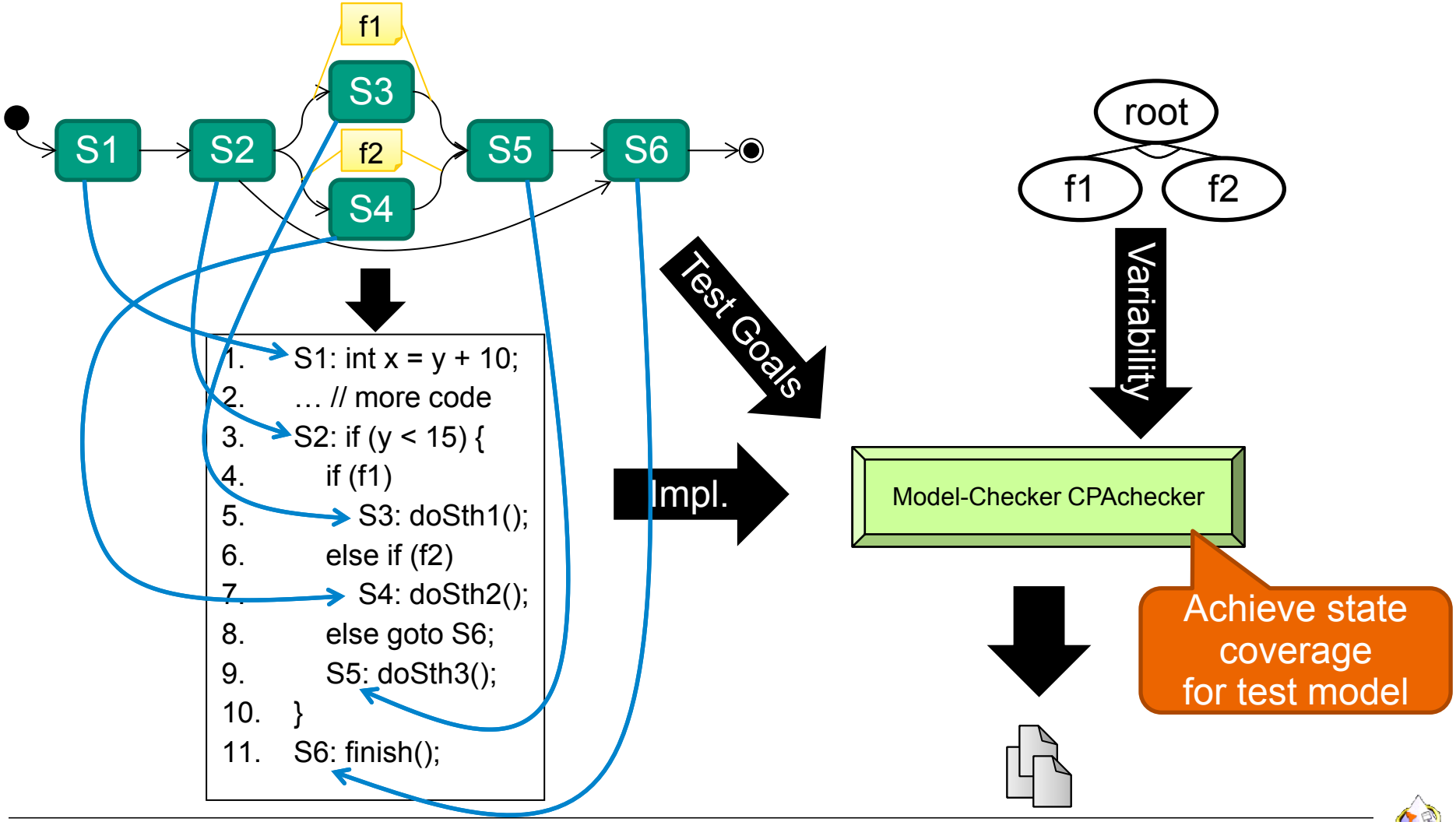
Config. / TC	S1-S2-S3	S1-S2-S4	S1-S2-S3-S5
(r, f1)	X		
(r, f2)		X	
(r, f1, f3)	X		X

[Cichos et al. 2011], [Cichos et al. 2012], [Lochau et al. 2014]

Using Symbolic Model Checking for Test Case Generation



Using Symbolic Model Checking for Test Case Generation



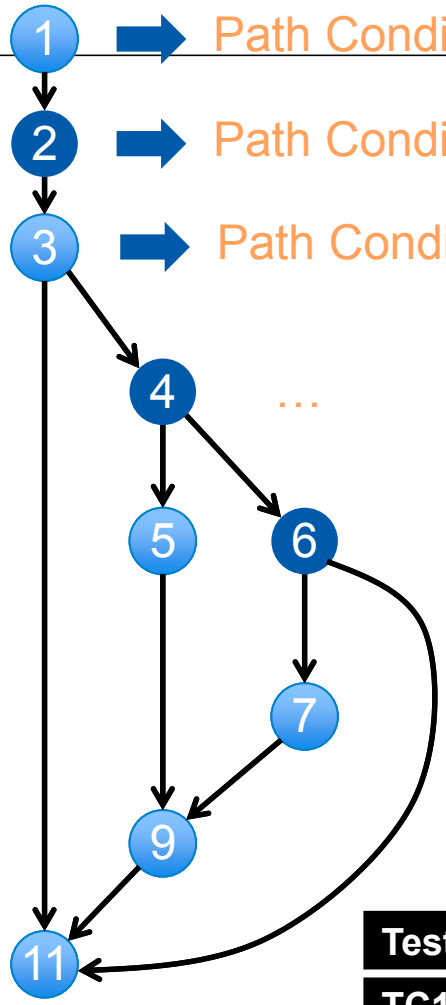
Test Case Derivation from Counterexamples

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

Assignment

Assumption



Path Condition

Path Condition

Path Condition

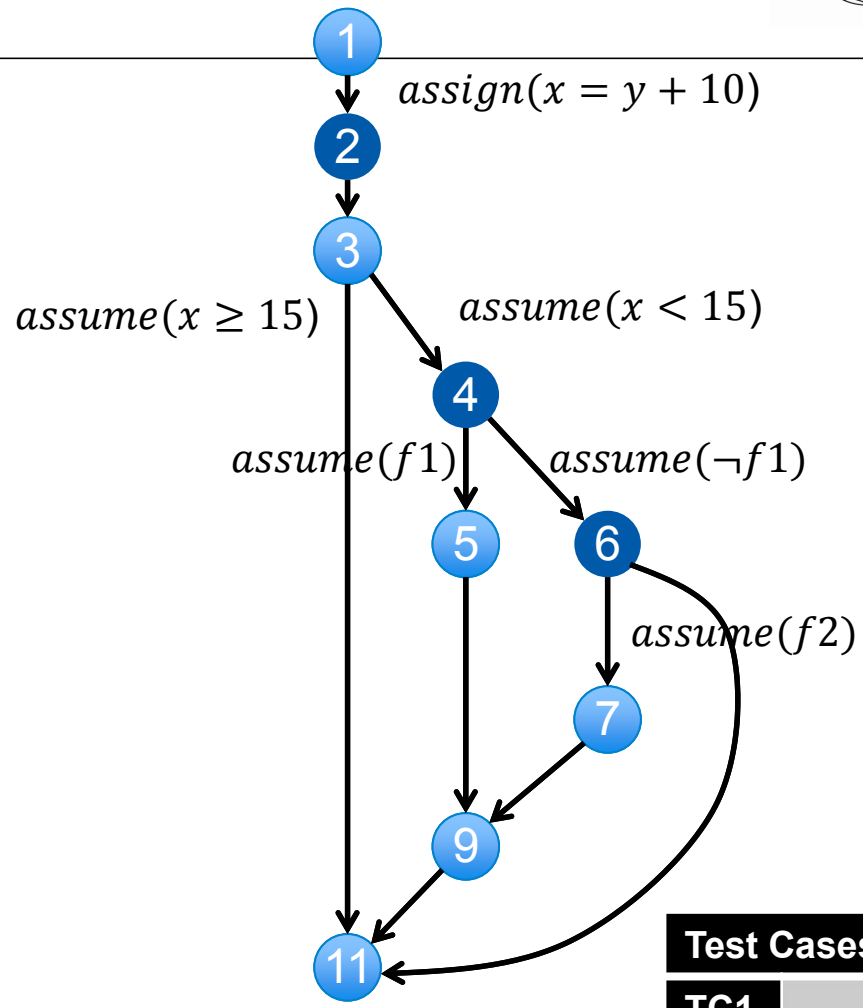
	(r,f1)	(r,f2)
S1		
S2		
S3		
S4		
S5		
S6		

Test Cases		
TC1		
TC2		
TC3		

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```



	(r,f1)	(r,f2)
S1		
S2		
S3		
S4		
S5		
S6		

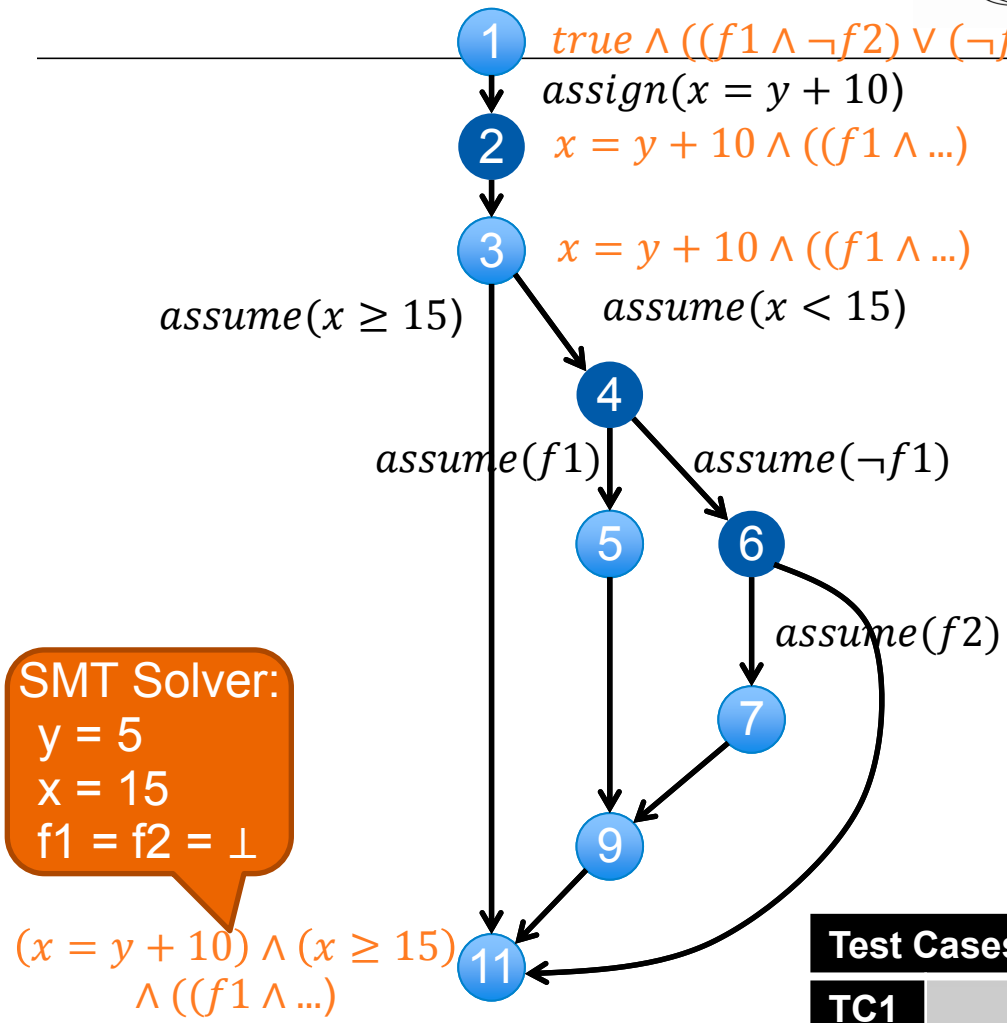
Test Cases		
TC1		
TC2		
TC3		

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1		
S2		
S3		
S4		
S5		
S6		



SMT Solver:
y = 5
x = 15
f1 = f2 = ⊥

Test Cases		
TC1		
TC2		
TC3		

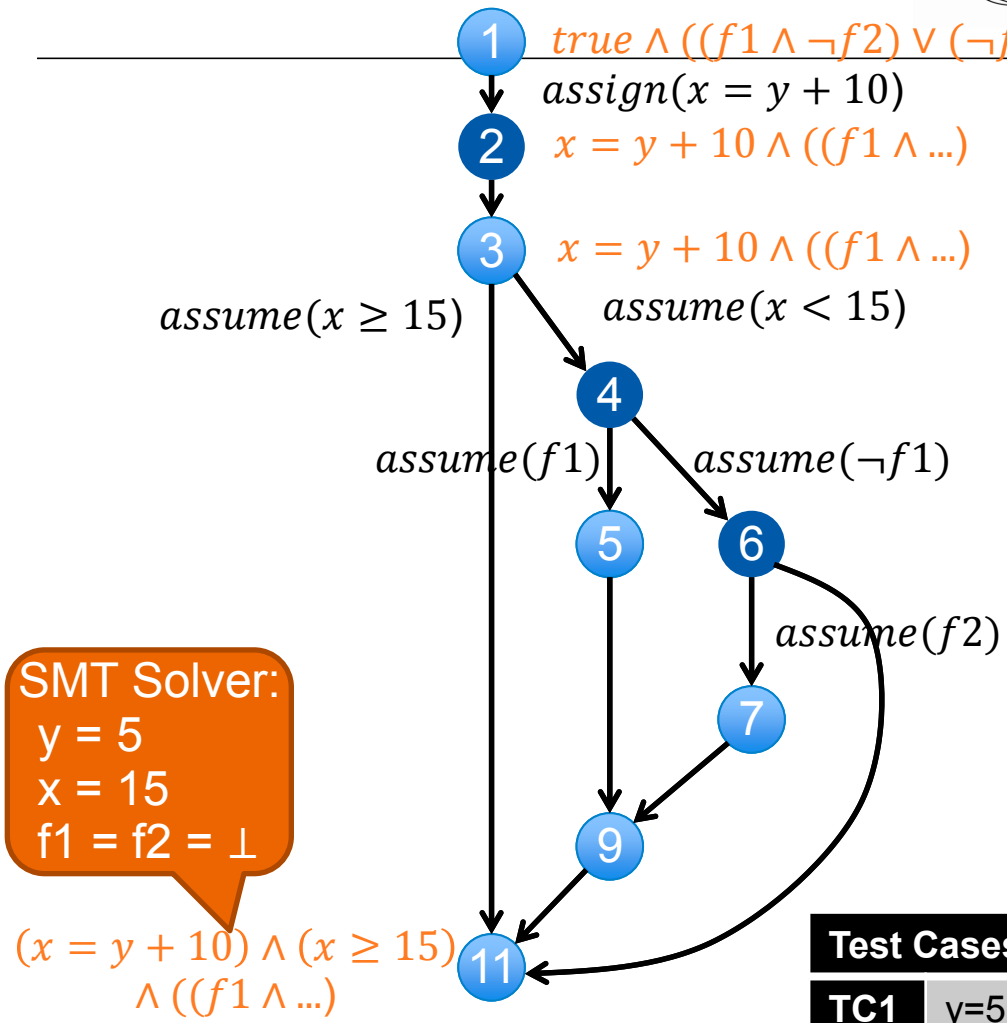
[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3		
S4		
S5		
S6	TC1	TC1



SMT Solver:
y = 5
x = 15
f1 = f2 = ⊥

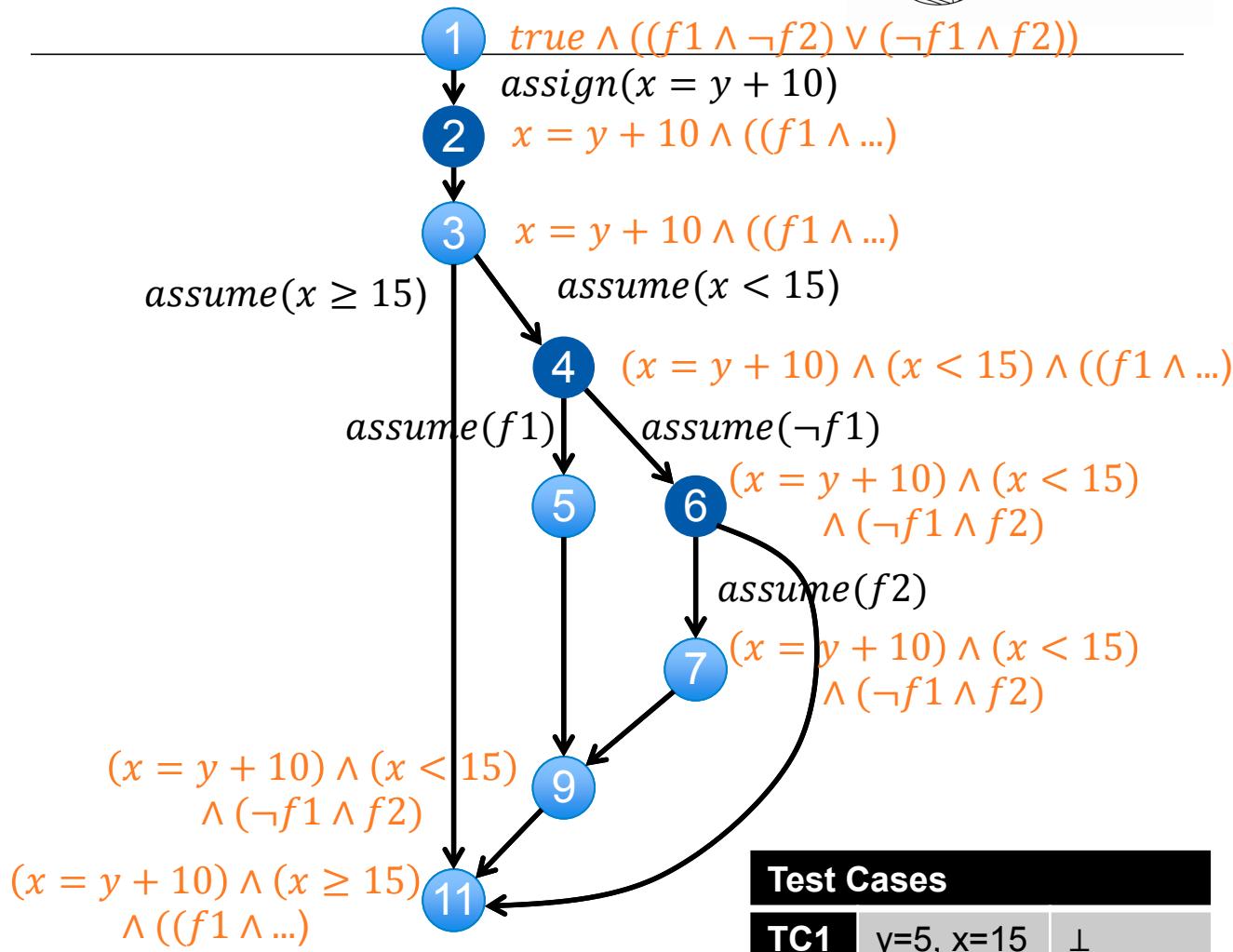
Test Cases		
TC1	y=5, x=15	⊥
TC2		
TC3		

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3		
S4		
S5		
S6	TC1	TC1



Test Cases		
TC1	y=5, x=15	⊥
TC2		
TC3		

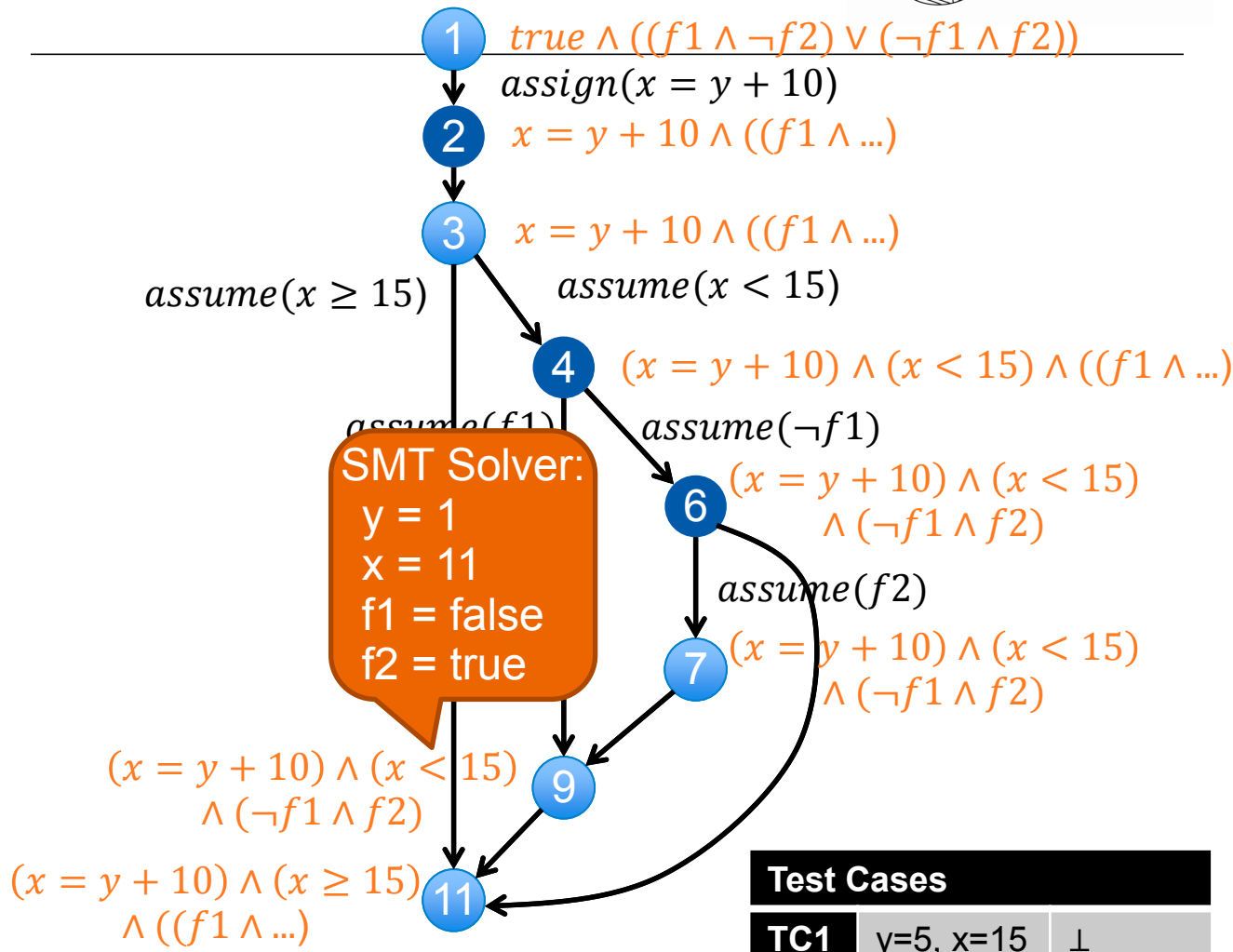
[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3		
S4		TC2
S5		TC2
S6	TC1	TC1



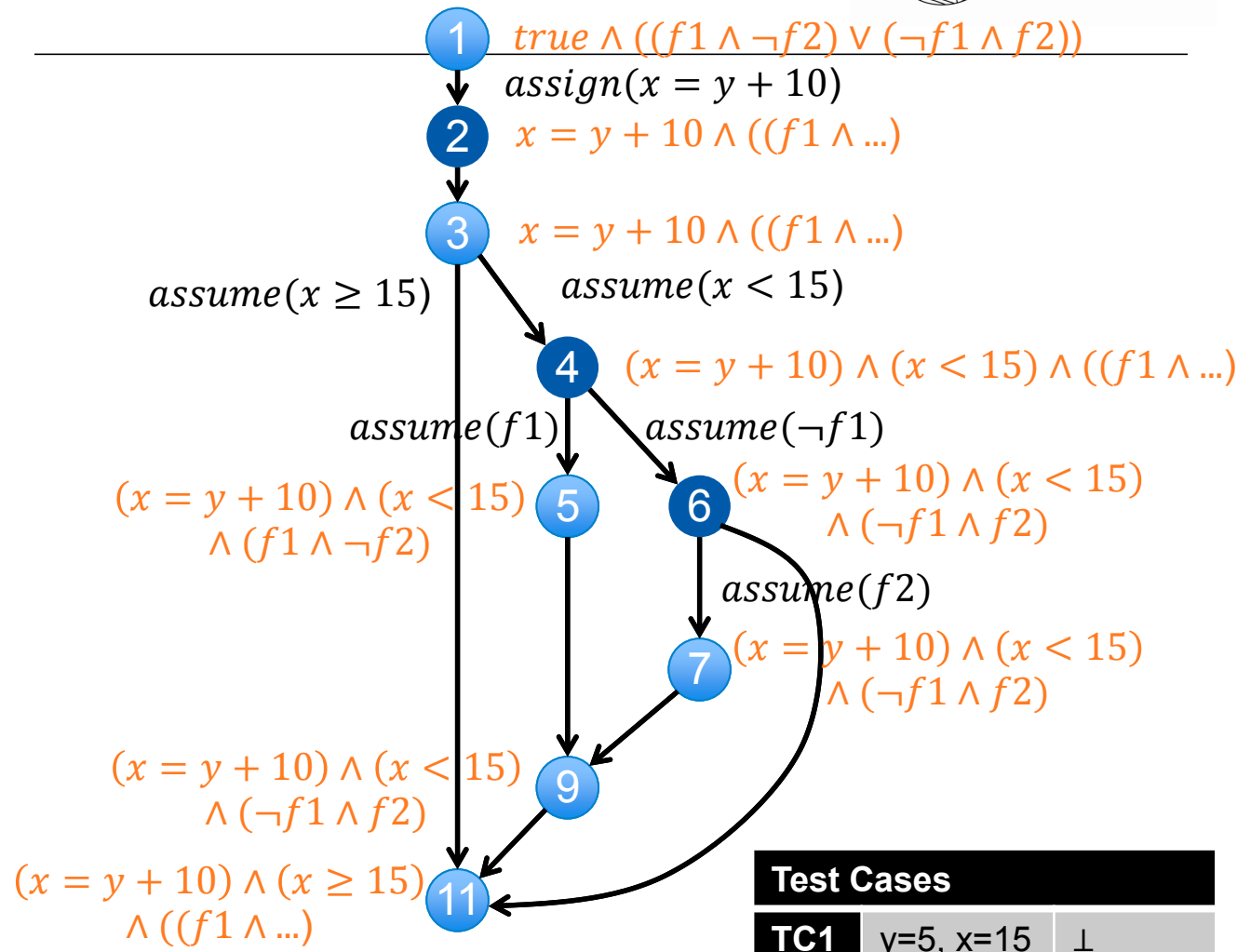
Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3		

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3		
S4		TC2
S5		TC2
S6	TC1	TC1



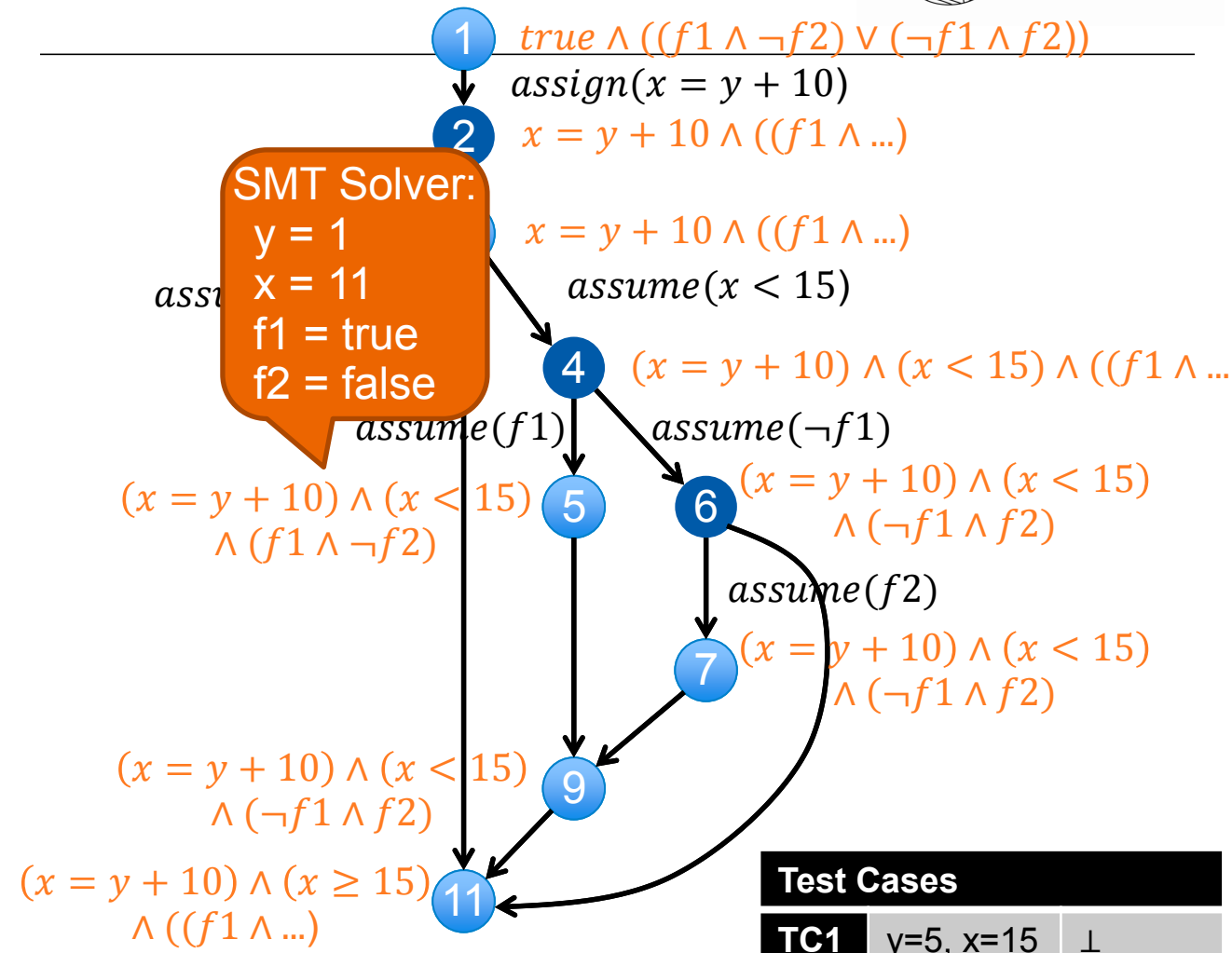
Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3		

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3	TC3	
S4		TC2
S5		TC2
S6	TC1	TC1



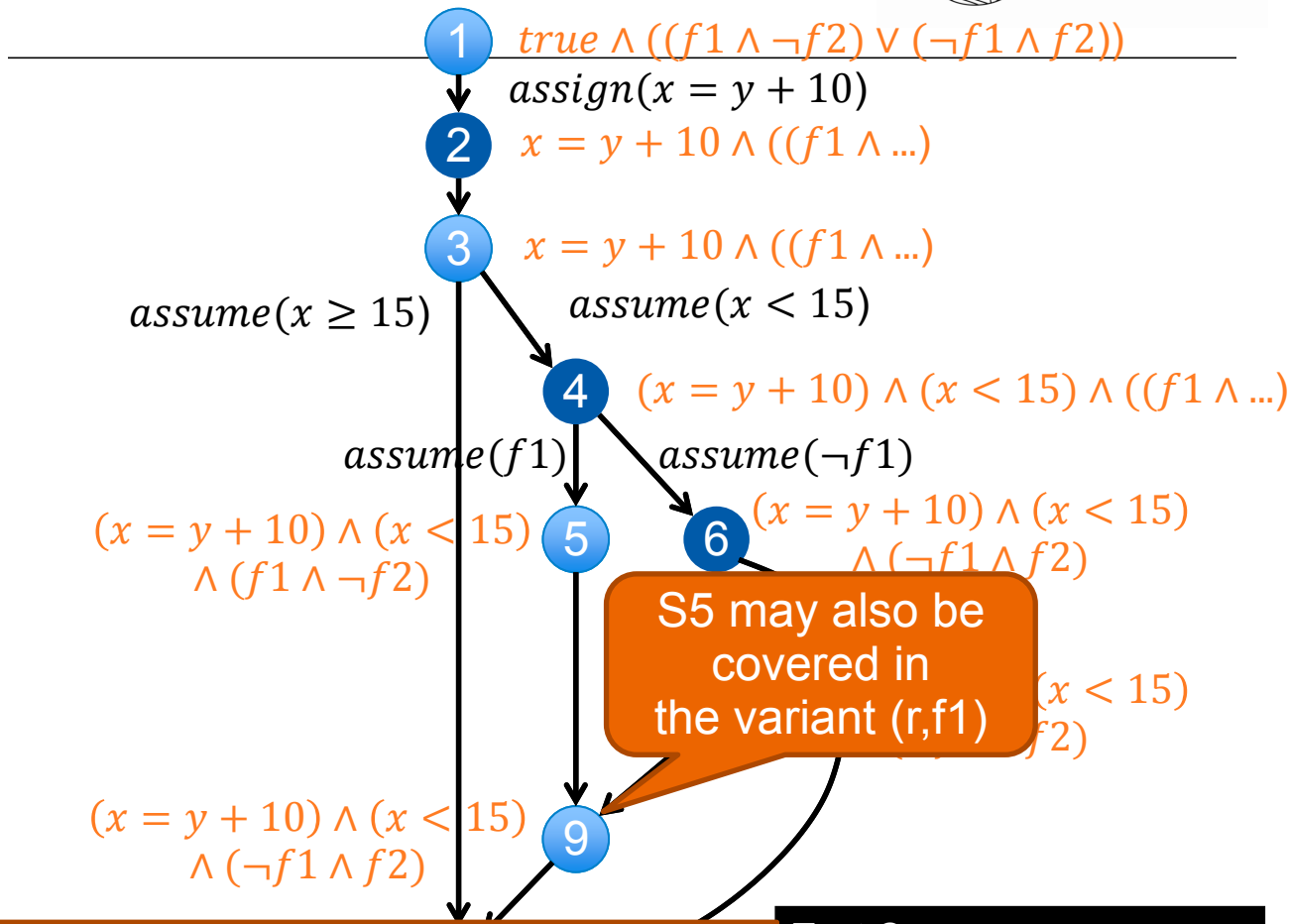
Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3	y=11, x=1	f1∧ ¬f2

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3	TC3	
S4		TC2
S5		TC2
S6	TC1	TC1



S5 may also be covered in the variant (r,f1)

A BDD analysis checks if a location may be reached by a path with another feature configuration.

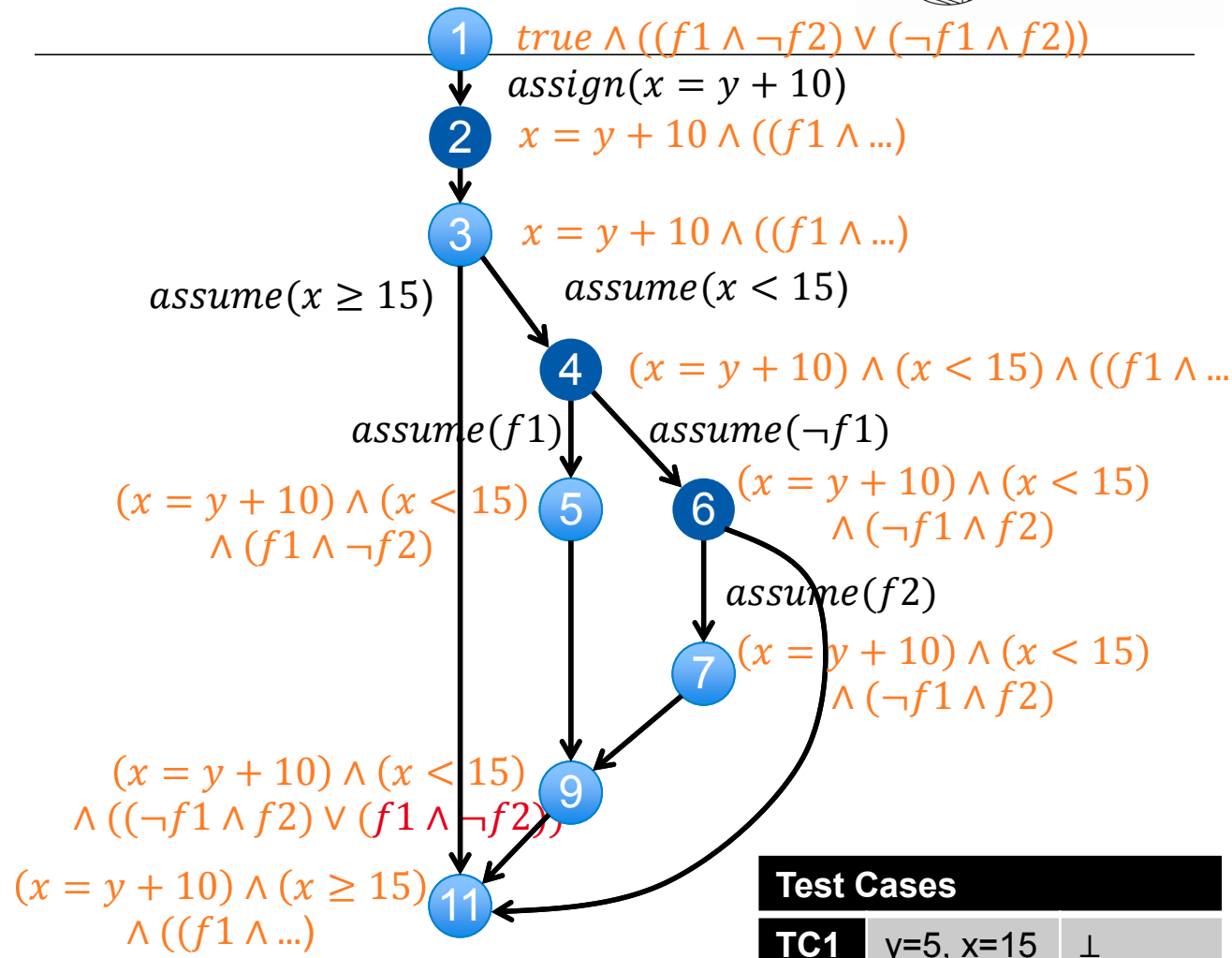
Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3	y=11, x=1	f1∧ ¬f2

[Beye

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3	TC3	
S4		TC2
S5		TC2
S6	TC1	TC1



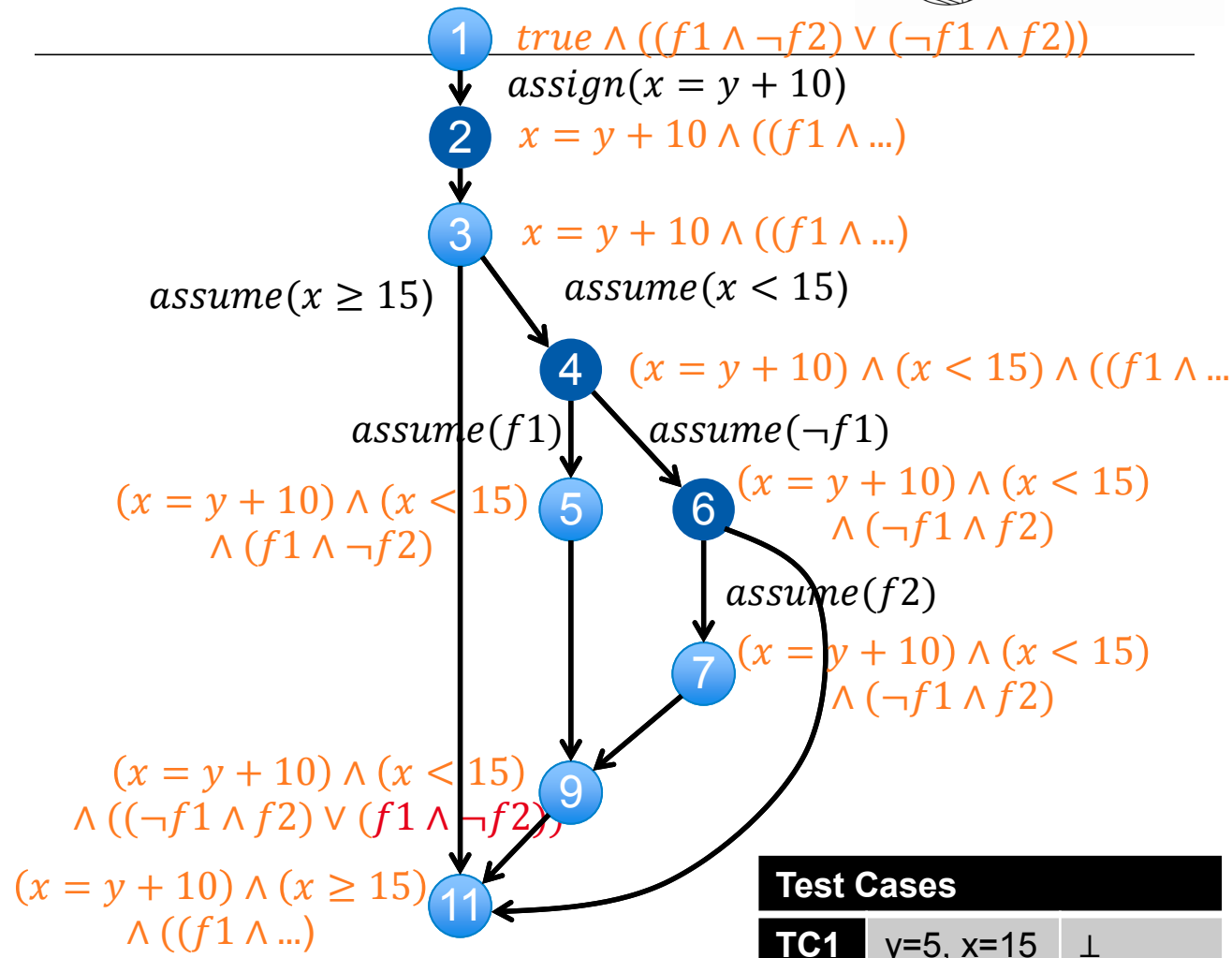
Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3	y=11, x=1	f1∧ ¬f2

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

Test Case Derivation from Counterexamples

```
1.   S1: int x = y + 10;
2.   … // more code
3.   S2: if (y < 15) {
4.       if (f1)
5.           S3: doSth1();
6.       else if (f2)
7.           S4: doSth2();
8.       else goto S6;
9.       S5: doSth3();
10.  }
11.  S6: finish();
```

	(r,f1)	(r,f2)
S1	TC1	TC1
S2	TC1	TC1
S3	TC3	
S4		TC2
S5	TC3	TC2
S6	TC1	TC1



Test Cases		
TC1	y=5, x=15	⊥
TC2	y=11, x=1	¬f1∧f2
TC3	y=11, x=1	⊥

[Beyer et al. 2004], [Rhein et al. 2011], [Apel et al. 2013]

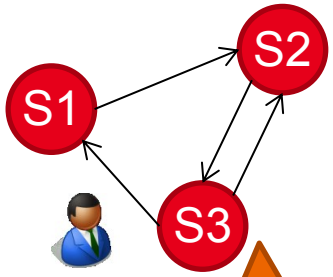
Testing Dynamic Software Product Lines

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Testing SPLs:** test that every derivable variant is correctly implemented

- **Testing DSPLs:** test that all possible reconfigurations are correct

Environment

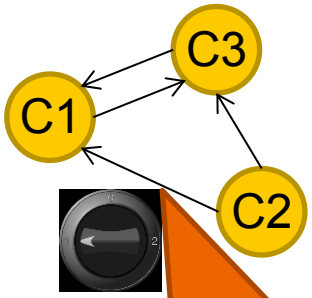


Non-deterministic influence on program variables

Program

```
1.  S1: int x = y + 10;
2.  … // more code
3.  S2: if (y < 15) {
4.      if (f1)
5.          S3: doSth1();
6.      else if (f2)
7.          S4: doSth2();
8.      else goto S6;
9.      S5: doSth3();
10. }
11. S6: finish();
```

Reconfiguration



Non-deterministic influence on feature variables

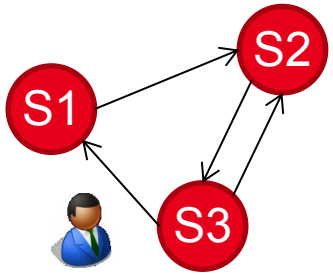
Non-deterministic behavior

Testing Dynamic Software Product Lines

- **Testing SPLs:** test that every derivable variant is correctly implemented

- **Testing DSPLs:** test that all possible reconfigurations are correct

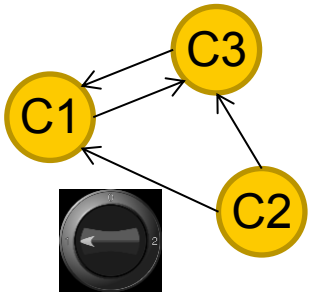
Environment



Program

```
1.      S1: int x = y +
        10;
2.      … // more code
3.      S2: if (y < 15) {
4.          if (f1)
5.              S3:
        doSth1();
6.          else if (f2)
7.              S4:
        doSth2();
8.          else goto S6;
9.          S5: doSth3();
10.     }
11.     S6: finish();
```

Reconfiguration



Non-deterministic behavior

- CPAchecker offers techniques to handle non-deterministic behavior

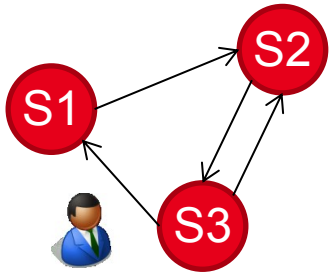
Testing Dynamic Software Product Lines

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Testing SPLs:** test that every derivable variant is correctly implemented

- **Testing DSPLs:** test that all possible reconfigurations are correct

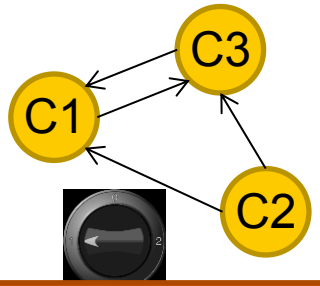
Environment



Program

```
1.    S1: int x = y + 10;
2.    … // more code
3.    S2: if (y < 15) {
4.        if (f1)
5.            S3: doSth1();
6.        else if (f2)
7.            S4: doSth2();
8.        else goto S6;
9.        S5: doSth3();
10.   }
11.   S6: finish();
```

Reconfiguration



Non-deterministic behavior

1. How to test reconfigurations efficiently?

- CPAchecker offers techniques to handle non-deterministic behavior

2. What is a suitable coverage criteria?

3. How to ensure liveness of the reconfiguration life cycle?

List of Literature

[Cichos et al. 2011] H. Cichos, S. Oster, M. Lochau, A. Schürr : *Model-Based Coverage-Driven Test Suite Generation for Software Product Lines*. MoDELS 2011: 425-439

[Cichos et al. 2012] H. Cichos, M. Lochau, S. Oster, A. Schürr : *Reduktion von Testsuiten für Software-Produktlinien*. Software Engineering 2012: 143-154

[Lochau et al. 2014] M. Lochau, J. Bürdek, S. Lity, M. Hagner, C. Legat, U. Goltz, A. Schürr : *Applying Model-based Software Product Line Testing Approaches to the Automation Engineering Domain*. AT Journal 2014 (submitted)

[Holzer et al. 2011] A. Holzer, M. Tautschnig, C. Schallhart, H. Veith : *An Introduction to Test Specification in FQL*. In S. Barner, I. Harris, D. Kroening, and O. Raz, editors, Hardware and Software: Verification and Testing, volume 6504 of Lecture Notes in Computer Science, pages 9–22. Springer Berlin Heidelberg, 2011.

[Beyer et al. 2011] D. Beyer and M. E. Keremoglu : *CPAchecker: A Tool for Configurable Software Verification*. In Proc. CAV, pages 184-190, Springer, 2011.

[Holzmann 1997] G. J. Holzmann : *The Model Checker Spin*, IEEE Trans. on Software Engineering, Vol. 23, No. 5, May 1997, pp. 279-295.

List of Literature (2/2)

[Beyer et al. 2004] D. Beyer, A. J. Chlipala, T. A. Henzinger, R. Jhala, and R. Majumdar. *Generating Test from Counterexamples*. In *Proc. of the 26th ICSE*, pages 326–335, 2004.

[Rhein et al. 2011] A. von Rhein, S. Apel, F. Raimondi : *Introducing Binary Decision Diagrams in the Explicit-State Verification of Java Code*. Java Pathfinder Workshop (co-located with ASE'11), 2011.

[Apel et al. 2013] S. Apel, A. von Rhein, P. Wendler, A. Größlinger, D. Beyer : *Strategies for Product-Line Verification: Case Studies and Experiments*. In Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE), pages 482–491. IEEE Computer Society, 2013.