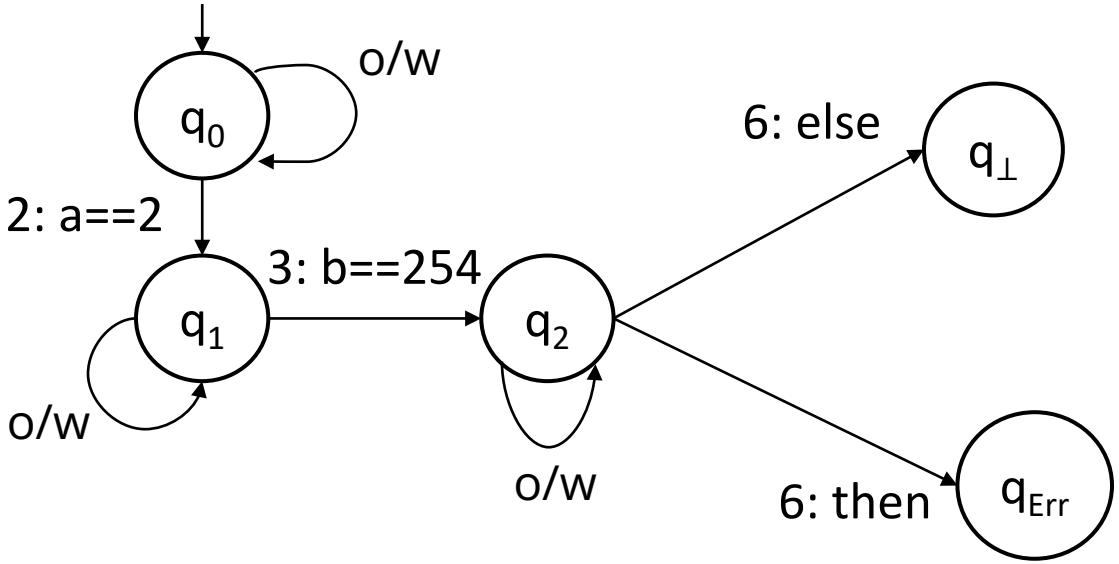
ConcurrentWitness2Test:
Test-Harnessing the Power of Concurrency

Levente Bajczi
Zsófia Ádám
Zoltán Micskei

Critical Systems Research Group

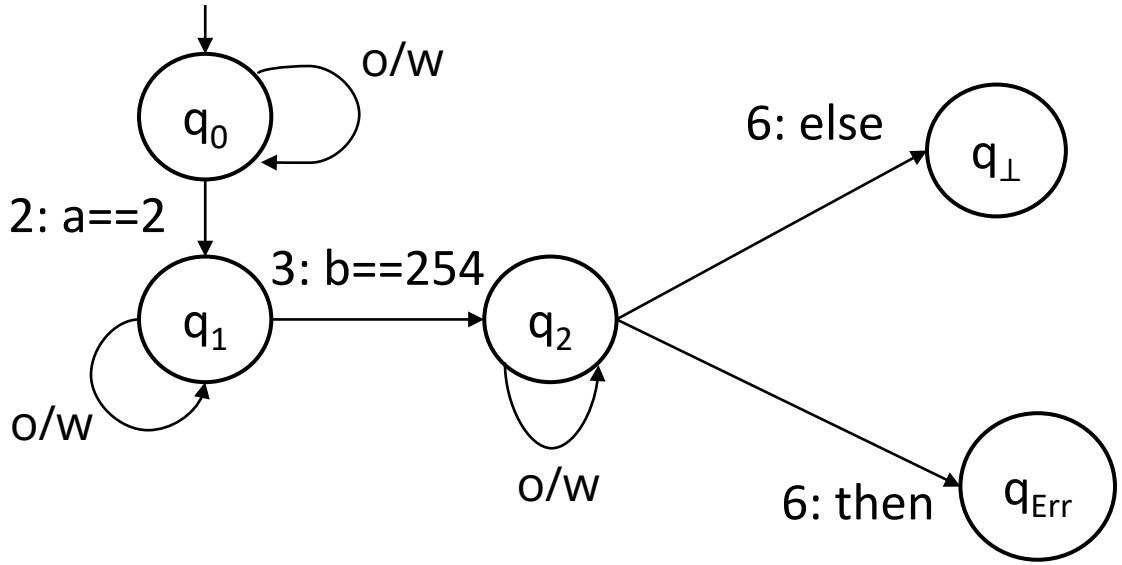
Test-harnesses in validation



(o/w = otherwise)

```
1   int main(void) {
2       unsigned char a = nondet_char();
3       unsigned char b = nondet_char();
4       unsigned char sum = a + b;
5       unsigned char mean = sum / 2;
6       if (mean < a / 2) {
7           error();
8       }
9       return 0;
10  }
```

Test-harnesses in validation

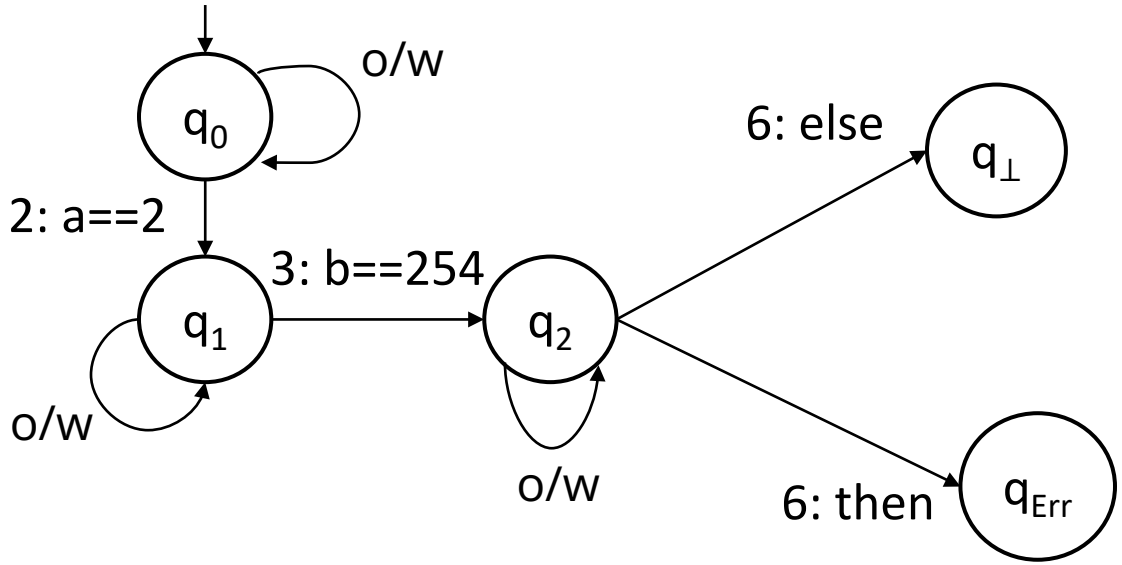


(o/w = otherwise)

```c
int main(void) {
    unsigned char a = nondet_char();
    unsigned char b = nondet_char();
    unsigned char sum = a + b;
    unsigned char mean = sum / 2;
    if (mean < a / 2) {
        error();
    }
    return 0;
}
```

```c
int idx = 0;
unsigned char nondet_char(void) {
    switch(idx) {
      case  0: ++idx; return 2;
      case  1: ++idx; return 254;
      default: return -1;
    }
}
```

Test-harnesses in validation



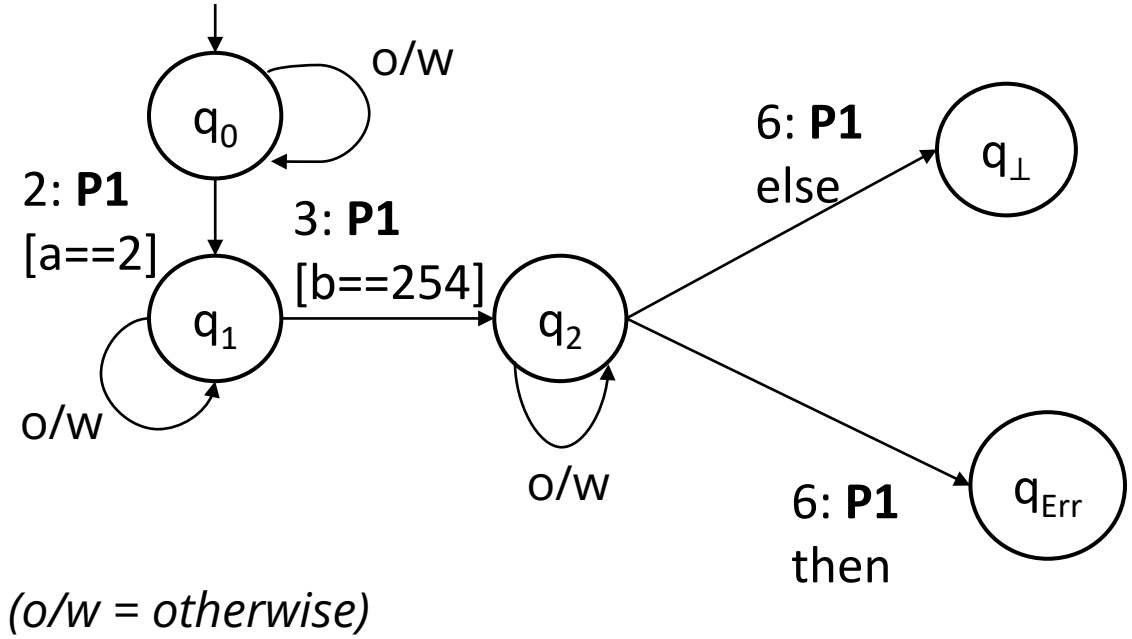
(o/w = otherwise)

Executable counterexample

```
1  int main(void) {
2      unsigned char a = nondet_char();
3      unsigned char b = nondet_char();
4      unsigned char sum = a + b;
5      unsigned char mean = sum / 2;
6      if (mean < a / 2) {
7          error();
8      }
9      return 0;
10 }
```

```
0  int idx = 0;
1  unsigned char nondet_char(void) {
2      switch(idx) {
3        case  0: ++idx; return 2;
4        case  1: ++idx; return 254;
5        default: return -1;
6      }
7  }
```

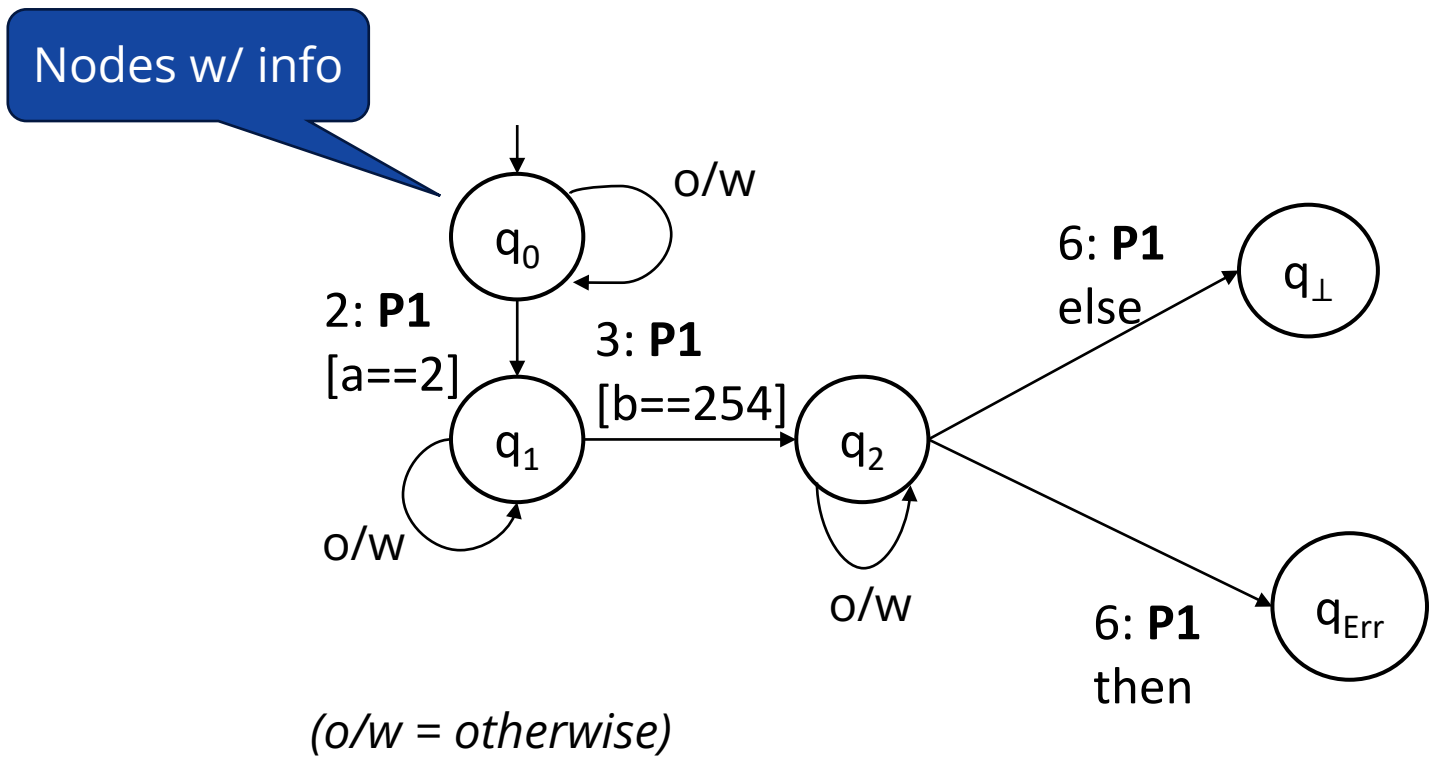
Multi-threaded programs and witnesses



(o/w = otherwise)

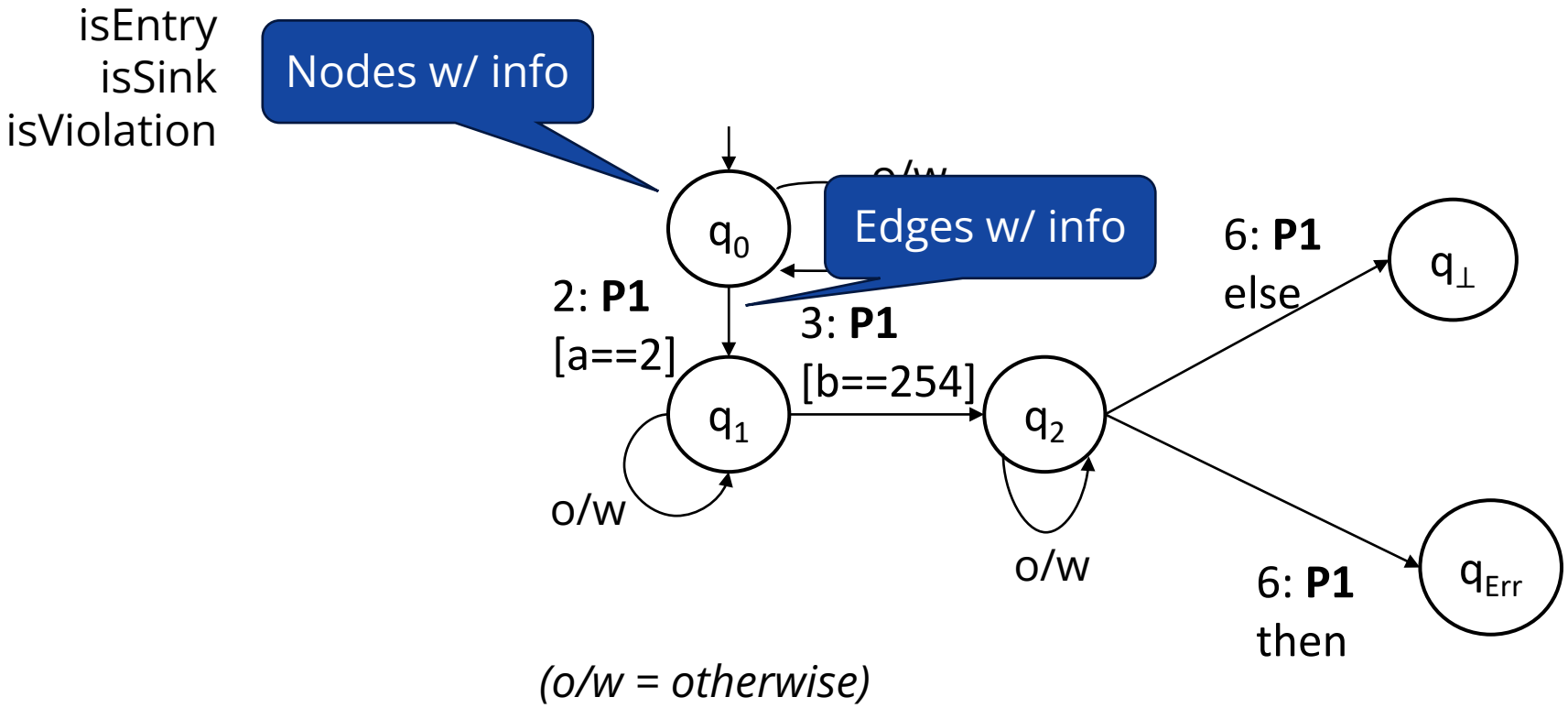
Multi-threaded programs and witnesses

isEntry
isSink
isViolation

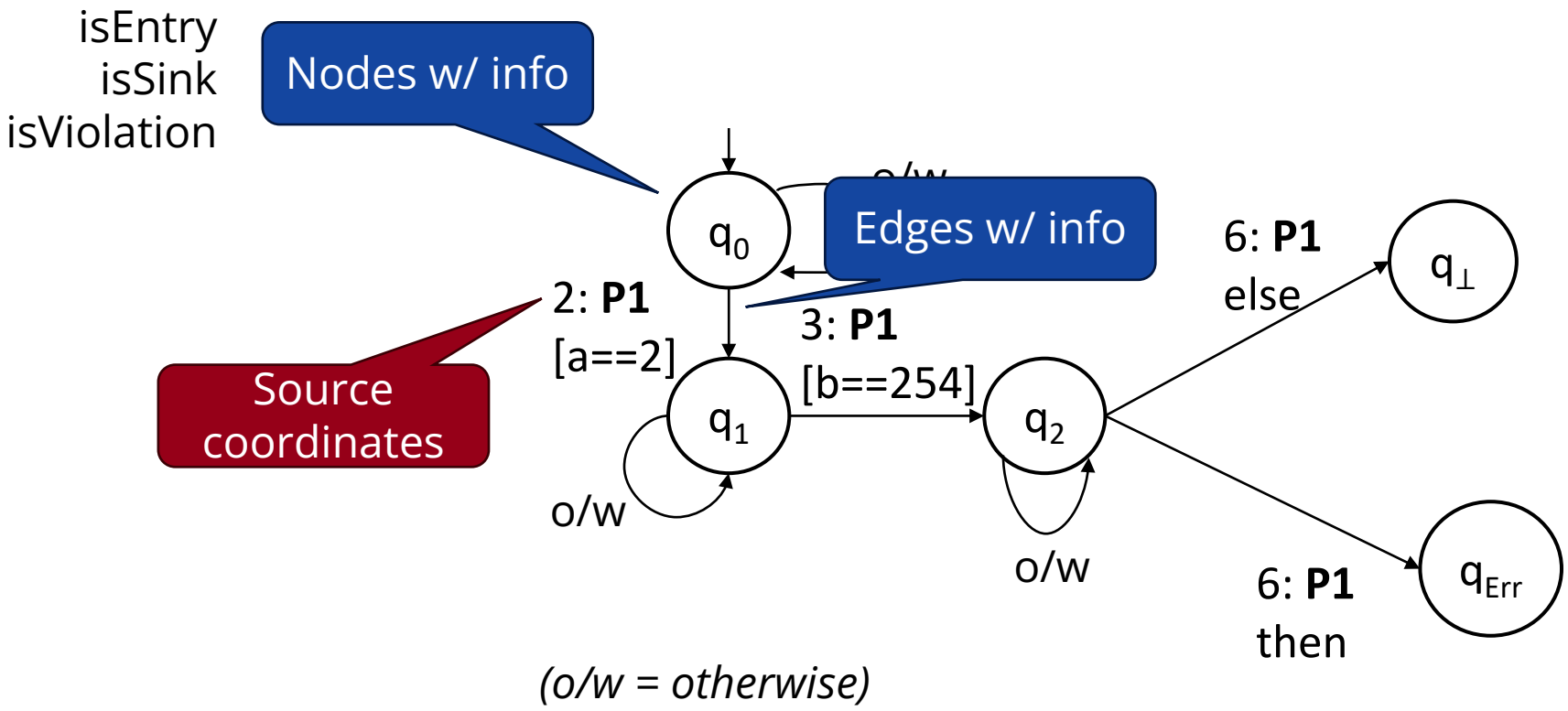


(o/w = otherwise)

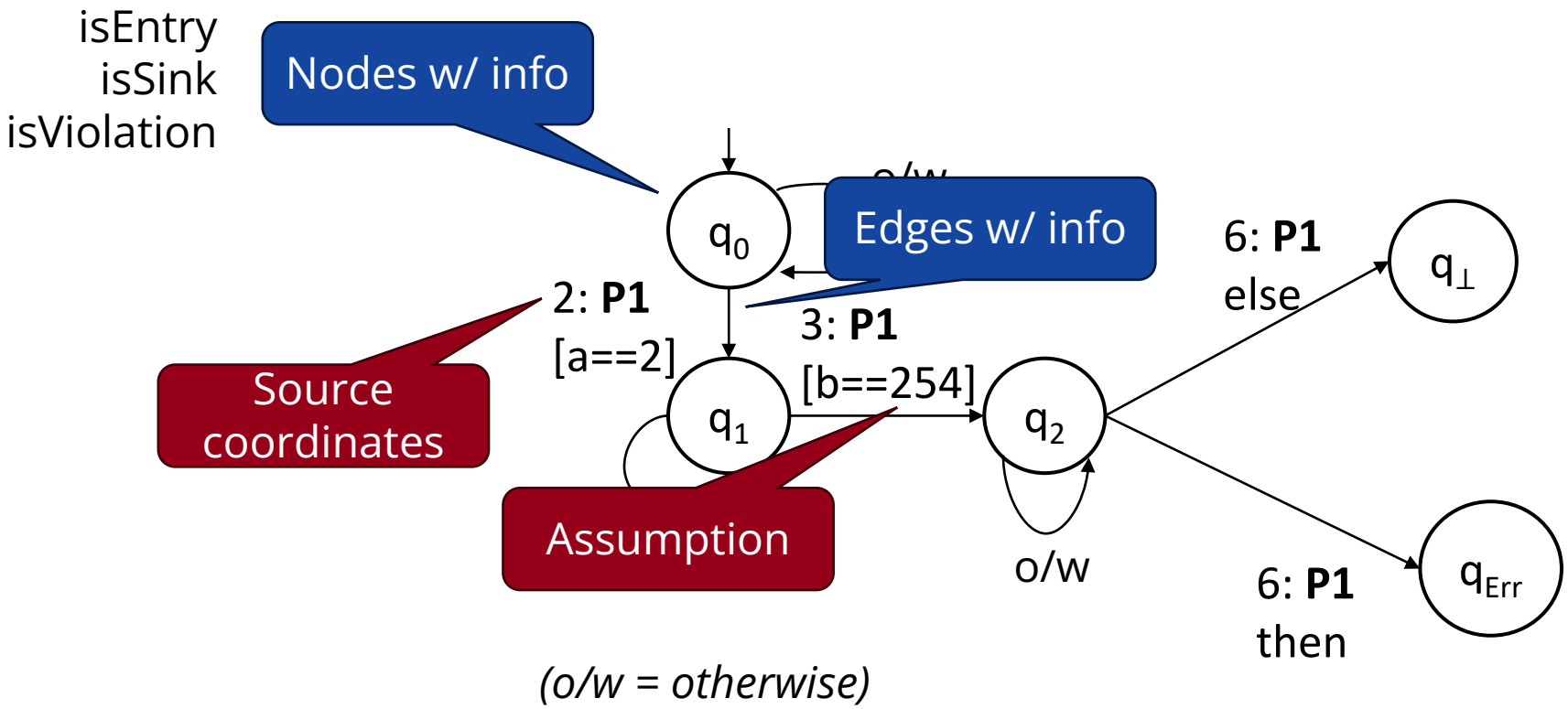
Multi-threaded programs and witnesses



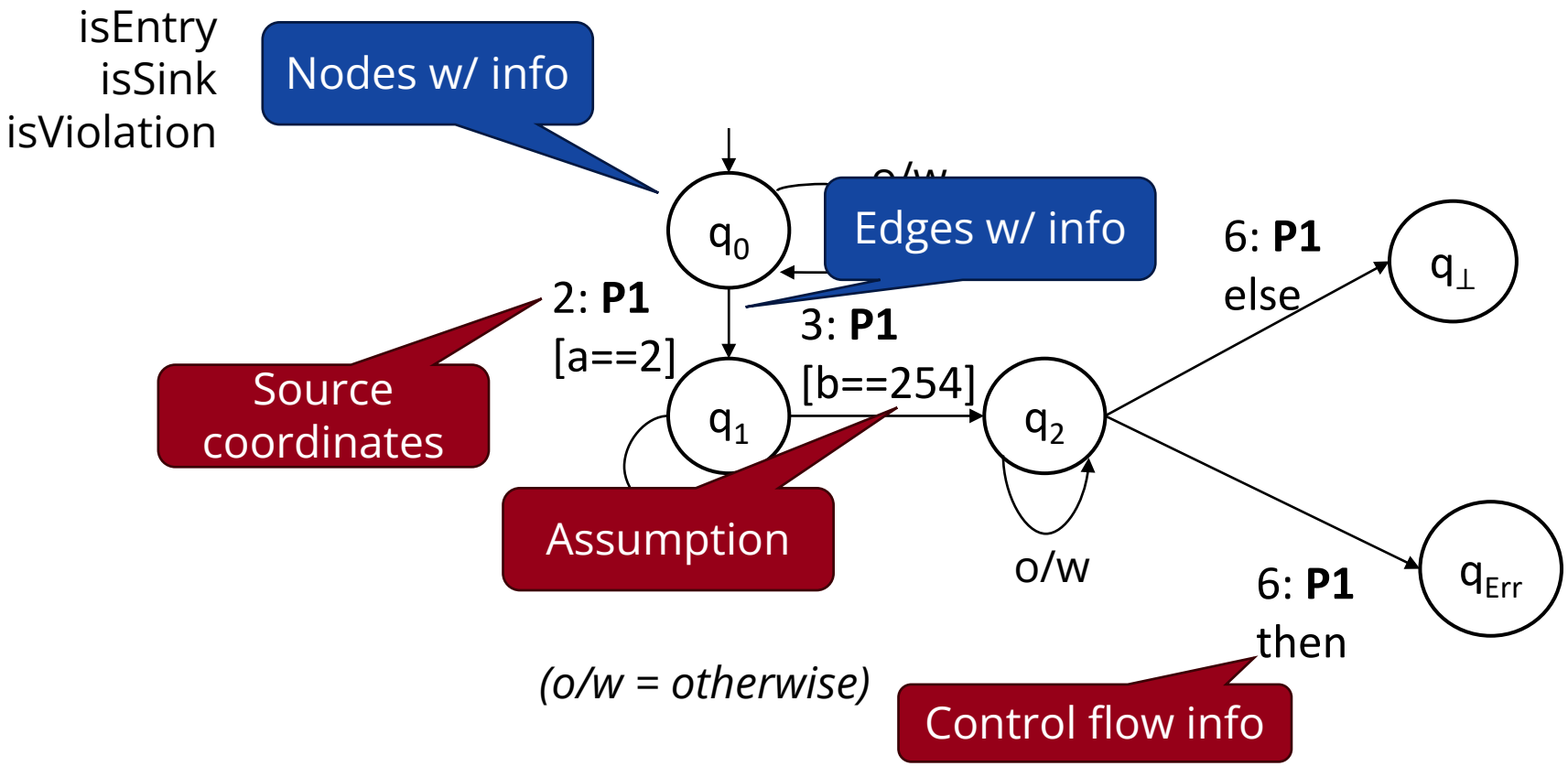
Multi-threaded programs and witnesses



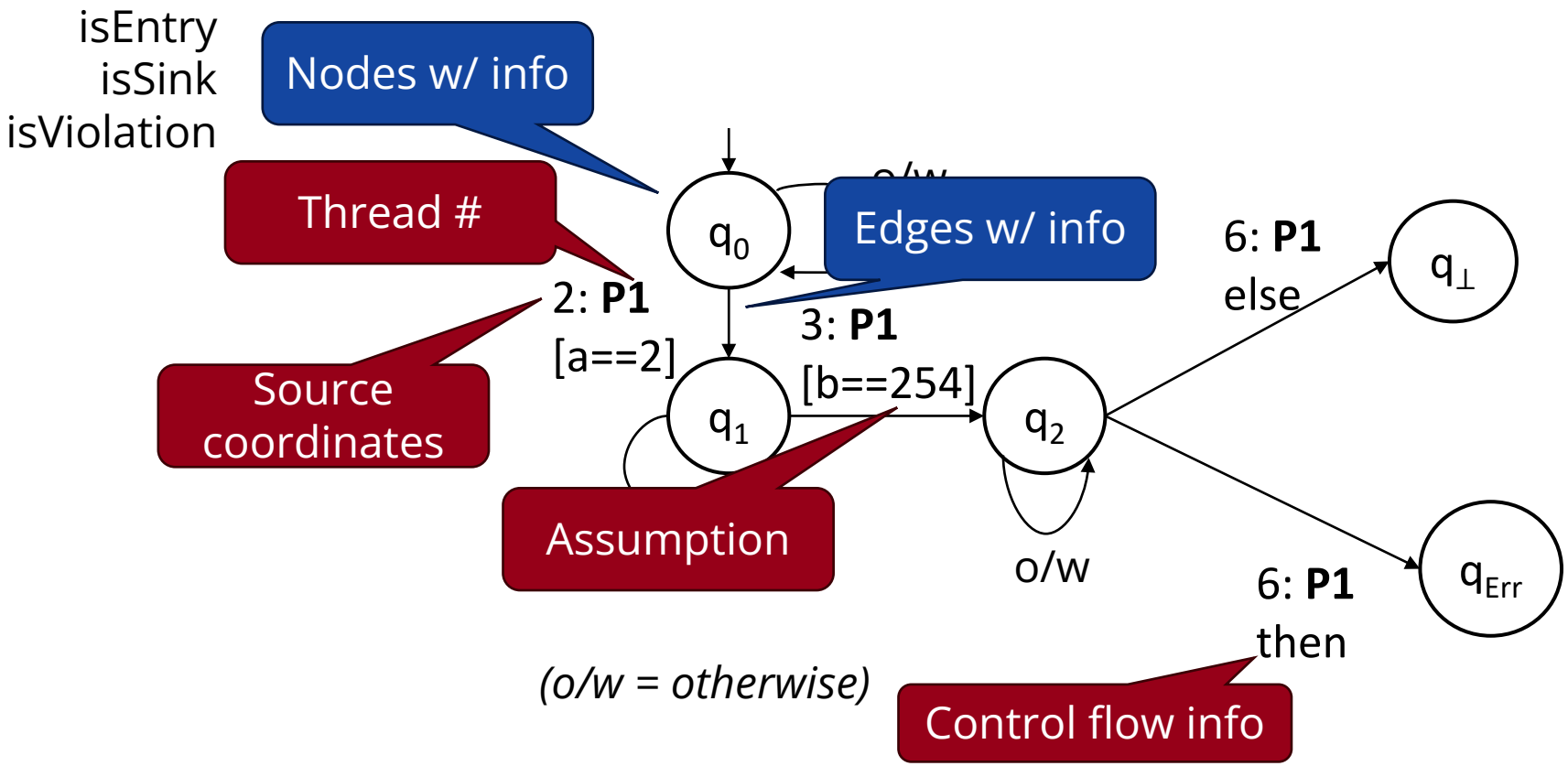
Multi-threaded programs and witnesses



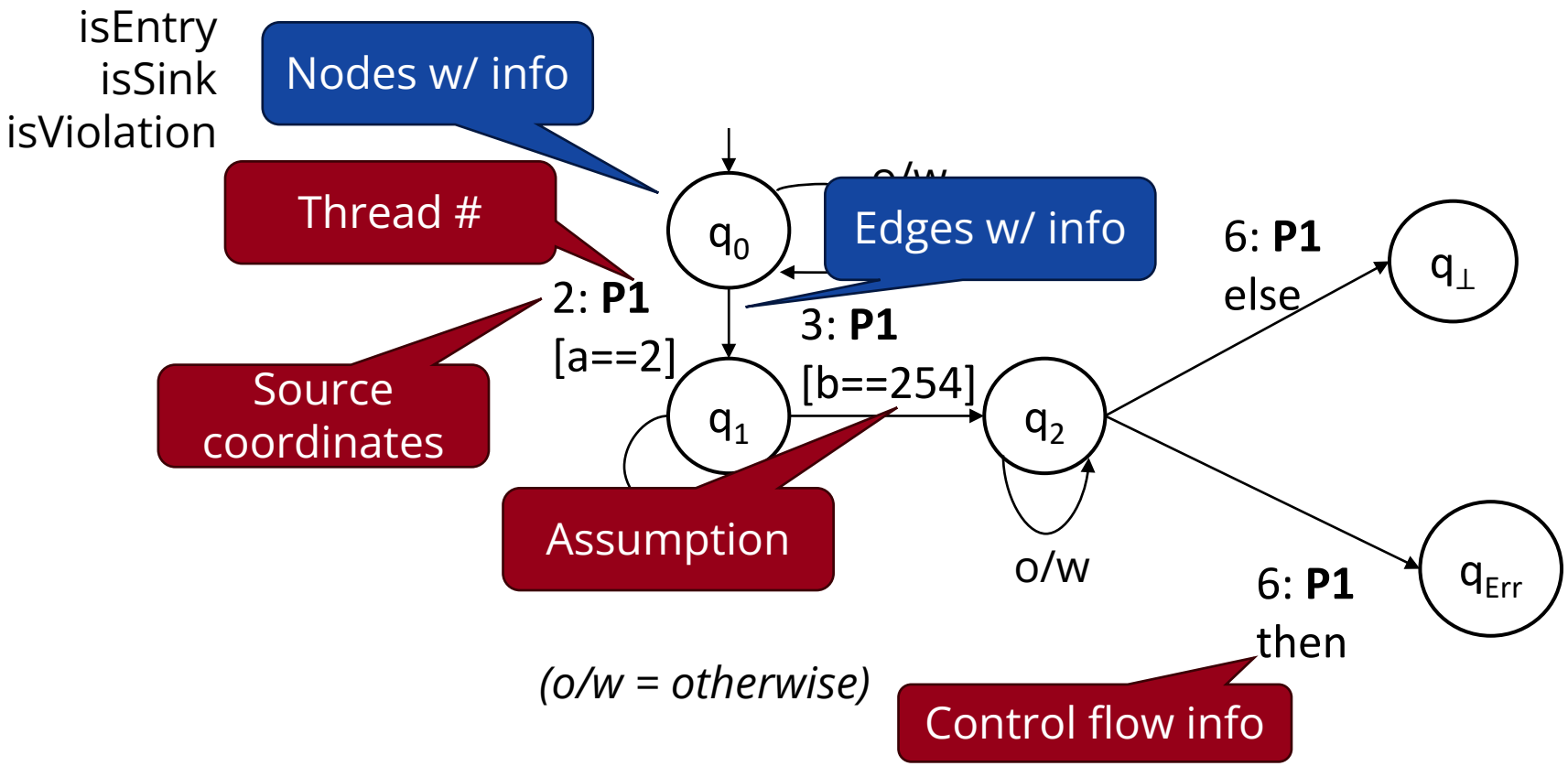
Multi-threaded programs and witnesses



Multi-threaded programs and witnesses



Multi-threaded programs and witnesses



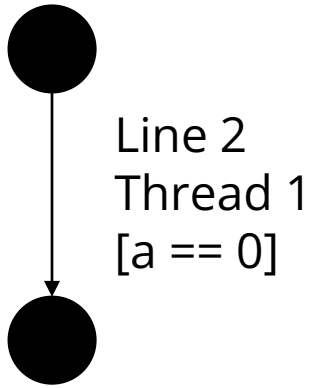
Nothing is mandatory → **flexibility** for verifiers, **complexity** for validators

Handling concurrency

```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```

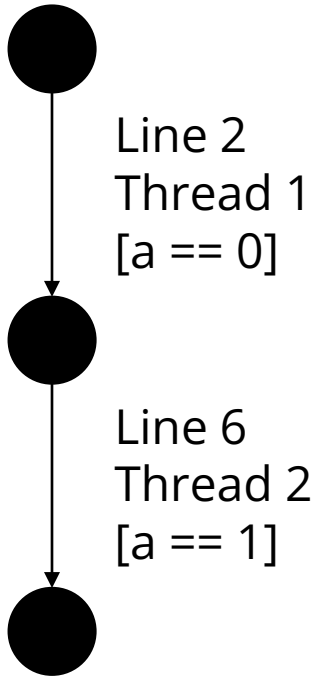
Handling concurrency

```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```



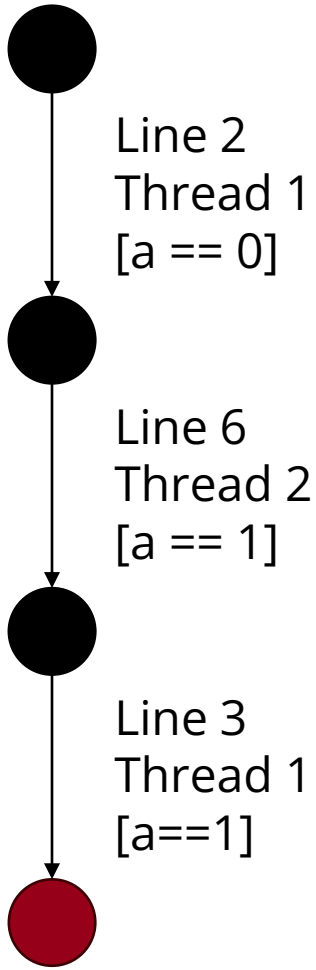
Handling concurrency

```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```



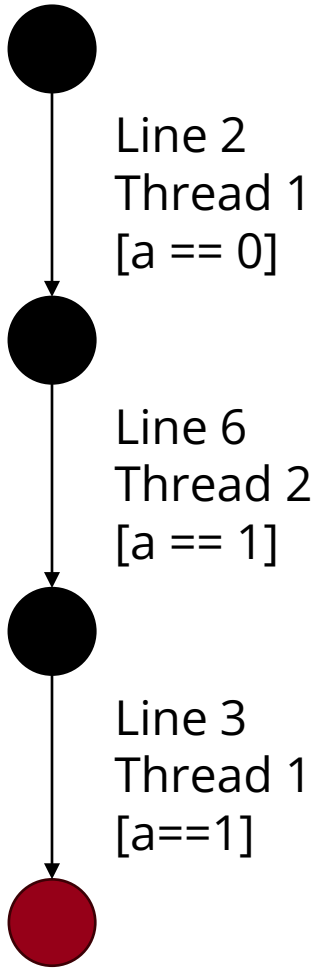
Handling concurrency

```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```



Handling concurrency

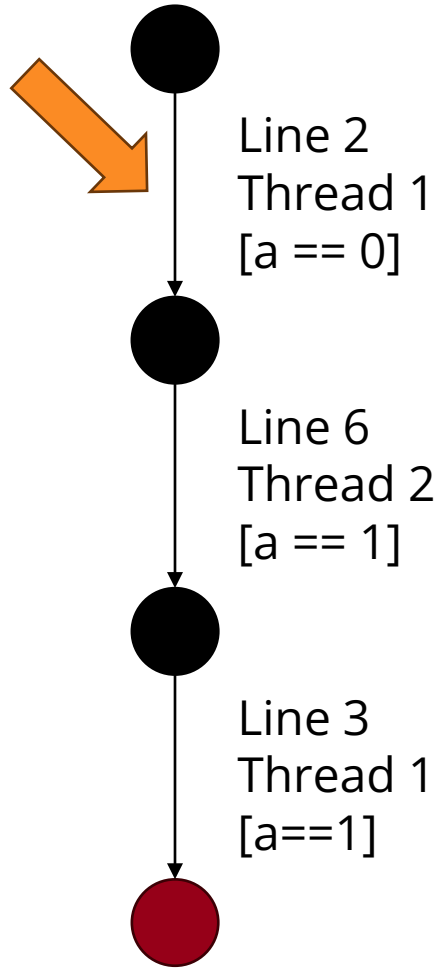
```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```




```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```

Handling concurrency

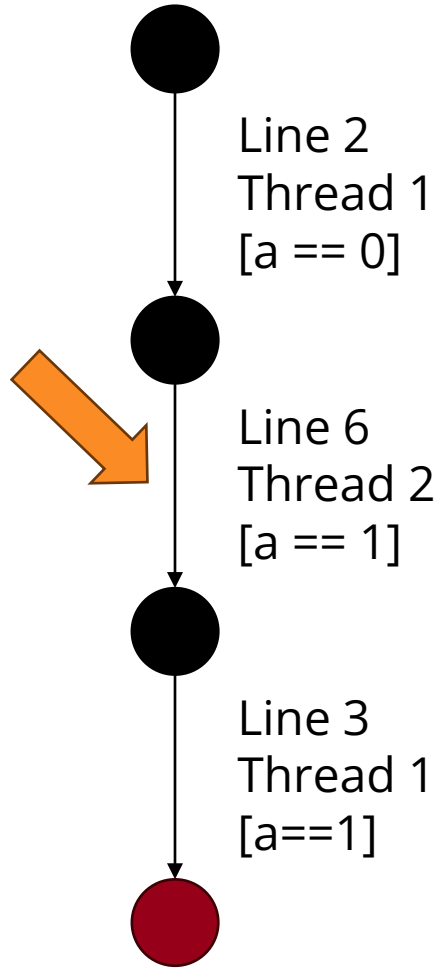
```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```




```
0  int a;

1  void P1(void) {
     yield(0);
2    a = 0;
     release(0);

3    assert(a == 0);

4  }

5  void P2(void) {

6    a = 1;

7  }
```

Handling concurrency

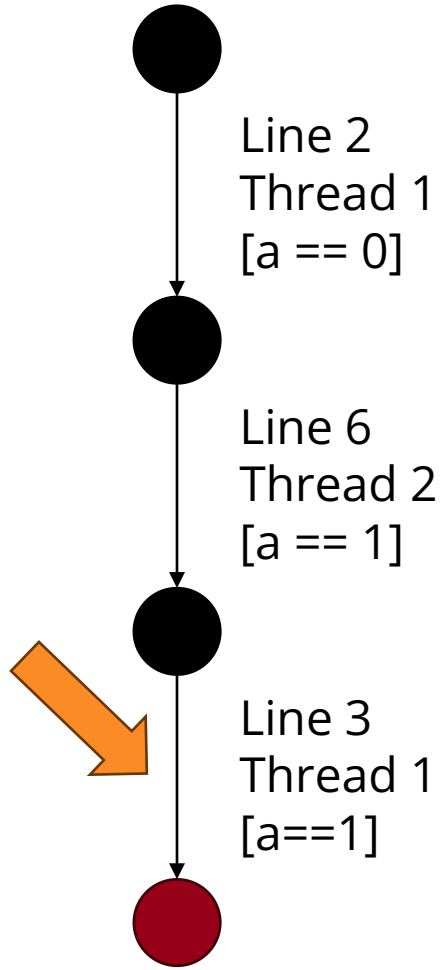
```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```




```
0  int a;

1  void P1(void) {
     yield(0);
2    a = 0;
     release(0);

3    assert(a == 0);

4  }

5  void P2(void) {
     yield(1);
6    a = 1;
     release(1);
7  }
```

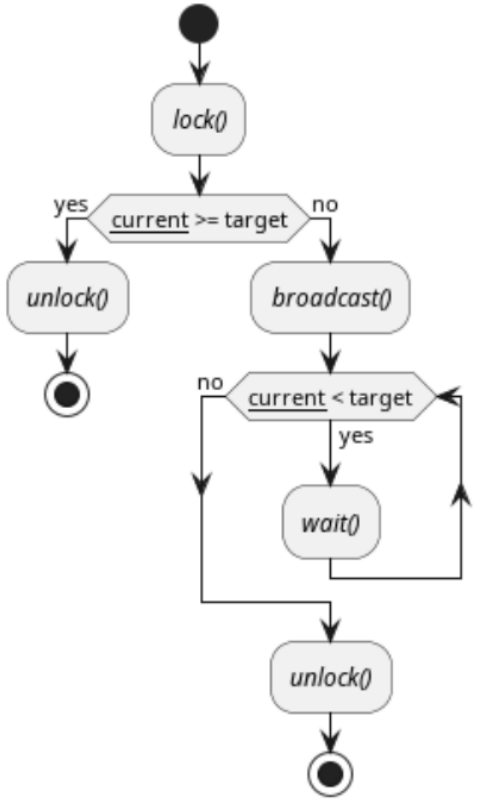
Handling concurrency

```
0  int a;

1  void P1(void) {
2    a = 0;
3    assert(a == 0);
4  }

5  void P2(void) {
6    a = 1;
7  }
```

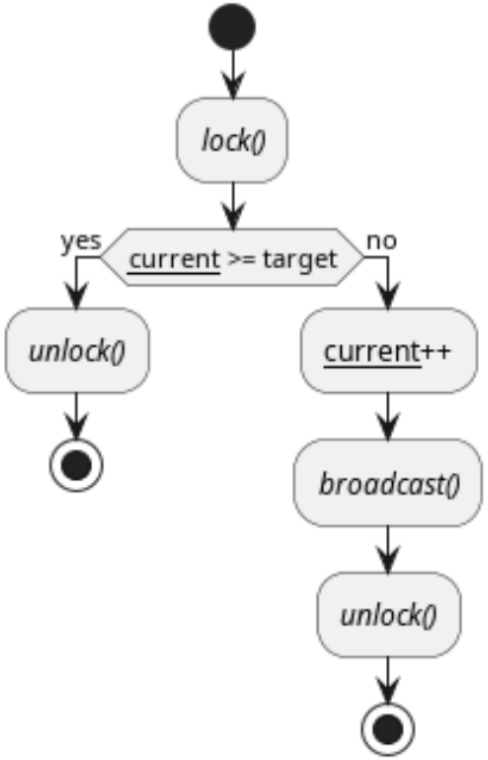



```
0  int a;

1  void P1(void) {
     yield(0);
2    a = 0;
     release(0);
     yield(2);
3    assert(a == 0);
     release(2);
4  }

5  void P2(void) {
     yield(1);
6    a = 1;
     release(1);
7  }
```

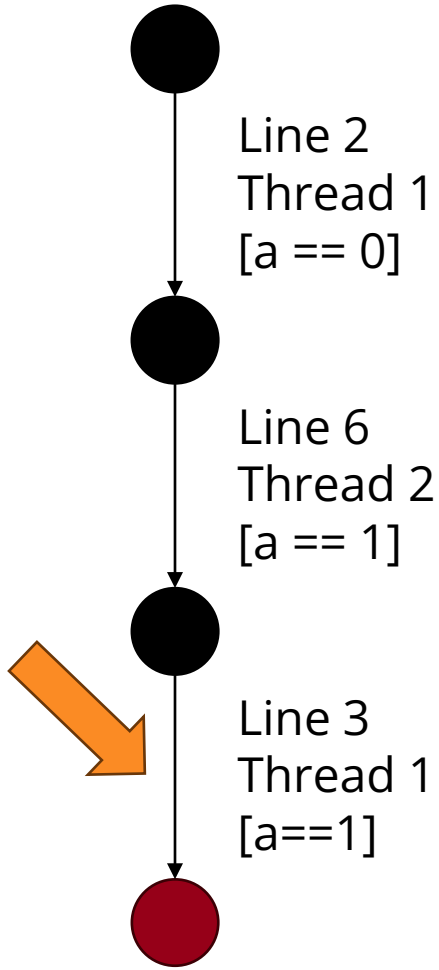
Handling concurrency



yield



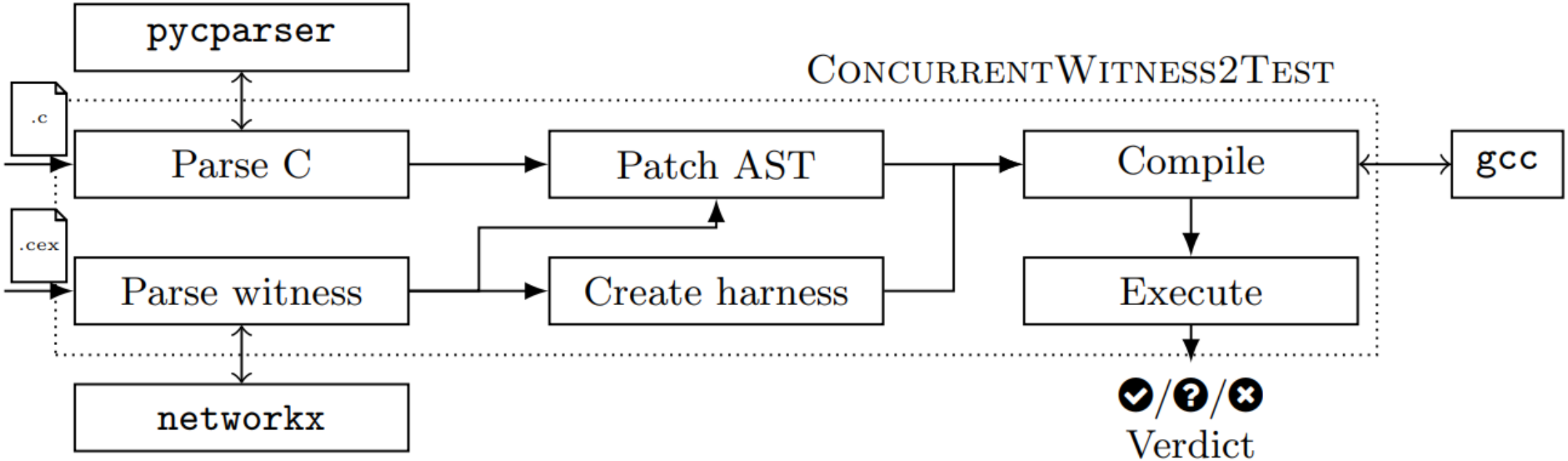
release




```
0  int a;

1  void P1(void) {
     yield(0);
2    a = 0;
     release(0);
     yield(2);
3    assert(a == 0);
     release(2);
4  }

5  void P2(void) {
     yield(1);
6    a = 1;
     release(1);
7  }
```

Software architecture

ftsrg/**ConcurrentWitness2Test**



Results

ftsrg/**ConcurrentWitness2Test**

	dartagnan	divine	theta	uautomizer	ugemcutter	utaipan
Confirmed	178	179 (2)	191	186	235	228
Refused	79	25 (1)	8	74	22	29
Error	193	111	96	168	194	170

Competition version

Results

ftsrg/**ConcurrentWitness2Test**

	dartagnan	divine	theta	uautomizer	ugemcutter	utaipan
Confirmed	178	179 (2)	191	186	235	228
Refused	79	25 (1)	8	74	22	29
Error	193	111	96	168	194	170

Competition version

10.5281/zenodo.10184336

Results

ftsrg/**ConcurrentWitness2Test**

	dartagnan	divine	theta	uautomizer	ugemcutter	utaipan
Confirmed	178	179 (2)	191	186	235	228
Refused	79	25 (1)	8	74	22	29
Error	193	111	96	168	194	170

Competition version

Up-to-date version

10.5281/zenodo.10184336

Results

ftsrg/**ConcurrentWitness2Test**

	dartagnan	divine	theta	uautomizer	ugemcutter	utaipan
Confirmed	178	179 (2)	191	186	235	228
Refused	79	25 (1)	8	74	22	29
Error	193	111	96	168	194	170

Competition version

Up-to-date version

2 unsupported tools
23 supported tools

10.5281/zenodo.10184336

Results

ftsrg/**ConcurrentWitness2Test**

	dartagnan	divine	theta	uautomizer	ugemcutter	utaipan
Confirmed	178	179 (2)	191	186	235	228
Refused	79	25 (1)	8	74	22	29
Error	193	111	96	168	194	170

Competition version

Most tools give underspecified witnesses:
not all paths lead to an error!

Up-to-date version

2 unsupported tools
23 supported tools

doi 10.5281/zenodo.10184336