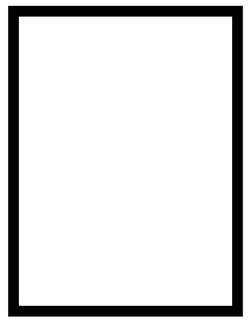
Finding and Understanding Bugs in Software Model Checkers

Chengyu Zhang, Ting Su, Yichen Yan, Fuyuan Zhang,
Geguang Pu, Zhendong Su

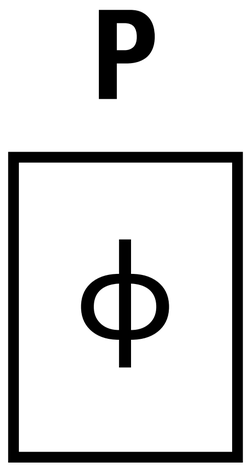
Software Model Checking

Software Model Checking

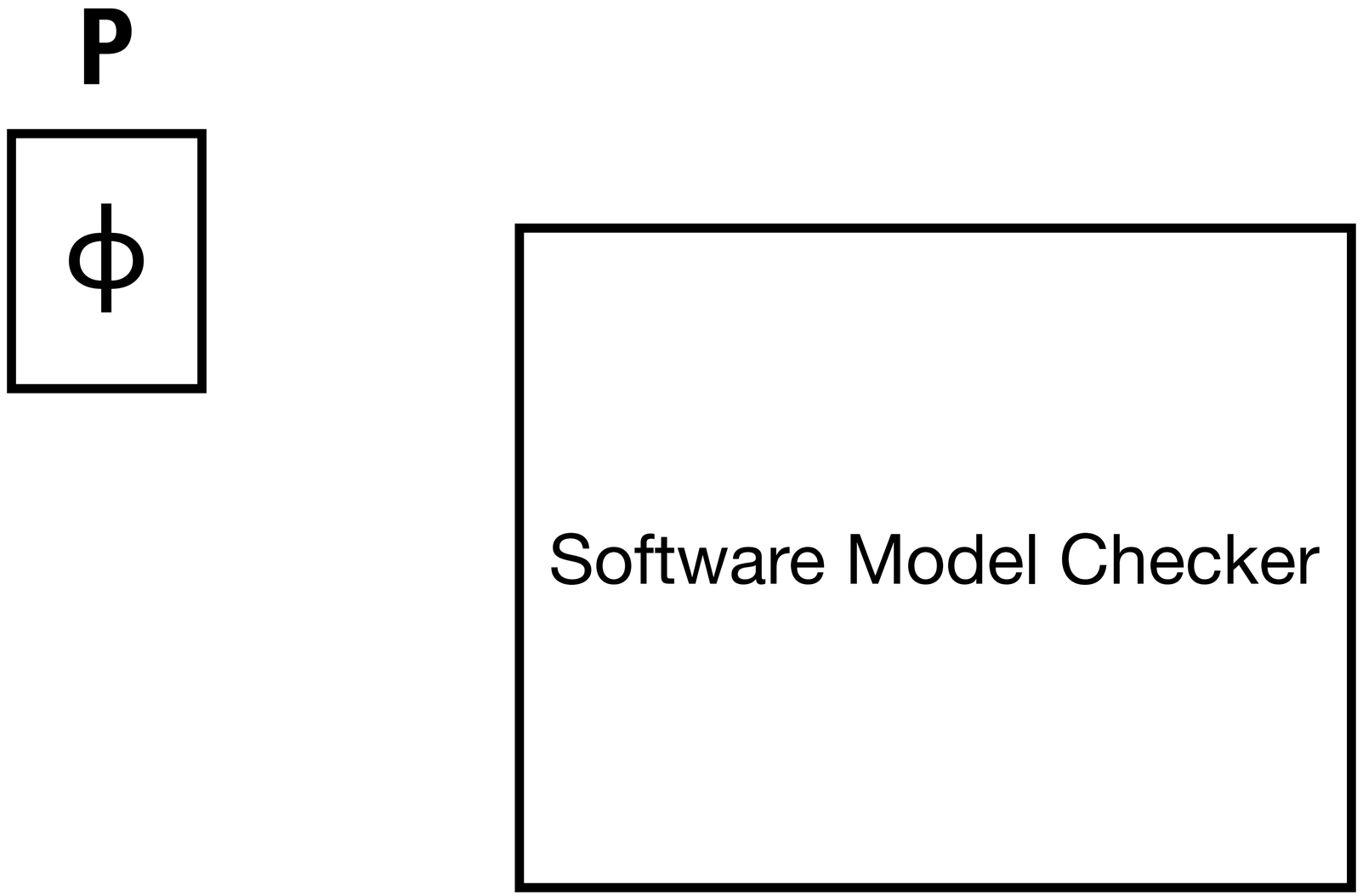
P



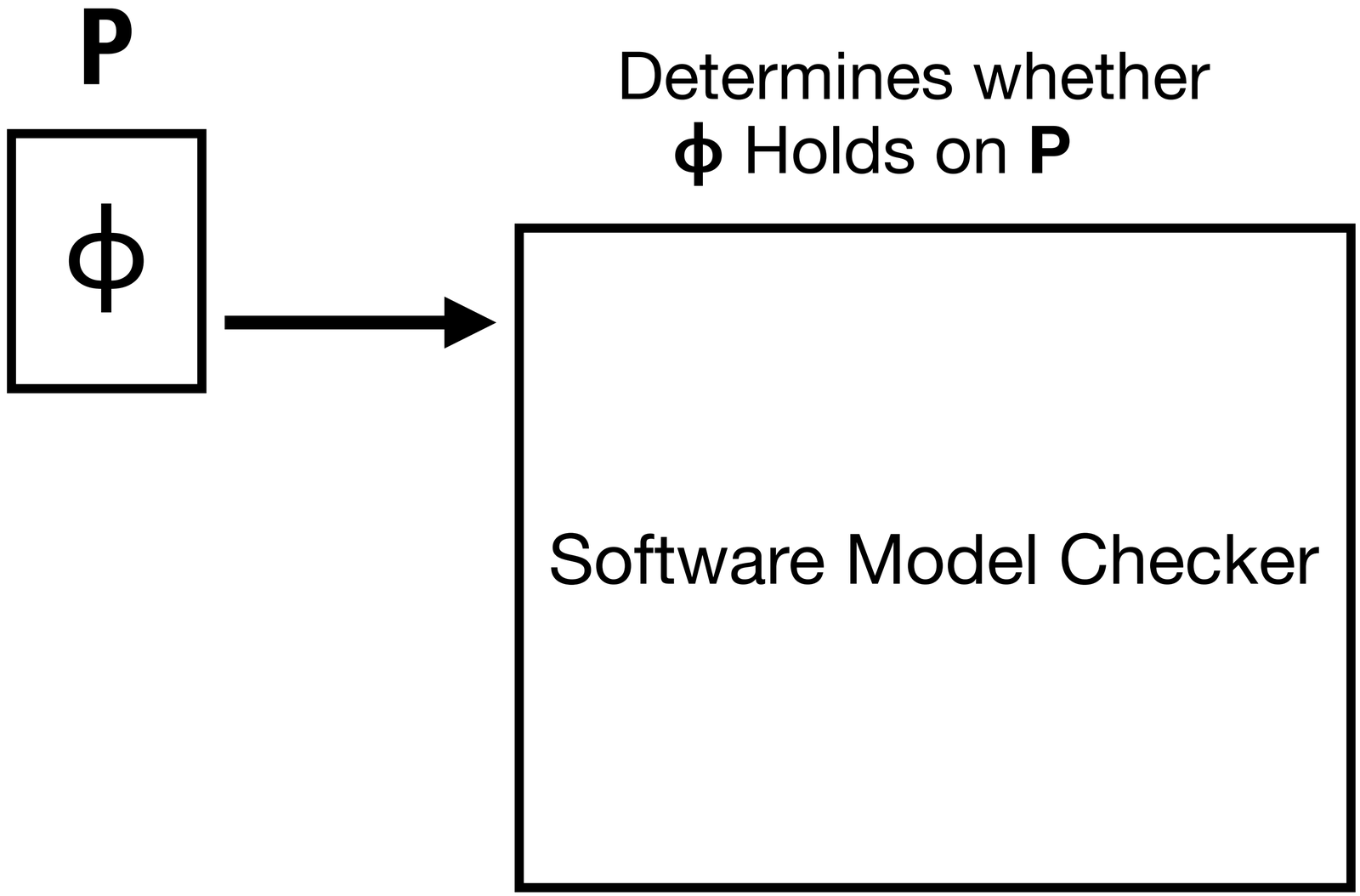
Software Model Checking



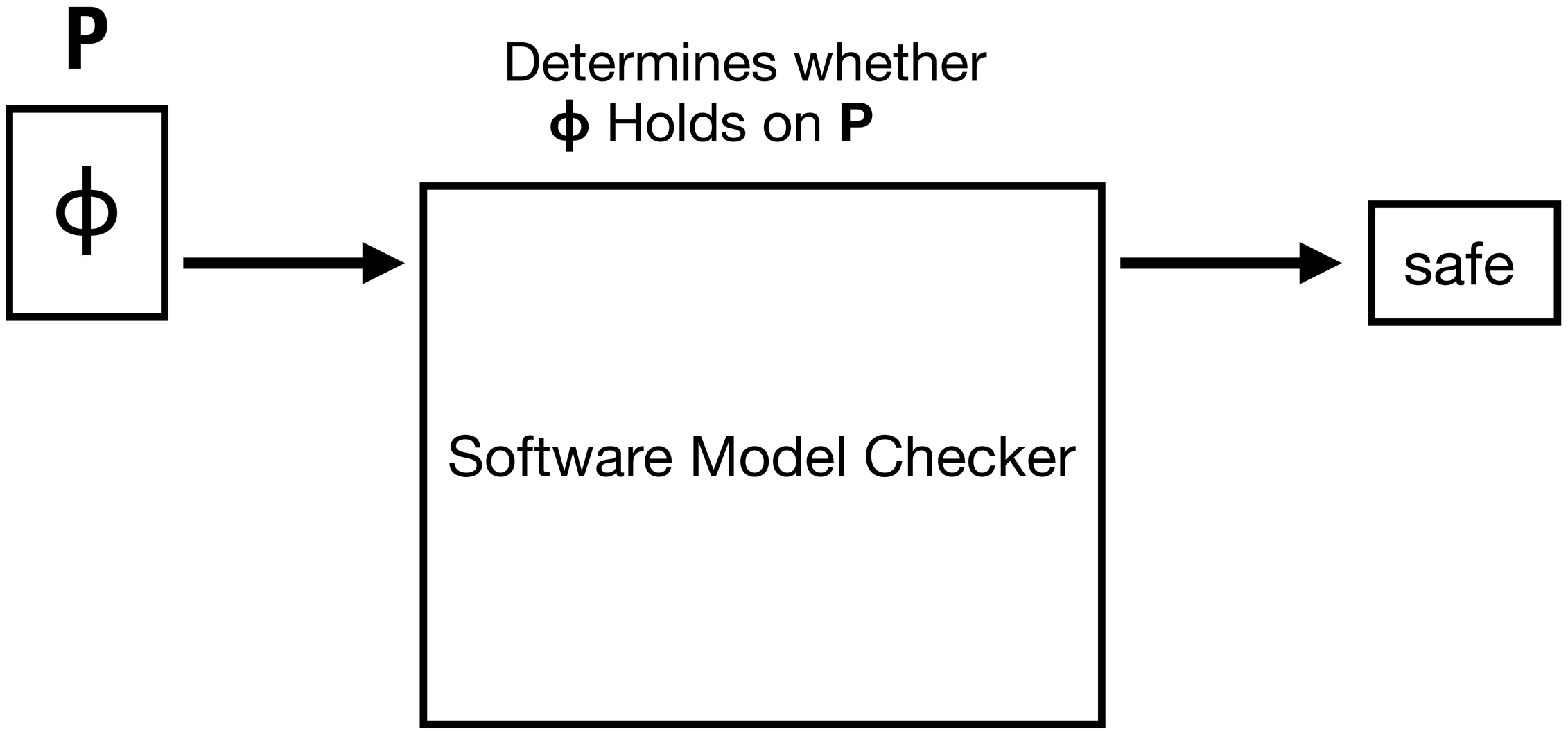
Software Model Checking



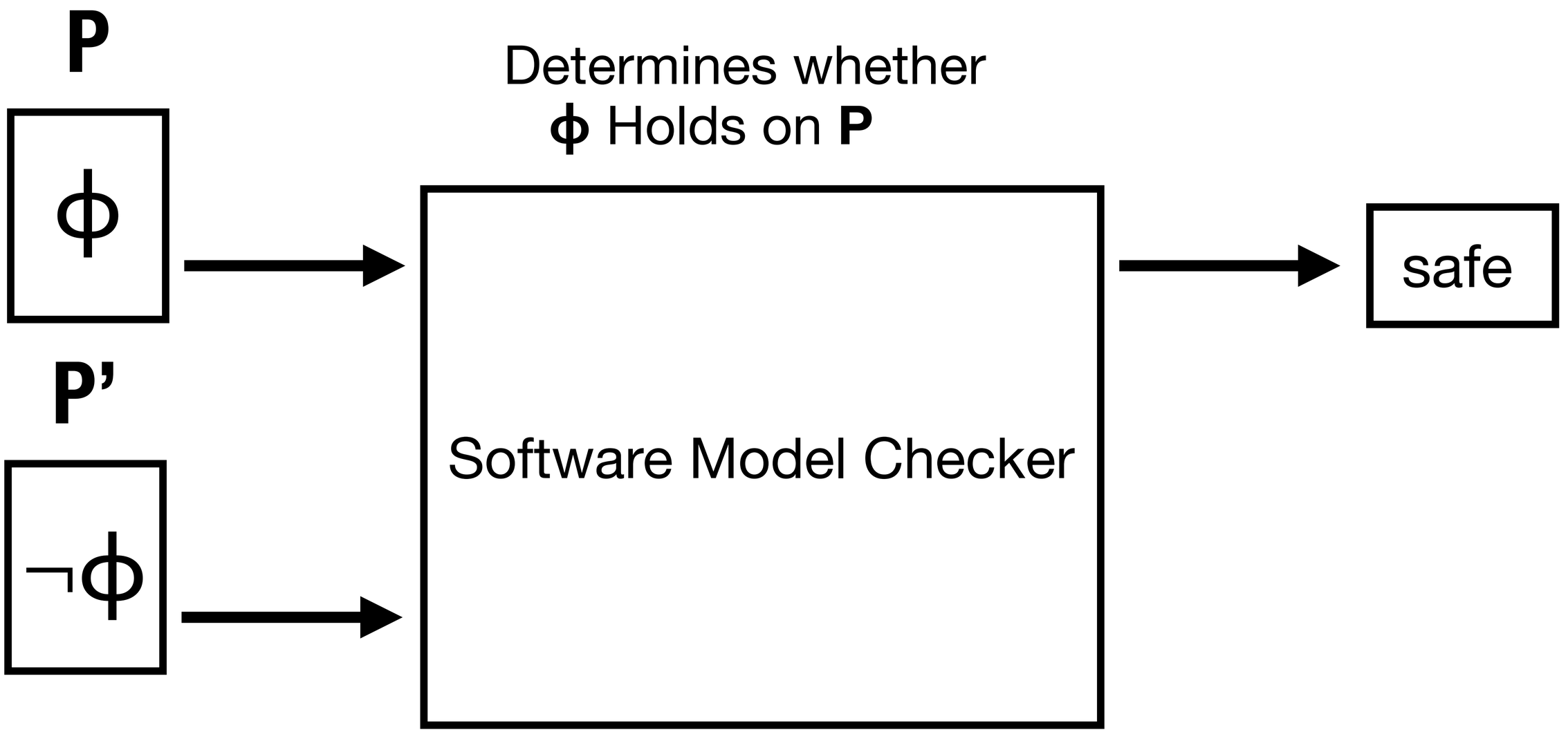
Software Model Checking



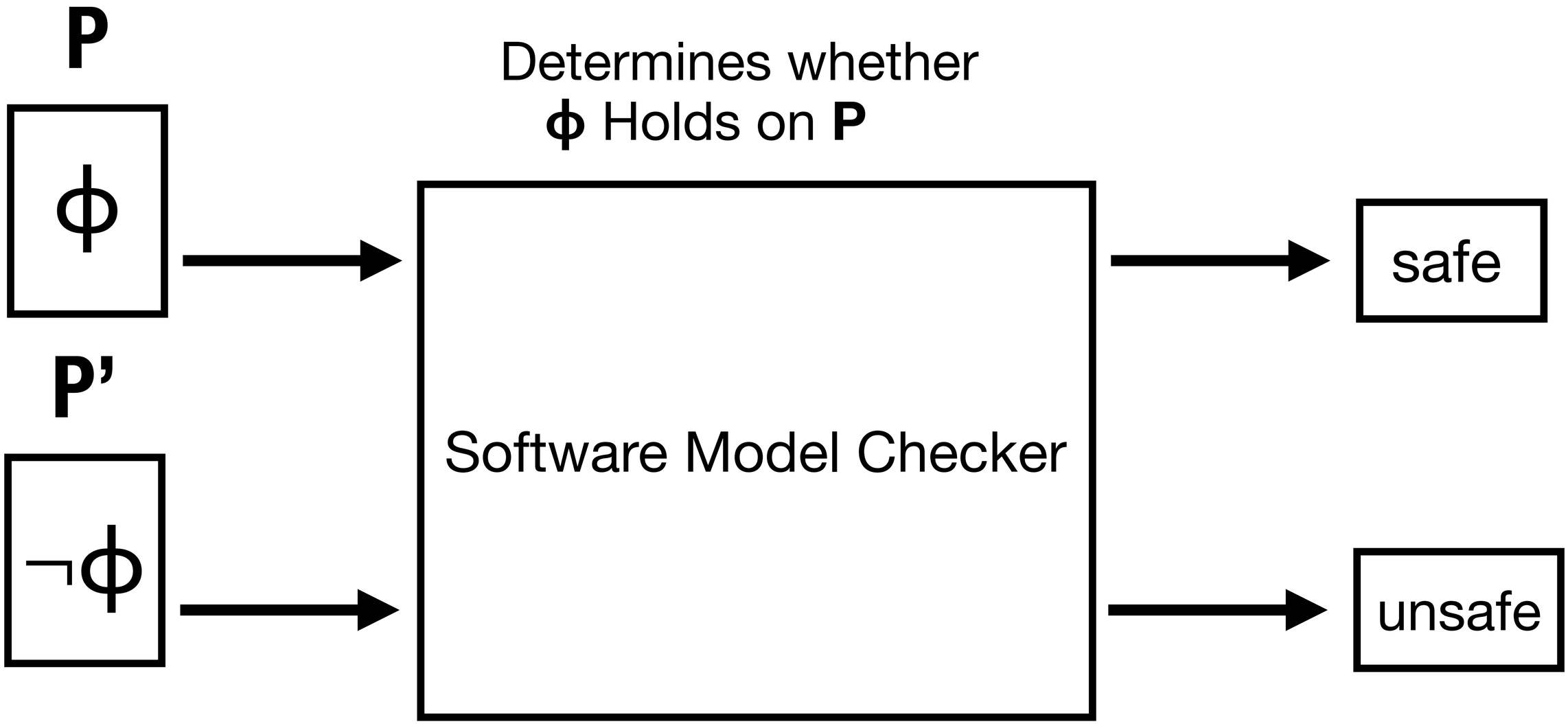
Software Model Checking



Software Model Checking



Software Model Checking

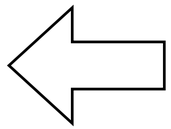


Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i > 0) {
       ……  //unreachable
      __VERIFIER_error();
  }
  return 0;
}
```

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i > 0) {
      ……  //unreachable
     __VERIFIER_error();
  }
  return 0;
}
```



Determines whether this function is reachable

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i > 0) {
       ……  //unreachable
      __VERIFIER_error();
  }
  return 0;
}
```

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i > 0) {
      ……  //unreachable
     __VERIFIER_error();
  }
  return 0;
}
```

SAFE!

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i == 0) {
       ……  //reachable
      __VERIFIER_error();
  }
  return 0;
}
```

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i == 0) {
       ……  //reachable
      __VERIFIER_error();
  }
  return 0;
}
```

UNSAFE!

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i == 0) {
       ……  //reachable
      __VERIFIER_error();
  }
  return 0;
}
```

software model checker:

SAFE!

Example: Reachability Safety Property

```
int main() {
  int i = 0;
  if (i == 0) {
       ……  //reachable
      __VERIFIER_error();
  }
  return 0;
}
```

software model checker:

SAFE!

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:0

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:0

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:1

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:1

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
      __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:1

UNSAFE

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
      __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

i:1

UNSAFE

Bug#529 in CPAchecker

```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
      __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

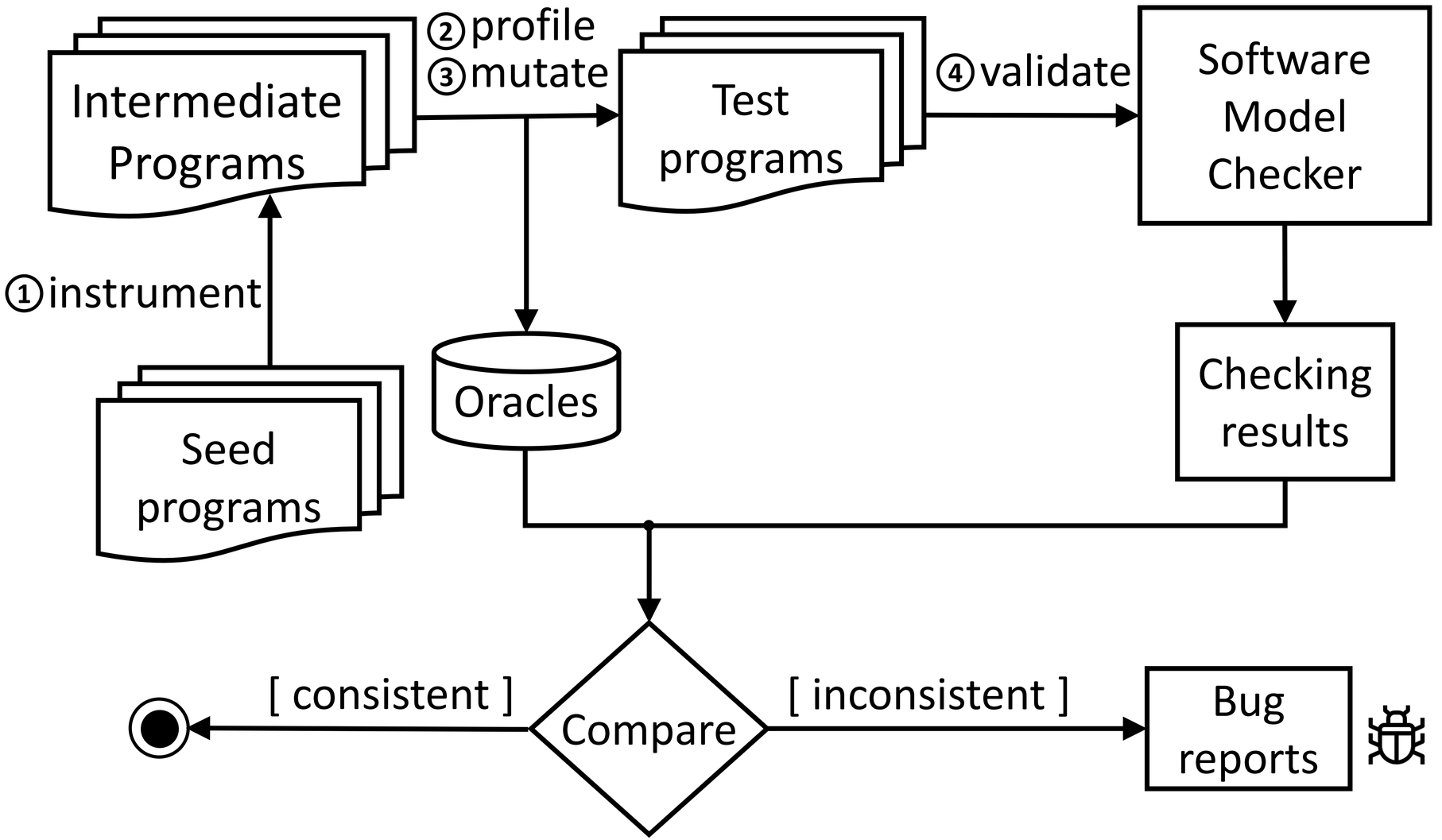
```
i:0
*(&i):1
```

Bug#529 in CPAchecker

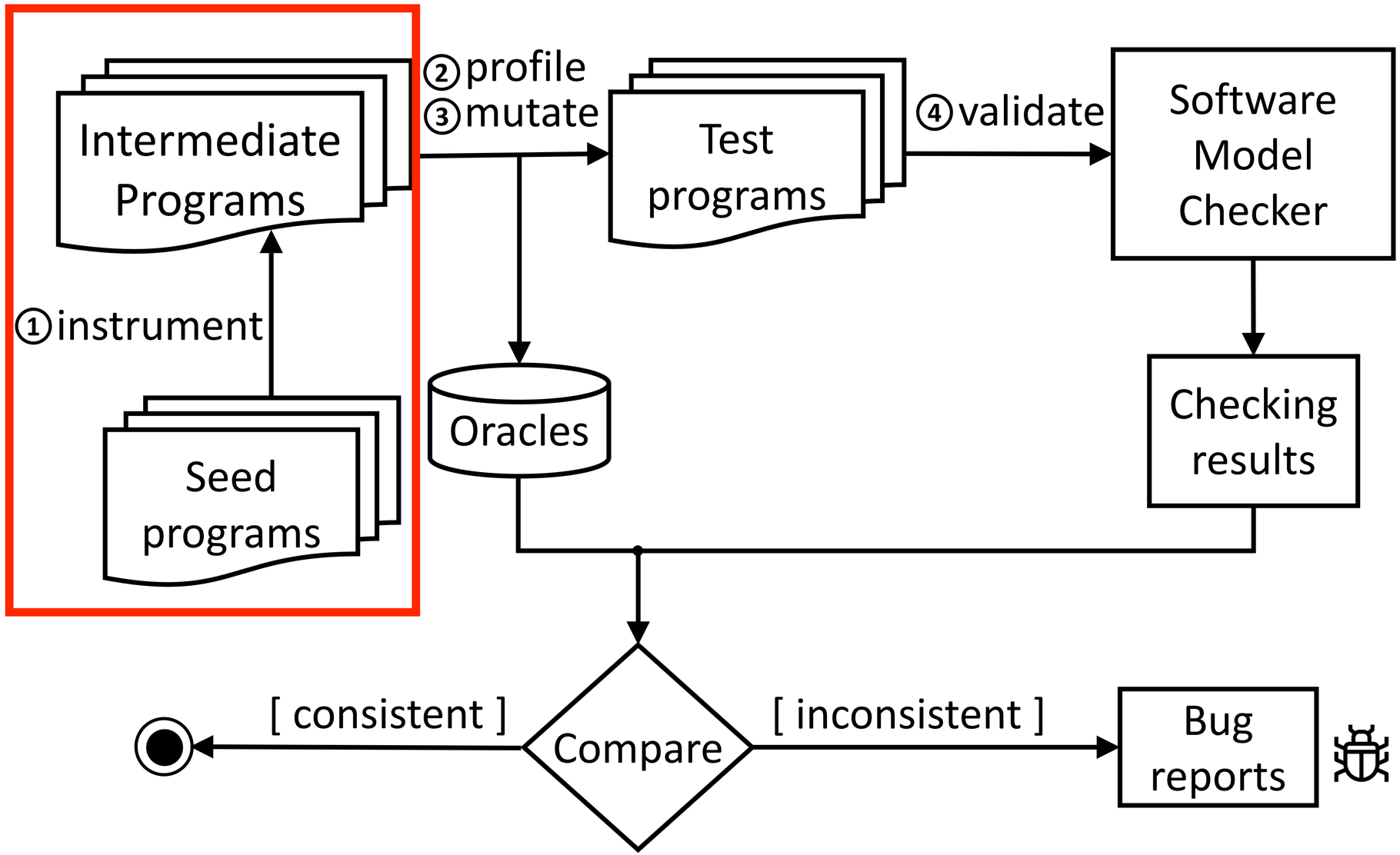
```
void main() {
  int i = 0;
  while (1) {
    if (i > 0) {
       __VERIFIER_error();
      break;
    }
    if (i == 0){
      *(&i) = *(&i) + 1;
    }
  }
}
```

```
i:0
*(&i):1
```

Workflow


Intermediate Programs
②profile
③mutate
Test programs
④validate
Software Model Checker
①instrument
Oracles
Checking results
Seed programs
[consistent]
Compare
[inconsistent]
Bug reports

Workflow


Intermediate Programs
②profile
③mutate
Test programs
④validate
Software Model Checker
①instrument
Seed programs
Oracles
Checking results
[consistent]
Compare
[inconsistent]
Bug reports

Approach I: Enumerative Reachability (ER)

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Approach I: Enumerative Reachability (ER)

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      __VERIFIER_error();
      if (i > 0) {
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
     }
   }
}
```

Intermediate Program 1

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
        __VERIFIER_error();
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
     }
   }
}
```

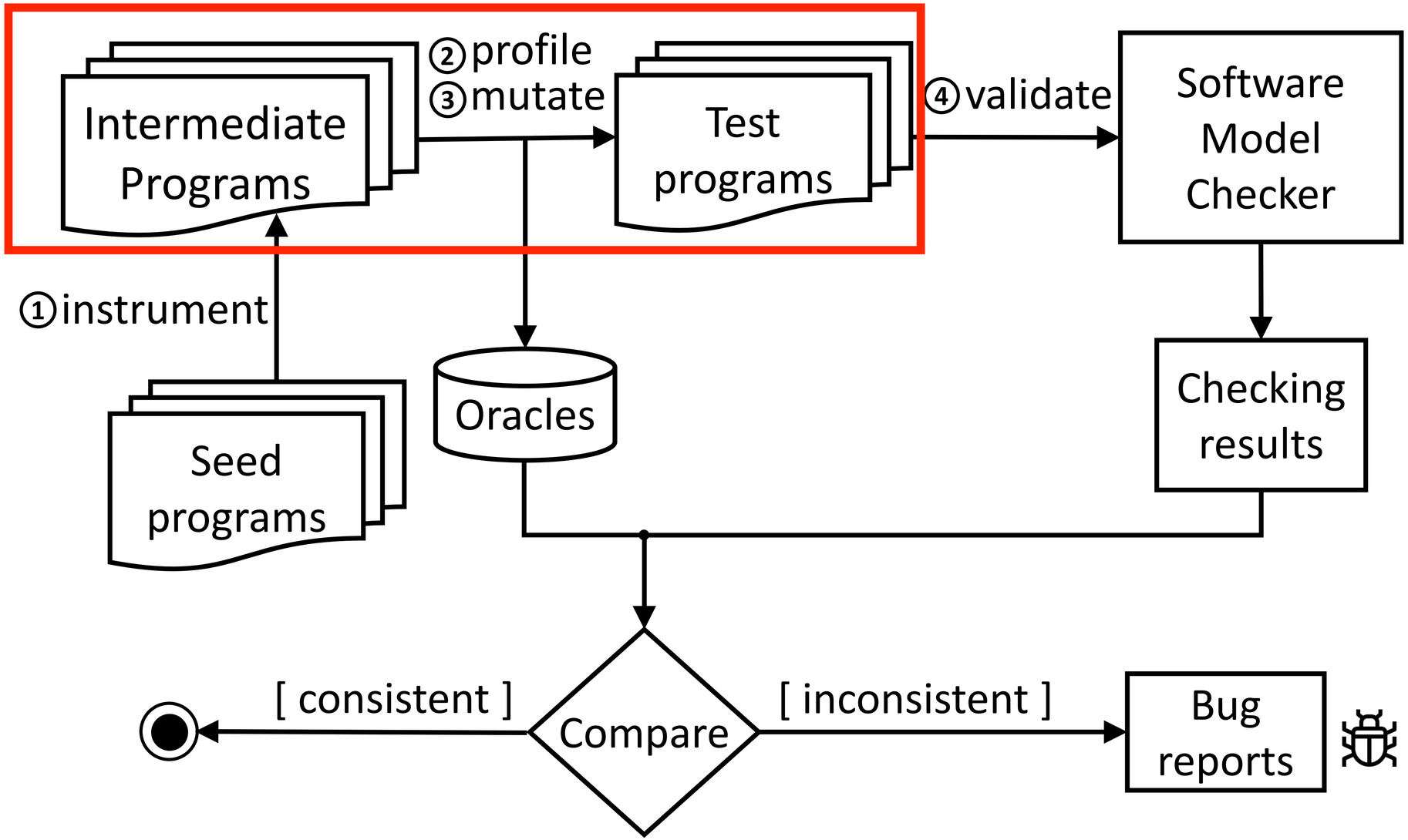
Intermediate Program 2

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
        __VERIFIER_error();
        *(&i) = *(&i) + 1;
     }
   }
}
```

Intermediate Program 3

Workflow


Intermediate Programs
②profile
③mutate
Test programs
④validate
Software Model Checker
①instrument
Seed programs
Oracles
Checking results
[consistent]
Compare
[inconsistent]
Bug reports

Approach I: Enumerative Reachability (ER)

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      __VERIFIER_error();
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Intermediate Program 1

Actual execution: unsafe

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         __VERIFIER_error();
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Intermediate Program 2

Actual execution: unsafe

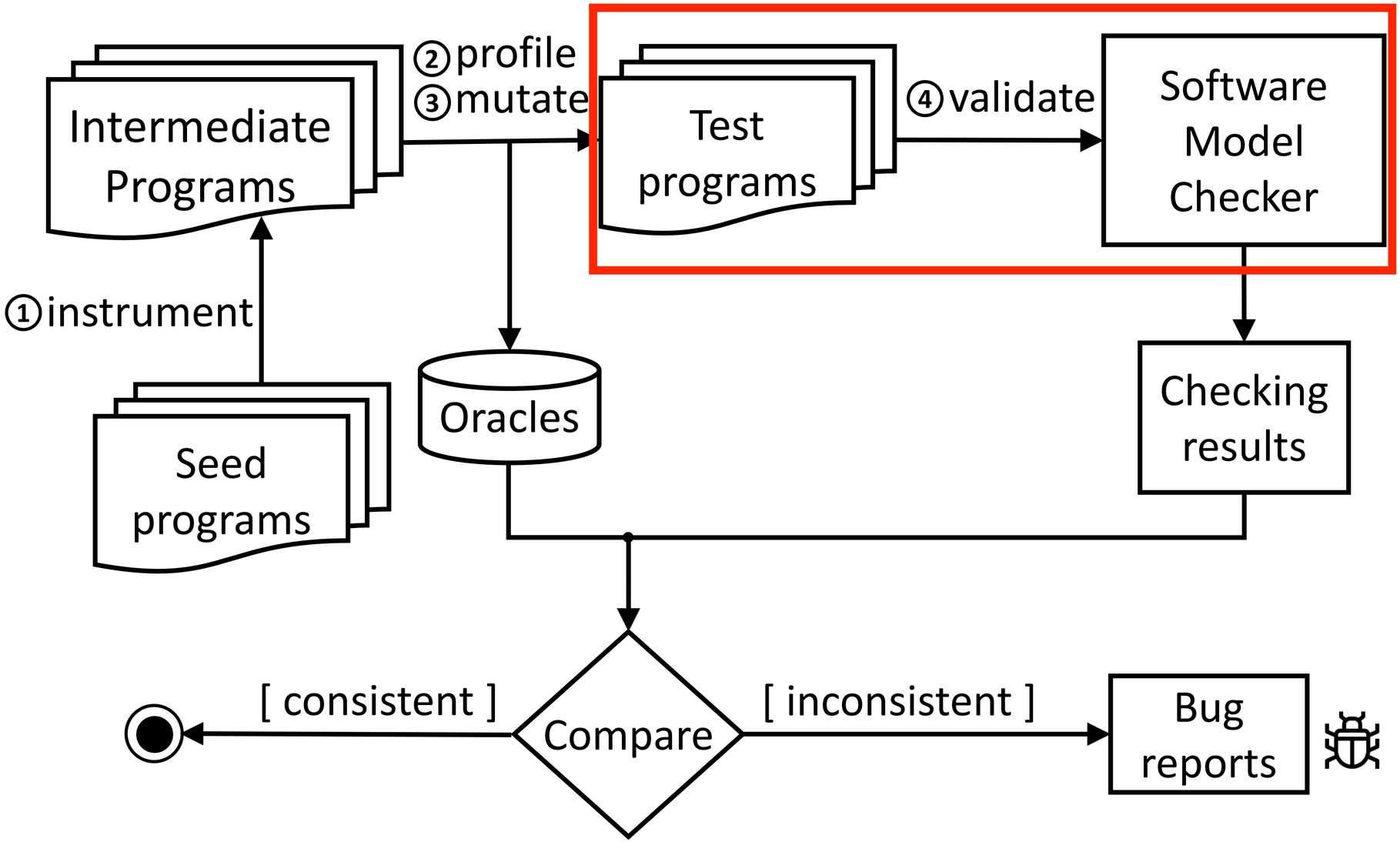
Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
         __VERIFIER_error();
    }
   }
}
```

Intermediate Program 3

Actual execution: unsafe

Workflow


②profile
③mutate
Intermediate Programs
Test programs
④validate
Software Model Checker
①instrument
Seed programs
Oracles
Checking results
[consistent]
Compare
[inconsistent]
Bug reports

Approach I: Enumerative Reachability (ER)

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      __VERIFIER_error();
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Test program 1

Actual execution: unsafe

Model checker: unsafe

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         __VERIFIER_error();
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Test program 2

Actual execution: unsafe

Model checker: safe

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
         __VERIFIER_error();
    }
   }
}
```

Test program 3

Actual execution: unsafe

Model checker: unsafe

Approach I: Enumerative Reachability (ER)

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      __VERIFIER_error();
         break;
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Test program 1

Actual execution: unsafe

Model checker: unsafe

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         __VERIFIER_error();
      }
      if (i == 0){
        *(&i) = *(&i) + 1;
    }
   }
}
```

Test program 2

Actual execution: unsafe

Model checker: safe

Bug#529 in CPAchecker

```
void main() {
   int i = 0;
   while (1) {
      if (i > 0) {
         break;
      }
      if (i == 0){
         __VERIFIER_error();
    }
   }
}
```

Test program 3

Actual execution: unsafe

Model checker: unsafe

Approach

- Approach I: Enumerative Reachability (ER)

Bug#534 in CPAchecker

```
int main(void){
  int a[3] = {1};
  int i = 0;
  while(i < 3){
    if(a[i] == 1) {
      ……
    }
    i++;
  }
  return 0;
}
```

a[3]:{1,0,0}

Bug#534 in CPAchecker

```
int main(void){
  int a[3] = {1};
  int i = 0;
  while(i < 3){
    if(a[i] == 1) {
      ……
    }
    i++;
  }
  return 0;
}
```

a[3]:{1,1,1}

Bug#534 in CPAchecker

```
int main(void){
  int a[3] = {1};
  int i = 0;
  while(i < 3){
    if(a[i] == 1) {
      ......
    }
    i++;
  }
  return 0;
}
```

a[3]:{1,1,1}: unsafe 

a[3]:{1,0,0}: unsafe

Bug#534 in CPAchecker

```
int main(void){
  int a[3] = {1};
  int i = 0;
  while(i < 3){
    if(a[i] == 1) {
      __VERIFIER_error();
    }
    i++;
  }
  return 0;
}
```

a[3]:{1,1,1}: unsafe 

a[3]:{1,0,0}: unsafe

Approach II: Enumerative **Counting** Reachability (ECR)

```c
int main(void){
   int a[3] = {1};
   int i = 0;

   while(i < 3){
     if(a[i] == 1) {

       ……
     }
     i++;
   }


   return 0;
}
```

```c
int main(void){
   int a[3] = {1};
   int i = 0;

   while(i < 3){

     if(a[i] == 1) {
       ……
     }
     i++;
   }


   return 0;
}
```

Approach II: Enumerative **Counting** Reachability (ECR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      if(a[i] == 1) {
         br++;
         ……
      }
      i++;
   }
   GetValue(br)

   return 0;
}
```

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      br++;
      if(a[i] == 1) {
         ……
      }
      i++;
   }
   GetValue(br)

   return 0;
}
```

Approach II: Enumerative **Counting** Reachability (ECR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      if(a[i] == 1) {
         br++;
         ……
      }
      i++;
   }
   GetValue(br)

   return 0;
}
```

Actual execution: br=1

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      br++;
      if(a[i] == 1) {
         ……
      }
      i++;
   }
   GetValue(br)

   return 0;
}
```

Actual execution: br=3

Approach II: Enumerative **Counting** Reachability (ECR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      if(a[i] == 1) {
         br++;
         ……
      }
      i++;
   }
   if(br != 1)
      __VERIFIER_error();
   return 0;
}
```

Actual execution: br=1

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      br++;
      if(a[i] == 1) {
         ……
      }
      i++;
   }
   if(br != 3)
      __VERIFIER_error();
   return 0;
}
```

Actual execution: br=3

Approach II: Enumerative **Counting** Reachability (ECR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      if(a[i] == 1) {
         br++;
         ……
      }
      i++;
   }
   if(br != 1)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      br++;
      if(a[i] == 1) {
         ……
      }
      i++;
   }
   if(br != 3)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe

Approach II: Enumerative **Counting** Reachability (ECR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      if(a[i] == 1) {
         br++;
         ……
      }
      i++;
   }
   if(br != 1)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe
Model checker: unsafe

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;
   while(i < 3){
      br++;
      if(a[i] == 1) {
         ……
      }
      i++;
   }
   if(br != 3)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe
Model checker: safe

Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)

Approach III: **Fused** Counting Reachability (FCR)

```
int main(void){
   int a[3] = {1};
   int i = 0;

   while(i < 3){

     if(a[i] == 1) {

       ……
     }
     i++;
   }


   return 0;
}
```

Approach III: **Fused** Counting Reachability (FCR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;int br2 = 0;
   while(i < 3){
      br1++;
      if(a[i] == 1) {
         br2++;
         ……
      }
      i++;
   }
   GetValue(br1)
   GetValue(br2)
   return 0;
}
```

Actual execution: br1=1; br2=3

Approach III: **Fused** Counting Reachability (FCR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;int br2 = 0;
   while(i < 3){
      br1++;
      if(a[i] == 1) {
         br2++;
         ……
      }
      i++;
   }
   if(br1 != 3 || br2 != 1)
      __VERIFIER_error();
   return 0;
}
```

Approach III: **Fused** Counting Reachability (FCR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;int br2 = 0;
   while(i < 3){
      br1++;
      if(a[i] == 1) {
         br2++;
         ……
      }
      i++;
   }
   if(br1 != 3 || br2 != 1)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe

Approach III: **Fused** Counting Reachability (FCR)

```
int main(void){
   int a[3] = {1};
   int i = 0;
   int br = 0;int br2 = 0;
   while(i < 3){
      br1++;
      if(a[i] == 1) {
         br2++;
         ……
      }
      i++;
   }
   if(br1 != 3 || br2 != 1)
      __VERIFIER_error();
   return 0;
}
```

Test oracle: safe

Model checker: unsafe

Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

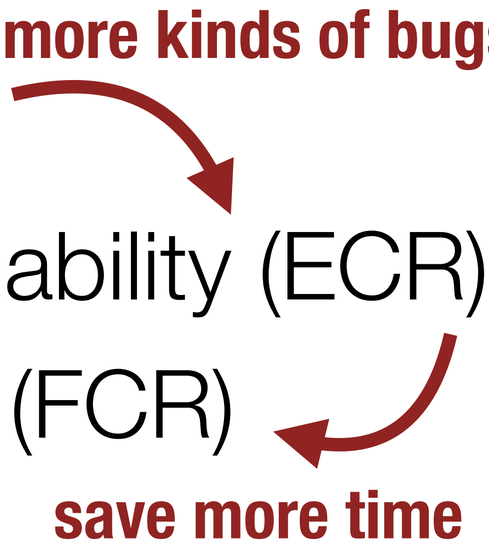
Approach

find more kinds of bugs

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)



Evaluation Setup

Evaluation Setup

GCC test suite

Evaluation Setup

GCC test suite

4,609 Files
219,636 Loc

Evaluation Setup

GCC test suite

4,609 Files
219,636 Loc

SeaHorn

IC3 based

Evaluation Setup

GCC test suite

4,609 Files

219,636 Loc

SeaHorn

IC3 based

CPA✓

CEGAR based

Evaluation Setup

GCC test suite

4,609 Files

219,636 Loc

SeaHorn

IC3 based

CPAchecker

CEGAR based

CBMC

BMC based

Evaluation Setup

GCC test suite

4,609 Files
219,636 Loc

IC3 based

CEGAR based

BMC based

RQ1: Can our approaches find bugs?

RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

RQ1: Can our approaches find bugs?

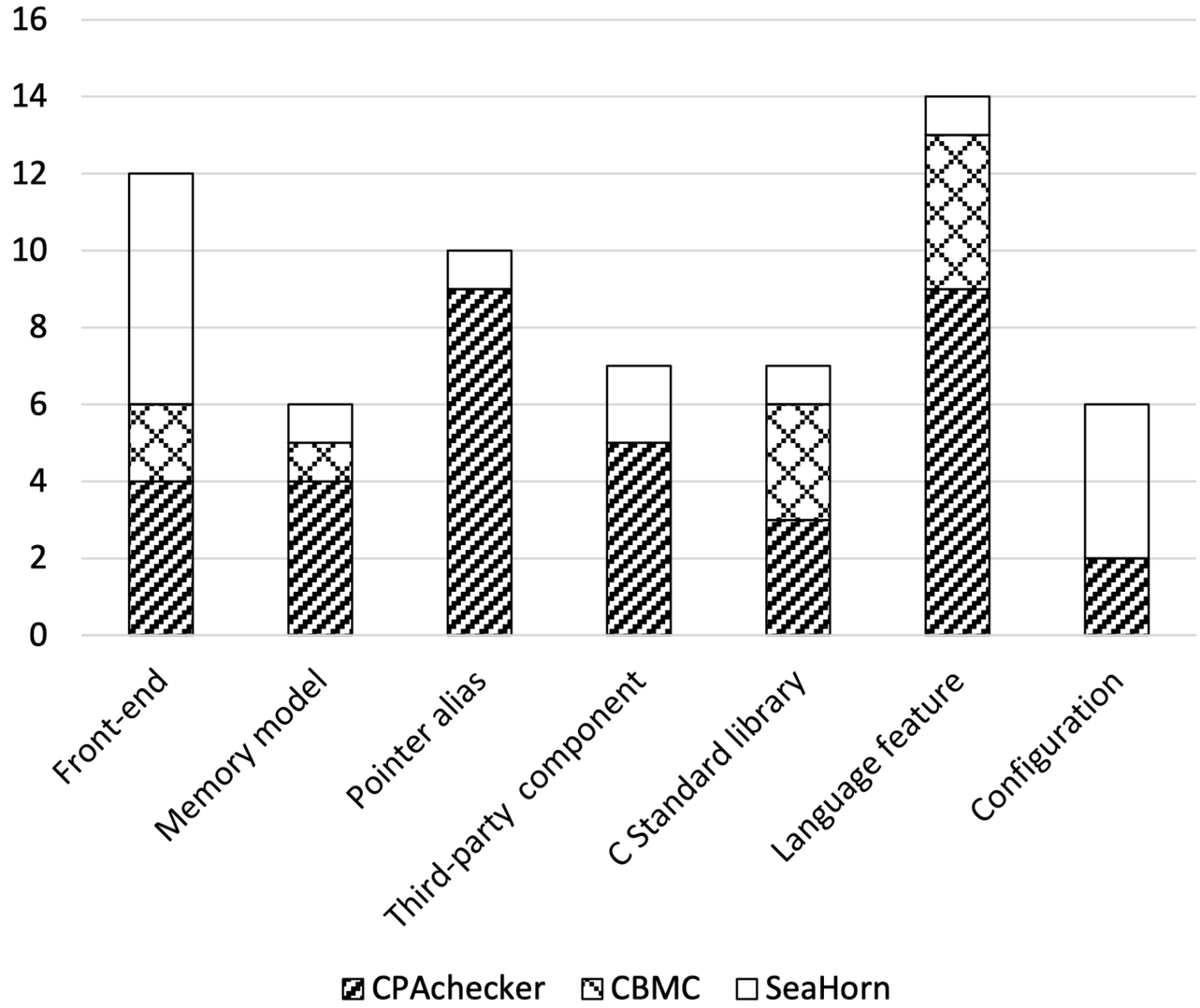
	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

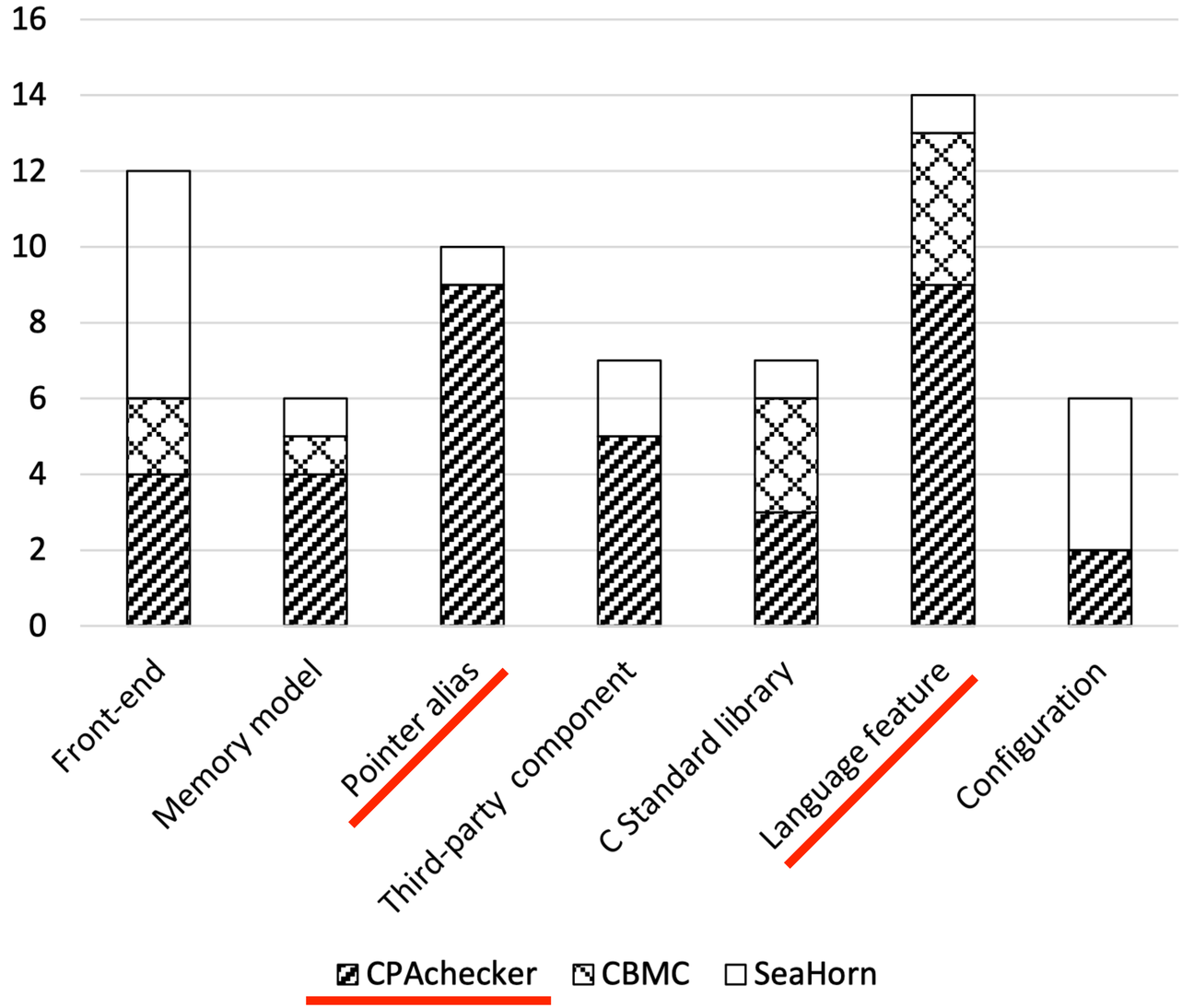
RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62



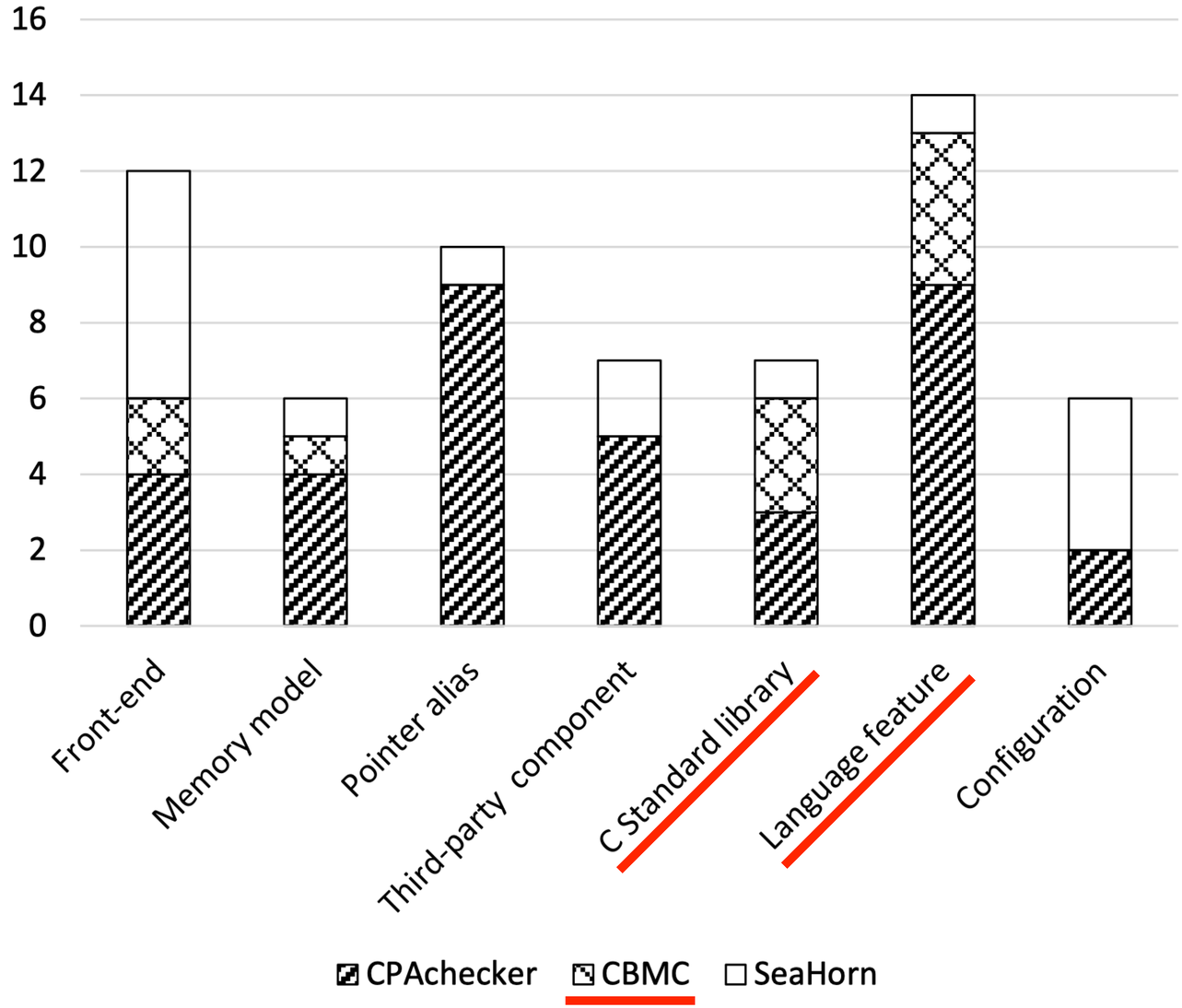
RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62



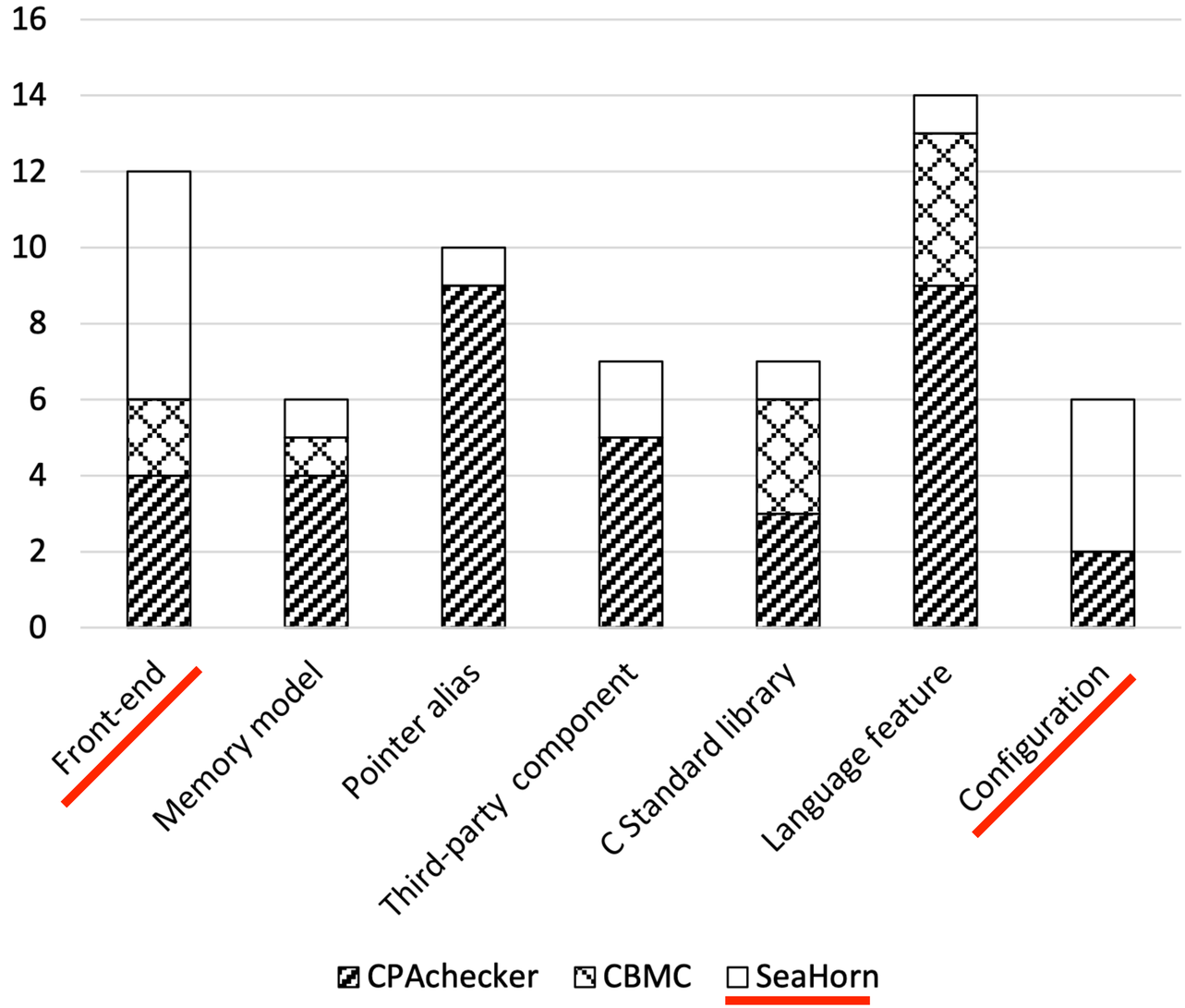
RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62



RQ1: Can our approaches find bugs?

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62



RQ2: How many bugs can be found by each approach?

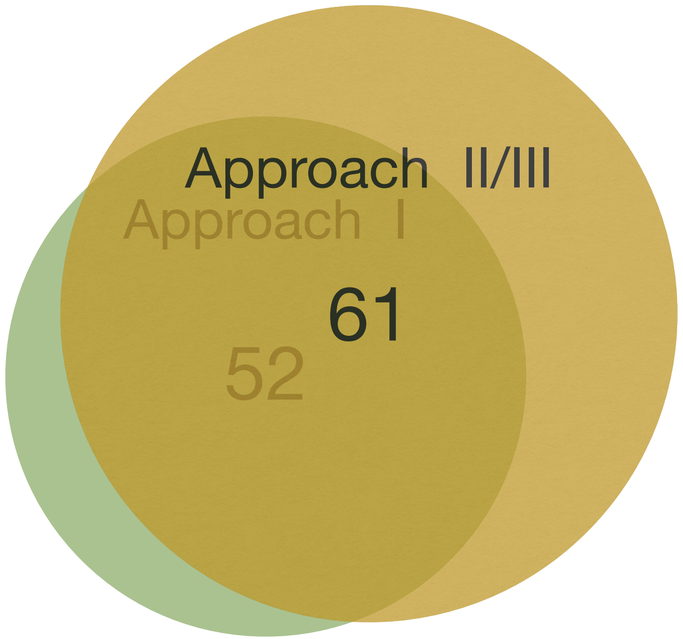
RQ2: How many bugs can be found by each approach?

RQ2: How many bugs can be found by each approach?

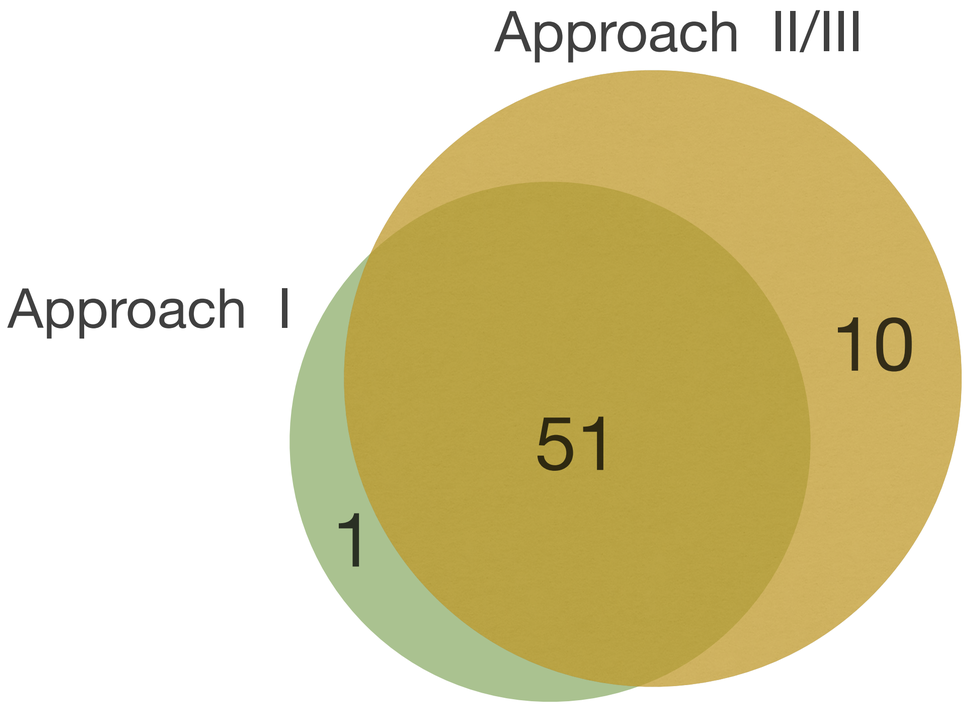
RQ2: How many bugs can be found by each approach?

RQ2: How many bugs can be found by each approach?

RQ2: How many bugs can be found by each approach?



RQ2: How many bugs can be found by each approach?



Bug#529 in CPAchecker (False negative in approach II/III)

```
void main() {
  int i = 0;

  while (1) {
     if (i > 0) {

        break;
     }
     if (i == 0){
       *(&i) = *(&i) + 1;
    }
  }


}
```

```
i:0
*(&i):1
```

Bug#529 in CPAchecker (False negative in approach II/III)

```
void main() {
  int i = 0;
  int br = 0;
  while (1) {
     if (i > 0) {
        br++;
        break;
     }
     if (i == 0){
       *(&i) = *(&i) + 1;
    }
  }
  if(br != 1)
     __VERIFIER_error();
}
```

```
i:0
*(&i):1
```

Test oracle: safe

Bug#529 in CPAchecker (False negative in approach II/III)

```
void main() {
  int i = 0;
  int br = 0;
  while (1) {
     if (i > 0) {
        br++;
        break;
     }
     if (i == 0){
       *(&i) = *(&i) + 1;
    }
  }
  if(br != 1)
     __VERIFIER_error();
}
```

```
i:0
*(&i):1
```

Test oracle: safe

Buggy model checker: safe

RQ3: How much time does each approach consume?

RQ3: How much time does each approach consume?

Approach instance	#Mutants	#Incorrect cases			Total time (h)		
		CPAchecker	CBMC	SeaHorn	CPAchecker	CBMC	SeaHorn
Approach I: ER	21,948	2,091	1,567	3,665	168.41	151.23	37.17
Approach II: ECR	21,948	5,972	4,690	6,752	258.03	132.08	95.35
Approach III: FCR	4,609	699	356	727	32.95	16.89	10.50

RQ3: How much time does each approach consume?

Approach instance	#Mutants	#Incorrect cases			Total time (h)		
		CPAchecker	CBMC	SeaHorn	CPAchecker	CBMC	SeaHorn
Approach I: ER	21,948	2,091	1,567	3,665	168.41	151.23	37.17
Approach II: ECR	21,948	5,972	4,690	6,752	258.03	132.08	95.35
Approach III: FCR	4,609	699	356	727	32.95	16.89	10.50

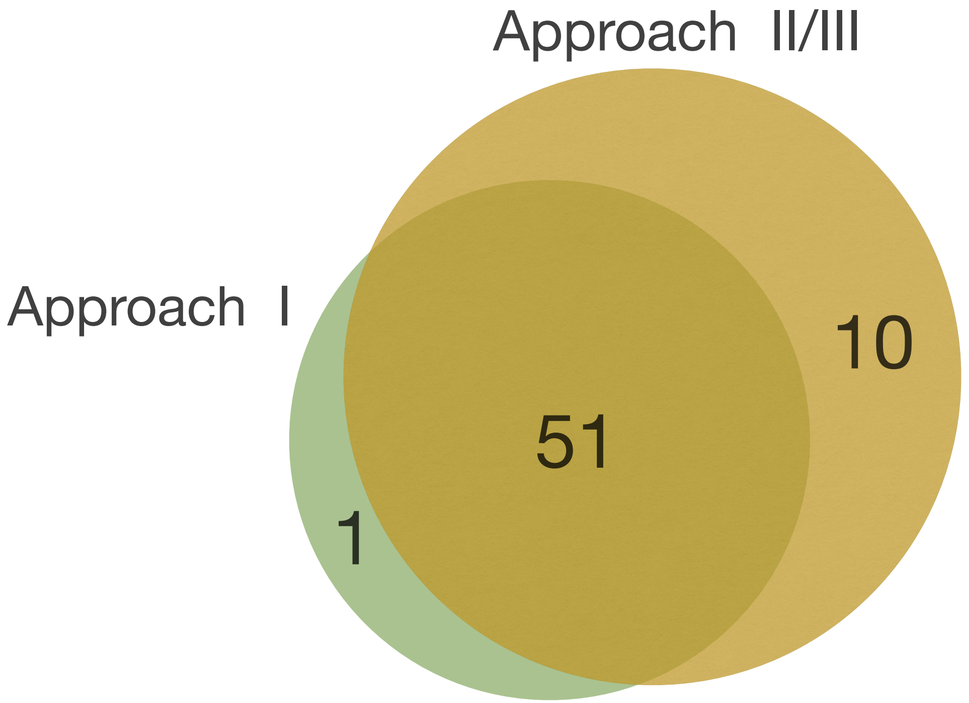
RQ3: How much time does each approach consume?

Save 89% of time

Approach instance	#Mutants	#Incorrect cases			Total time (h)		
		CPAchecker	CBMC	SeaHorn	CPAchecker	CBMC	SeaHorn
Approach I: ER	21,948	2,091	1,567	3,665	168.41	151.23	37.17
Approach II: ECR	21,948	5,972	4,690	6,752	258.03	132.08	95.35
Approach III: FCR	4,609	699	356	727	32.95	16.89	10.50

RQ3: How much time does each approach consume?

Approach instance	#Mutants	#Incorrect cases			Total time (h)		
		CPAchecker	CBMC	SeaHorn	CPAchecker	CBMC	SeaHorn
Approach I: ER	21,948	2,091	1,567	3,665	168.41	151.23	37.17
Approach II: ECR	21,948	5,972	4,690	6,752	258.03	132.08	95.35
Approach III: FCR	4,609	699	356	727	32.95	16.89	10.50



Assorted Bug Samples

- Front-end
- Memory model
- Pointer alias
- Third-party component
- C standard library
- Language feature
- Configuration

https://github.com/MCFuzzer/MCFuzz/issues

Example: Front-end related bug in CPAchecker

```
void f(int a, int b){
  if (a == b)
    __VERIFIER_error();
}
int main(){
  int d = 0;
  int c = 4;
  int e = 2;
  f (d=c&&e, 1);
  return 0;
}
```

Example: Front-end related bug in CPAchecker

```
void f(int a, int b){
  if (a == b)
    __VERIFIER_error();
}
int main(){
  int d = 0;
  int c = 4;
  int e = 2;
  f (d=c&&e, 1);
  return 0;
}
```

d = c&&e = 1

Test oracle: unsafe

Example: Front-end related bug in CPAchecker

```
void f(int a, int b){
  if (a == b)
    __VERIFIER_error();
}
int main(){
  int d = 0;
  int c = 4;
  int e = 2;
  f (d=c&&e, 1);
  return 0;
}
```

```
__CPAchecker_TMP_0 = c&&e
d = __CPAchecker_TMP_0
```

Test oracle: unsafe

Example: Front-end related bug in CPAchecker

```
void f(int a, int b){
  if (a == b)
    __VERIFIER_error();
}
int main(){
  int d = 0;
  int c = 4;
  int e = 2;
  f (d=c&&e, 1);
  return 0;
}
```

```
d = __CPAchecker_TMP_0
__CPAchecker_TMP_0 = c&&e
```

Test oracle: unsafe

Example: Front-end related bug in CPAchecker

```
void f(int a, int b){
  if (a == b)
    __VERIFIER_error();
}
int main(){
  int d = 0;
  int c = 4;
  int e = 2;
  f (d=c&&e, 1);
  return 0;
}
```

```
d = __CPAchecker_TMP_0
__CPAchecker_TMP_0 = c&&e
```

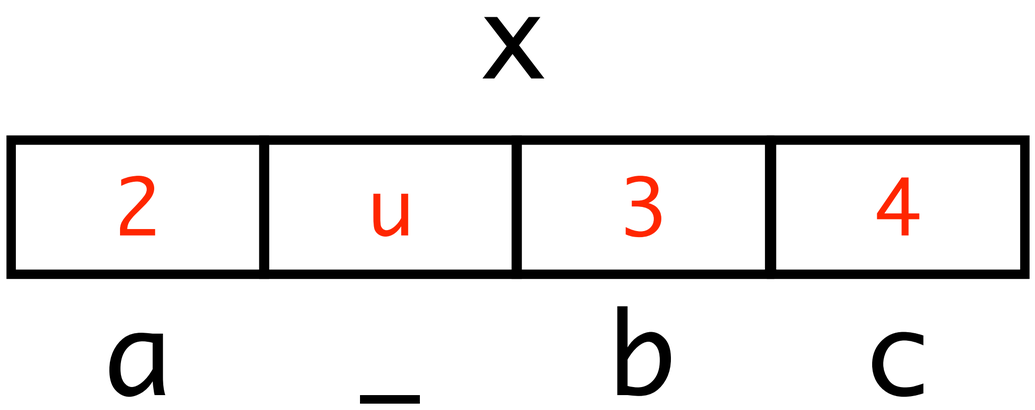
Test oracle: unsafe

: safe

Example: Language feature related bug in CBMC

```
struct {
int a:4; int :4;
int b:4; int c:4;
} x = { 2,3,4 };
int main (){
  if (x.b != 3)
    __VERIFIER_error();
  return 0;
}
```

Test oracle: safe



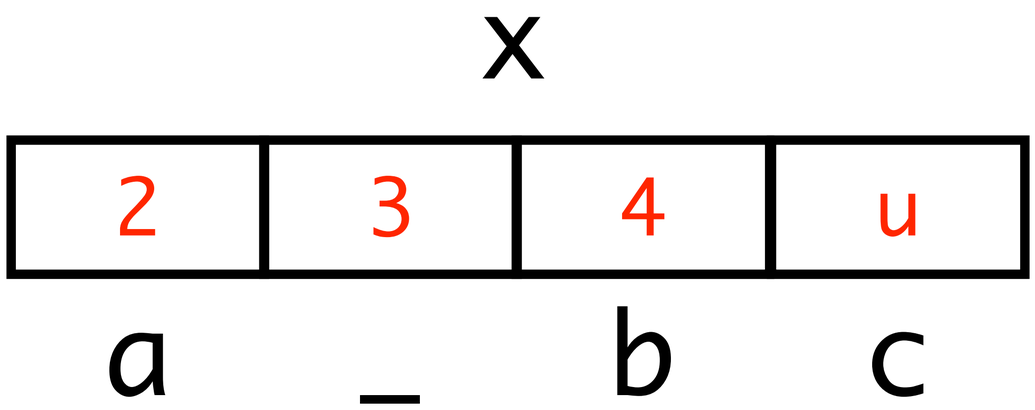
"Unnamed members of objects of structure type do not participate in initialization."

-- C standard

Example: Language feature related bug in CBMC

```
struct {
int a:4; int :4;
int b:4; int c:4;
} x = { 2,3,4 };
int main (){
  if (x.b != 3)
    __VERIFIER_error();
  return 0;
}
```

Test oracle: safe



"Unnamed members of objects of structure type do not participate in initialization."

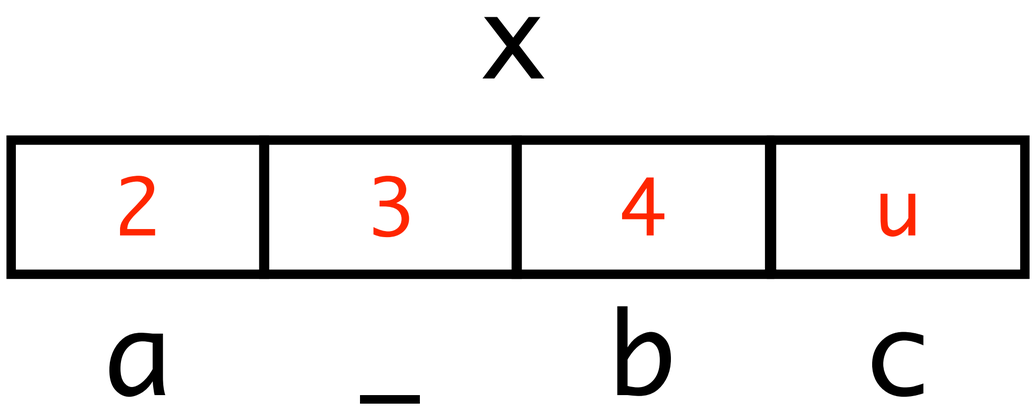
-- C standard

Example: Language feature related bug in CBMC

```
struct {
int a:4; int :4;
int b:4; int c:4;
} x = { 2,3,4 };
int main (){
  if (x.b != 3)
    __VERIFIER_error();
  return 0;
}
```

Test oracle: safe

CBMC : unsafe



"Unnamed members of objects of structure type do not participate in initialization."

-- C standard

Example: Configuration related bug in Seahorn

```
void test(int x,int y,
          int q){
  if ((x / y) != q )
    __VERIFIER_error();
}
int main (){
  test (7, 6, 1);
  test (-7, -6, 1);
  return 0;
}
```

```
7/6 == 1
-7/-6 == 1
```

Test oracle: safe

Example: Configuration related bug in Seahorn

```
void test(int x,int y,
          int q){
  if ((x / y) != q )
    __VERIFIER_error();
}
int main (){
  test (7, 6, 1);
  test (-7, -6, 1);
  return 0;
}
```

```
7/6 == 1
-7/-6 == 1
```

Test oracle: safe


```
sea pf file.c
```

Example: Configuration related bug in Seahorn

```
void test(int x,int y,
          int q){
  if ((x / y) != q )
    __VERIFIER_error();
}
int main (){
  test (7, 6, 1);
  test (-7, -6, 1);
  return 0;
}
```

```
7/6 == 1
-7/-6 == 1
```

Test oracle: safe

```
sea pf file.c
```

SeaHorn : unsafe

Example: Configuration related bug in Seahorn

```
void test(int x,int y,
          int q){
  if ((x / y) != q )
    __VERIFIER_error();
}
int main (){
  test (7, 6, 1);
  test (-7, -6, 1);
  return 0;
}
```

```
7/6 == 1
-7/-6 == 1
```

Test oracle: safe


```
sea pf
-inline file.c
```

Example: Configuration related bug in Seahorn

```
void test(int x,int y,
          int q){
  if ((x / y) != q )
    __VERIFIER_error();
}
int main (){
  test (7, 6, 1);
  test (-7, -6, 1);
  return 0;
}
```

```
7/6 == 1
-7/-6 == 1
```

Test oracle: safe

```
sea pf
-inline file.c
```

SeaHorn : safe

Evaluation on SV-COMP benchmarks

Evaluation on SV-COMP benchmarks

- SV-COMP benchmarks
  - Benchmarks for software verification competition.

Evaluation on SV-COMP benchmarks

- SV-COMP benchmarks
  - Benchmarks for software verification competition.
  - Selected 1106 files from ReachSafety and SoftwareSystems.

Evaluation on SV-COMP benchmarks

- SV-COMP benchmarks
  - Benchmarks for software verification competition.
  - Selected 1106 files from ReachSafety and SoftwareSystems.
- Result
  - Found 5 bugs via SV-COMP benchmark.

Evaluation on SV-COMP benchmarks

- SV-COMP benchmarks
  - Benchmarks for software verification competition.
  - Selected 1106 files from ReachSafety and SoftwareSystems.
- Result
  - Found 5 bugs via SV-COMP benchmark.
- Undefined behavior
  - It is not the bug, but leads to false alarms.

Usability of software model checkers

Usability of software model checkers

✓ Should not give wrong checking result.

Usability of software model checkers

- ✓ Should not give wrong checking result.
- ✓ Should not let user to choose the right configuration.

Usability of software model checkers

- ✓ Should not give wrong checking result.
- ✓ Should not let user to choose the right configuration.
- ✓ Better user manual.

Developers' Feedback

kuhar commented on Jul 26 • edited

Hello,

FYI, the MCFuzz issues filled against SeaHorn are now in its test suite and run in CI after every commit:
https://github.com/seahorn/seahorn/tree/master/test/mcfuzz

As of today, SeaHorn passes 9 out of the 16 test-cases provided.

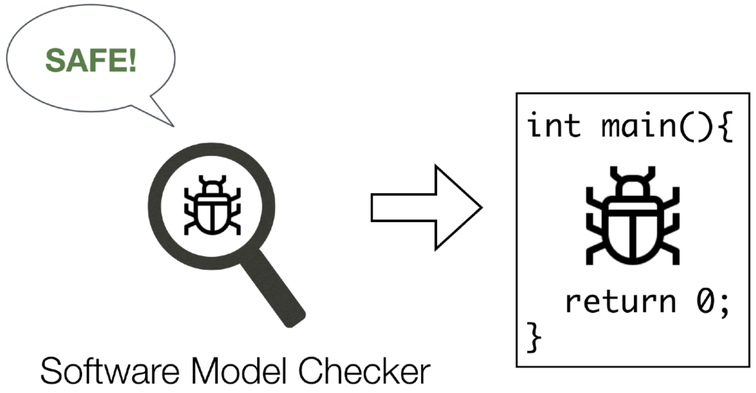
kroening commented on Jan 6 — Member

Many thanks; I've made a PR with a test for this as PR #3695.

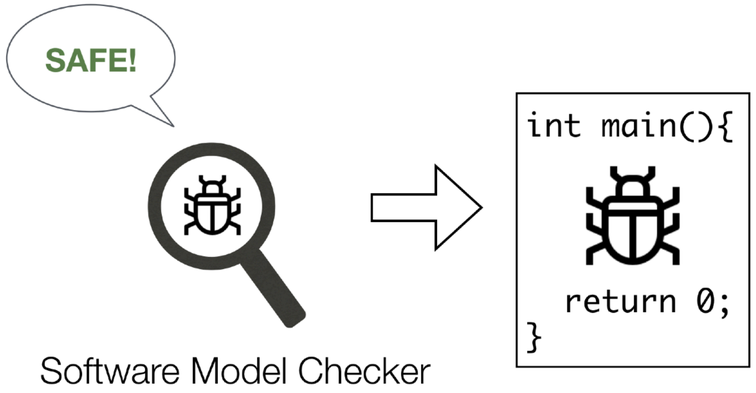
Again, thank you very much for finding and reporting these cases!
This is very helpful for us.

Kind regards,
Philipp

Software model checker may have bugs



Software model checker may have bugs



Proposed the approaches to find the bug

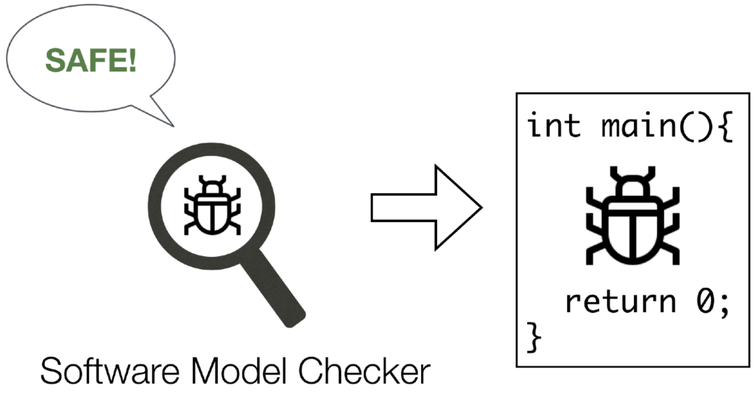
Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

find more kinds of bugs

save more time

Software model checker may have bugs



Proposed the approaches to find the bug

Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

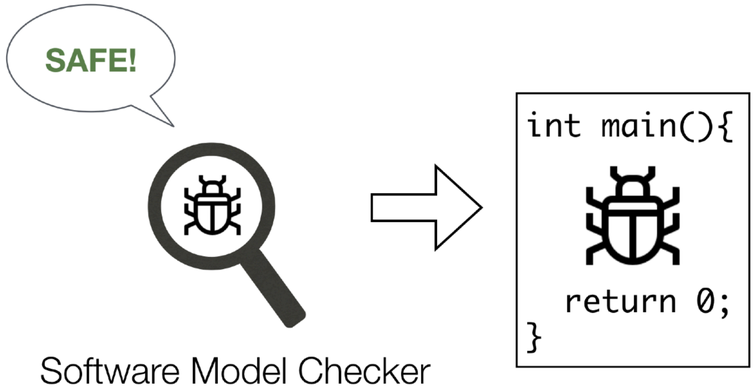
find more kinds of bugs

save more time

Totally found 62 bugs in three model checkers

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

Software model checker may have bugs



Proposed the approaches to find the bug

Approach

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

find more kinds of bugs

save more time

Totally found 62 bugs in three model checkers

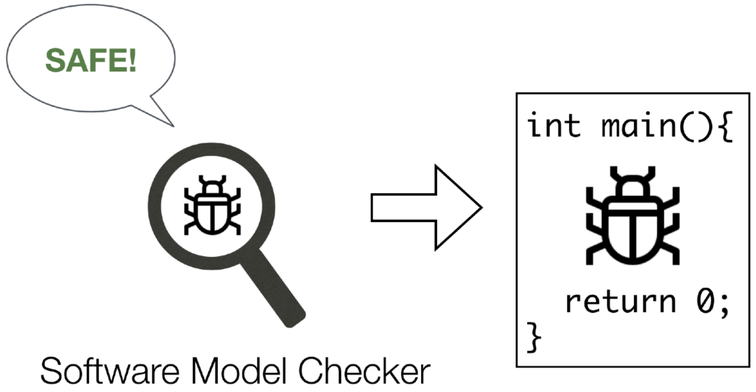
	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

Categorized the bugs into 7 categories

Bug categories

- Front-end
- Memory model
- Pointer alias
- Third-party component
- C standard library
- Language feature
- Configuration

Software model checker may have bugs



Proposed the approaches to find the bug

Approach

find more kinds of bugs

- Approach I: Enumerative Reachability (ER)
- Approach II: Enumerative **Counting** Reachability (ECR)
- Approach III: **Fused** Counting Reachability (FCR)

save more time

Thank you !

Totally found 62 bugs in three model checkers

	CPAchecker	CBMC	SeaHorn	Total
Fixed	14	4	2	20
Confirmed	22	2	14	38
Unconfirmed	0	4	0	4
Total	36	10	16	62

Categorized the bugs into 7 categories

Bug categories

- Front-end
- Memory model
- Pointer alias
- Third-party component
- C standard library
- Language feature
- Configuration