Difference Verification with Conditions

Thomas Lemberger

Joint work with Dirk Beyer and Marie-Christine Jakobs

LMU Munich and TU Darmstadt

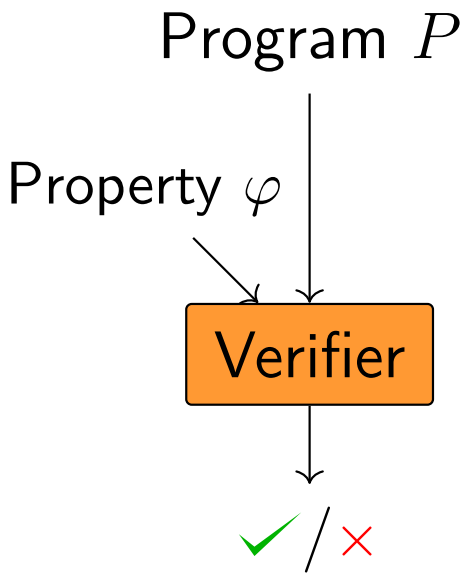


SEFM '20

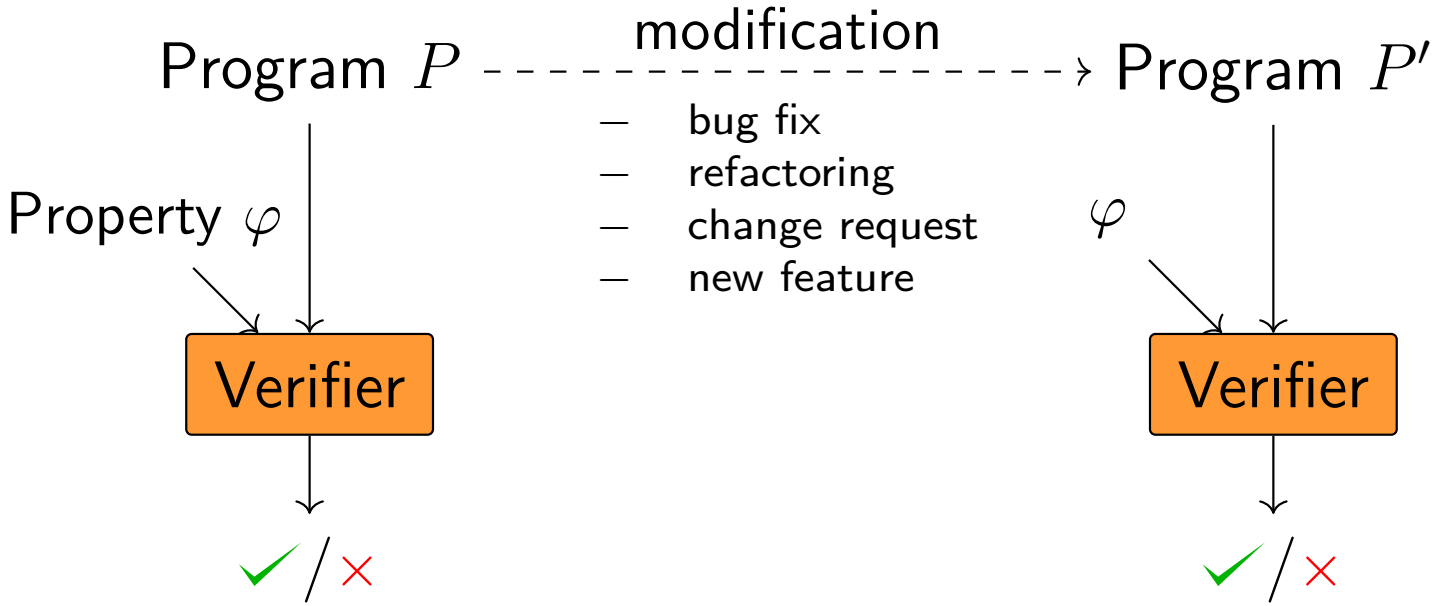
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

SoSy-Lab Software Systems

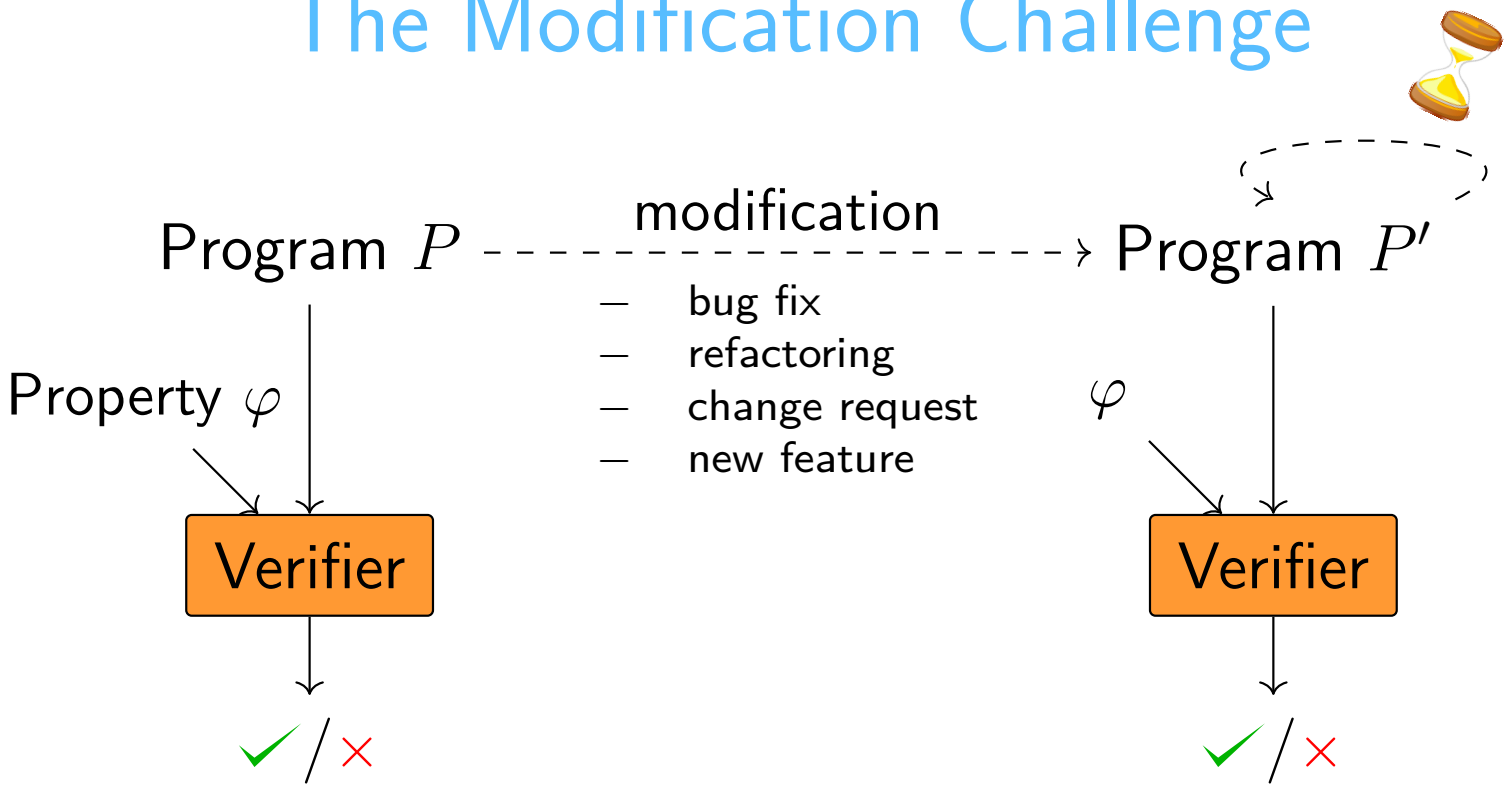
The Modification Challenge



The Modification Challenge



The Modification Challenge



Challenge:

- Programs change frequently
- Reverification necessary after each change
- Reverification must keep up with changes

$\Rightarrow$ Verification of every modified program **infeasible!**

Program Modifications

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program Modifications

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```

Program Modifications

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```

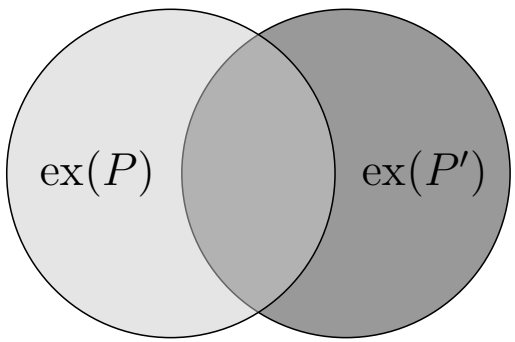
Program Modifications

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



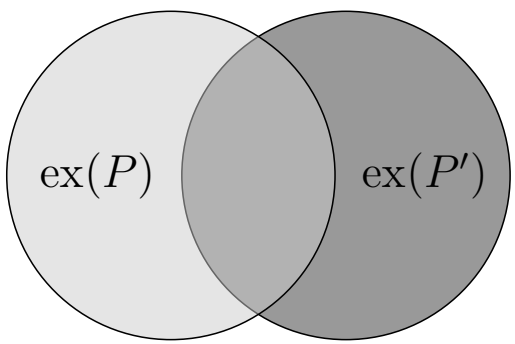
Program Modifications

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

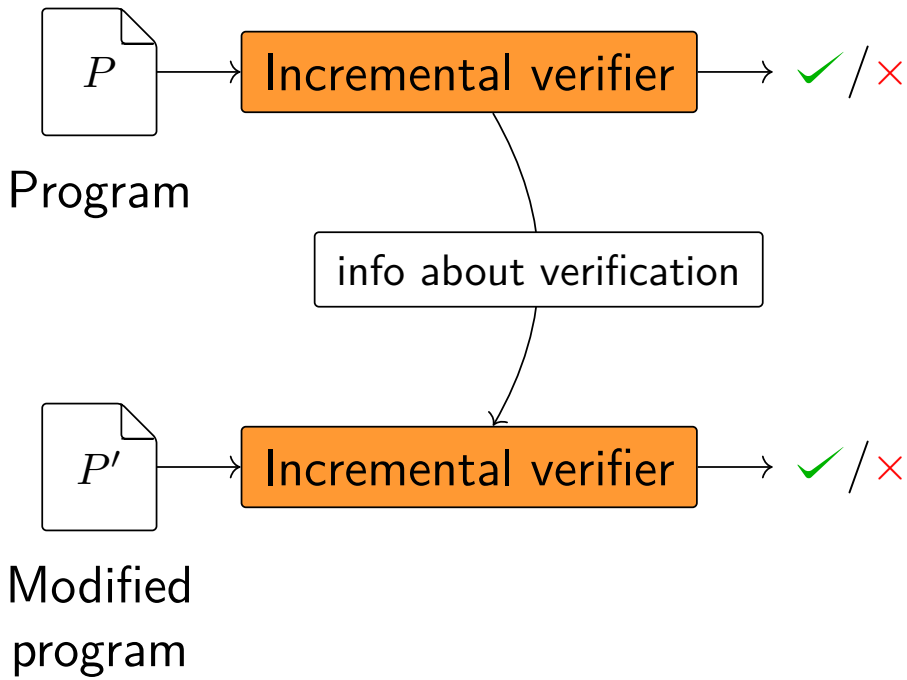
```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



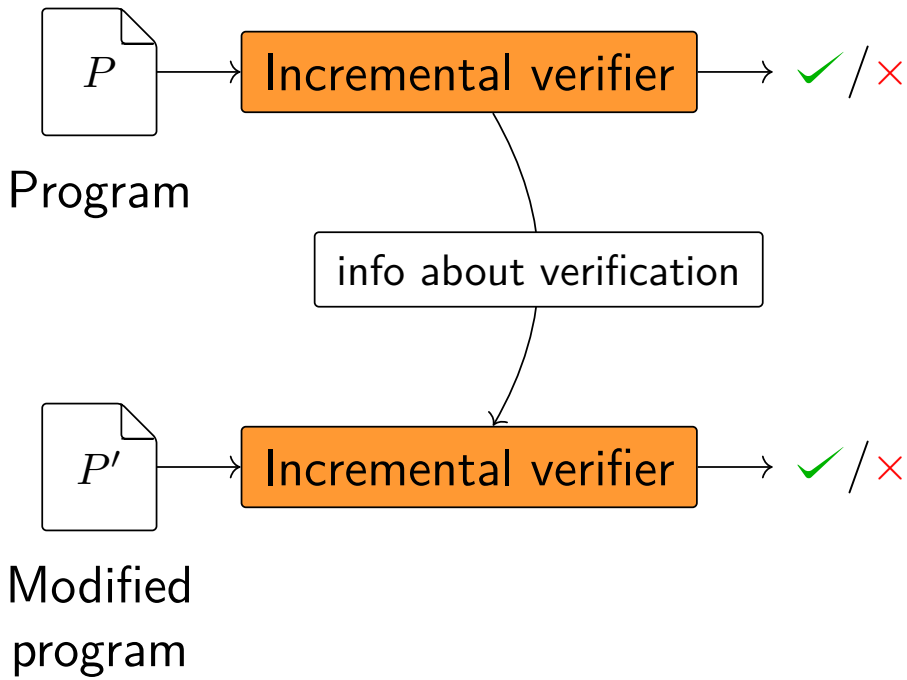
⇒ verification of unchanged program executions wastes resources

⇒ **better:** reuse information obtained in previous verification run(s)

Incremental Verification



Incremental Verification



Types of info:

- Function summaries
- Invariants
- Caches
- ...

Weaknesses of Existing Approaches

Existing approaches...

- require initial full verification
- require same verifier in each step
- tailored to specific verification approach

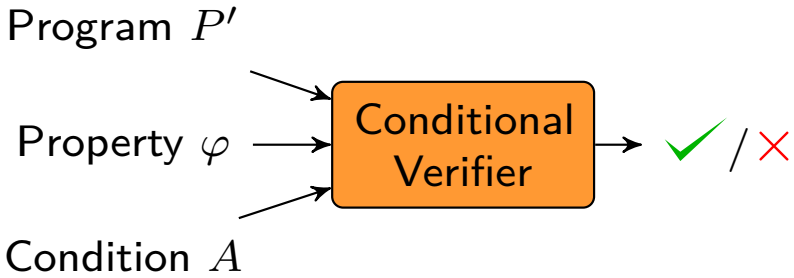
Difference Verification with Conditions...

- requires no initial verification ✓
- doesn't care or know about switches between verifiers ✓
- is independent of any specific verification approach ✓

Difference Verification with Conditions

- Use verifier-independent *condition* to exchange information
- Construct condition from original and modified program (no information from previous runs used)

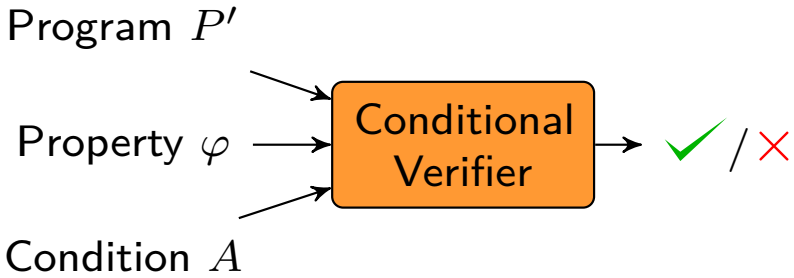
Conditional Model Checking[1]



- Condition guides verifier
- Restricts relevant program state space to explore

[1] D. Beyer, T. A. Henzinger, M. E. Keremoglu, P. Wendler: Conditional Model Checking, 2012.

Conditional Model Checking[1]



- Condition guides verifier
- Restricts relevant program state space to explore

- Condition is automaton with two types of edge labels:
  - *Source-code guards* restrict syntactic paths ("do not explore if-branch")
  - *State-space guards* restrict possible program states ("assume x $>$ 0")

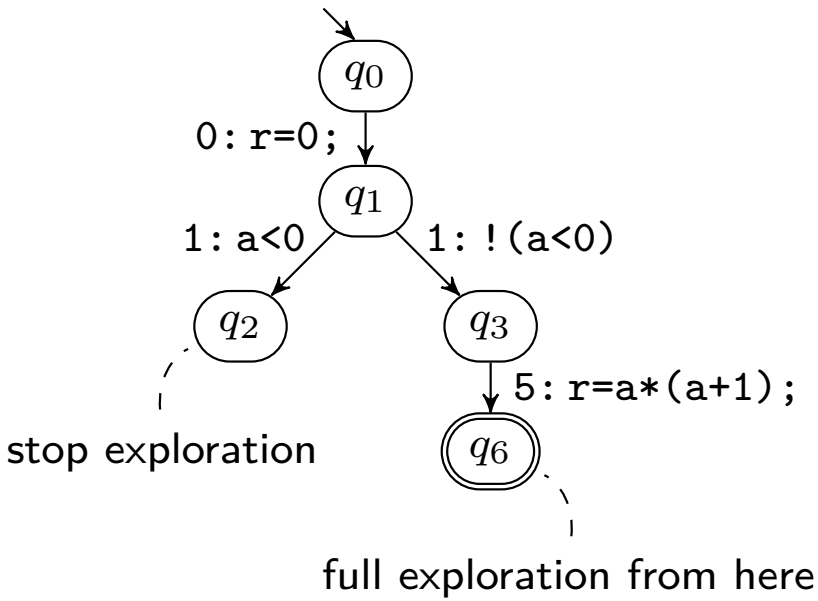
[1] D. Beyer, T. A. Henzinger, M. E. Keremoglu, P. Wendler: Conditional Model Checking, 2012.

Example Difference Condition

Program P'

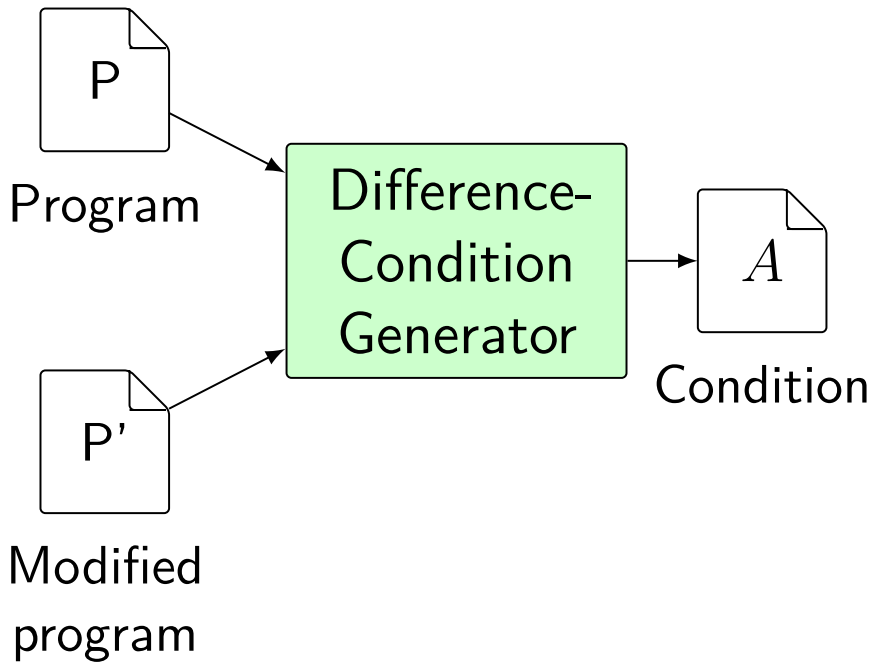
```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```

Difference-Condition A



Here only source-code guards

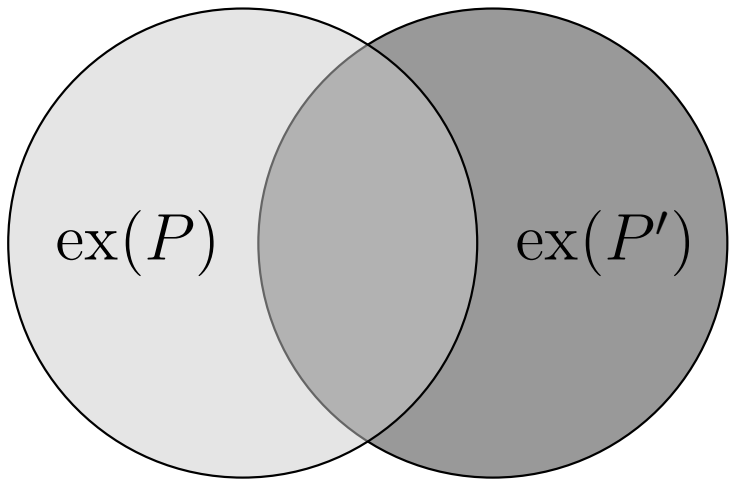
Difference-Condition Generation



Difference-Condition Generation

Soundness Criterion:

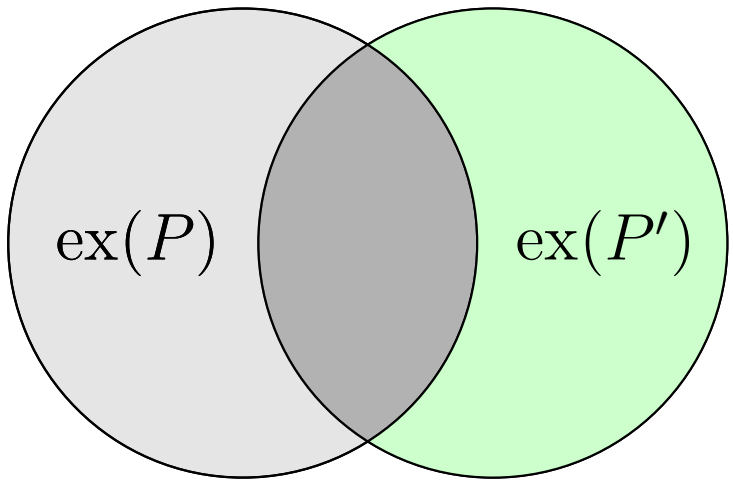
The generated condition does not exclude any modified execution, i.e., none from $ex(P') \setminus ex(P)$.



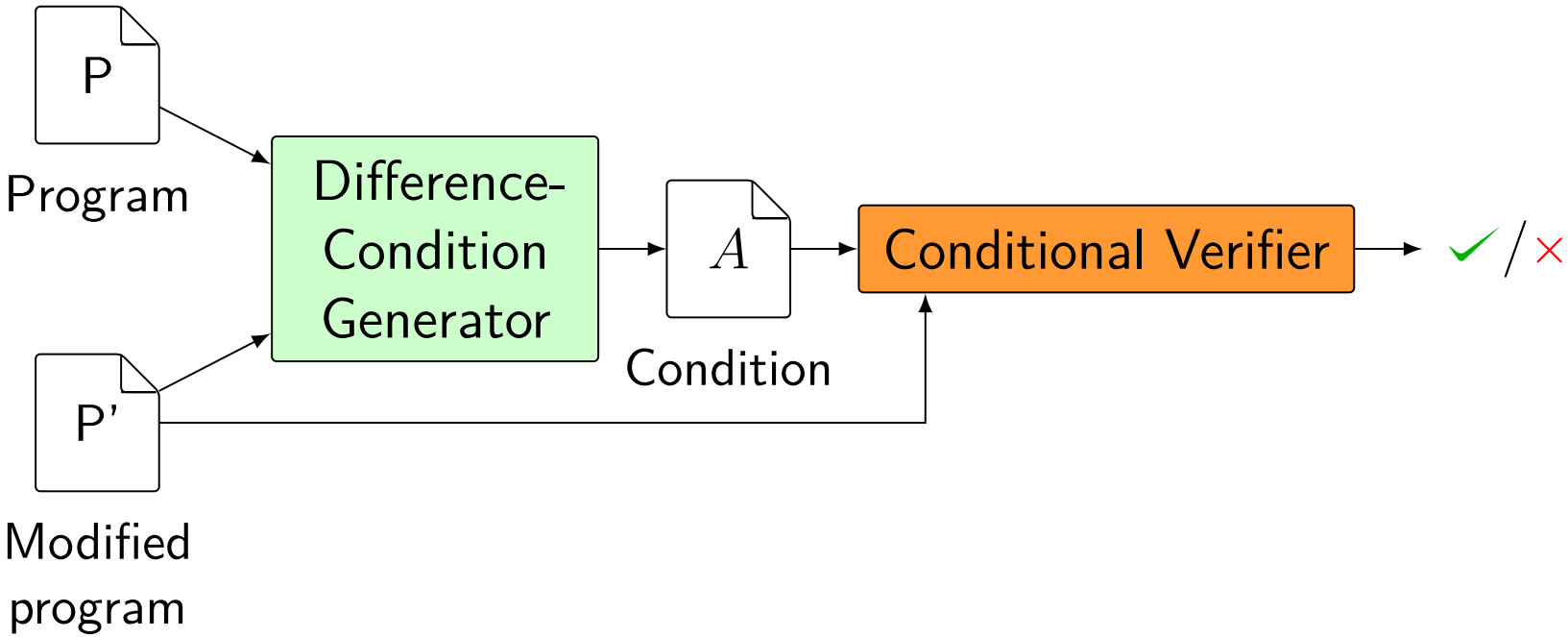
Difference-Condition Generation

Soundness Criterion:

The generated condition does not exclude any modified execution, i.e., none from $ex(P') \setminus ex(P)$.

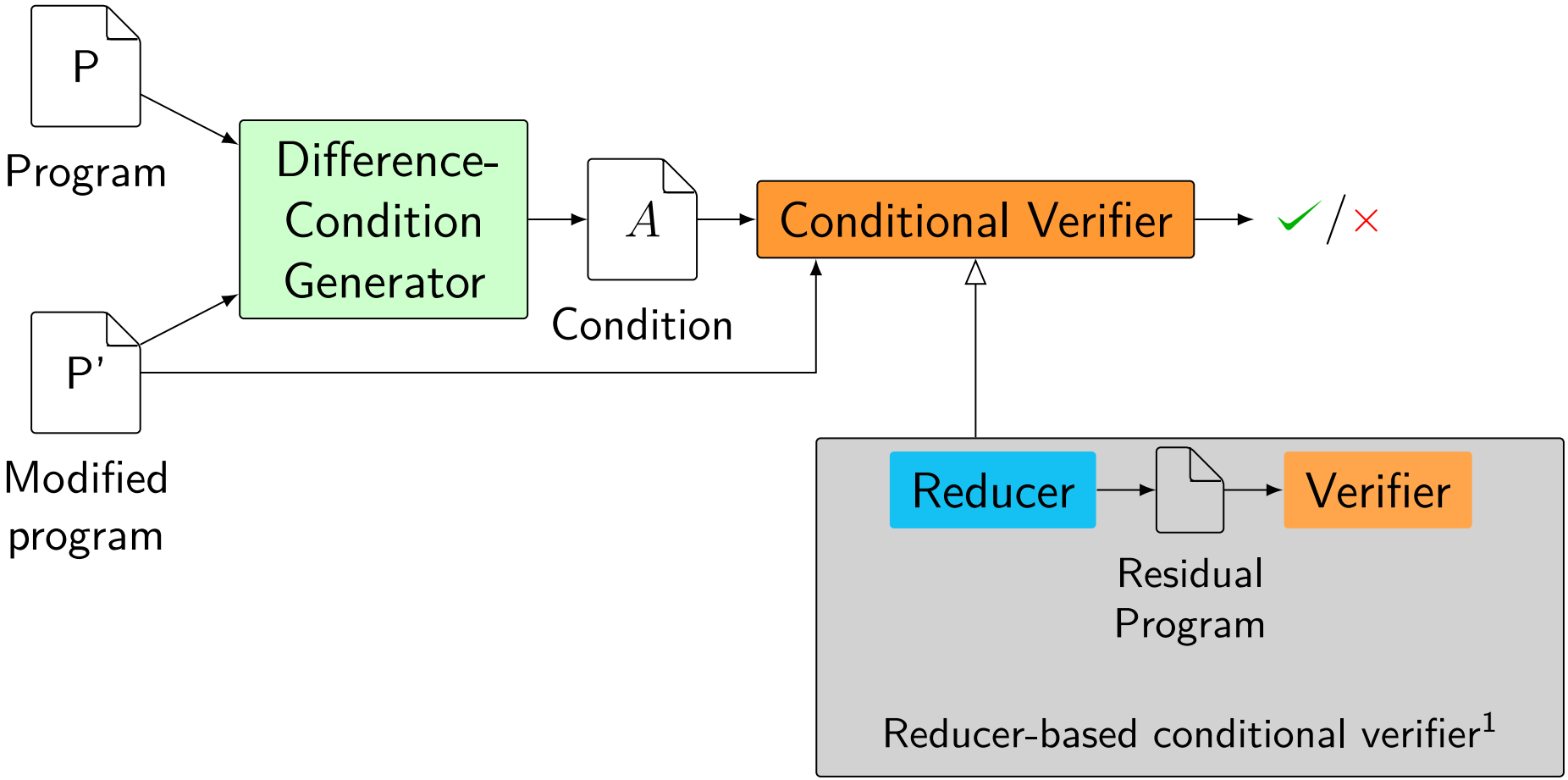


Difference Verification with Conditions



[1] D. Beyer, M.-C. Jakobs, T. Lemberger, H. Wehrheim: Reducer-Based Construction of Conditional Verifiers

Difference Verification with Conditions



[1] D. Beyer, M.-C. Jakobs, T. Lemberger, H. Wehrheim: Reducer-Based Construction of Conditional Verifiers

Example for difference-condition generator:

DiffCond

Syntactic Difference-Condition Generation

DiffCond: Syntactic Difference-Condition Generator

1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

DiffCond: Syntactic Difference-Condition Generator

1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```

DiffCond: Syntactic Difference-Condition Generator

1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
→ 0 r=0;
  1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
      else
  5   r=a+a+1;
  6   r=r/2;
  7 assert r>0;
```

Program P'

```
→ 0 r=0;
  1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
      else
  5   r=a*(a+1);
  6   r=r/2;
  7 assert r>0;
```

(l_0, l_0')

DiffCond: Syntactic Difference-Condition Generator

1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
  0 r=0;
→ 1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
      else
  5   r=a+a+1;
  6   r=r/2;
  7 assert r>0;
```

Program P'

```
  0 r=0;
→ 1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
      else
  5   r=a*(a+1);
  6   r=r/2;
  7 assert r>0;
```

(l_0, l_0')

0: r=0; ↓

(l_1, l_1')

DiffCond: Syntactic Difference-Condition Generator

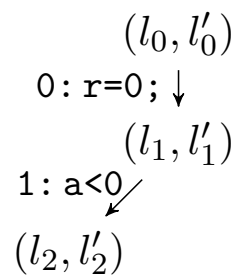
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
  0 r=0;
  1 if(a<0)
→ 2   while(a<0)
  3     r=r-a;
  4     a=a+1;
    else
  5   r=a+a+1;
  6   r=r/2;
  7 assert r>0;
```

Program P'

```
  0 r=0;
  1 if(a<0)
→ 2   while(a<0)
  3     r=r-a;
  4     a=a+1;
    else
  5   r=a*(a+1);
  6   r=r/2;
  7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

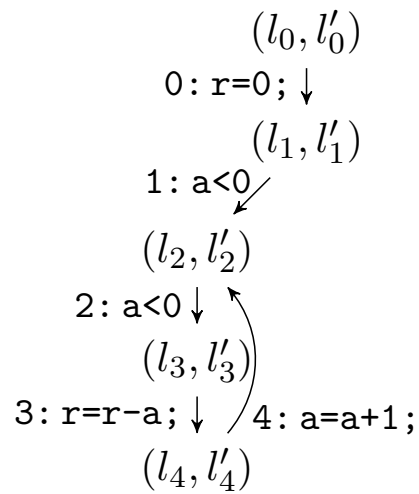
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

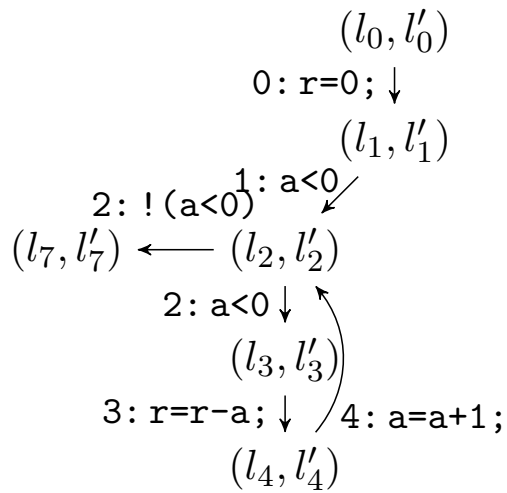
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a+a+1;
6   r=r/2;
→ 7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
5   r=a*(a+1);
6   r=r/2;
→ 7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

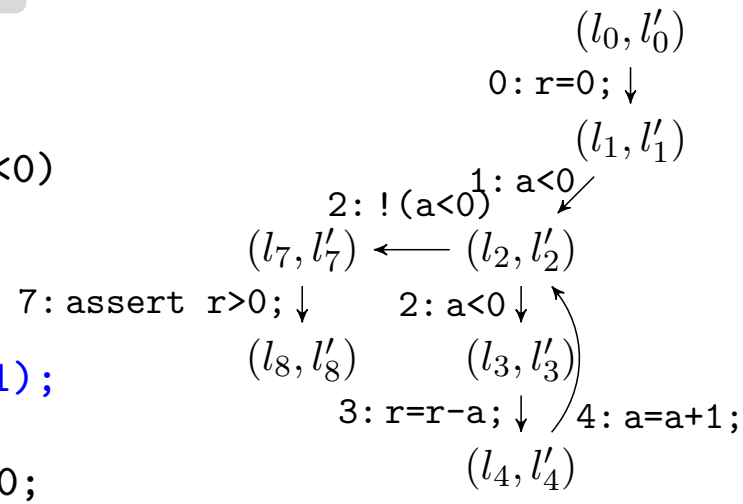
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
  0 r=0;
→ 1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
    else
  5   r=a+a+1;
  6   r=r/2;
  7 assert r>0;
```

Program P'

```
  0 r=0;
→ 1 if(a<0)
  2   while(a<0)
  3     r=r-a;
  4     a=a+1;
    else
  5   r=a*(a+1);
  6   r=r/2;
  7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

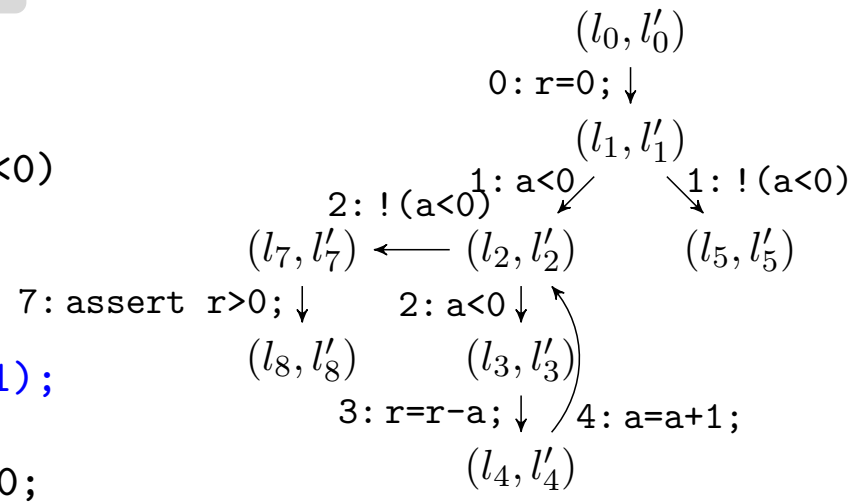
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
→ 5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
    else
→ 5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

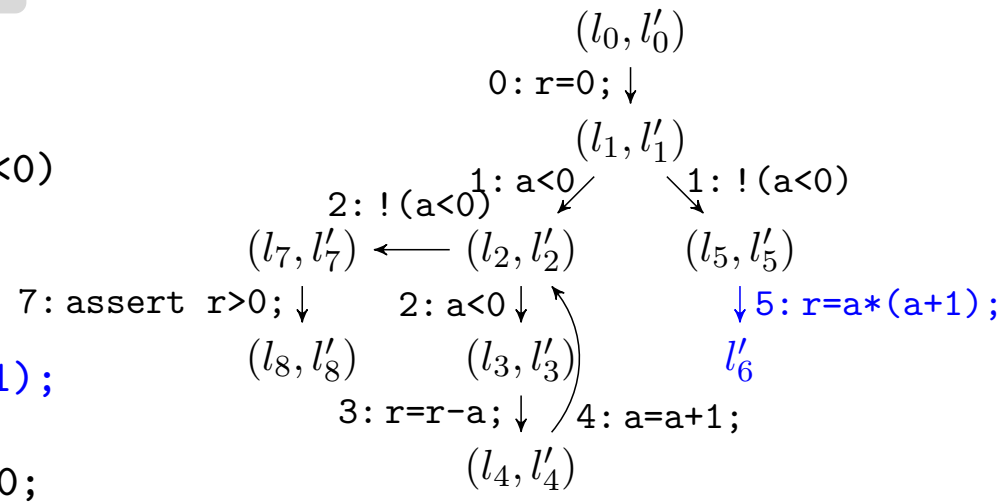
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



DiffCond: Syntactic Difference-Condition Generator

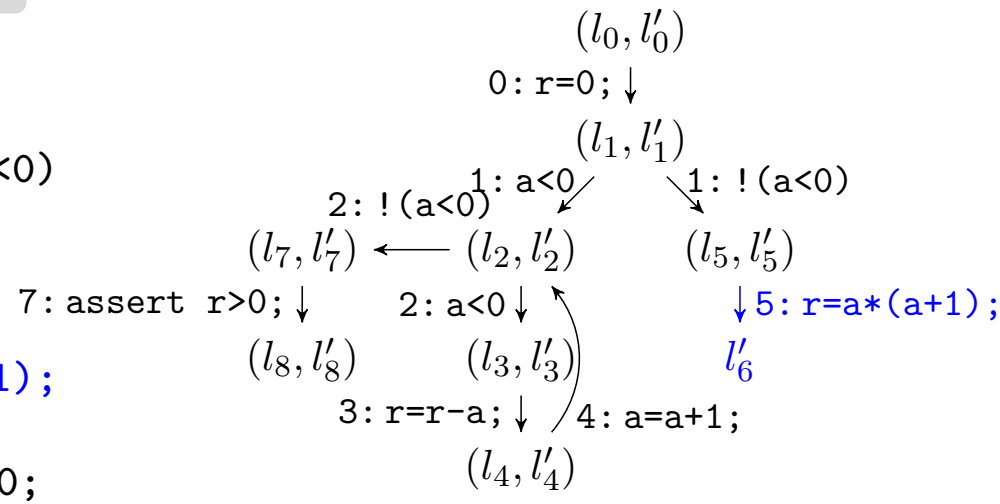
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



2. Generate condition from parallel composition

DiffCond: Syntactic Difference-Condition Generator

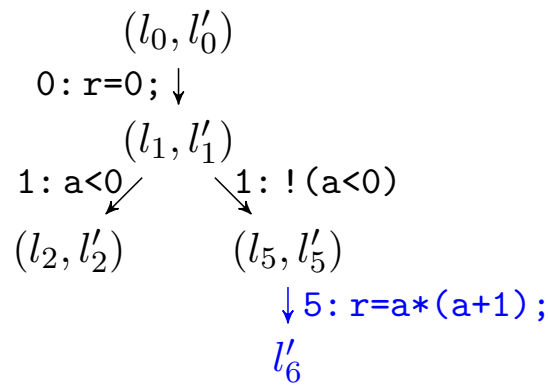
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



2. Generate condition from parallel composition

DiffCond: Syntactic Difference-Condition Generator

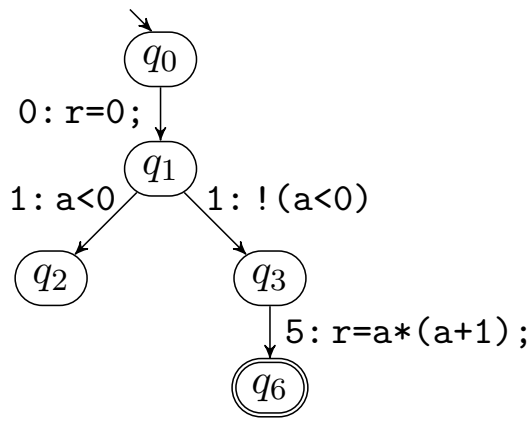
1. Detect differences
   - Compute the parallel composition of original and modified program
   - At each composed path, stop if modified program deviates

Program P

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a+a+1;
6   r=r/2;
7 assert r>0;
```

Program P'

```
0 r=0;
1 if(a<0)
2   while(a<0)
3     r=r-a;
4     a=a+1;
  else
5   r=a*(a+1);
6   r=r/2;
7 assert r>0;
```



2. Generate condition from parallel composition

Conceptual Limitation of DiffCond

- Modifications propagate
  - Changes in global variables
  - Looping programs

$\Rightarrow$ Works best on loosely coupled systems

Program P^{∞}

```
0 while (1) {
1   r=0;
2   if(a<0)
3     while(a<0)
4       r=r-a;
5       a=a+1;
    else
6     r=a+a+1;
7     r=r/2;
8   assert r>0;
9 }
```

Evaluation

Evaluation

- Full reverification $\Leftrightarrow$ difference verification with conditions
  1. with native conditional verifier
  2. with off-the-shelf verifier turned into conditional verifier with reducer-based approach

Evaluation

- Full reverification $\Leftrightarrow$ difference verification with conditions
  1. with native conditional verifier
  2. with off-the-shelf verifier turned into conditional verifier with reducer-based approach

- Implementation for C programs
- Artifact:
  `https://gitlab.com/sosy-lab/research/data/difference-data`

Evaluation

- Full reverification $\Leftrightarrow$ difference verification with conditions
  1. with native conditional verifier
  2. with off-the-shelf verifier turned into conditional verifier with reducer-based approach

- Implementation for C programs
- Artifact:
  `https://gitlab.com/sosy-lab/research/data/difference-data`
- Verifiers used:
  - CPAchecker with predicate abstraction (conditional verifier)
  - CPA-Seq
  - Ultimate Automizer

Evaluation Setting

Evaluation Environment

- Similar to Software-Verification Competition SV-COMP
- Per run: 15 min. time limit, 15 GB memory

Evaluation Setting

Evaluation Tasks

- 10 426 reverification tasks (P, P')
- Artifically generated from *sv-benchmarks*
- Each program P is combination of two strongly coupled programs:

```
int main1() {
  // original program 1
}

int main2() {
  // original program 2
}
```

```
// Combination
int main() {
  if(nondet_bool())
    main1();
  else
    main2();
}
```

Evaluation Setting

Evaluation Tasks

- 10 426 reverification tasks (P, P')
- Artifically generated from *sv-benchmarks*
- Each program P is combination of two strongly coupled programs:

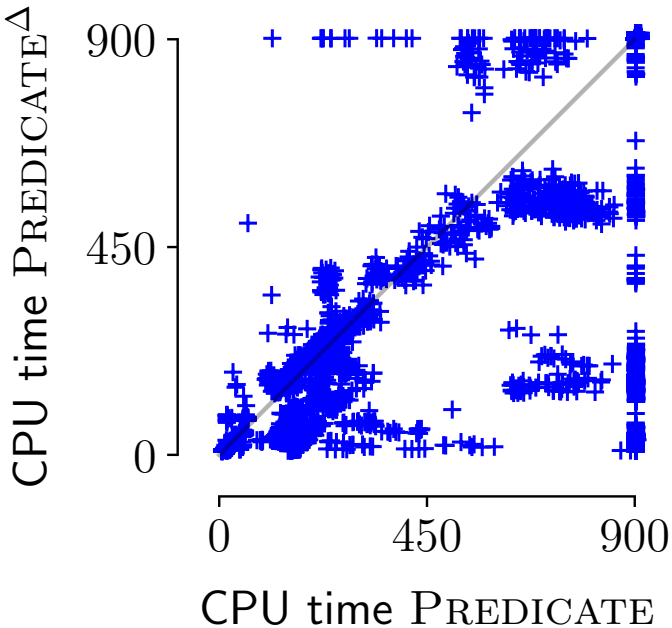
```
int main1() {
  // original program 1
}

int main2() {
  // original program 2
}
```

```
// Combination
int main() {
  if(nondet_bool())
    main1();
  else
    main2();
}
```

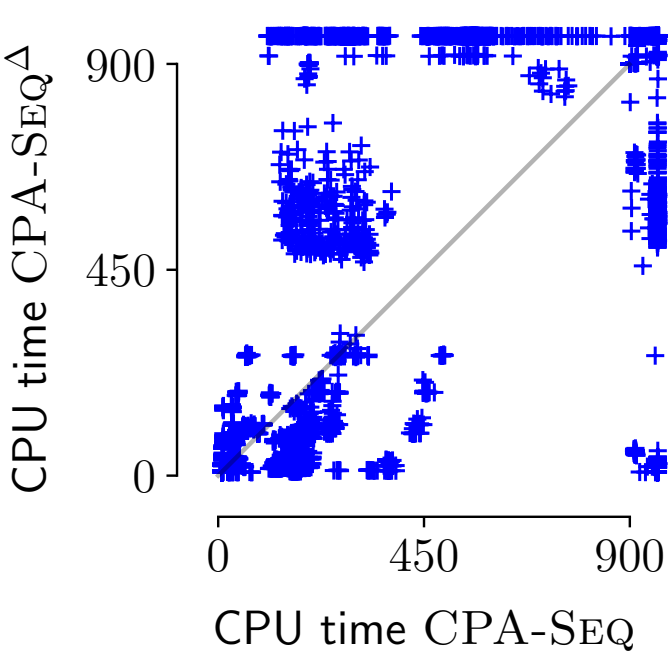
- Program Modifications $P \rightarrow P'$: Existing modifications
  - Fix bug
  - Introduce bug
  - Stay safe

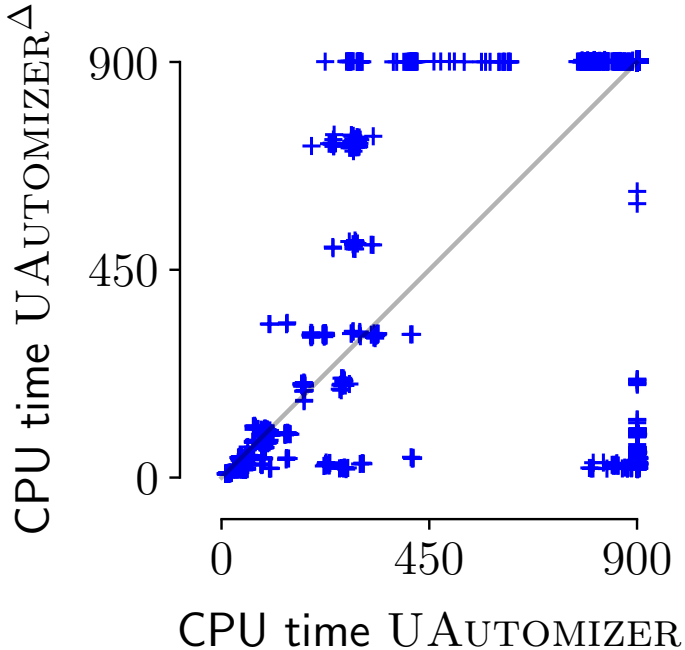
1. Reverification with native conditional verifier



- CPU time (in s) per task
- PREDICATE: CPAchecker with predicate abstraction
- Δ: Difference verification with conditions

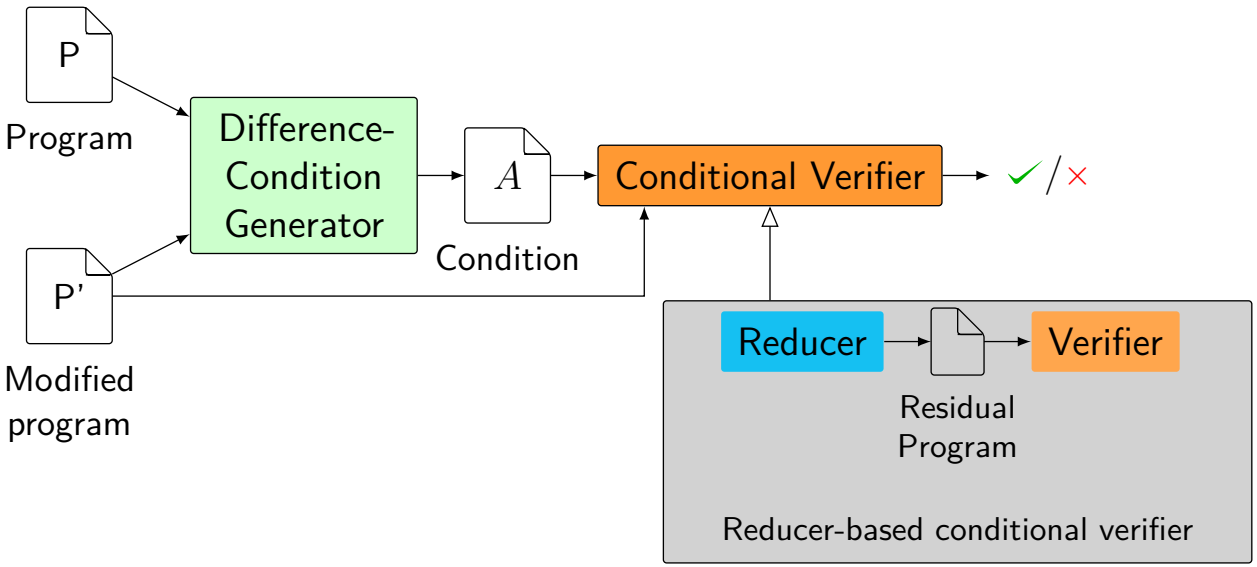
2. Reverification with off-the-shelf verifier





- CPU time (in s) per task
- Δ: Difference verification with conditions
- Difference verification with conditions suffers from residual programs

Conclusion



- requires no initial verification
- allows changing the verifier at any point
- independent of any specific verification approach