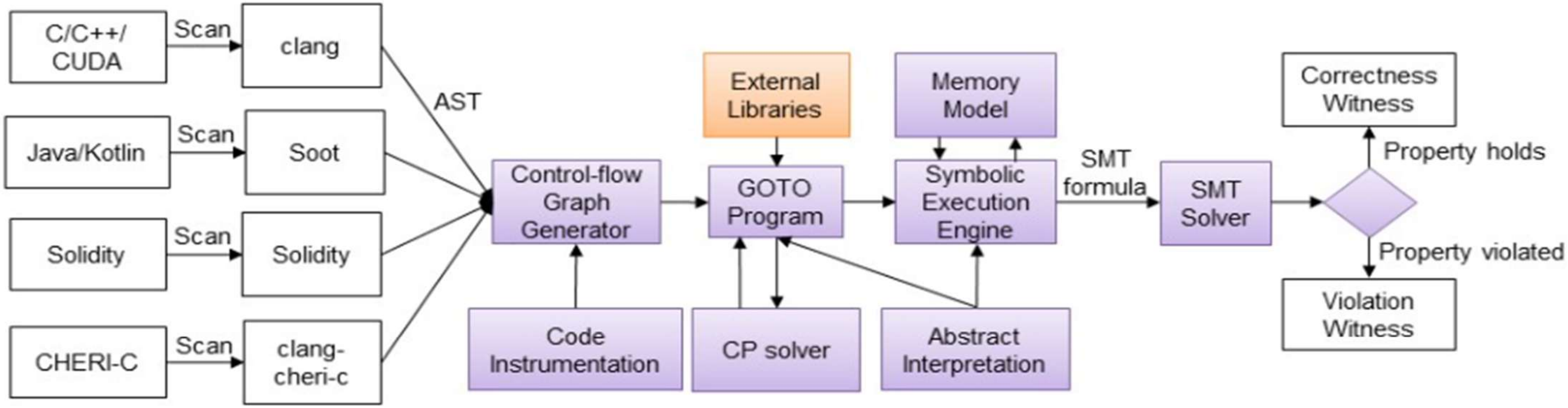
ESBMC v7.7: Efficient Concurrent Software Verification with Scheduling, Incremental SMT and Partial Order Reduction

Tong Wu, Xianzhiyu Li, Edoardo Manino, Rafael Sá Menezes, Mikhail R. Gadelha, Shale Xiong, Norbert Tihanyi, Pavlos Petoumenos, Lucas C. Cordeiro

University of Manchester
Department of Computer Science

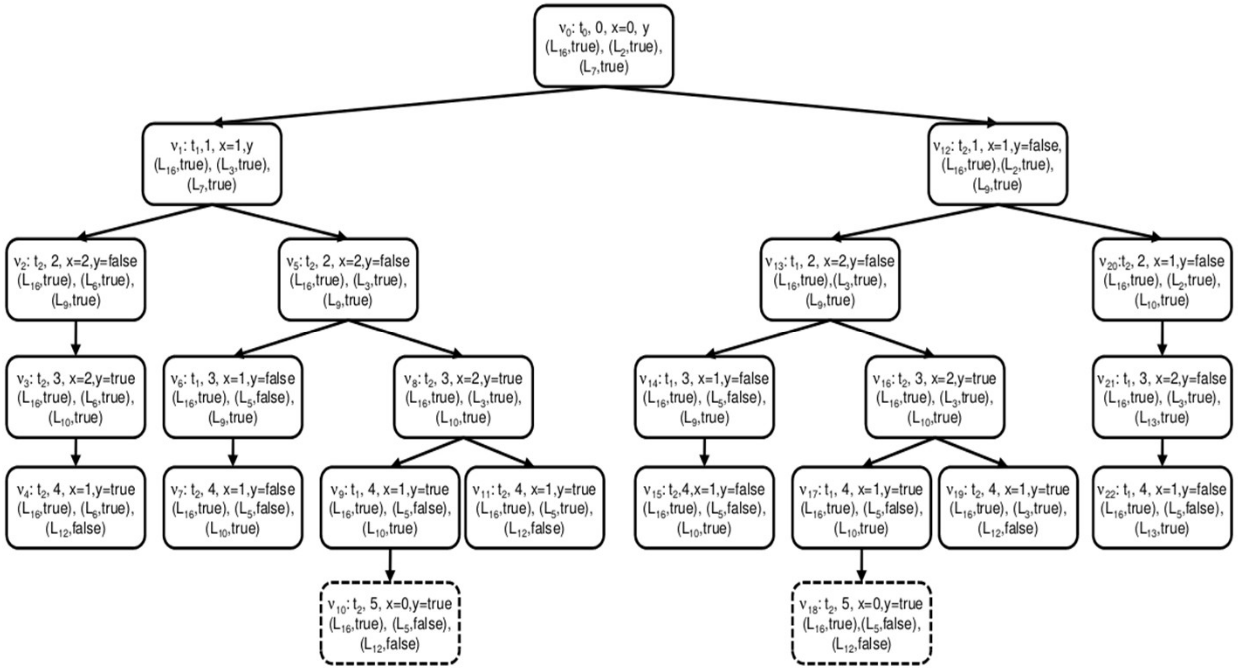
ESBMC: Software Verification Platform

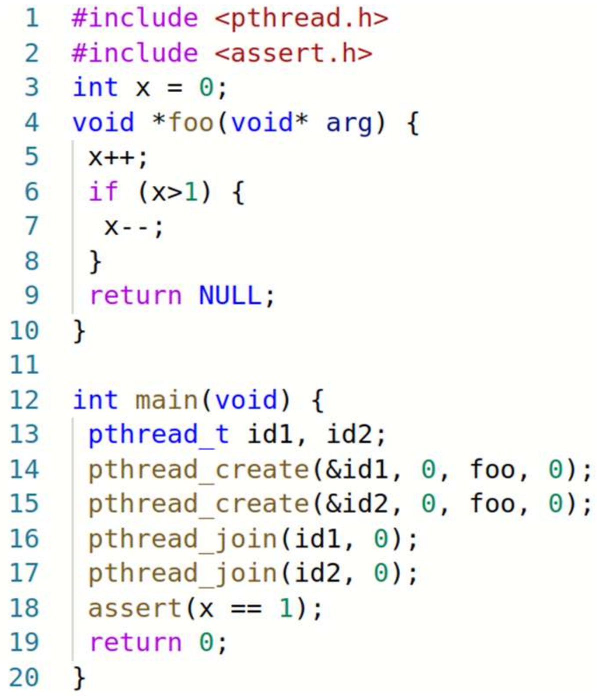
Logic-based automated reasoning for checking the **safety** and **security** of **single- and multi-threaded programs**



Combines BMC, *k*-induction, abstract interpretation, CP/SMT solving towards correctness proof and bug hunting
www.esbmc.org

Concurrency in ESBMC


ν_0: t_0, 0, x=0, y (L$_{16}$,true), (L$_2$,true), (L$_7$,true)
ν_1: t_1,1, x=1,y (L$_{16}$,true), (L$_3$,true), (L$_7$,true)
ν_{12}: t_2,1, x=1,y=false, (L$_{16}$,true),(L$_2$,true), (L$_9$,true)
ν_2: t_2, 2, x=2,y=false (L$_{16}$,true), (L$_6$,true), (L$_9$,true)
ν_5: t_2, 2, x=2,y=false (L$_{16}$,true), (L$_3$,true), (L$_9$,true)
ν_{13}: t_1, 2, x=2,y=false (L$_{16}$,true),(L$_3$,true), (L$_9$,true)
ν_{20}:t_2, 2, x=1,y=false (L$_{16}$,true), (L$_2$,true), (L$_{10}$,true)
ν_3: t_2, 3, x=2,y=true (L$_{16}$,true), (L$_6$,true), (L$_{10}$,true)
ν_6: t_1, 3, x=1,y=false (L$_{16}$,true), (L$_5$,false), (L$_9$,true)
ν_8: t_2, 3, x=2,y=true (L$_{16}$,true), (L$_3$,true), (L$_{10}$,true)
ν_{14}: t_1, 3, x=1,y=false (L$_{16}$,true), (L$_5$,false), (L$_9$,true)
ν_{16}: t_2, 3, x=2,y=true (L$_{16}$,true), (L$_3$,true), (L$_{10}$,true)
ν_{21}: t_1, 3, x=2,y=false (L$_{16}$,true), (L$_3$,true), (L$_{13}$,true)
ν_4: t_2, 4, x=1,y=true (L$_{16}$,true), (L$_6$,true), (L$_{12}$,false)
ν_7: t_2, 4, x=1,y=false (L$_{16}$,true), (L$_5$,false), (L$_{10}$,true)
ν_9: t_1, 4, x=1,y=true (L$_{16}$,true), (L$_5$,false), (L$_{10}$,true)
ν_{11}: t_2, 4, x=1,y=true (L$_{16}$,true), (L$_5$,true), (L$_{12}$,false)
ν_{15}: t_2,4,x=1,y=false (L$_{16}$,true), (L$_5$,false), (L$_{10}$,true)
ν_{17}: t_1, 4, x=1,y=true (L$_{16}$,true), (L$_5$,false), (L$_{10}$,true)
ν_{19}: t_2, 4, x=1,y=true (L$_{16}$,true), (L$_3$,true), (L$_{12}$,false)
ν_{22}: t_1, 4, x=1,y=false (L$_{16}$,true), (L$_5$,false), (L$_{13}$,true)
ν_{10}: t_2, 5, x=0,y=true (L$_{16}$,true), (L$_5$,false), (L$_{12}$,false)
ν_{18}: t_2, 5, x=0,y=true (L$_{16}$,true),(L$_5$,false), (L$_{12}$,false)



```c
#include <pthread.h>
#include <assert.h>
int x = 0;
void *foo(void* arg) {
 x++;
 if (x>1) {
  x--;
 }
 return NULL;
}

int main(void) {
 pthread_t id1, id2;
 pthread_create(&id1, 0, foo, 0);
 pthread_create(&id2, 0, foo, 0);
 pthread_join(id1, 0);
 pthread_join(id2, 0);
 assert(x == 1);
 return 0;
}
```

Main assumptions

1. **Sequential consistency** allow us to model thread schedule as a sequential program
2. **Bound on context switches*** avoids state explosion due to exponential number of interleavings

*The default bound on the number of context switches is 3

Concurrency in ESBMC

ESBMC v1.17 (2012)

Main author: Lucas C. Cordeiro
Context-bounded model checking

ESBMC v3.0 (2016)

New Clang frontend

ESBMC v6.4 (2020)

Minor concurrency improvements

ESBMC v6.9 (2022)

Minor concurrency improvements

ESBMC v7.2 (2023)

Minor concurrency improvements

ESBMC v7.3 (2024)

Support for CUDA concurrency

ESBMC v7.7 (2025)

Main author: Tong Wu
This presentation!

Concurrency in ESBMC

ESBMC v1.17 (2012)
Main author: Lucas C. Cordeiro
Context-bounded model checking

ESBMC v3.0 (2016)
New Clang frontend

ESBMC v6.4 (2020)
Minor concurrency improvements

ESBMC v6.9 (2022)
Minor concurrency improvements

ESBMC v7.2 (2023)
Minor concurrency improvements

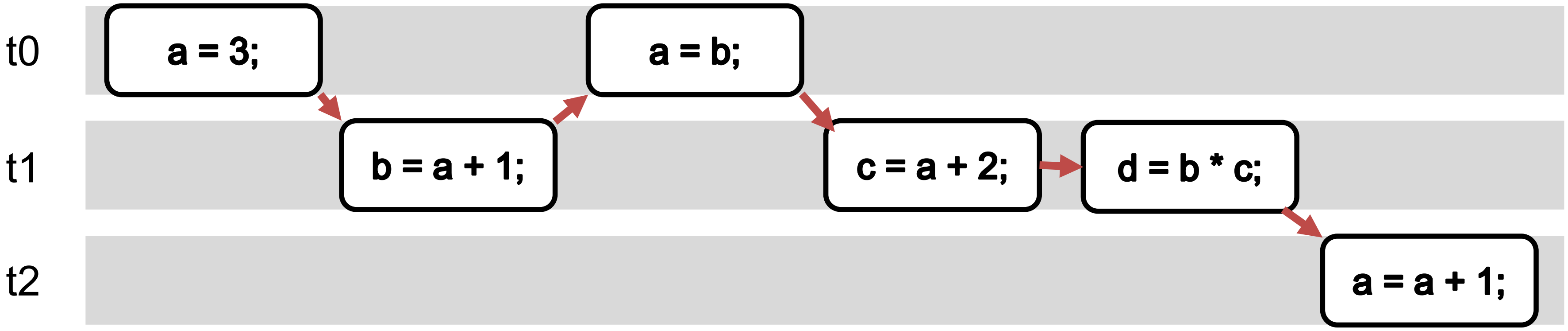
ESBMC v7.3 (2024)
Support for CUDA concurrency

ESBMC v7.7 (2025)
Main author: Tong Wu
This presentation!

Improvements

1. Reverse Priority Scheduling
2. Incremental SMT Solving
3. Partial Order Reduction
4. Data Races
5. Pthread Operational Models

Reverse Priority Scheduling



t0

a = 3;

a = b;

t1

b = a + 1;

c = a + 2;

d = b * c;

t2

a = a + 1;

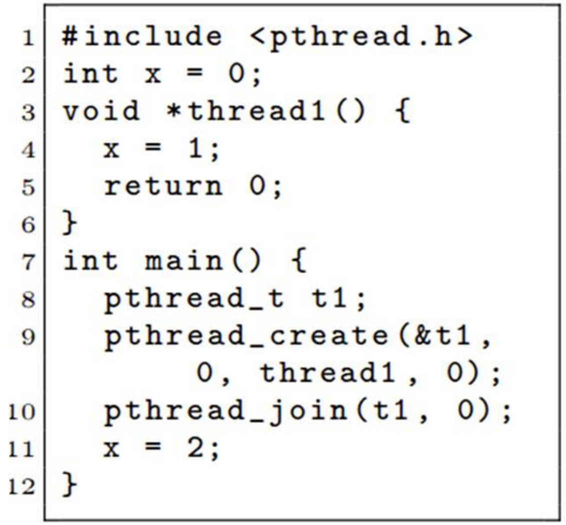
During every context switch, the scheduler

1. Check which thread **tc** is currently active
2. Tries to switch to a **newer** created thread **ti > tc**
3. If none are eligible, tries to continue with **current** thread **tc**
4. Otherwise, selects an **older** thread **ti < tc**

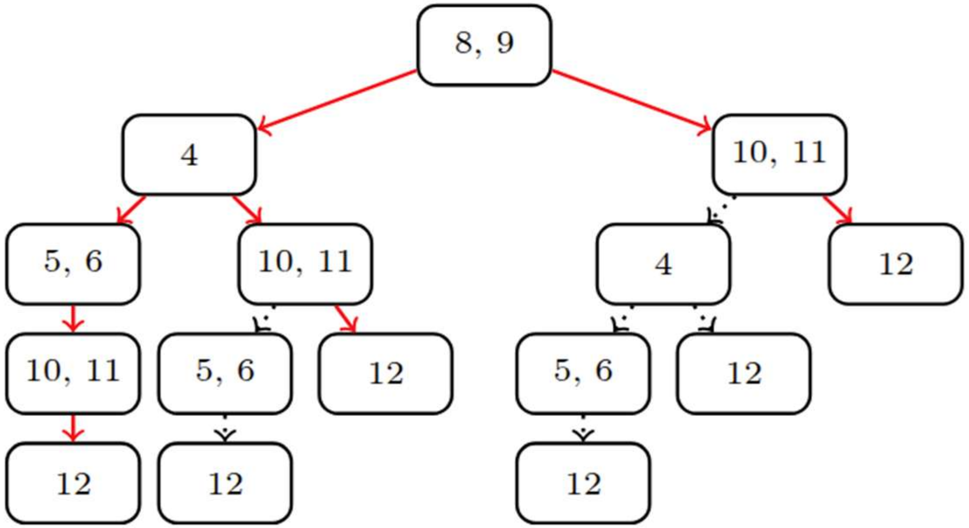
Previous ESBMC versions always prioritized t0

This prioritizes interleavings with **newly-created** threads, enabling ESBMC to explore new execution paths **earlier**, and find bugs **six times faster**.

Incremental SMT Solving



```
 1  #include <pthread.h>
 2  int x = 0;
 3  void *thread1() {
 4    x = 1;
 5    return 0;
 6  }
 7  int main() {
 8    pthread_t t1;
 9    pthread_create(&t1,
          0, thread1, 0);
10    pthread_join(t1, 0);
11    x = 2;
12  }
```



Default ESBMC behaviour

1. Call the SMT solver **once** it reaches the end of an interleaving
2. By that time, the interleaving may be long **unfeasible**

Incremental SMT mode

1. Call the SMT solver **repeatedly** after every assumption and state guard
2. Push & pop SMT interface **shares information** across calls

Removes an average of **53%** interleavings

Other Improvements

Partial order reductions

1. ESBMC removes equivalent interleavings with **optimal partial order reduction**
2. More accurate analysis of shared variables that are accessed by **pointers**
3. Reduce verification time for proving correctness by **40%**

MANCHESTER
1824
The University of Manchester

Other Improvements

Partial order reductions

1. ESBMC removes equivalent interleavings with **optimal partial order reduction**
2. More accurate analysis of shared variables that are accessed by **pointers**
3. Reduce verification time for proving correctness by **40%**

Data races:

1. Track memory via **memory addresses**, instead of variable names
2. Force a context switch during flag updates to **expose data races earlier**
3. Reduces the number of incorrect verdict by **9%**

MANCHESTER 1824
The University of Manchester

Other Improvements

Partial order reductions

1. ESBMC removes equivalent interleavings with **optimal partial order reduction**
2. More accurate analysis of shared variables that are accessed by **pointers**
3. Reduce verification time for proving correctness by **40%**

Data races:

1. Track memory via **memory addresses**, instead of variable names
2. Force a context switch during flag updates to **expose data races earlier**
3. Reduces the number of incorrect verdict by **9%**

Pthread operational models

1. We reduced the number of unnecessary context switches
2. Monitor only **user program variables** which are shared by **at least two threads**
3. Correctly solves 8% additional verification instances

Ablation Study

Individual Technique	Correct True	Correct False	Incorrect True	Incorrect False
Reverse Scheduling	**+66**	**+18**	+8	+7
Incremental SMT	-3	**+1**	+3	+6
POR	**+16**	-20	+15	**0**
Data Races	-5	**+16**	**-9**	**-14**
Pthread OM	**+31**	**+3**	+5	+3
All Techniques*	**+97**	**+21**	+13	+10

Compared against ESBMC v7.7 without the corresponding technique(s)

*except for incremental SMT, which is disabled in ESBMC v7.7 by default