

ULTIMATE AUTOMIZER

SV-COMP 2026

Manuel Bentele Max Barth Marcel Ebbinghaus Jan Körner Daniel Dietsch
Matthias Heizmann Dominik Klumpp Frank Schüssele Andreas Podelski

April 13, 2026



ULTIMATE

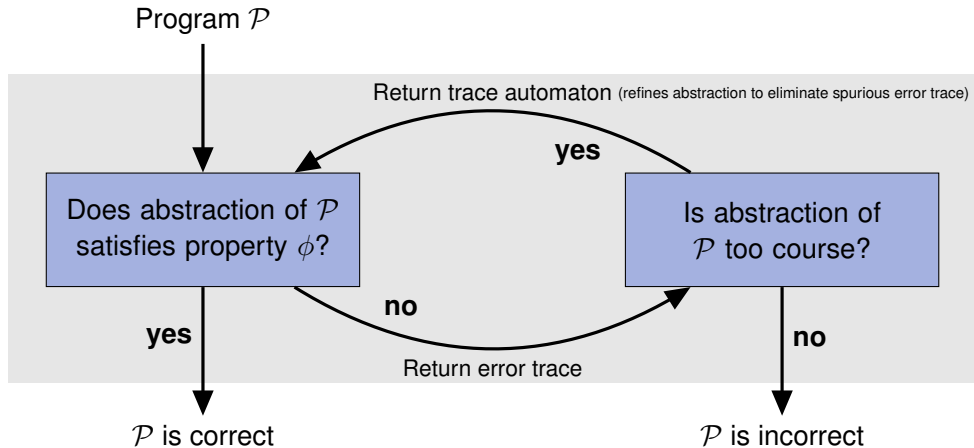


automata-based software verification

for **non-reachability**, **memory safety**, **termination**, **overflows**, and **race detection**.

Verification Approach

Software Model Checking via Trace Abstraction



Verification Challenge

Represent the Memory of a Program

```
static const uint32_t st_var = 0xEDODDCBA;

void main()
{
    const uint8_t st_arr[4] = {0xBA, 0xDC, 0x0D, 0xED};

    uint8_t *dy_arr = malloc(4 * sizeof(uint8_t));

    memcpy(dy_arr, st_arr, 4 * sizeof(uint8_t));

    for (uint8_t i = 0; i < 4; i++) {
        assert(dy_arr[i] == ((st_var >> (8 * i)) & 0xFF));
        printf("%02x", dy_arr[i]);
    }

    free(dy_arr);
}
```

■ Memory blocks

- Static allocated
- Dynamic allocated

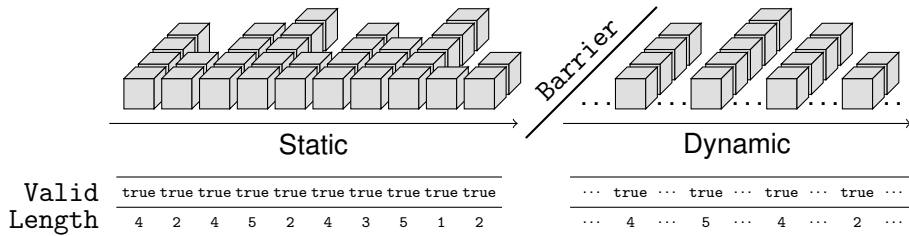
■ Memory operations

- Management
- Modification

Memory Modeling

How ULTIMATE AUTOMIZER Represents Memory

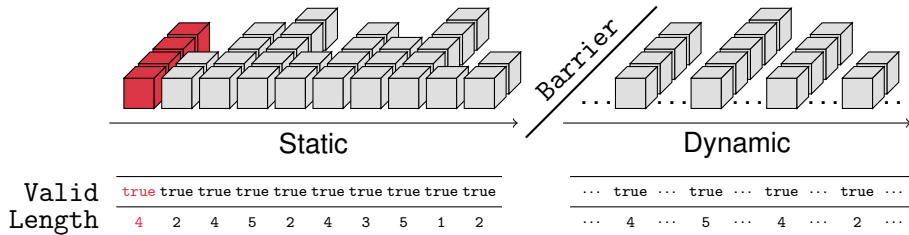
(Two-Dimensional Model)



Memory Modeling

How ULTIMATE AUTOMIZER Represents Memory

(Two-Dimensional Model)

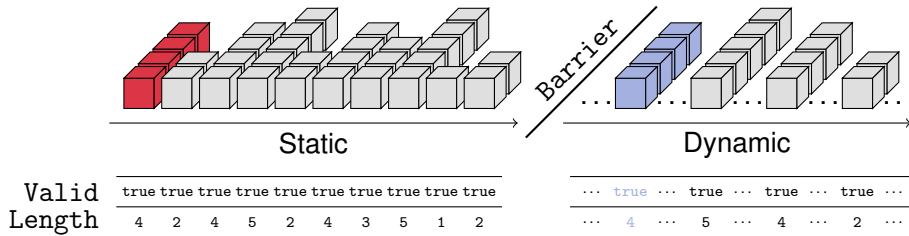


```
const uint8_t st_arr[4] = {0xBA, 0xDC, 0x0D, 0xED};
```

Memory Modeling

How ULTIMATE AUTOMIZER Represents Memory

(Two-Dimensional Model)

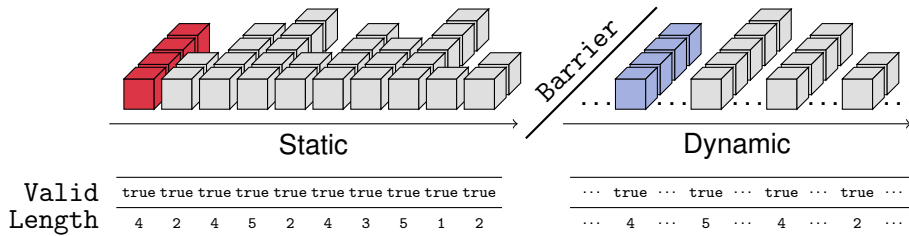


```
uint8_t *dy_arr = malloc(4 * sizeof(uint8_t));
```

Memory Modeling

How ULTIMATE AUTOMIZER Represents Memory

(Two-Dimensional Model)

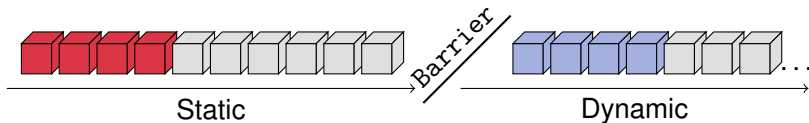


Such a complex model for reachability analysis?

Recent Feature

Introduction of a Simplified Memory Model

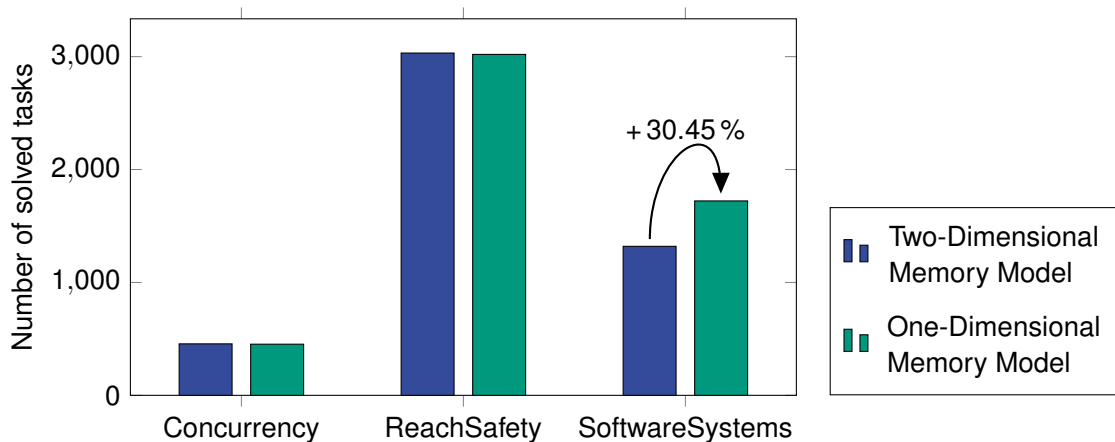
(One-Dimensional Model)



Performance Evaluation

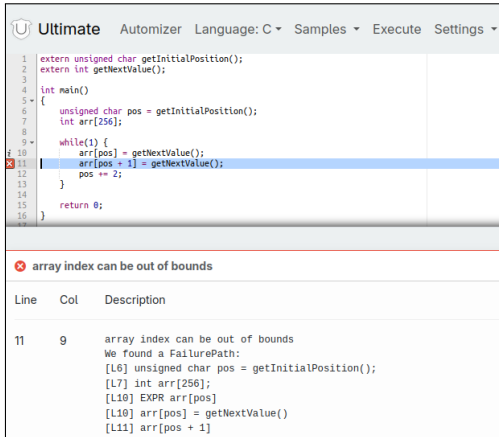
Comparison of One-Dimensional Memory Model

(SV-COMP 2026 Benchmarks)



Explore the Tool

Web Interface of ULTIMATE AUTOMIZER



The screenshot displays the Ultimate Automizer web interface. At the top, there is a navigation bar with the 'Ultimate' logo and menu items: 'Automizer', 'Language: C', 'Samples', 'Execute', and 'Settings'. Below the navigation bar is a code editor showing a C program. The code is as follows:

```
1 extern unsigned char getInitialPosition();
2 extern int getNextValue();
3
4 int main()
5 {
6     unsigned char pos = getInitialPosition();
7     int arr[256];
8
9     while(1) {
10         arr[pos] = getNextValue();
11         arr[pos + 1] = getNextValue();
12         pos += 2;
13     }
14
15     return 0;
16 }
```

Line 11 is highlighted in blue. Below the code editor, a red error message is displayed: 'array index can be out of bounds'. Below the error message is a table with the following columns: 'Line', 'Col', and 'Description'.

Line	Col	Description
11	9	array index can be out of bounds We found a FailurePath: [L6] unsigned char pos = getInitialPosition(); [L7] int arr[256]; [L10] EXPR arr[pos] [L10] arr[pos] = getNextValue() [L11] arr[pos + 1]



<https://ultimate-pa.org>