The VerCors Verifier

a Progress Report

Highlights:

- Supports **many** input languages
  - Extensions & new ones!
  - Parallel & concurrent
- **Architecture** improvements
  - Easy to use & extend
- Many case studies (see paper)

Open for collaborations!

utwente.nl/vercors

Lukas Armborst[1], Pieter Bos[1], **Lars B. van den Haak**[2], Marieke Huisman[1], Robert Rubbens[1], Ömer Şakar[1], Philip Tasche[1]

[1]University of Twente, The Netherlands
[2]Eindhoven University of Technology, The Netherlands

SCAN ME

Why verify Parallel & Concurrent software?

Subtle **bugs**

- E.g. Data races

Hard to **verify**

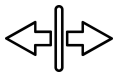
Parallel programs needed for **performance**!

VerCors

Supports many languages:

- PVL
- Improved:
  - Java, C/C++
- Added:
  - OpenCL, CUDA, SYCL, Halide, SystemC, Java BIP, LLVM,

Prototypal Verification Language



Based on Separation Logic

Focus on **practicality**

- Verify program *as-is*
  - Add annotations

Pallas

Mercedes

MAXIMAL RELIABILITY OF CONCURRENT AND DISTRIBUTED SOFTWARE

CHEOPS

VERIFIED CONSTRUCTION OF CORRECT AND OPTIMISED PARALLEL SOFTWARE

Example: Verification of Parallel Add

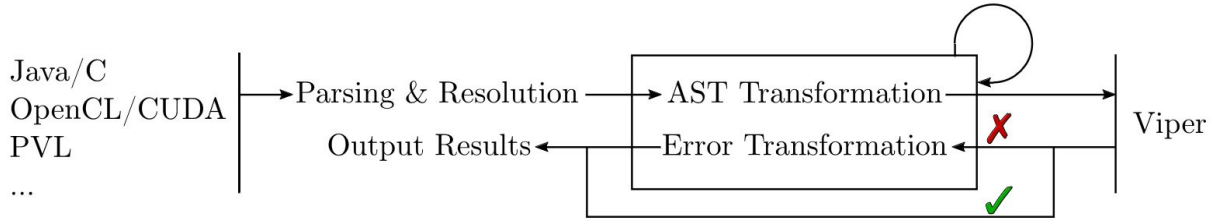
```
1  context_everywhere a != null && a.length == n;
2  context_everywhere b != null && b.length == n;
3  context_everywhere c != null && c.length == n;
4  context (∀* int i=0..n; Perm(a[i], write));
5  context (∀* int i=0..n; Perm(b[i], read));
6  context (∀* int i=0..n; Perm(c[i], read));
7  ensures (∀ int i=0..n; a[i] == b[i] + c[i]);
8  decreases;
9  void add(int[] a, int[] b, int[] c, int n){
10 par(int tid=0..n)
11   context Perm(a[tid], write);
12   context Perm(b[tid], read);
13   context Perm(c[tid], read);
14   ensures a[tid] == b[tid] + c[tid];
15 {
16   a[tid] = b[tid] + c[tid];
17 }
18 }
```

Parallel block

Annotations:

- Well formedness
  - No out of bounds accesses
- Permissions
  - No data races
- Functional correctness
- Termination

Internal structure & Improvements



- Easy to extend:
  - **Decomposed** in 80 steps
  - **Desugar** to intermediate AST
  - Allows **reuse** for all languages
    - Easy to add new languages
- Clear progress of Verification Goal

- Build on **Viper** (*ETH Zürich*)
- Precise error reporting
  - Link **verification failure** to **source code**

Example errors: No write permission

```
========================================
At arrays.pvl
----------------------------------------
   13    context Perm(c[tid], read);
   14    ensures a[tid] == b[tid] + c[tid]; {
         [------------------------
   15    a[tid] = b[tid] + c[tid];
         ------------------------]
   16  }}
   17
----------------------------------------
Insufficient permission to assign to field.
(https://utwente.nl/vercors#assignFieldFailed)
========================================
```

Example errors: Incorrect specification/implementation

```
---------------------------------------
   12    context Perm(b[tid], read);
   13    context Perm(c[tid], read);
                 [-------------------------
   14    ensures a[tid] == b[tid] + c[tid];
                 -------------------------]
   15    {
   16    a[tid] = b[tid] + c[tid] + 1;
---------------------------------------
[2/2] ... this expression may be false
(https://utwente.nl/vercors#parPostFailed:false)
```

Improved support for GPUs

```
1  /*@
2   ...
3   context get_global_size(0)==n && get_local_size(0)==32;
4   context a != NULL && \pointer_length(a) == n;
5   context b != NULL && \pointer_length(b) == n;
6   context c != NULL && \pointer_length(c) == n;
7   requires Perm(&bs[get_local_id(0)], write);
8   context Perm(&a[\gtid], write);
9   context Perm(&b[\gtid], read);
10  context Perm(&c[\gtid], read);
11  ensures a[\gtid] == b[\gtid] + c[\gtid];
12  decreases;
13 @*/
14 __kernel void add(global int a[],
15   global int b[], global int c[], int n) {
16  int gtid = get_global_id(0), ltid = get_local_id(0);
17  __local int bs[32];
18  bs[ltid] = b[gtid];
19  a[gtid] = bs[gtid] + c[gtid];
20 }
```

Specify shared memory per thread block

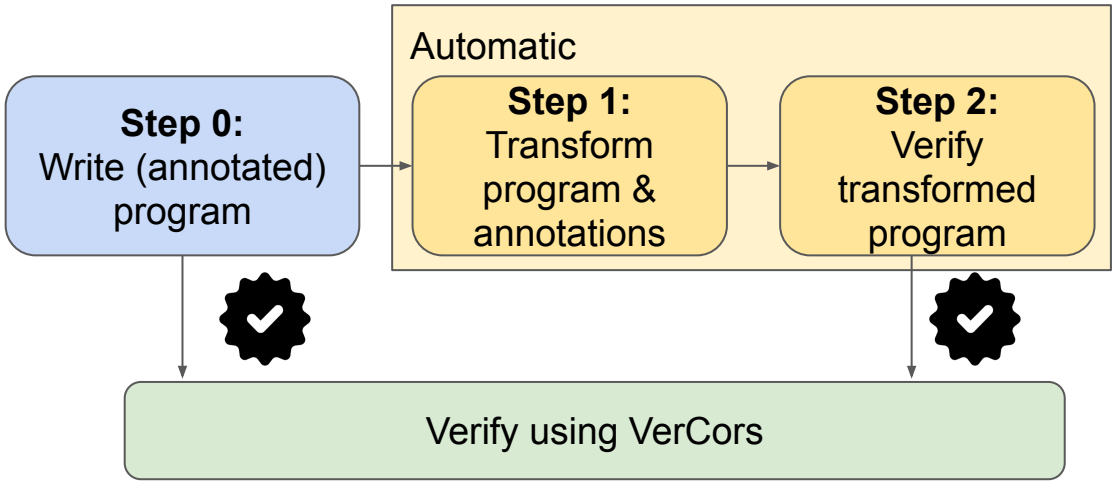
Specify contract per thread

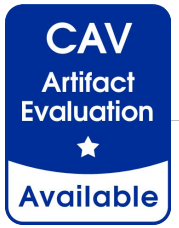
Implicit nested parallel block

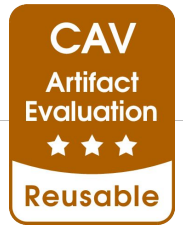
Store in shared memory

Annotation aware program transformers

- **VeyMont**
  - Restricted sequential program
  - Parallelise program
- **Alpinist**
  - Optimize GPU programs
  - Specify optimizations







The VerCors Verifier

a Progress Report

Highlights:

- Supports **many** input languages
  - Improvements & new ones!
  - Parallel & concurrent
- **Architecture** improvements
  - Easy to use & extend
- Many case studies (see paper)

Open for collaborations!

utwente.nl/vercors

Lukas Armborst[1], Pieter Bos[1], **Lars B. van den Haak**[2], Marieke Huisman[1], Robert Rubbens[1], Ömer Şakar[1], Philip Tasche[1]

[1]University of Twente, The Netherlands
[2]Eindhoven University of Technology, The Netherlands

SCAN ME

Backup slide - Complete GPU example

```
1  /*@
2   context get_local_size(1) == 1 && get_local_size(2) == 1;
3   context get_num_groups(1) == 1 && get_num_groups(2) == 1;
4   context get_global_size(0)==n && get_local_size(0)==32;
5   context a != NULL && \pointer_length(a) == n;
6   context b != NULL && \pointer_length(b) == n;
7   context c != NULL && \pointer_length(c) == n;
8   requires Perm(&bs[get_local_id(0)], write);
9   context Perm(&a[\gtid], write);
10  context Perm(&b[\gtid], read);
11  context Perm(&c[\gtid], read);
12  ensures a[\gtid] == b[\gtid] + c[\gtid];
13  decreases;
14 @*/
15 __kernel void add(global int a[],
16   global int b[], global int c[], int n) {
17  int gtid = get_global_id(0), ltid = get_local_id(0);
18  __local int bs[32];
19  bs[ltid] = b[gtid];
20  a[gtid] = bs[ltid] + c[gtid];
21 }
22
```

Backup slide - Encoded GPU example

```
1   ...
2   ensures (∀ int i=0..n; a[i] == b[i] + c[i]);
3 void add(int[] a, int[] b, int[] c, int n, int blockSize, int gridSize){
4  par(int blockId=0..gridSize)
5    context (∀* int i=0..blockSize; Perm(a[blockId*32 + i], write));
6    context (∀* int i=0..blockSize; Perm(b[blockId*32 + i], read));
7    context (∀* int i=0..blockSize; Perm(c[blockId*32 + i], read));
8    ensures (∀* int i=0..blockSize; a[blockId*32 + i] == b[blockId*32+i]+c[blockId*32+i]);{
9    int[32] bs = new int[32];
10   par(int threadId=0..blockSize)
11     context Perm(bs[threadId], write);
12     context Perm(a[blockId*32 + threadId], write);
13     context Perm(b[blockId*32 + threadId], read);
14     context Perm(c[blockId*32 + threadId], read);
15     ensures a[blockId*32 + threadId] == b[blockId*32+threadId] + c[blockId*32+threadId];{
16     int gtid = blockId* 32 + threadId;
17     int ltid = threadId;
18     bs[ltid] = b[gtid];
19     a[gtid] = bs[ltid] + c[gtid];
20 } } }
21
```

Example errors: Violated array bounds

```
=====================================
At arrays.pvl
-------------------------------------
    9  void add(int[] a, int[] b, int[] c, int n){
   10    par(int tid=0..n+1)
               [--------------------
   11    context Perm(a[tid], write);
               --------------------]
   12    context Perm(b[tid], read);
   13    context Perm(c[tid], read);
-------------------------------------
Index may be negative, or exceed the length of the array.
(https://utwente.nl/vercors#arrayBounds)
=====================================
```