Control Explicit—Data Symbolic Model Checking

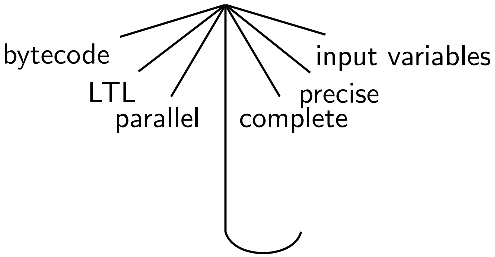
Petr Bauch

POPL Student Session

23 January 2013

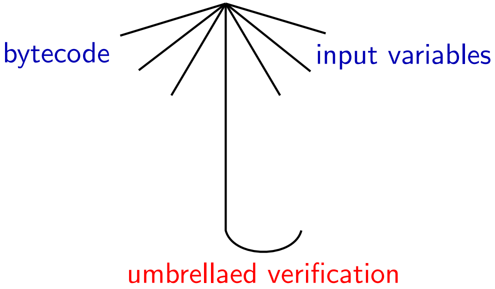
Introducing

complete and precise verification of parallel software
against temporal specification



umbrellaed verification

Real Code



Real Code

C

```
int a, b;
cin >> a >> b;
if ( a > 3*b )
  a = a*b;
```

automated

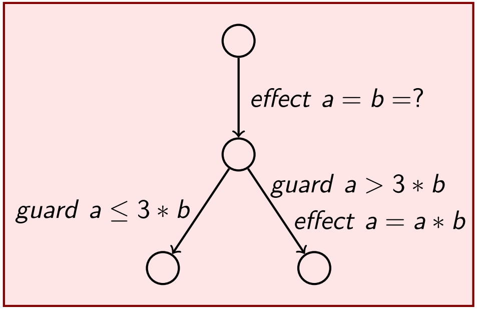
LLVM

```
%6 = load i32* %a, align 4
%7 = load i32* %b, align 4
%8 = mul nsw i32 3, %7
%9 = icmp sgt i32 %6, %8
br i1 %9, label %10, label %14
```

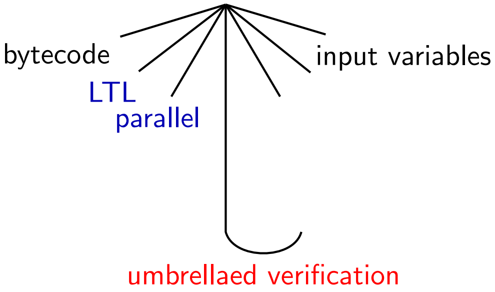
- no modelling
- input variables
- Peano arithmetic

manual

DVE



Parallel Programs



Parallel Programs

Thread 1

```
global a, b;
if ( a > 3*b )
   a = a*b;
```

Thread 2

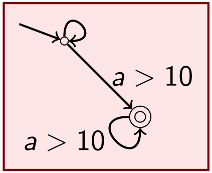
```
global a, b;
a = a+b;
while ( a > 3 )
   b--;
```

Thread 3

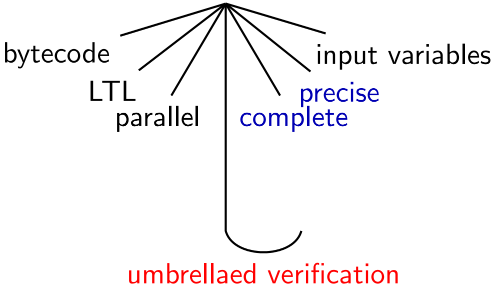
```
global a, b;
while ( true )
   b = b+a;
```

- thread interleaving
- temporal specification
- Linear Temporal Logic

$FG(a > 10)$



Complete and Precise



Two Sources of Nondeterminism

Thread 1

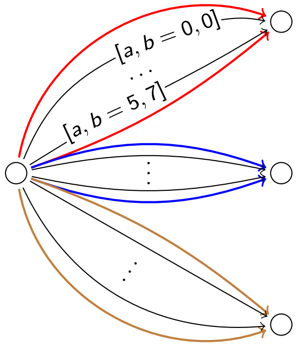
```
global a, b;
if ( a > 3*b )
   a = a*b;
```

Thread 2

```
global a, b;
a = a+b;
while ( a > 3 )
   b--;
```

Thread 3

```
global a, b;
while ( true )
   b = b+a;
```



1. control flow
2. data flow

Two Model Checking Strategies

Explicit

- states stored explicitly
- set of visited states
- parallel processing
- distributed storage

vs.

Symbolic

- one symbolic representation
- fix point computation
- BDDs, SAT, SMT

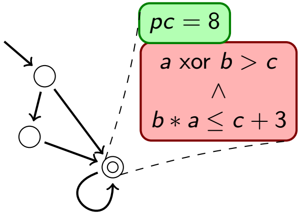
Better Together

Control Explicit ——— Data Symbolic

Model Checking

DiVinE

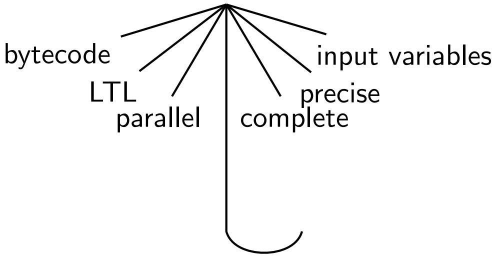
- store states explicitly
- parallelise computation



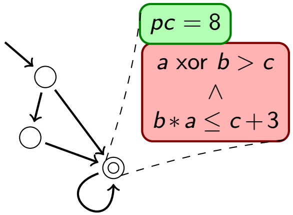
$\mathcal{BV}$ solver

- successor computation
- state matching

Closing Remarks



umbrellaed verification



Control Explicit—Data Symbolic
Model Checking