1/2DIVINE + DIVINE = 1 and 1/4DIVINE

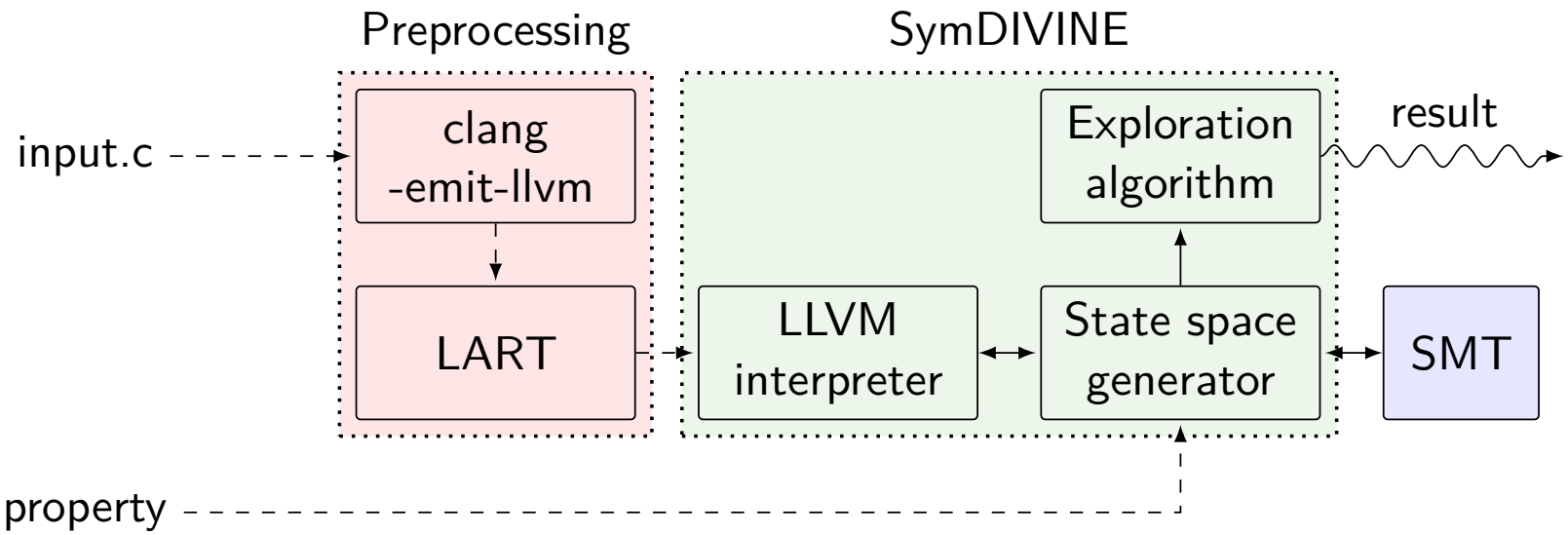
Henrich Lauko

Masaryk University

Brno, Czech Republic

April 28, 2016

Recapitulation SymDIVINE



State space generation example SymDIVINE

Recapitulation DIVINE

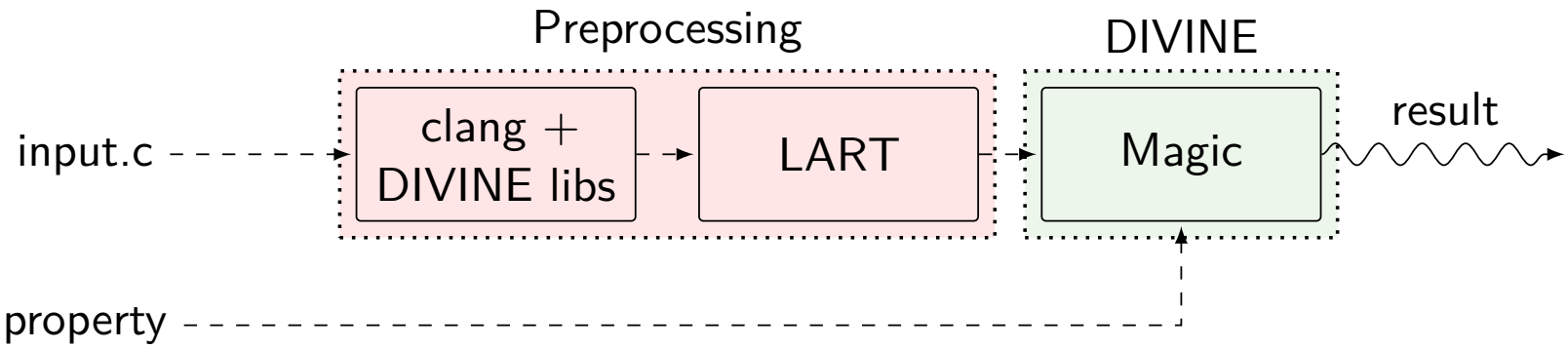
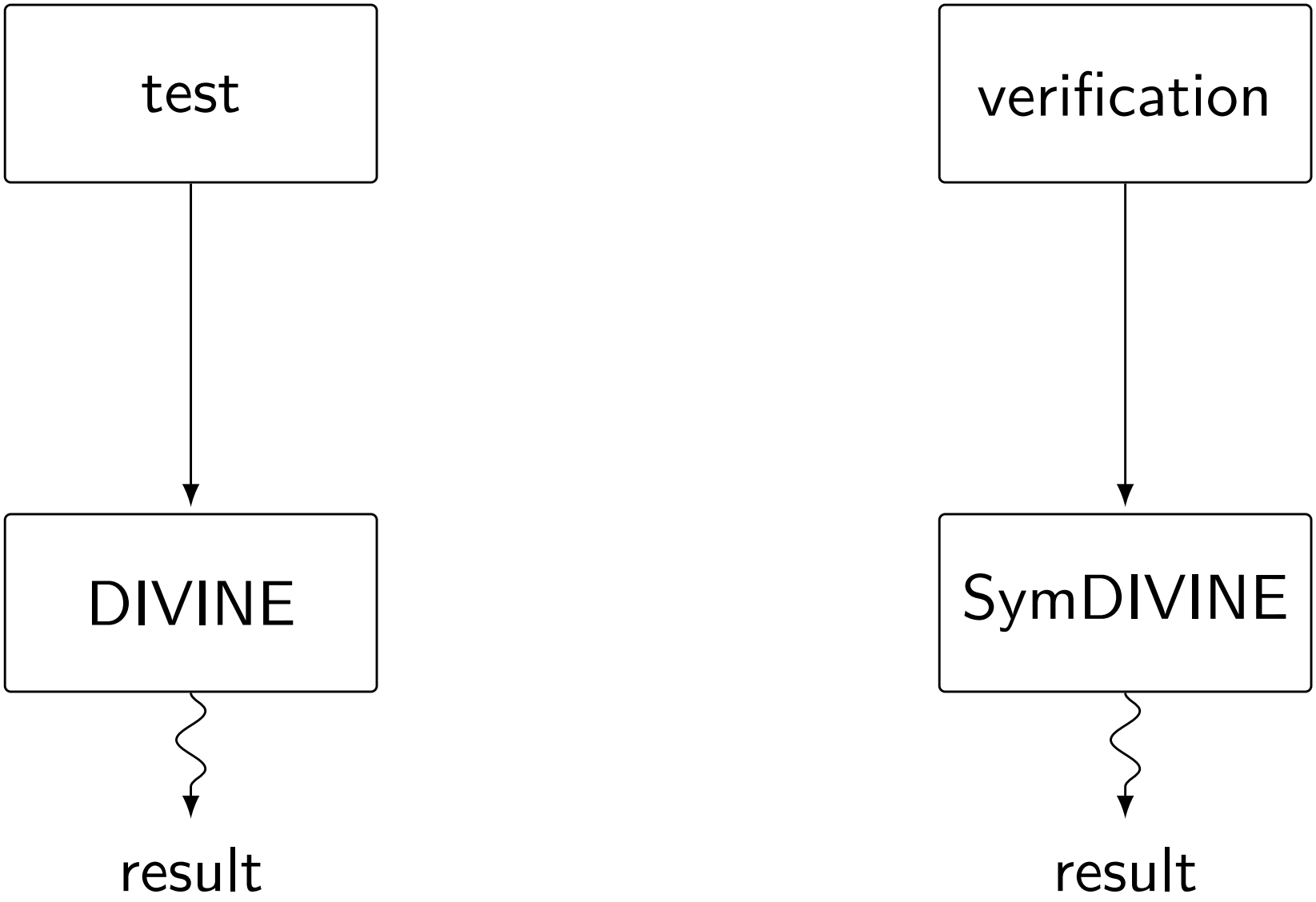
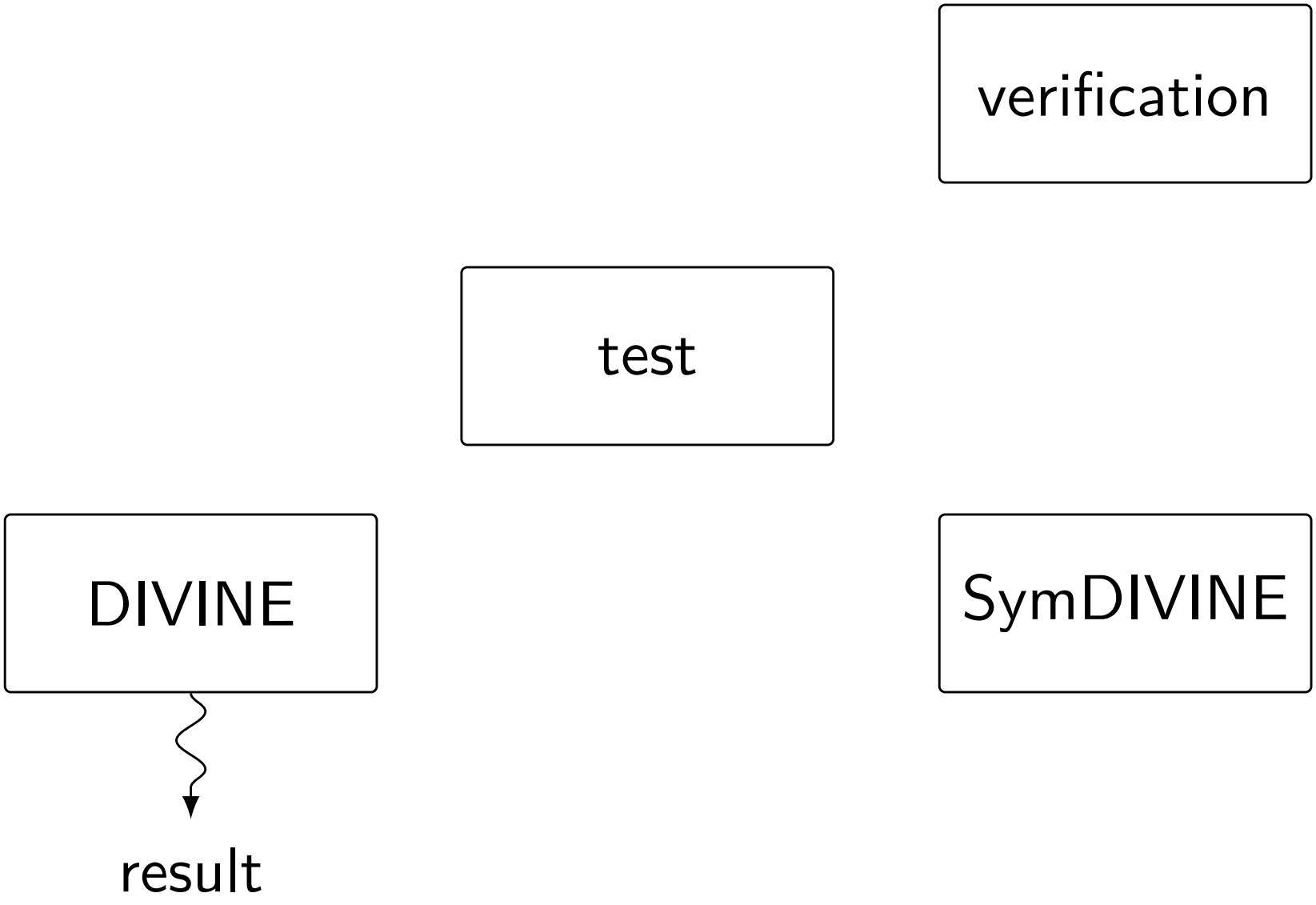


Figure 1: Mornfall

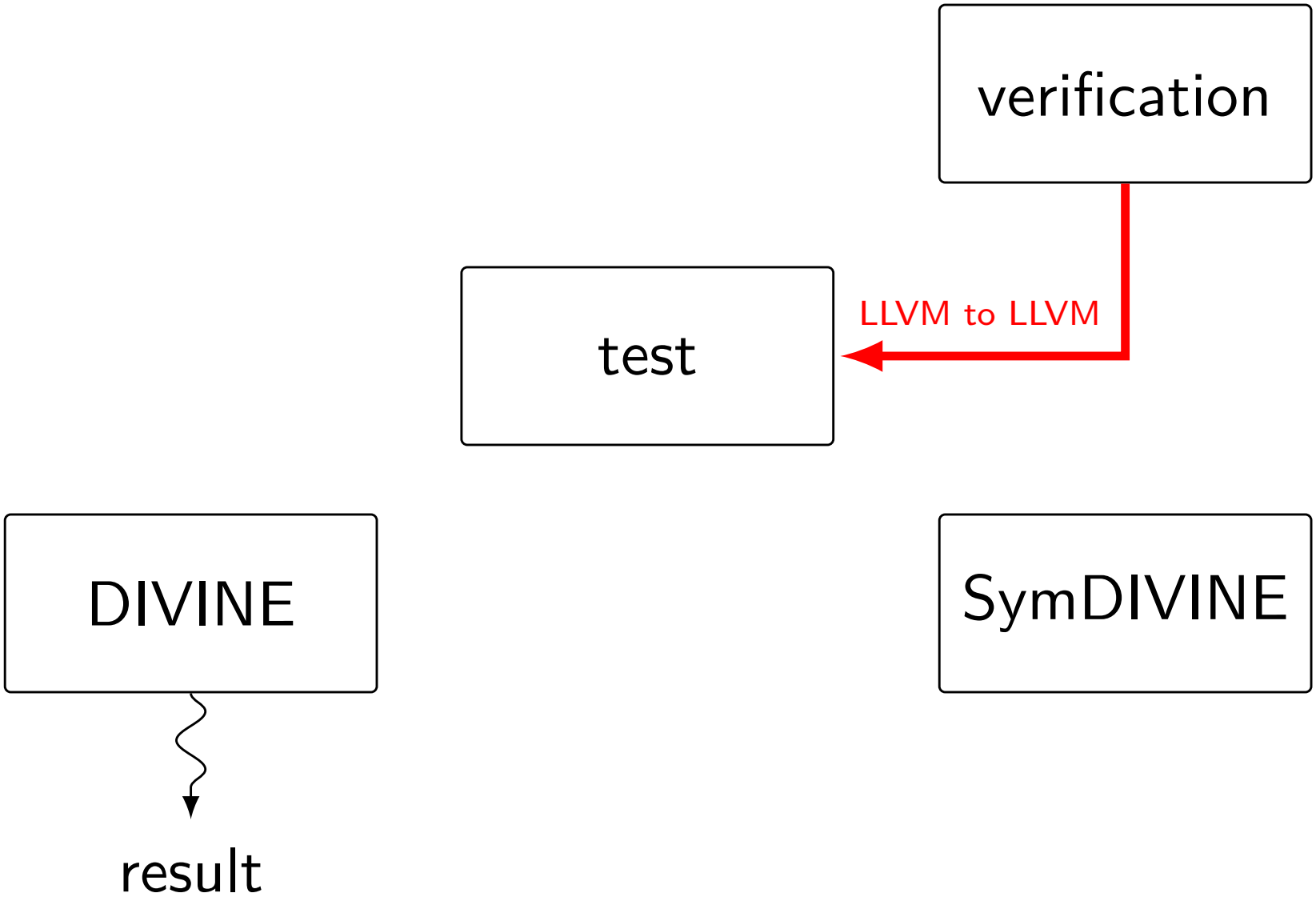
Big idea



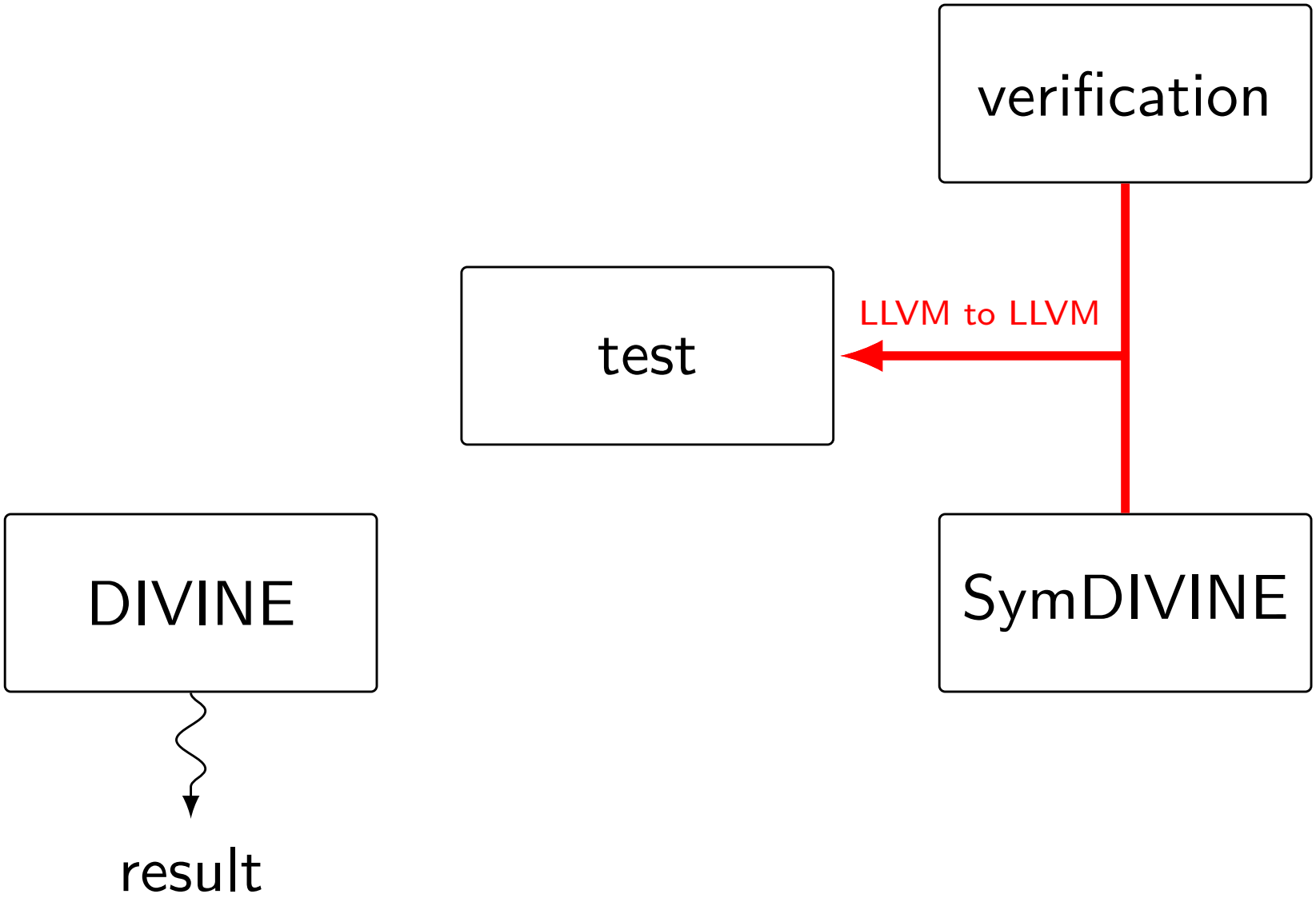
Big idea



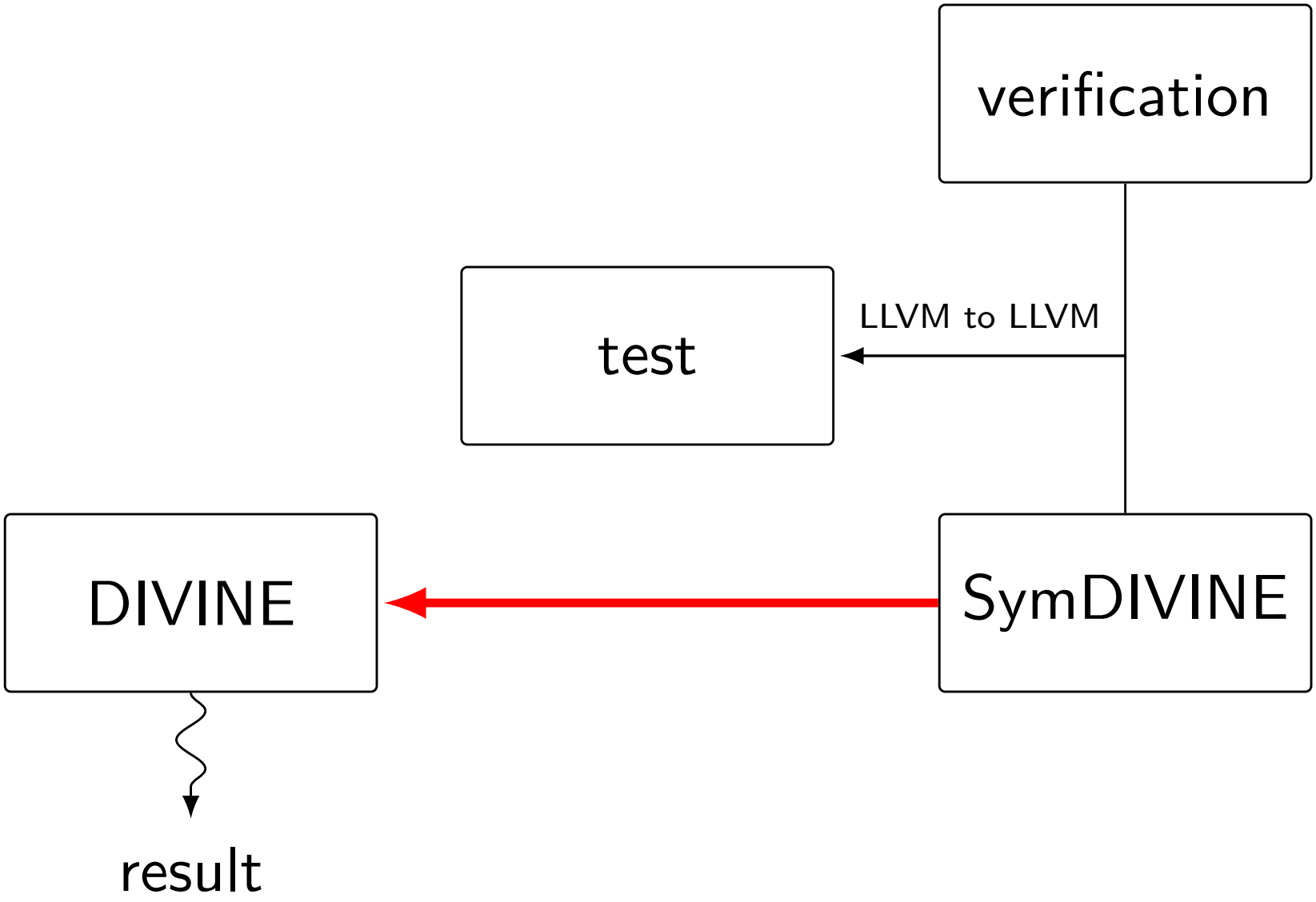
Big idea



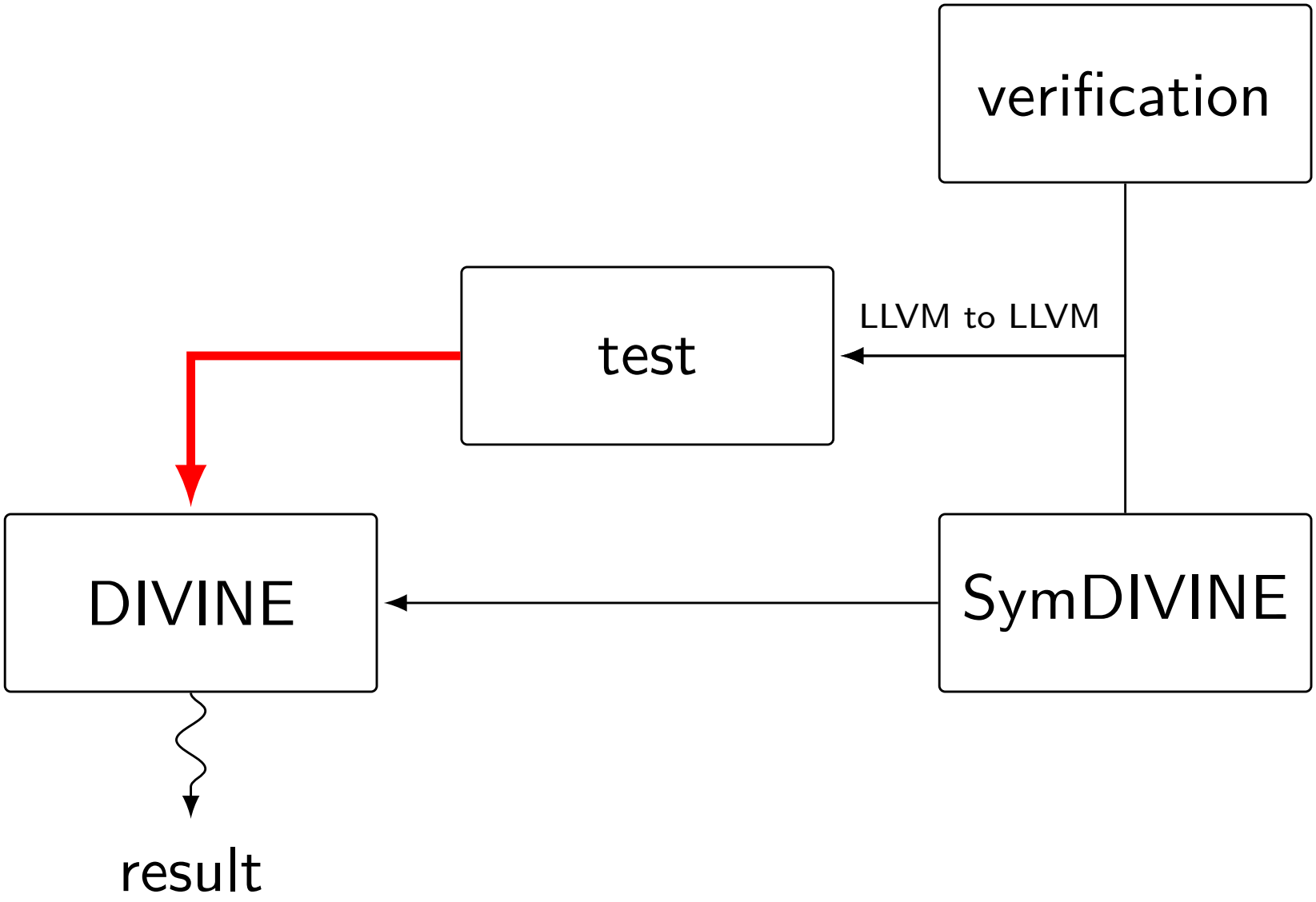
Big idea



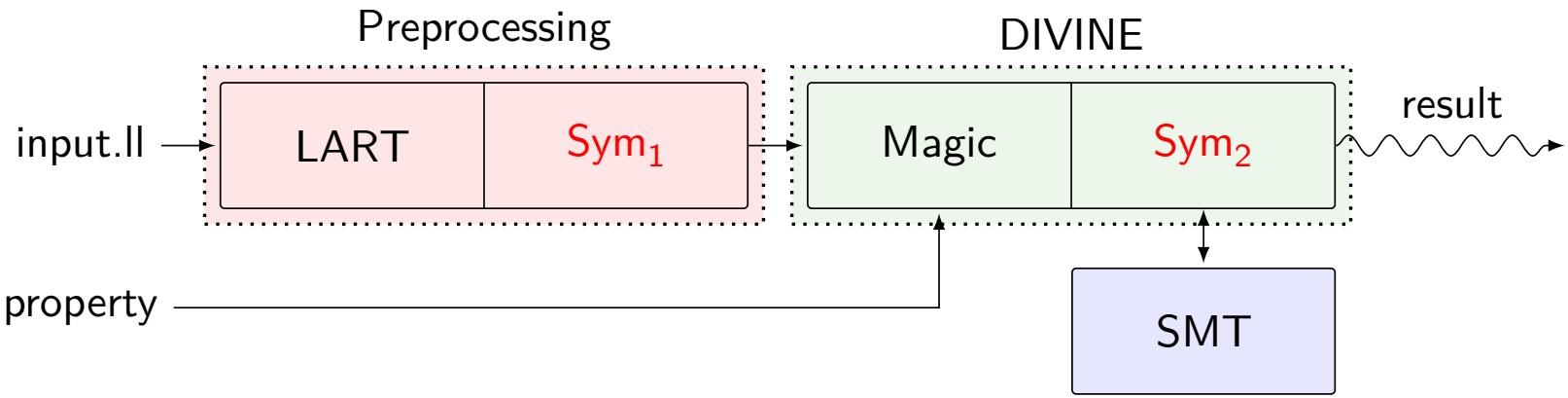
Big idea



Big idea

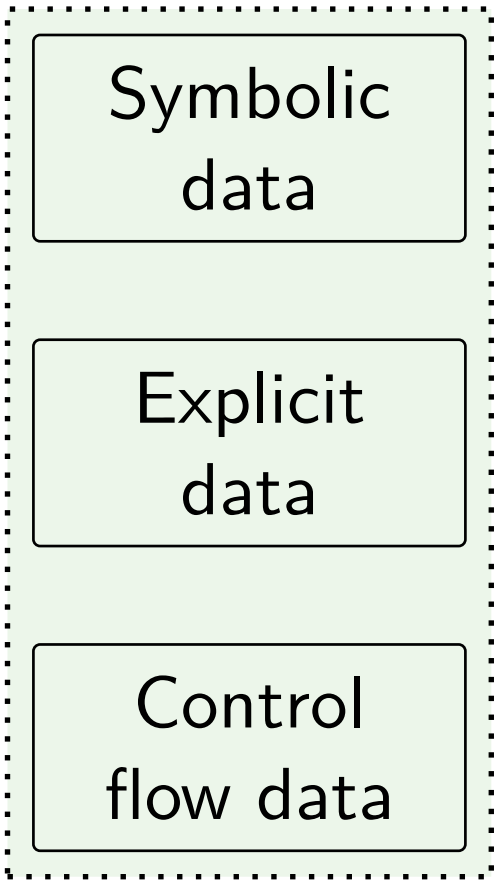


Integration

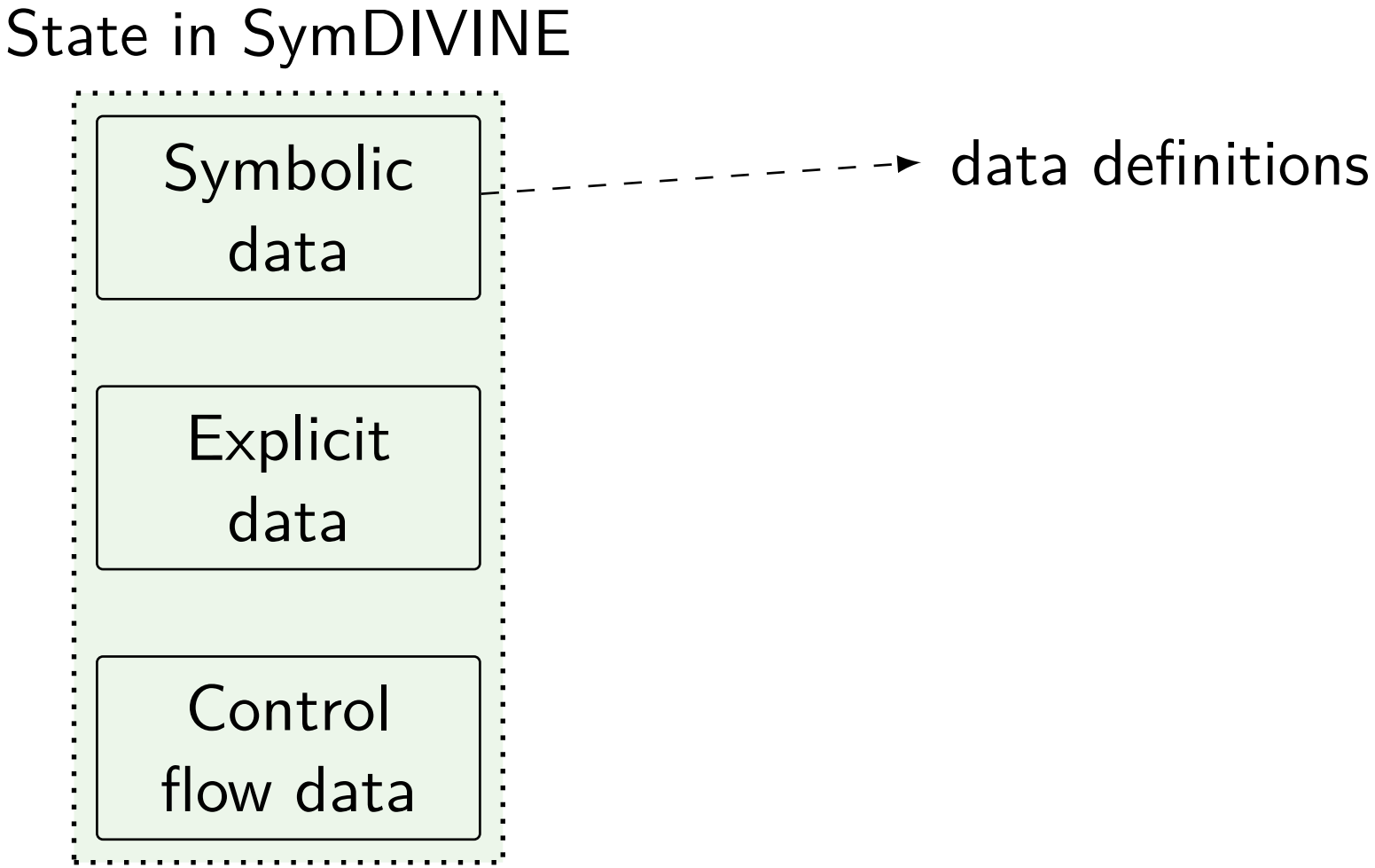


Data representation (Sym_1)

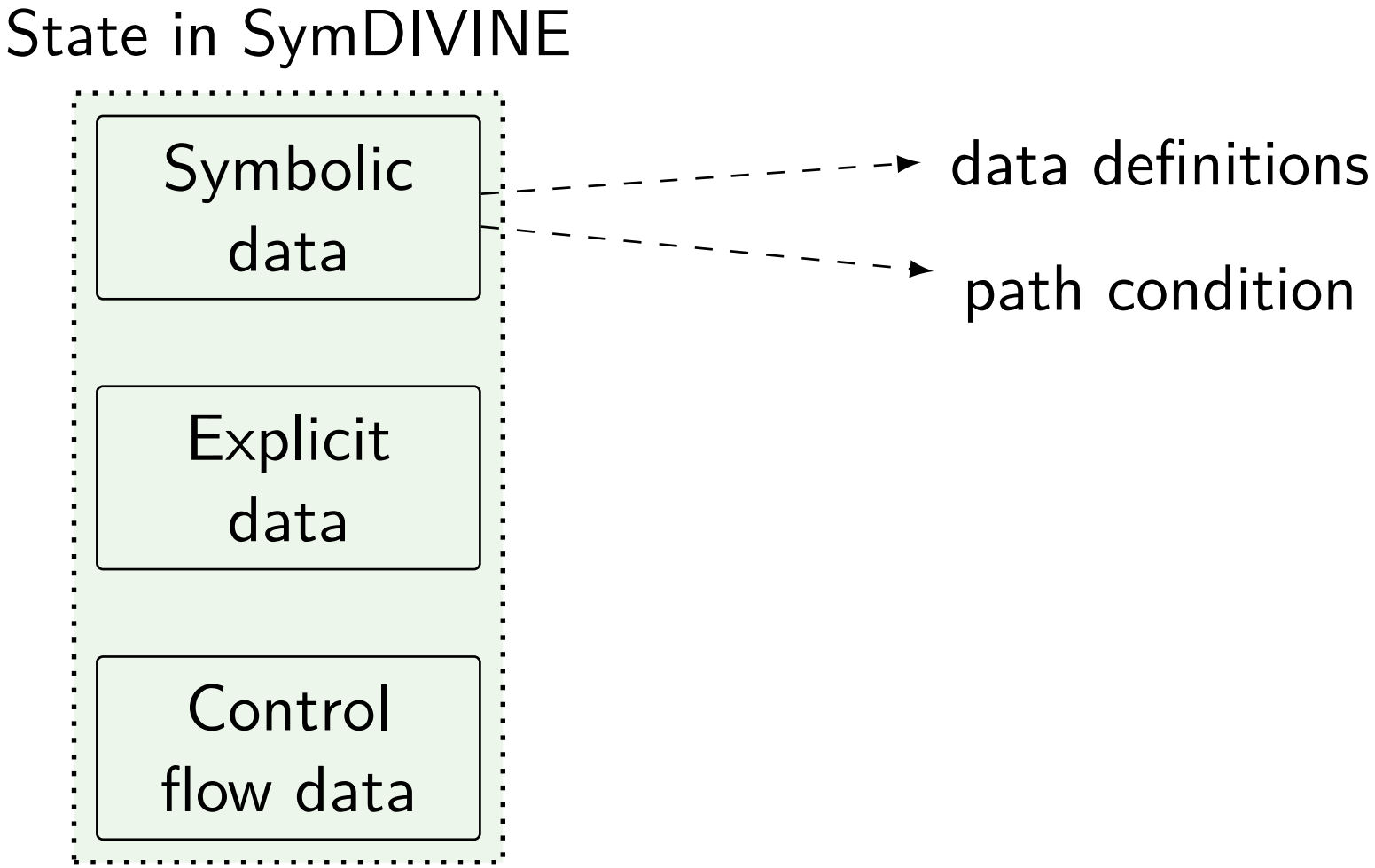
State in SymDIVINE



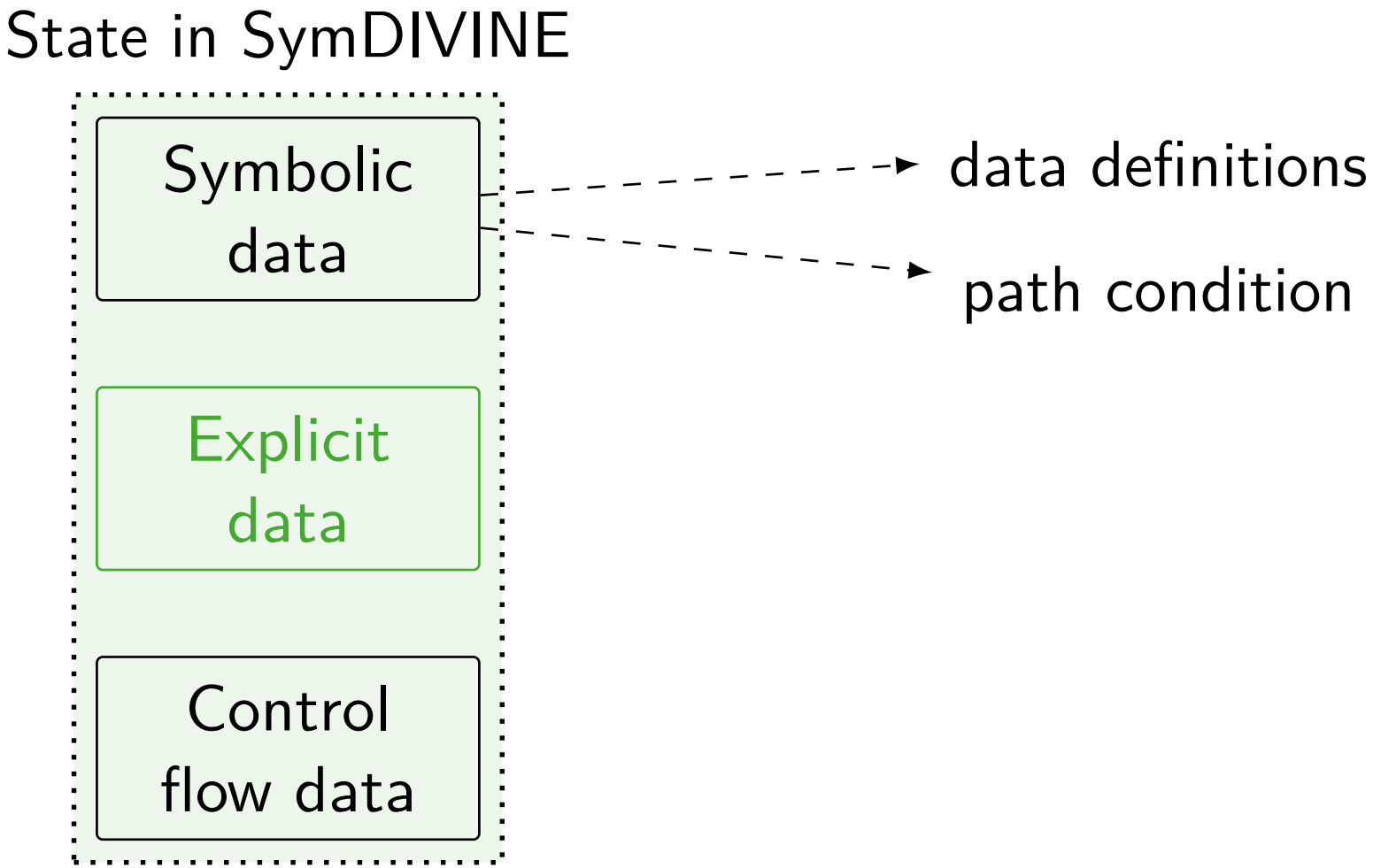
Data representation (Sym_1)



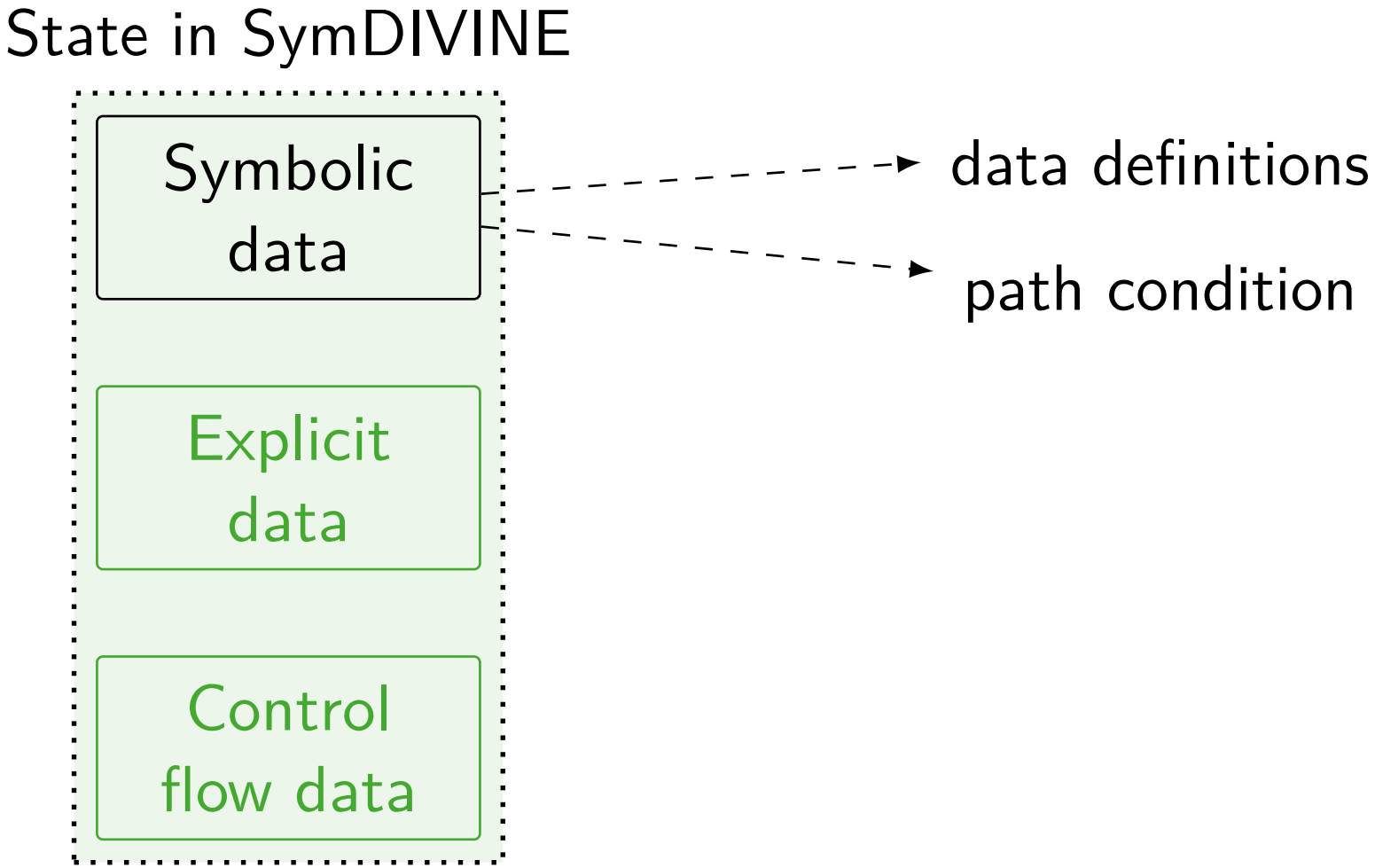
Data representation (Sym_1)



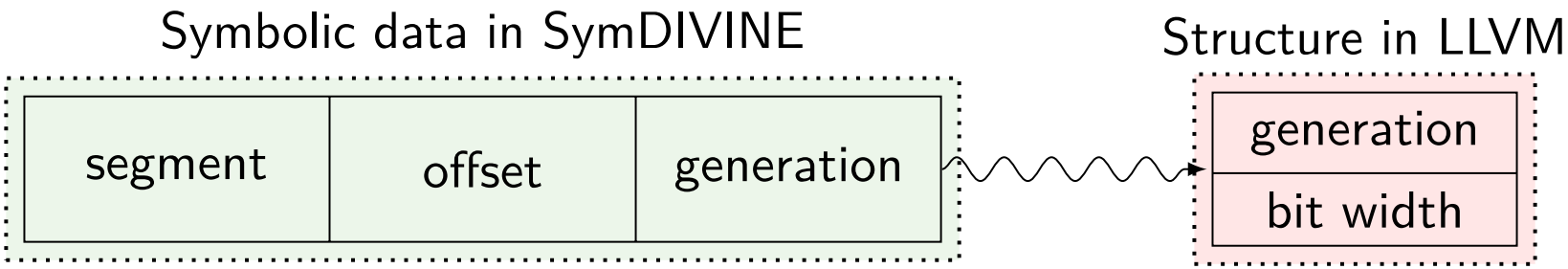
Data representation (Sym_1)



Data representation (Sym_1)



Data representation (Sym_1)



- segment and offset represented as address of structure
- using bit width in SMT solver
- propagation in AST

Data representation example (Sym_1)

```
1 int useless_function(int x, int y) {
2     int z = x + 1;
3     if (x < y) {
4         return y - z;
5     } else {
6         return y;
7     }
8 }
```

...

```
1 int x = __VERIFIER_nondet_int();
2 int y = 1;
3 int res = useless_function(x, y);
```

Data representation example (Sym_1)

```
declarations = {z = x + 1, nd_y = 1};
path_condition = {true};
```

```
nondet_int useless_function(nondet_int x, int y) {
    nondet_int z = x + 1;
    nondet_int nd_y = y; //exlicit value
    if ( x < nd_y) {
        return nd_y - z ;
    } else {
        return nd_y;
    }
}
```

```
nondet_int x;
int y = 1;
nondet_int res = useless_function(x, y);
```

- data replacement and propagation in AST

Data representation example (Sym_1)

```
declarations = {z = x + 1, nd_y = 1};
path_condition = {true};
```

```
nondet_int useless_function(nondet_int x, int y) {
    nondet_int z = plus(x, 1);
    nondet_int nd_y = y; //exlicit value
    if (less(x,nd_y)) {
        return minus(nd_y,z);
    } else {
        return nd_y;
    }
}
```

```
nondet_int x;
int y = 1;
nondet_int res = useless_function(x, y);
```

- operations on nondeterministic data and function duplication

Control representation example (Sym_1)

```
declarations = {z = x + 1, nd_y = 1};
path_condition = {x < nd_y};
```

```
nondet_int useless_function(nondet_int x, int y) {
    nondet_int z = plus(x, 1);
    nondet_int nd_y = y; //exlicit value
    bool choice = __divine_choice(2);
    if (choice) {
        change_pc(less(x, nd_y));
        nondet_int ret = minus(nd_y,z);
        cleanup(ret);
        return ret;
    } else {
        change_pc(not less(x, nd_y));
        nondet_int ret = nd_y;
        cleanup(ret);
        return ret;
```

Function duplication example (Sym_1)

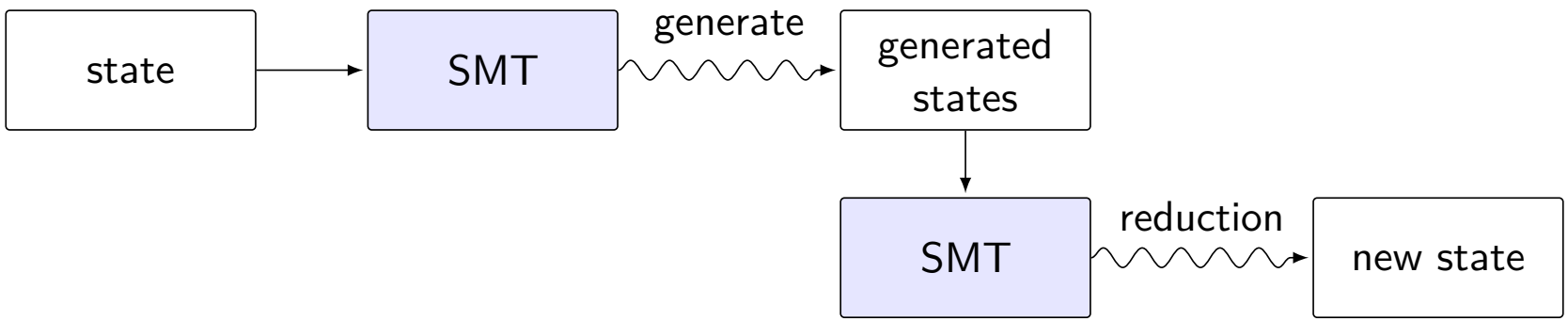
```
1 int function(int x, int y) { ... }
```

```
1 nondet_int function(nondet_int x, int y) { ... }
```

```
1 int function(nondet_int x, nondet_int y) { ... }
```

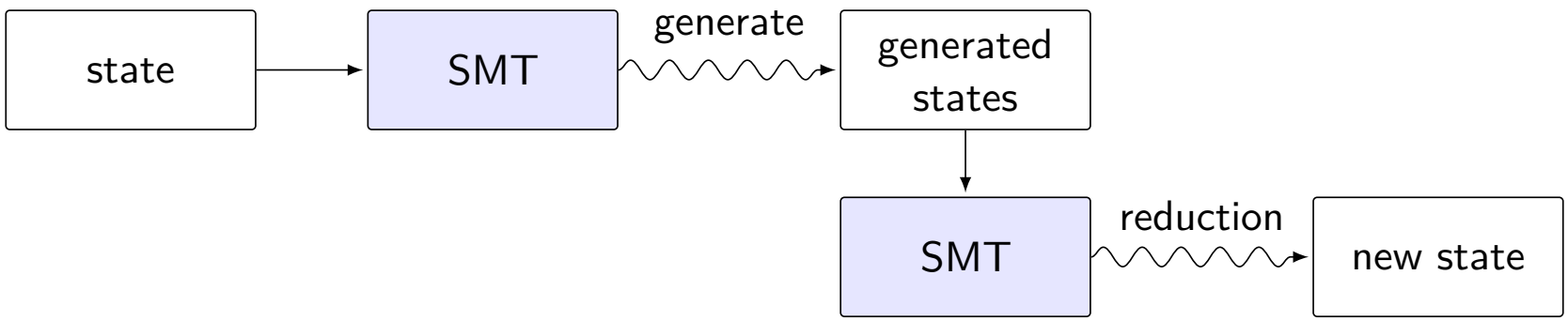
```
1 int x = __VERIFIER_nondet_int();
2 int y = 1, z = 0;
3
4 int res1 = function(x, y);
5 int res2 = function(y, z);
```

DIVINE state space generation Sym_2



- Before generating – checking path condition.

DIVINE state space generation Sym_2

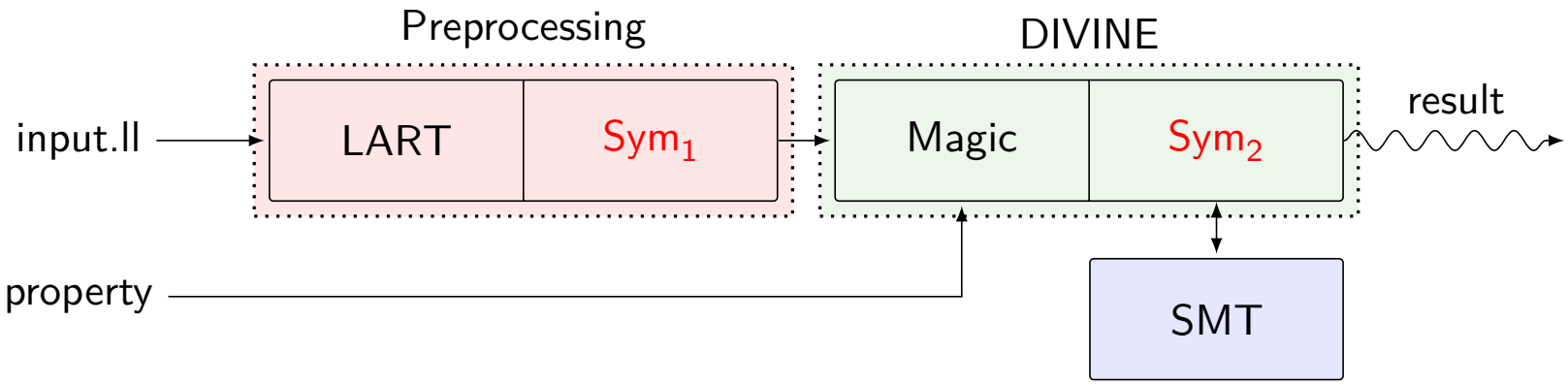


- Checking equality of symbolic part.

DIVINE keeping states Sym_2

- hashing explicit part
- linearly chaining symbolic parts to hash table position

Summary


Preprocessing
DIVINE
input.ll
LART
Sym1
Magic
Sym2
result
property
SMT

Questions and Lunch!