To-Be-Recorded Analysis In Clad. Summary

Petro Zarytskyi

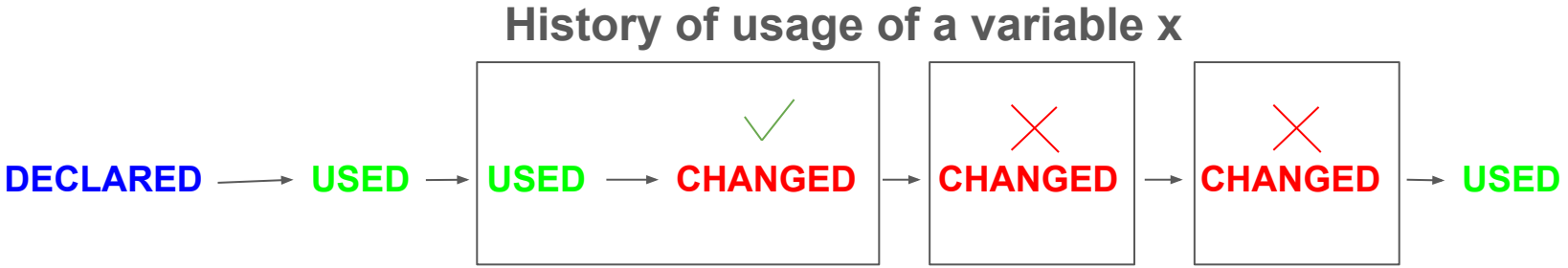
Mentors: Vassil Vassilev, David Lange

A quick reminder of how TBR analysis works

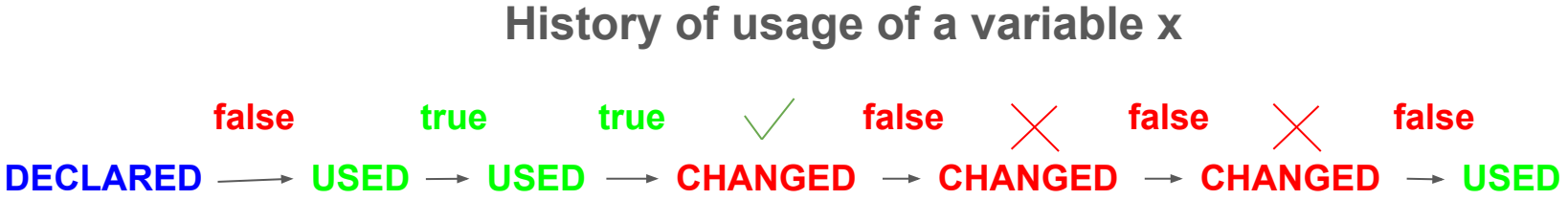
History of usage of a variable x

DECLARED → USED → USED → CHANGED → CHANGED → CHANGED → USED

A quick reminder of how TBR analysis works



A quick reminder of how TBR analysis works



Overview

Modes	used for analysing expressions and finding used variables (data-flow)
VarData	stores the information about one variable
VarDatas graph	used to handle control-flow

Modes

marking mode

y;

no variables are changed, therefore, the marking mode is off

y = x * x;

because of assignment, the marking mode is turned on for RHS

Linear analysis

```
y = x * x;
```



```
_d_x += _d_y * x + x * _d_y;
_d_y = 0;
```

```
y = 2 * x + 3 * z;
```

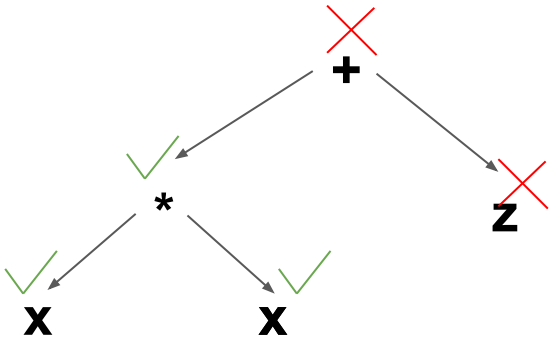


```
_d_x += 2 * _d_y;
_d_z += 3 * _d_y;
_d_y = 0;
```

Modes

$y = x * x + z;$

non-linear mode



by default, the RHS of the assignment operator is in linear mode

addition is not able to affect linearity itself

a product becomes non-linear when both terms are no constant

Data types

VarData — stores all the necessary information about one variable (in trivial cases, it is represented with bool)

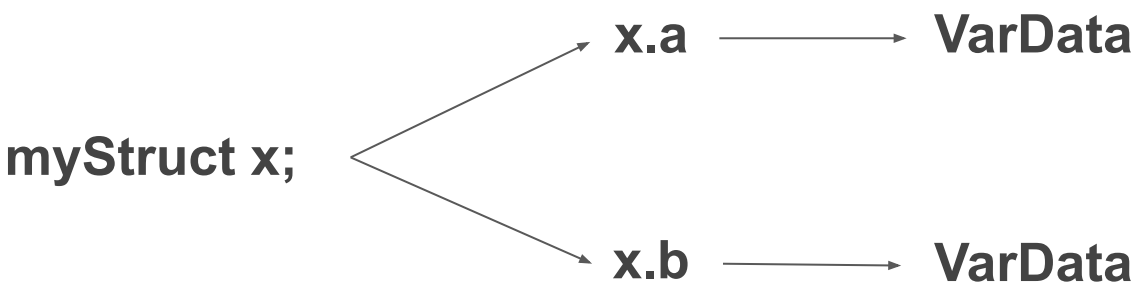
VarsData — stores information about all the variables (this is a map from VarDecl* to VarData)

FundType VarData

double x; ⟶ bool

ObjType VarData

```
struct myStruct {
   type1 a;
   type2 b;
};
```



RefType VarData

double& x = y; VarData
(corresponds to y)

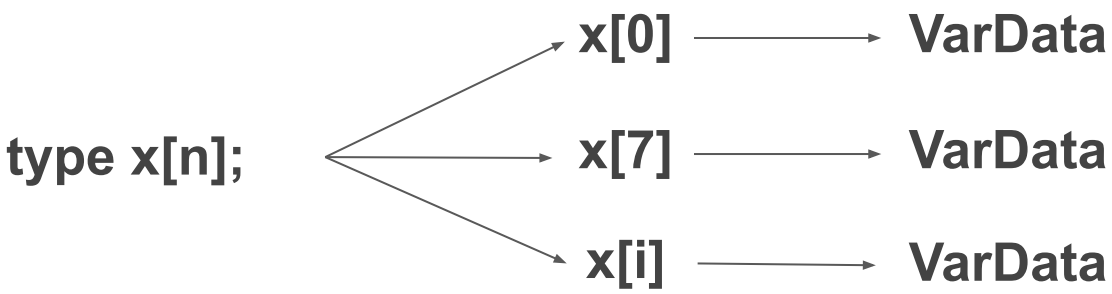
RefType VarData

double& x = y; → VarData (corresponds to y)

~~double& x = (cond ? y : z);~~

~~double& t = arr[k];~~

ArrType VarData



Non-constant indices

```
x = y * x[k];
x[0] = 1;
```

this could be any element of x

here, we have to be conservative and save x[0]

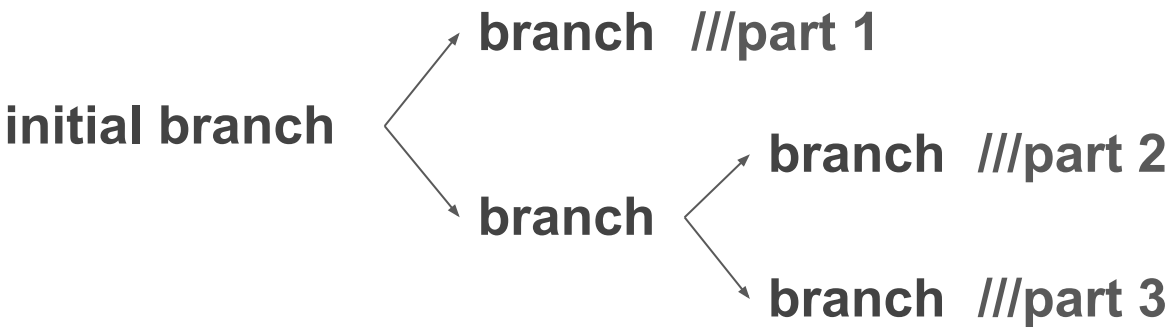
reqStack

std::vector<std::vector<VarsData>>

reqStack

std::vector<std::vector<VarsData>>

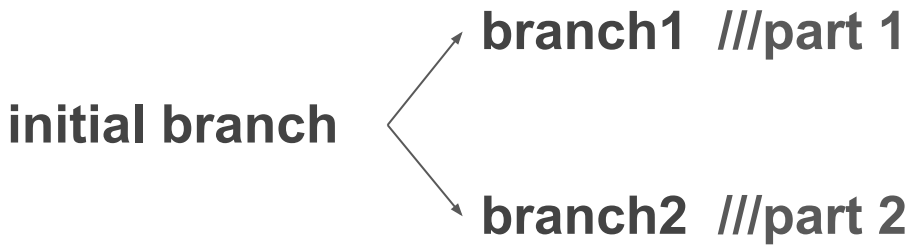
```
if (cond1) {
   ///part 1
} else {
   if (cond2) {
      ///part 2
   } else {
      ///part 3
   }
}
```



How are branches merged?

std::vector<std::vector<VarsData>>

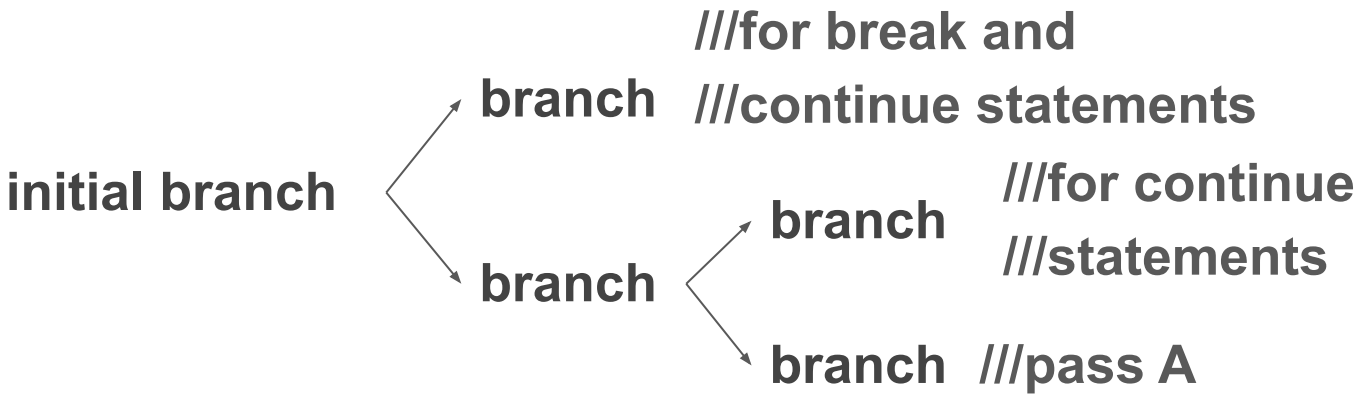
```
if (cond1) {
   ///part 1
} else {
   ///part 2
}
```



mergedBranch[VD] = branch1[VD] || branch2[VD]

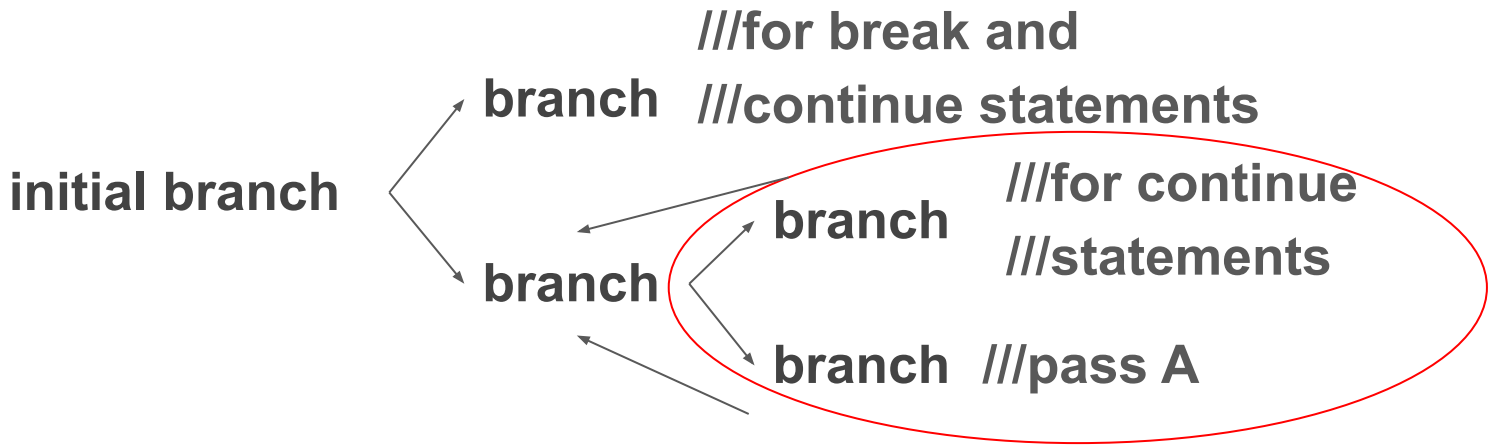
What about loops?

```
while (cond)
    ///A
```



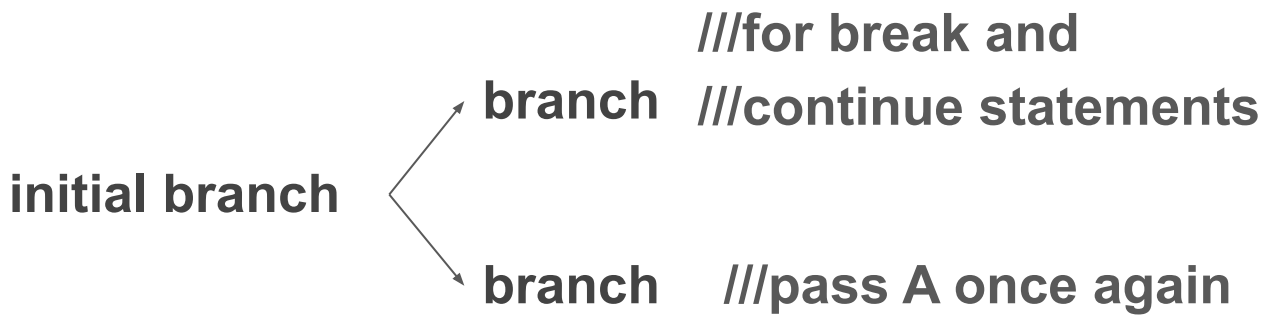
What about loops?

```
while (cond)
    ///A
```



What about loops?

```
while (cond)
    ///A
```



What should be implemented in future

- Calling functions should make the analysis proceed to analysing the function
- Add reliable support for references
- Add support for pointers