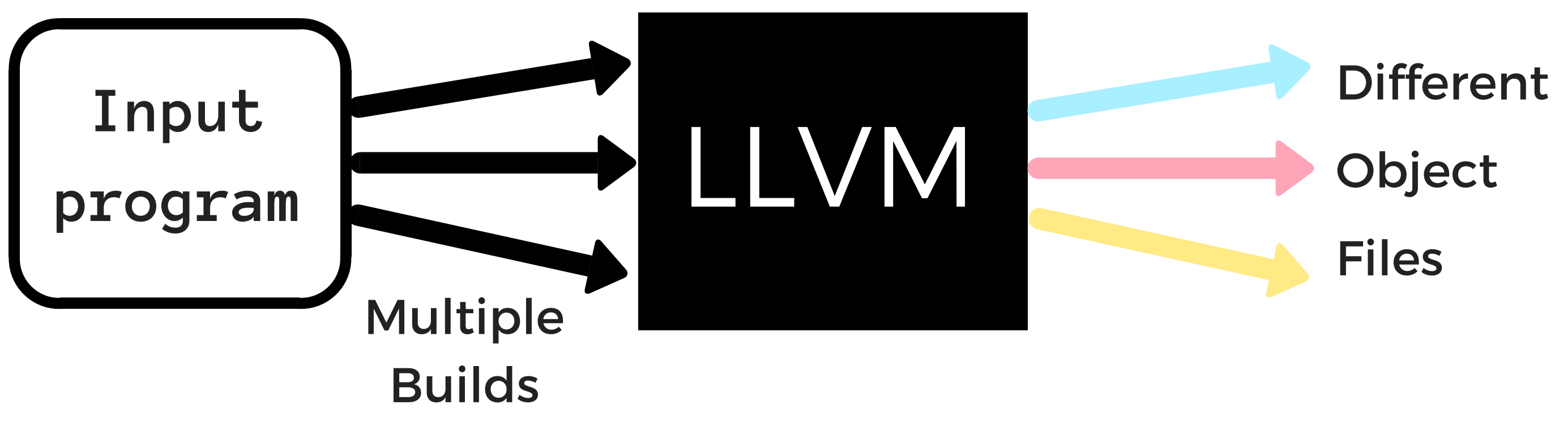
NON-DETERMINISM IN LLVM CODE GENERATION

Mandeep Singh Grang (Engineer, Senior) • Qualcomm Innovation Center, Inc., San Diego, CA

1. THE PROBLEM

LLVM generates non-deterministic code



2. WHERE IS THE PROBLEM OBSERVED?

i. Between back-to-back runs of the same toolchain

ii. Between asserts and non-asserts toolchains

iii. Between toolchains hosted on different operating systems

3. WHAT CAUSES THIS NON-DETERMINISM?

i Iteration of unordered containers

```
int x = 1, y = 2, z = 3, w = 4;

DenseMap<int *, int> Map;
Map[&x] = x;
Map[&y] = y;
Map[&z] = z;
Map[&w] = w;

for (auto &I : Map)
 dbgs() << I.second;

Output of 3 runs: 1 2 3 4
                  2 1 4 3
                  2 3 4 1
```

The DenseMap hashes on pointer keys which are different from run-to-run. This makes the iteration order non-deterministic.

ii Using non-stable sorting functions

```
using IntPair = std::pair<int, int>;
IntPair x = {1, 1}, y = {1, 2},
        z = {1, 3}, w = {1, 4};
SmallVector<IntPair, 4> V = { x, y, z, w };

std::sort(V.begin(), V.end(),
  [] (IntPair a, IntPair b) {
    return a.first < b.first; });

for (auto I : V)
  dbgs() << I.second;

Output: Depends on the underlying std::sort
algorithm, which could be non-stable.
```

Sorting order of keys with the same values is non-deterministic.

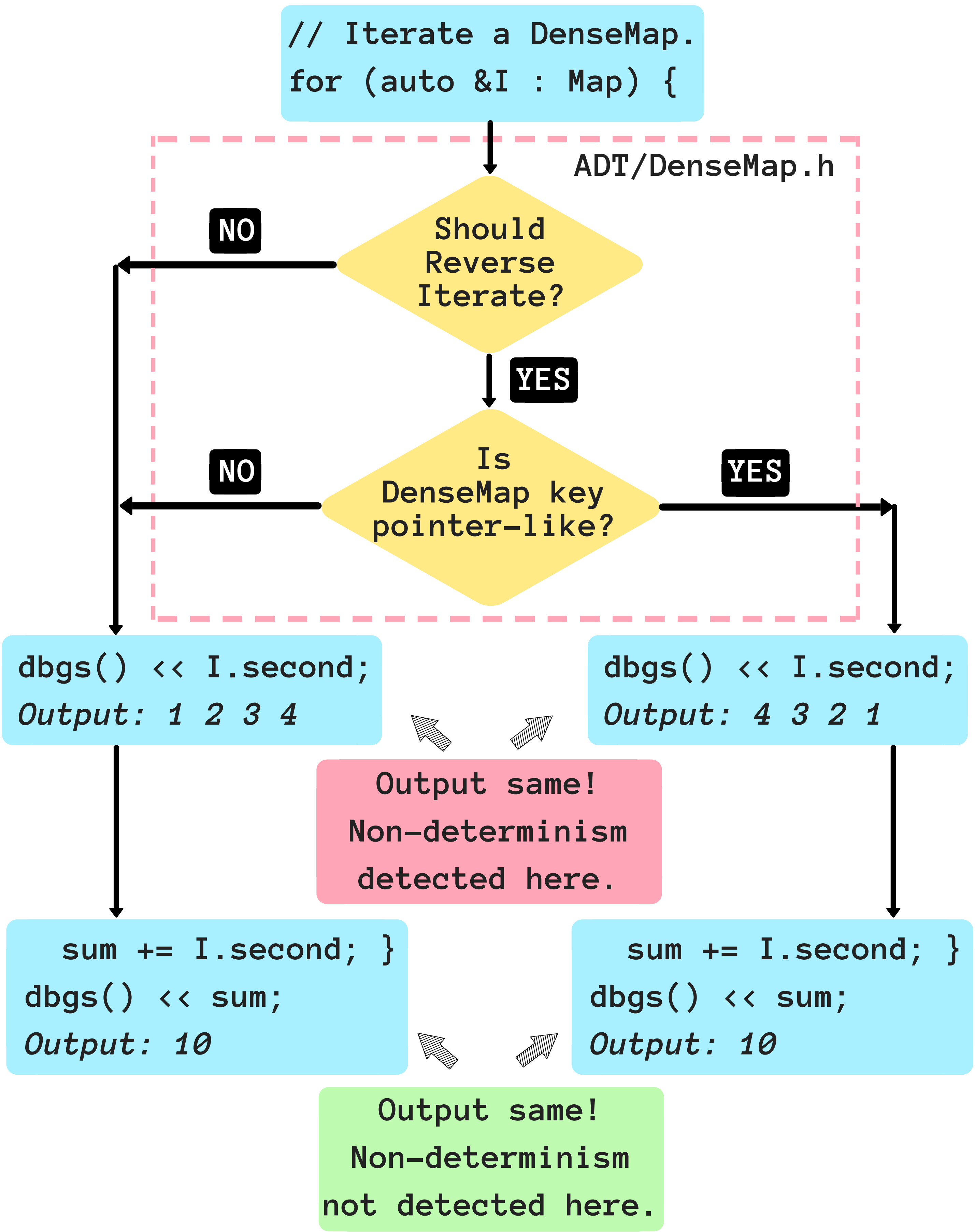
4. WHY IS IT A PROBLEM?

The generated code might not necessarily be "wrong". However, it may cause:

- Hard-to-reproduce bugs
- Unexpected runtime crashes
- Unpredictable performance

5. HOW DO YOU UNCOVER INSTANCES OF NON-DETERMINISM?

- The idea is to force "reverse iteration" of unordered containers and check if the generated code changes.
- Reverse iteration is limited to containers with pointer-like keys.



```
// Iterate a DenseMap.
for (auto &I : Map) {
```

```
dbgs() << I.second;
Output: 1 2 3 4
```

```
dbgs() << I.second;
Output: 4 3 2 1
```

Output same! Non-determinism detected here.

```
 sum += I.second; }
dbgs() << sum;
Output: 10
```

```
 sum += I.second; }
dbgs() << sum;
Output: 10
```

Output same! Non-determinism not detected here.

- Currently, the following containers support reverse iteration:

  `SmallPtrSet, DenseMap and DenseSet`

- To enable reverse iteration by default, set the build macro:

  `-DLLVM_REVERSE_ITERATION:BOOL=ON`

- A "Reverse Iteration" buildbot has been setup:

  `http://lab.llvm.org:8011/builders/reverse-iteration`

6. HOW DO YOU FIX/AVOID NON-DETERMINISM?

i Sort the container before iteration

```
SmallPtrSet<T *, 4> S;
std::sort(S.begin(), S.end(), [] (T *A, T *B)
{ return A->x < B->x; });
for (auto &I : S);
```

ii Use a stronger sort predicate

```
std::sort(S.begin(), S.end(), [] (T *A, T *B)
{ return A->x < B->x && A->y < B->y; });
```

iii Use a stable sort function

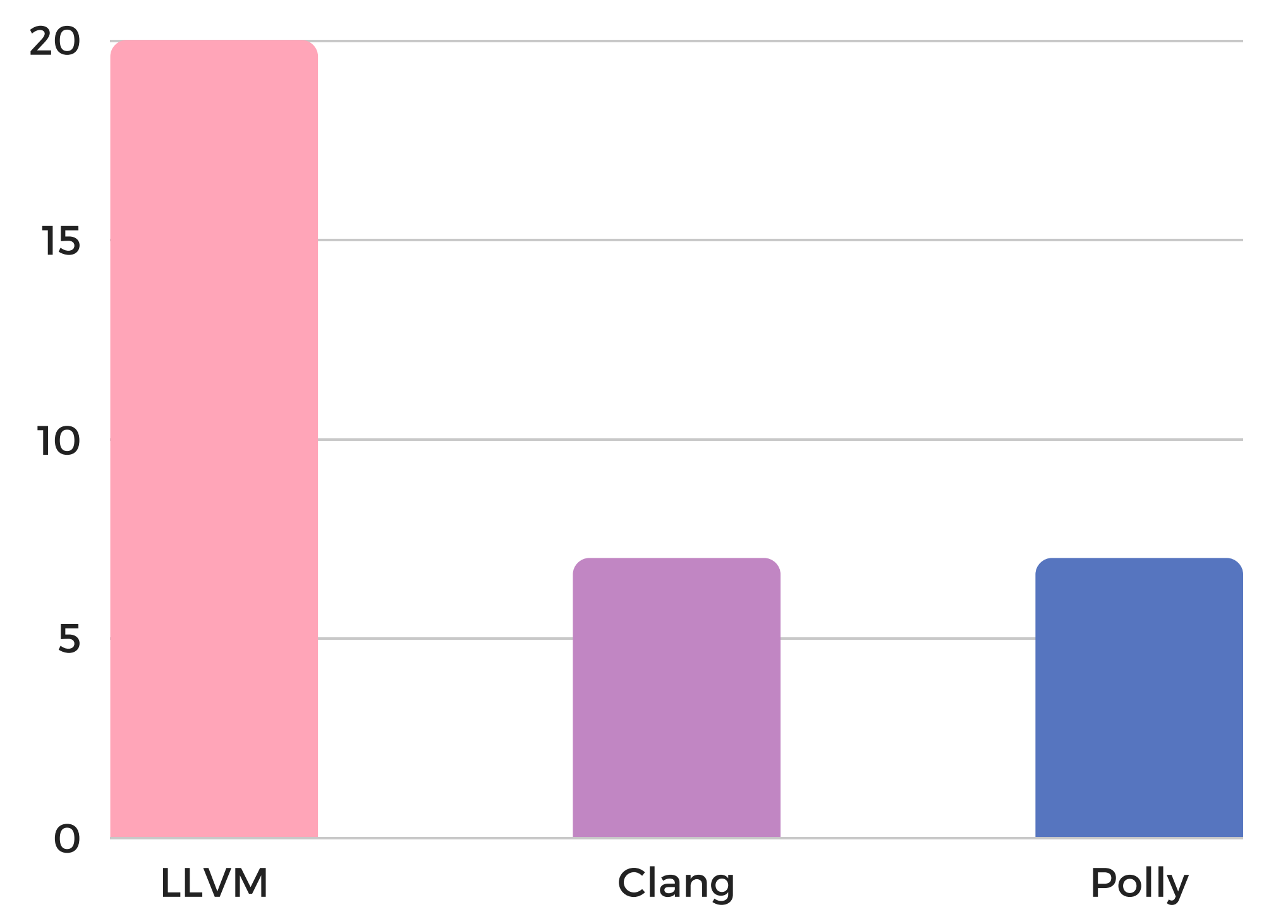
```
std::stable_sort(S.begin(), S.end(),
[] (T *A, T *B) { return A->x < B->x; });
```

iv Use an ordered container

```
SmallSetVector<T *, 4> S;
for (auto &I : S);
```

⚠ Using ordered containers instead of unordered ones can increase compile time.

7. HOW MANY ISSUES HAVE BEEN DETECTED SO FAR?



ninja check failures detected by reverse iteration (Oct 2016 - Oct 2017)

LLVM Developers' Meeting 2017, San Jose, CA