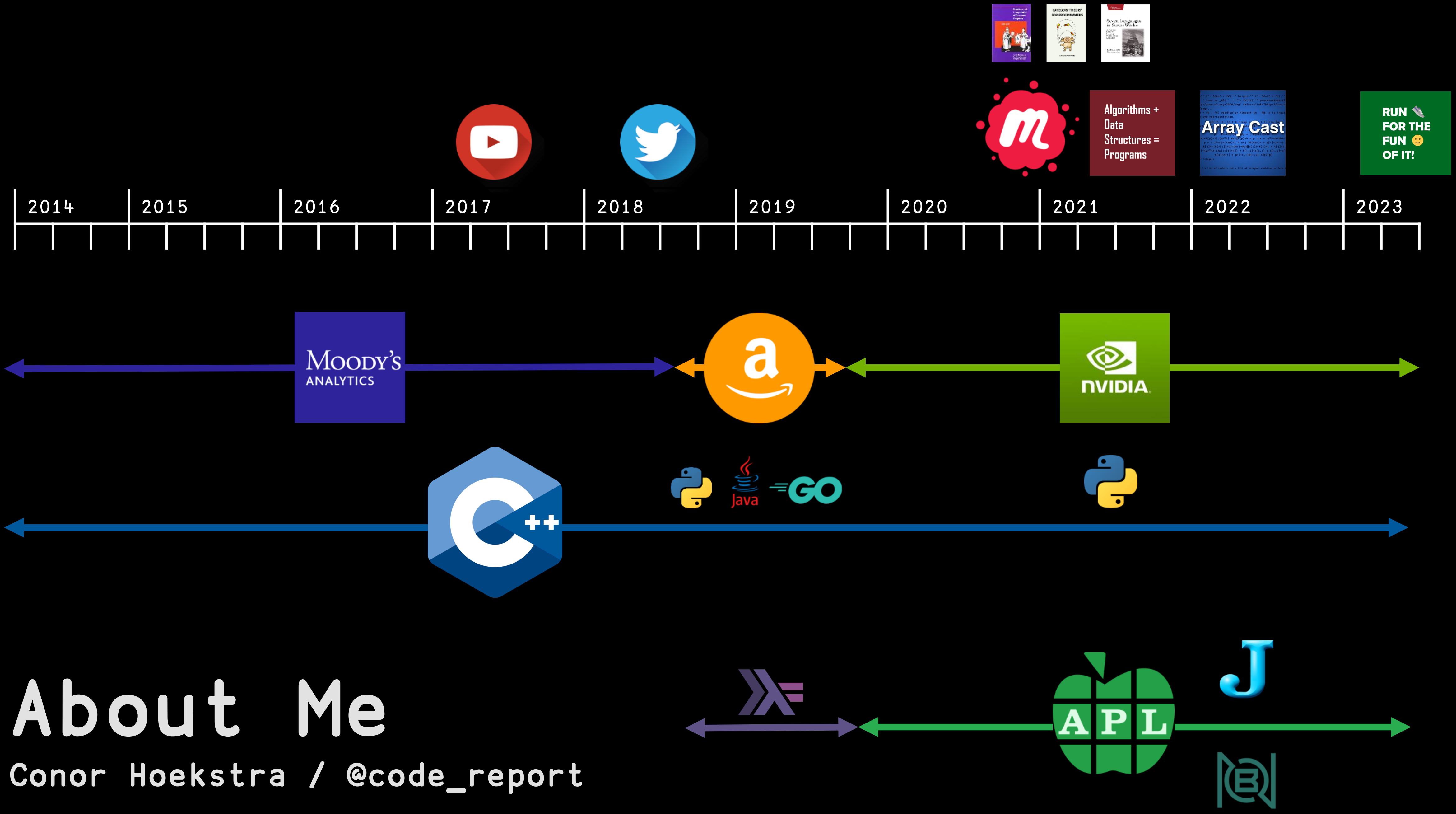
C++ vs Haskell vs BQN

Conor Hoekstra

code_report | codereport

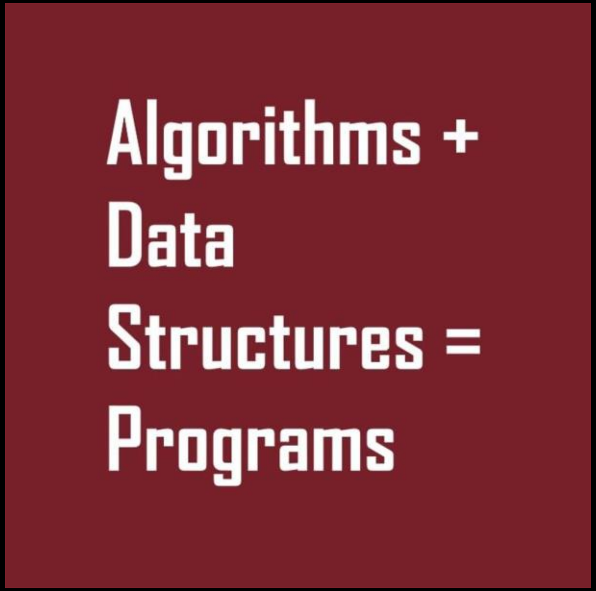


About Me

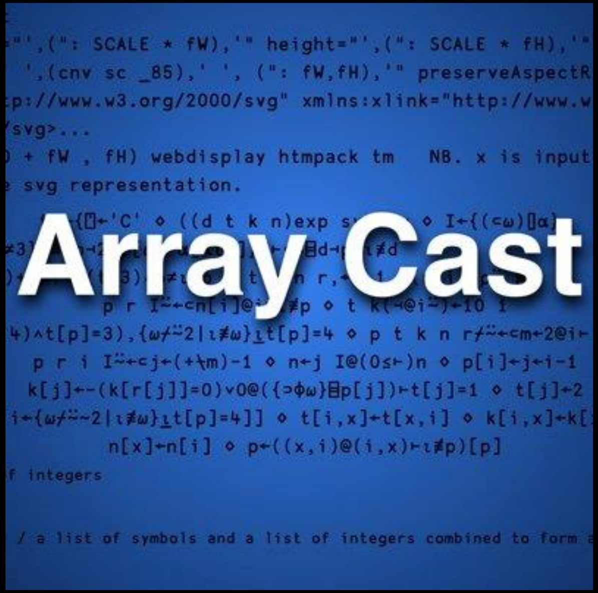
Conor Hoekstra / @code_report

319 Videos

32 (20) Talks



136 Episodes
@adspthepodcast



56 Episodes
@arraycast

10 Episodes
@conorhoekstra

https://github.com/codereport/Content

```
[1, 2, 3, 4, 5]
```

```
[1, 3, 5]
```

C++14 Function Deduced Return Type

```
auto filter_odds(std::vector<int> nums) {
    auto odds = std::vector<int>{};
    for (auto const num : nums)
        if (num % 2 == 1)
        odds.push_back(num);
    return odds;
}
```

C++11 auto

C++11 range-based for

```cpp
auto filter_odds(std::vector<int> nums) {
    auto odds = std::vector<int>{};
    std::copy_if(
        nums.begin(), nums.end(),
        std::back_inserter(odds),
        [](auto e) { return e % 2 == 1; });
    return odds;
}
```

C++11 lambda

```cpp
auto filter_odds(std::vector<int> nums) {
    auto odds = std::vector<int>{};
    std::copy_if(
        nums.begin(), nums.end(),
        std::back_inserter(odds),
        [](auto e) { return e % 2 == 1; });
    return odds;
}
```

C++14 generic lambda

```cpp
auto filter_odds(std::vector<int> nums) {
    auto odds = std::vector<int>{};
    std::ranges::copy_if(
        nums, std::back_inserter(odds),
        [](auto e) { return e % 2 == 1; });
    return odds;
}
```

C++20 range overload

```cpp
auto filter_odds(std::vector<int> nums) {
  return nums
       | std::views::filter(
           [](auto e) { return e % 2 == 1; });
}
```

C++20 Ranges

```haskell
filterOdd xs = filter (\e -> mod e 2 == 1) xs
filterOdd xs = filter odd xs
filterOdd = filter odd
```

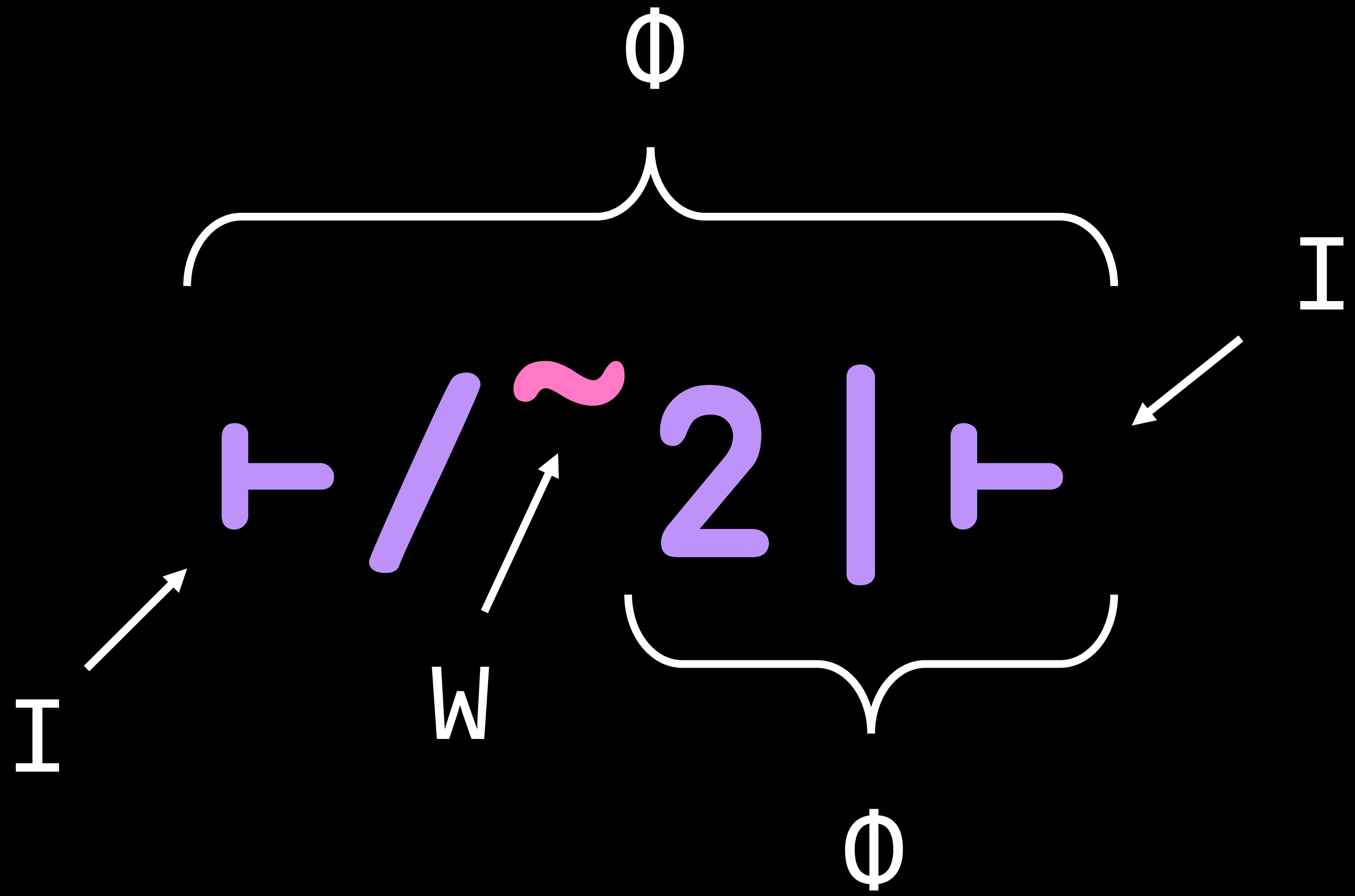
```
filter odd
```

$$\{(2 \mid \mathbb{x})/\mathbb{x}\}$$

$$\underbrace{2 \multimap | / \overset{I}{\vdash}}_{\Phi}$$

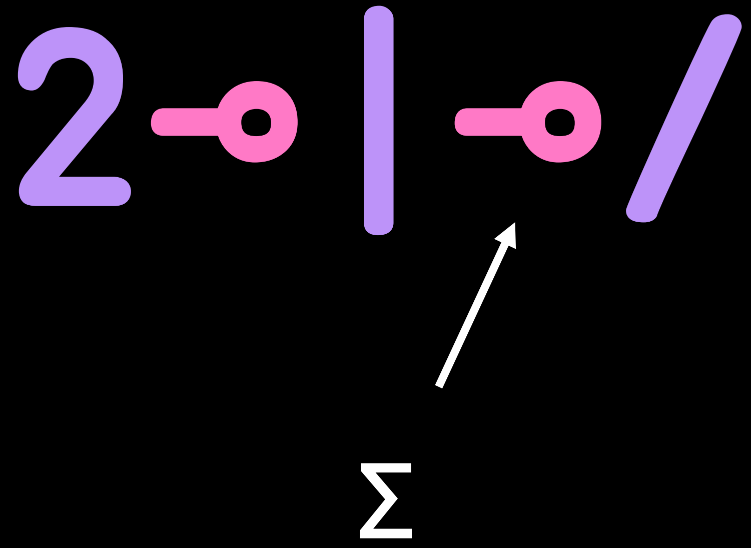
W

```
{𝕩/˜2|𝕩}
```


Φ
I
⊢/~2|⊢
I
W
Φ

I

$$\underbrace{\texttt{2⊸|/⊢}}_{\Phi}$$


2
Σ

combinatorylogic.com
bqnpad.mechanize.systems
tryapl.org

Thank You

https://github.com/codereport/Content/Talks

Conor Hoekstra

code_report

codereport