Expressions speed up

… when you are too lazy to calculate everything

What is optimised

- General logic expression: **A & B & C & D**
- Complex logic: **(A & !B) I C & D**
- Arithmetic logic: **A + B + C + D > 2**
- All above: **F & (A + B + C + D > 2)**

SA approach

- Evaluate all rules (terribly slow)
- Use only some of them
- Use regexps for everything

Old rspamd

- Reverse Polish Notation

A & B & C -> A B C & &

- Evaluated rules, applying basic optimisations:

```
A & B & C & D
0 & 1 & 1 & 0
A & B & C & D
```

If A equal to 0 there is no need to evaluate other components

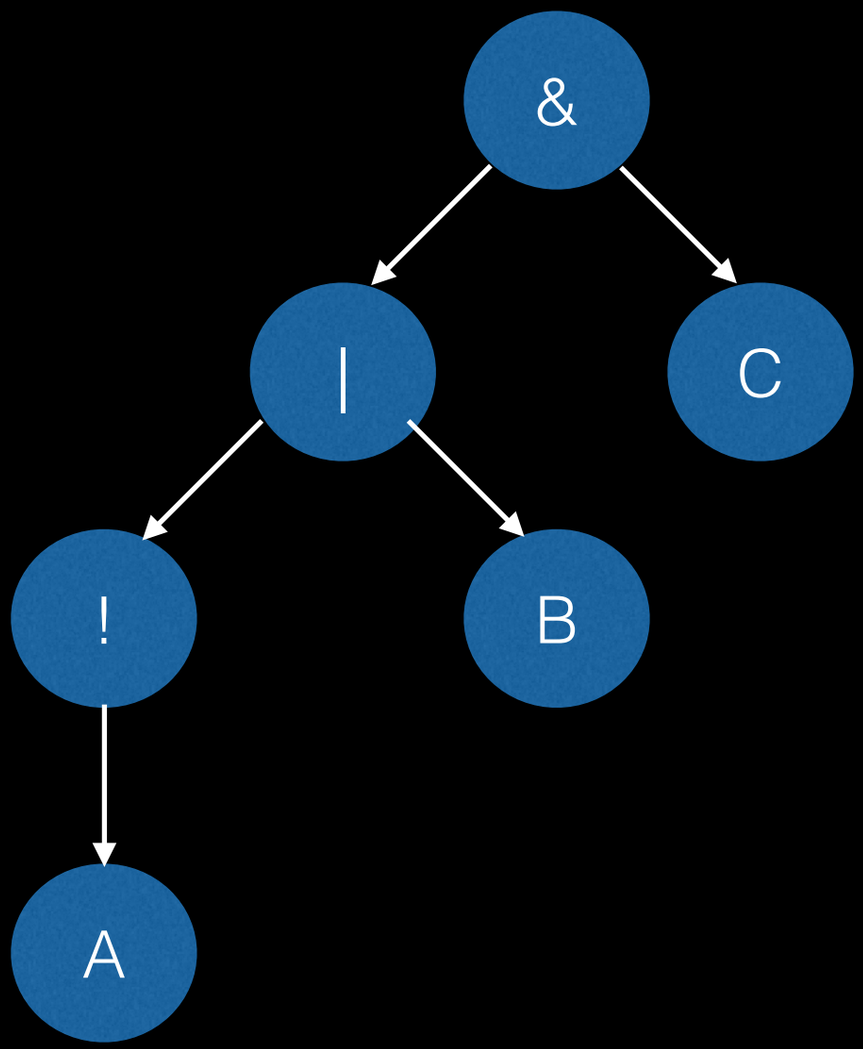
Can we do better?

- We want to organise evaluations to execute faster ops before expensive ops
- We want to have a generic evaluation of the arguments to decide when to stop and return

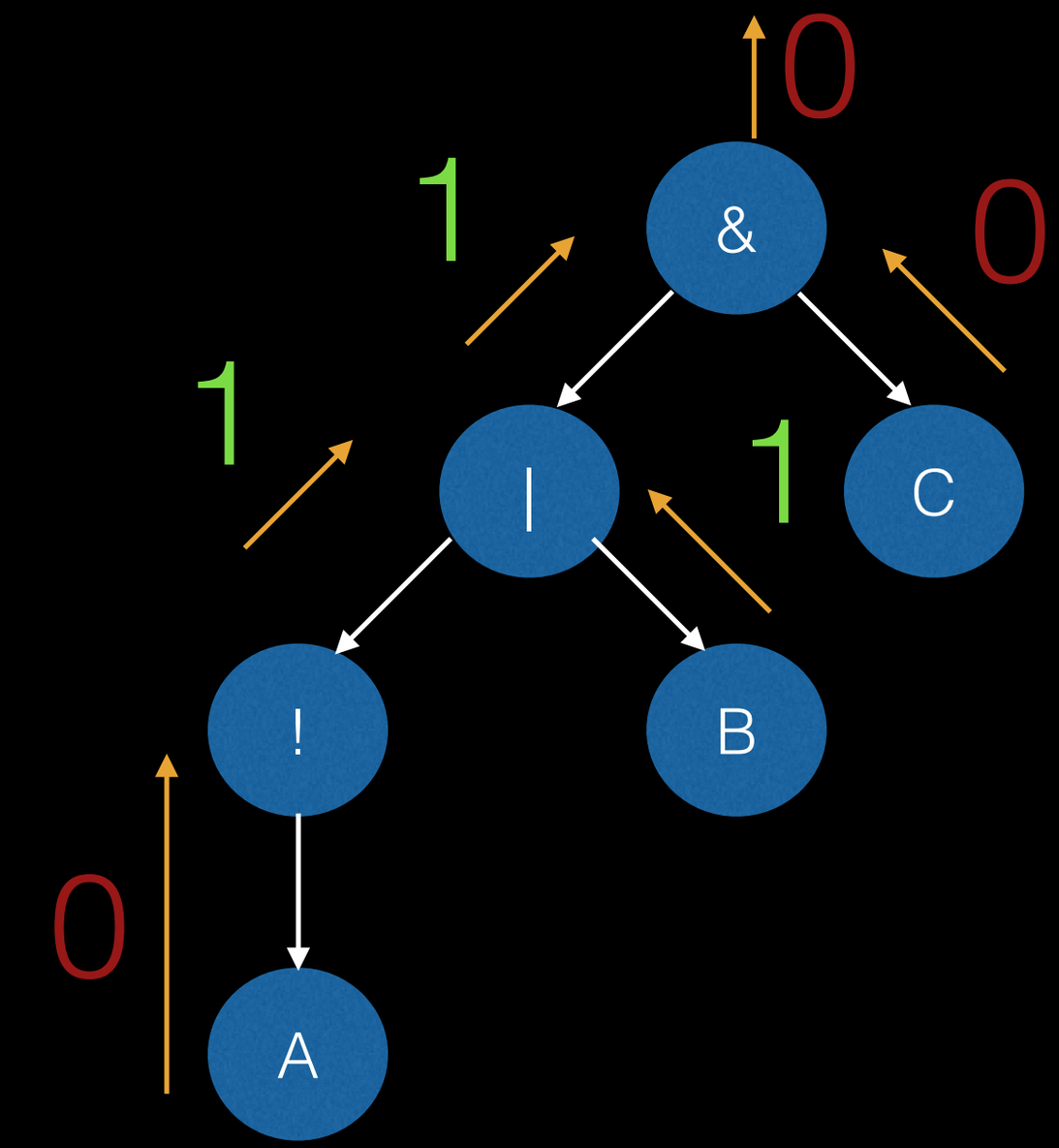
Solution: AST

- Abstract Syntax Tree - a tree of expressions
- Optimize branches in the tree by execution time and frequency
- Apply greedy algorithm to minimise calculations
- Be as lazy as possible (laziness is good!)

AST building

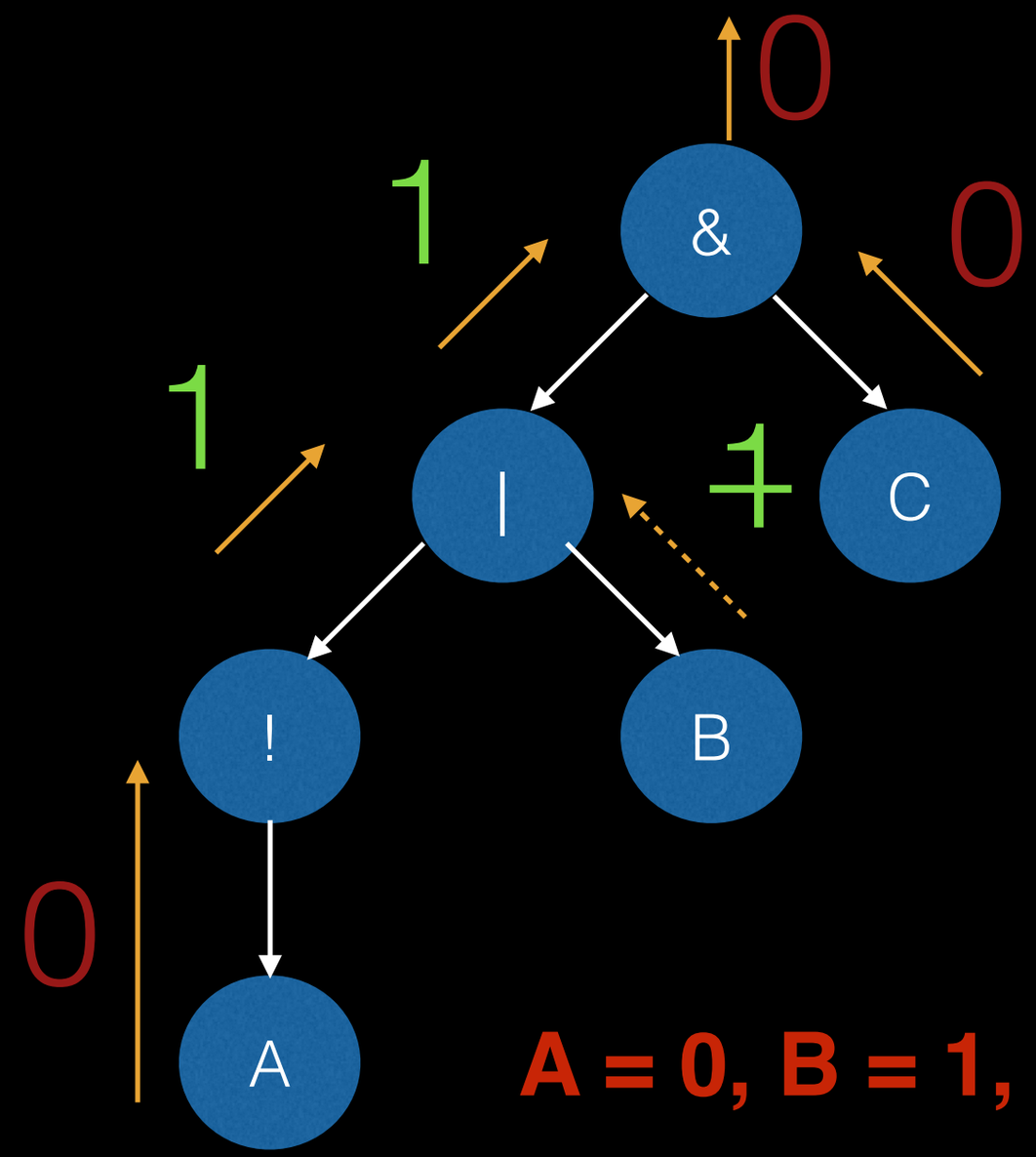


AST eval (naive)



A = 0, B = 1, C = 0

AST branches cut

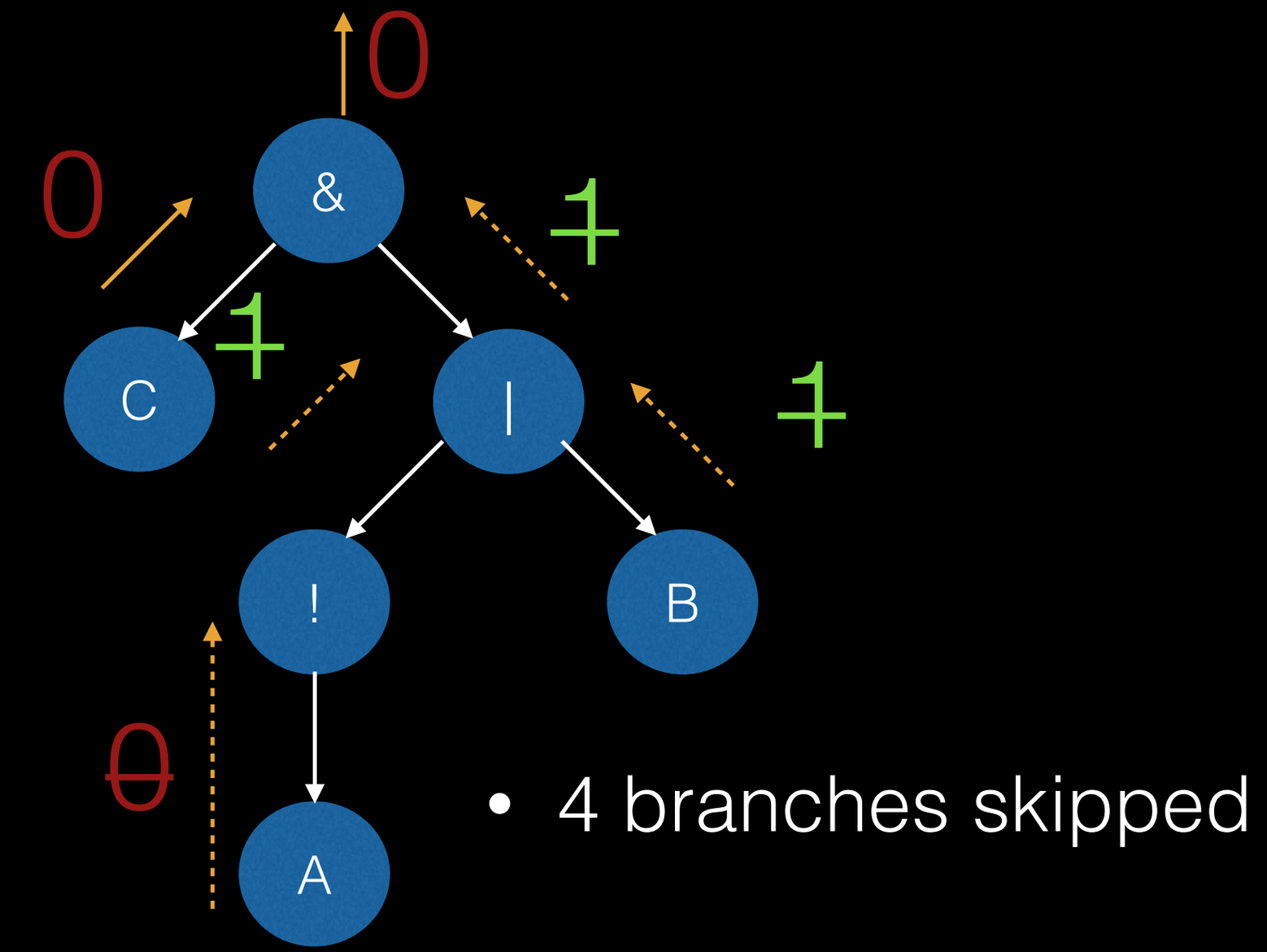


- 1 branch skipped

Can we do better?

- In the previous slide we cut merely a single branch
- Not good, still have to evaluate too many unnecessary stuff

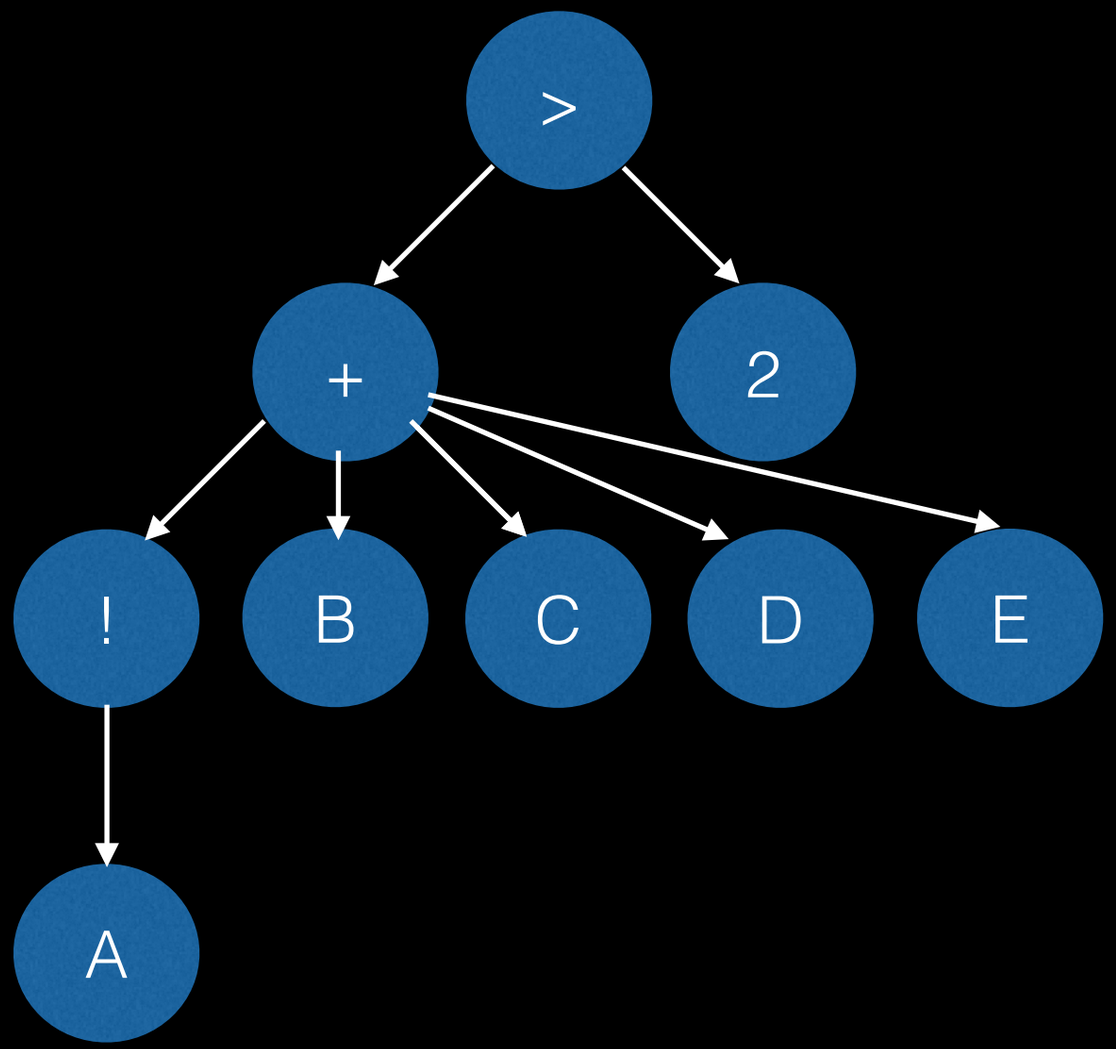
AST branches reorder



AST branches reorder

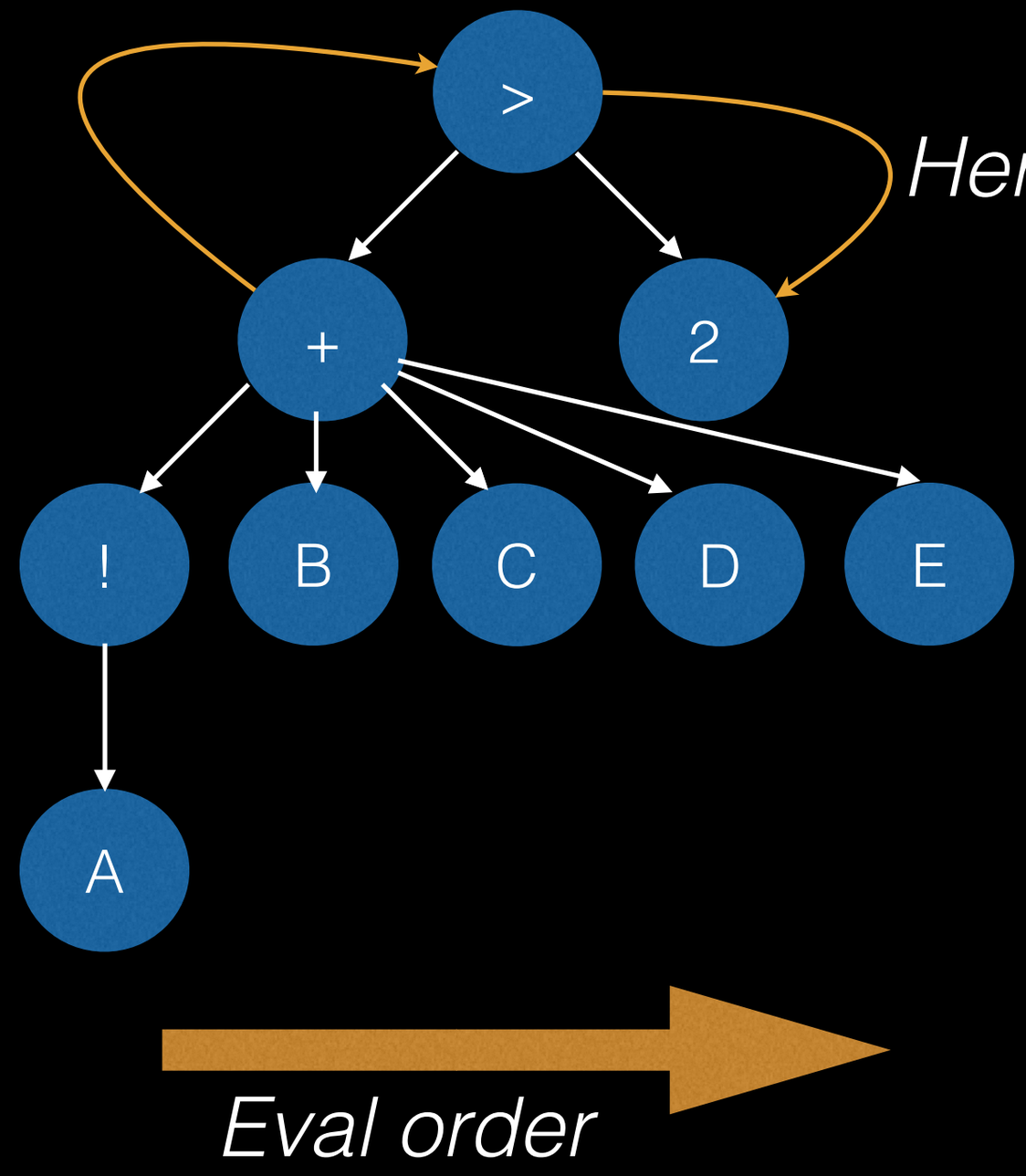
- Prioritise branches with fewer operations in the underneath levels
- Skip unnecessary evaluations
- Reduce the total running time of the expression

N-ary operations



N-ary optimizations

What do we compare?

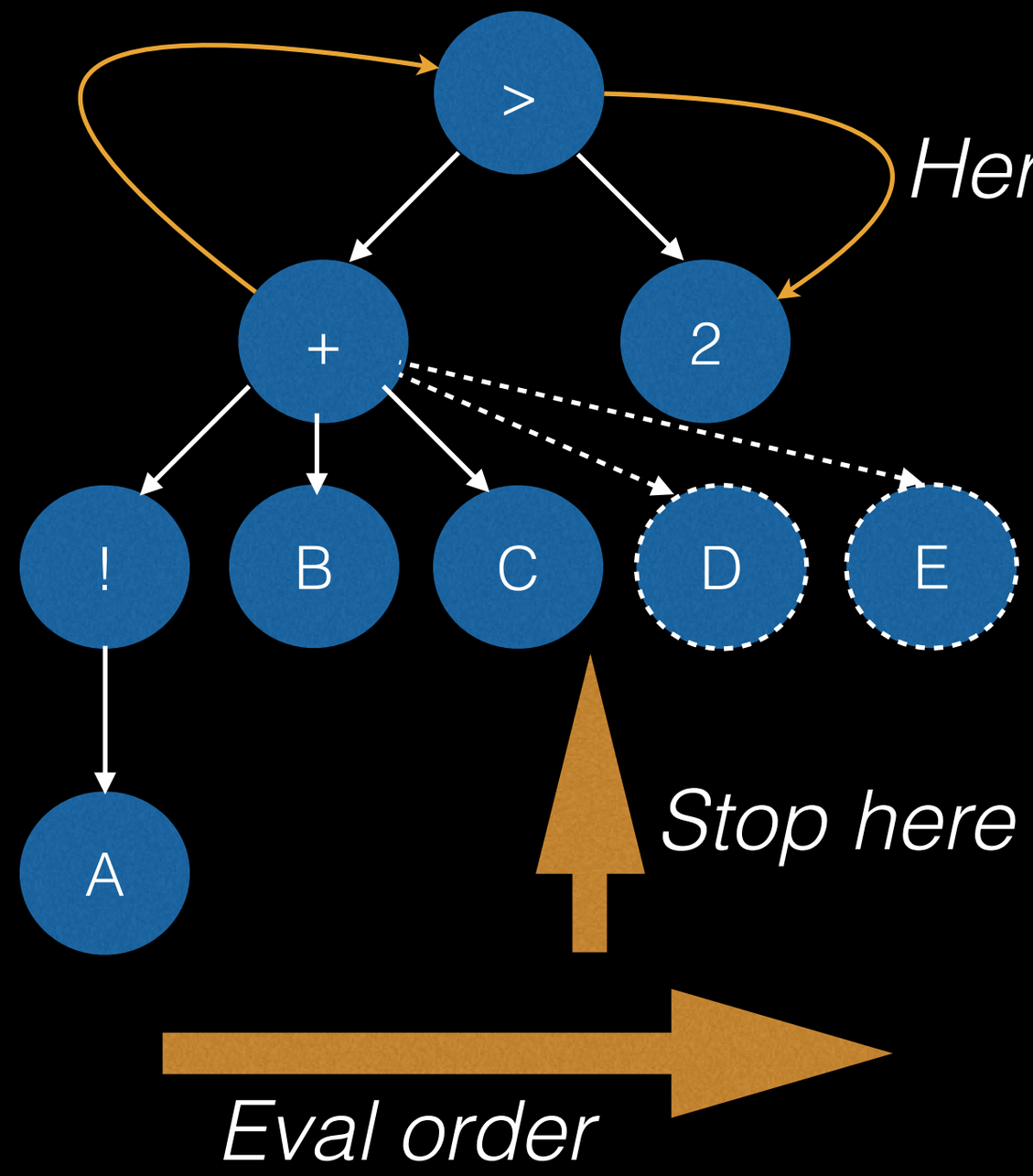


Here is our limit

Eval order

N-ary optimizations

What do we compare?



Here is our limit

Stop here

Eval order

Results

- Rspamd with RPN: 200ms on a normal message, 1.6 seconds on stupid large text message (10 Mb of text)
- Rspamd with AST: 40ms on a normal message, 400ms on stupid large text message
- SA: ??? (timeout?)

Further steps

- Greedy algorithm to optimize execution time:
  - calculate frequency and average time of a component
  - minimize expression by applying greedy formula: min(freq / avg_time) for each component

Learn dynamically

- We need to re-evaluate order of AST in the real time
- Solution: periodically evaluate atoms weights and resort tree using the same greedy algorithm
- Average time and cost is already evaluated

Laziness is the source of the progress