

HASKELL MEETUP - 27/04/2017

QUICKCHECK BY EXAMPLE

GOALS FOR TODAY

- ▶ Quick recap on QuickCheck
- ▶ How to find good properties
- ▶ Learning from contradiction

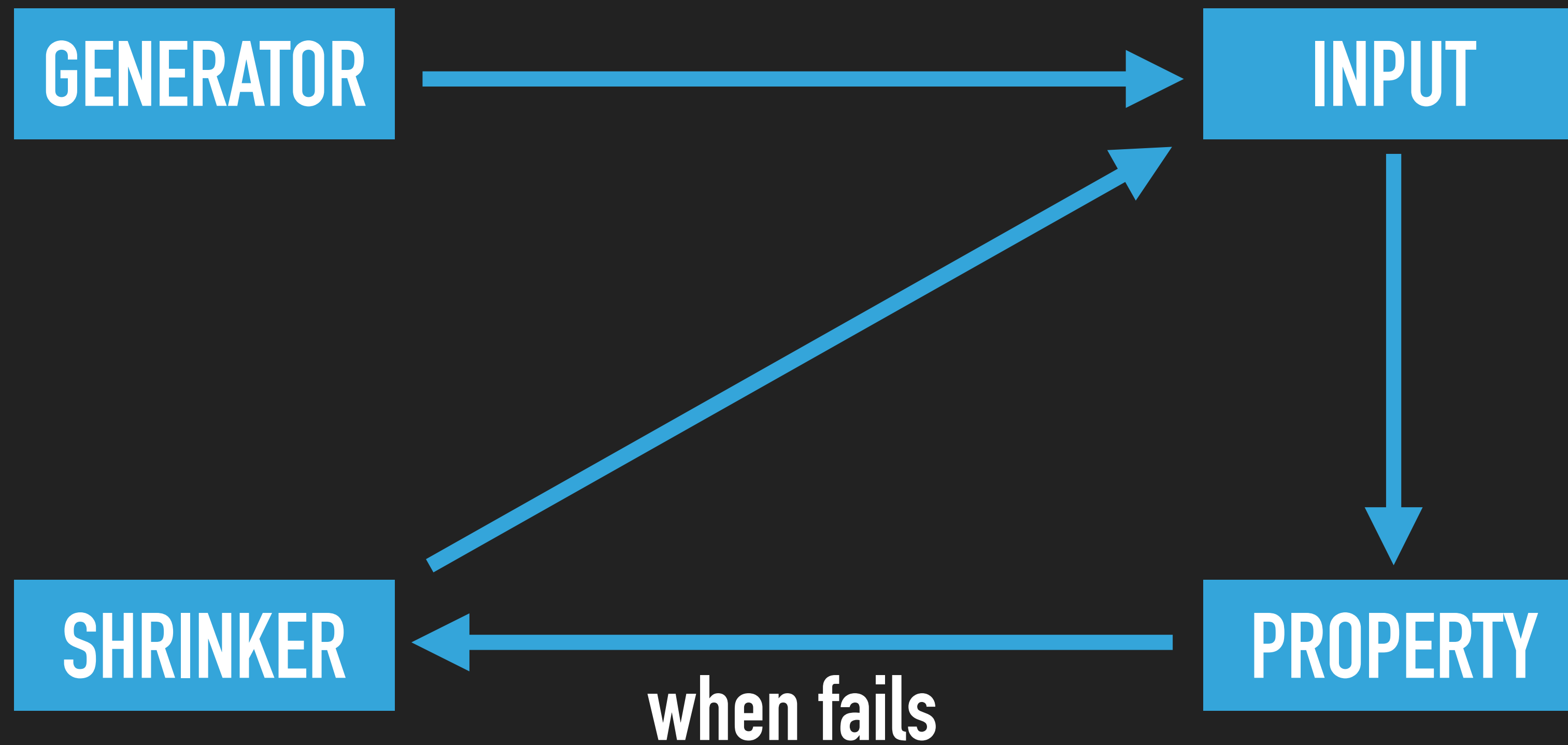
QUICK RECAP

WHAT IS QUICKCHECK?

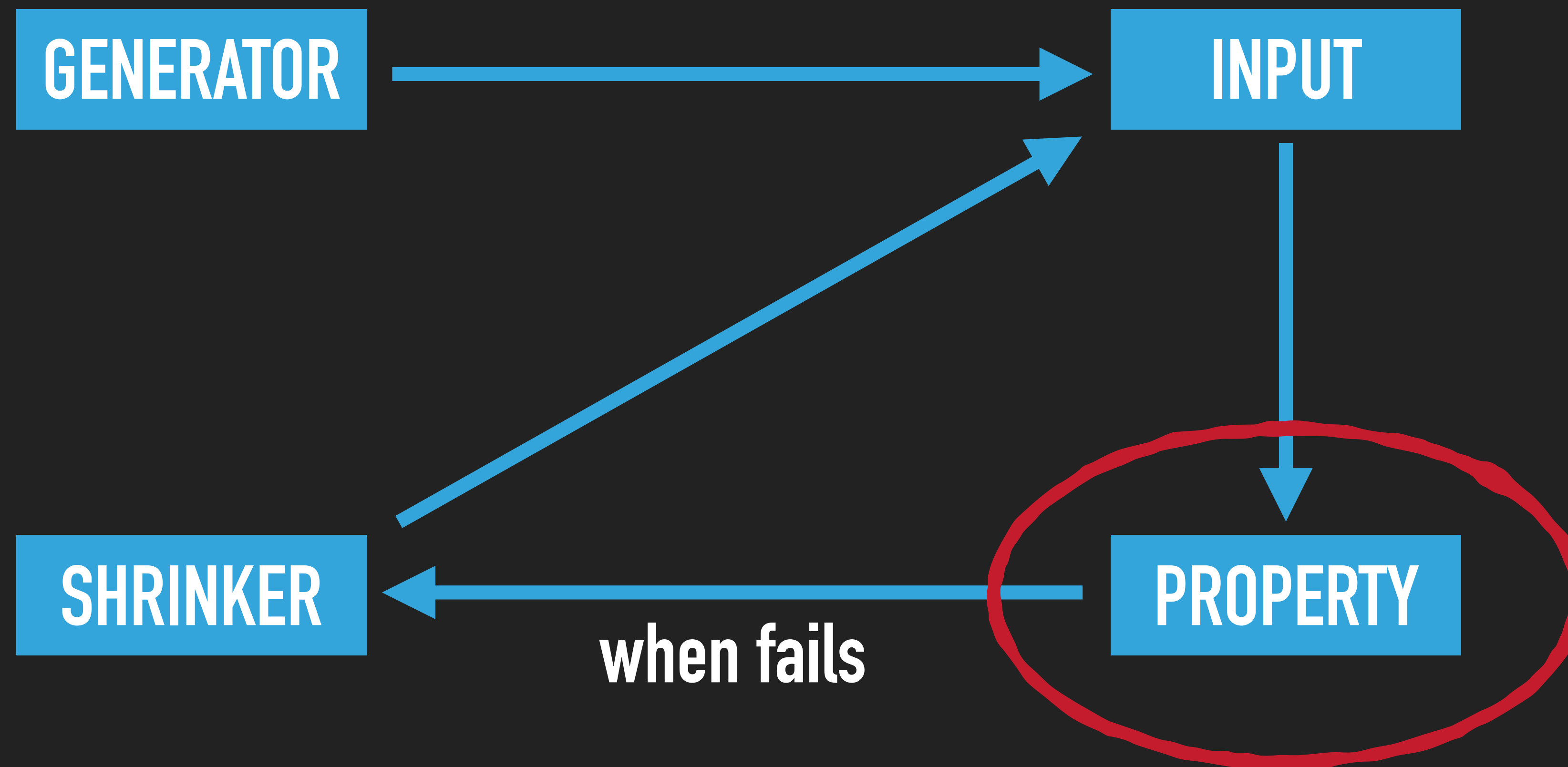
PROPERTY BASED TESTING - 101

- ▶ Tests invariants on your code
- ▶ By generating random inputs
- ▶ Shrinking the counter-examples

PROPERTY BASED TESTING - 101



PROPERTY BASED TESTING - 101



IN THEORY...

FINDING PROPERTIES

FINDING PROPERTIES

- ▶ Test against **reference** algorithm
 - ▶ Example system sort
- ▶ **Round-trip**: $\text{do} + \text{undo} = \text{identity}$
 - ▶ Parse & Serialize

FINDING PROPERTIES

- ▶ Relations between input & output
- ▶ Relations between functions
- ▶ Look at equivalence classes

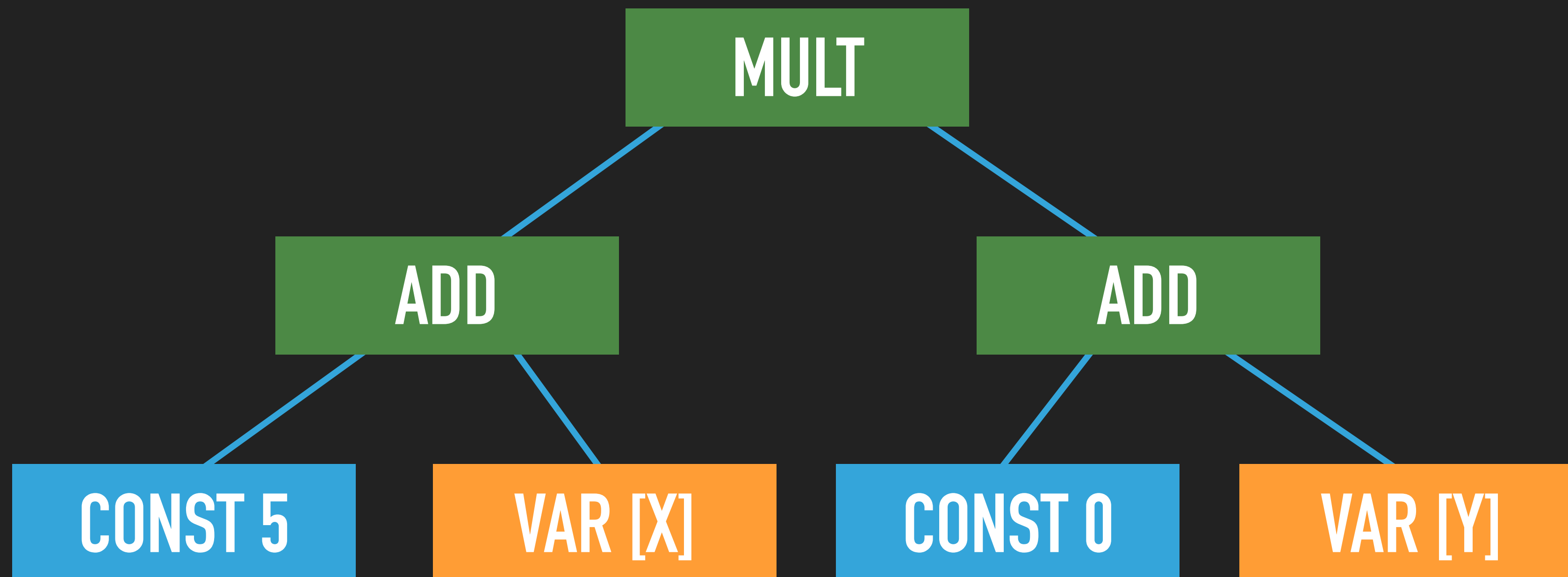
IN PRACTICE...

QUICKCHECK BY EXAMPLE

ARITHMETIC DSL

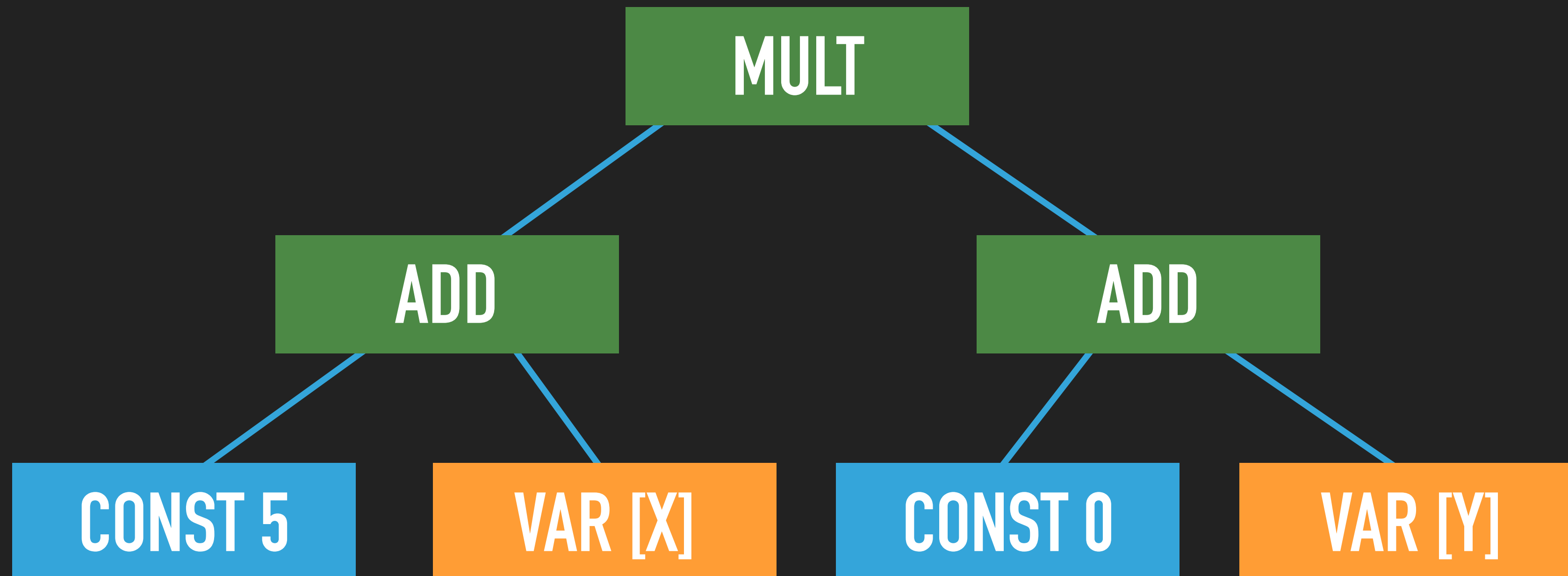
- ▶ Integer, Variable, (+) and (*)
- ▶ Goal: testing interpreters
 - ▶ Dependency
 - ▶ Evaluation
 - ▶ Optimization

ARITHMETIC EXPRESSION

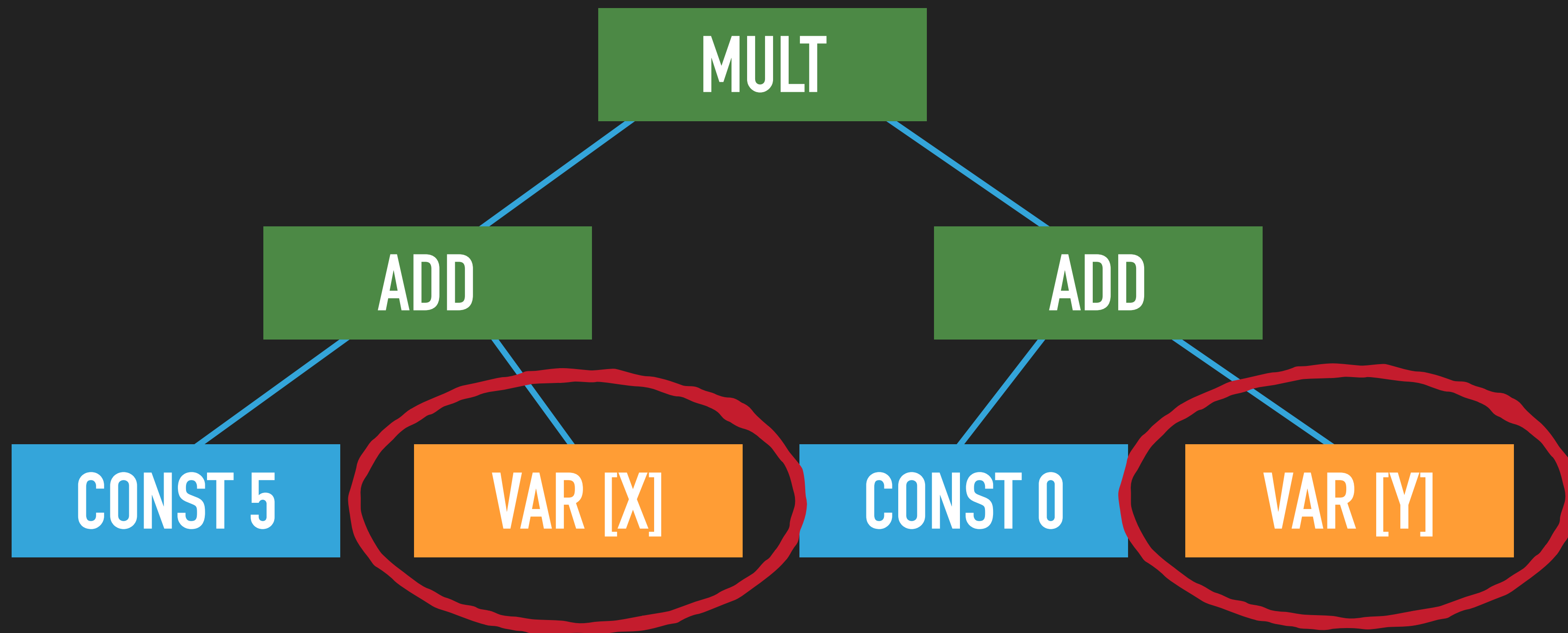


$(5 + X) * (0 + Y)$

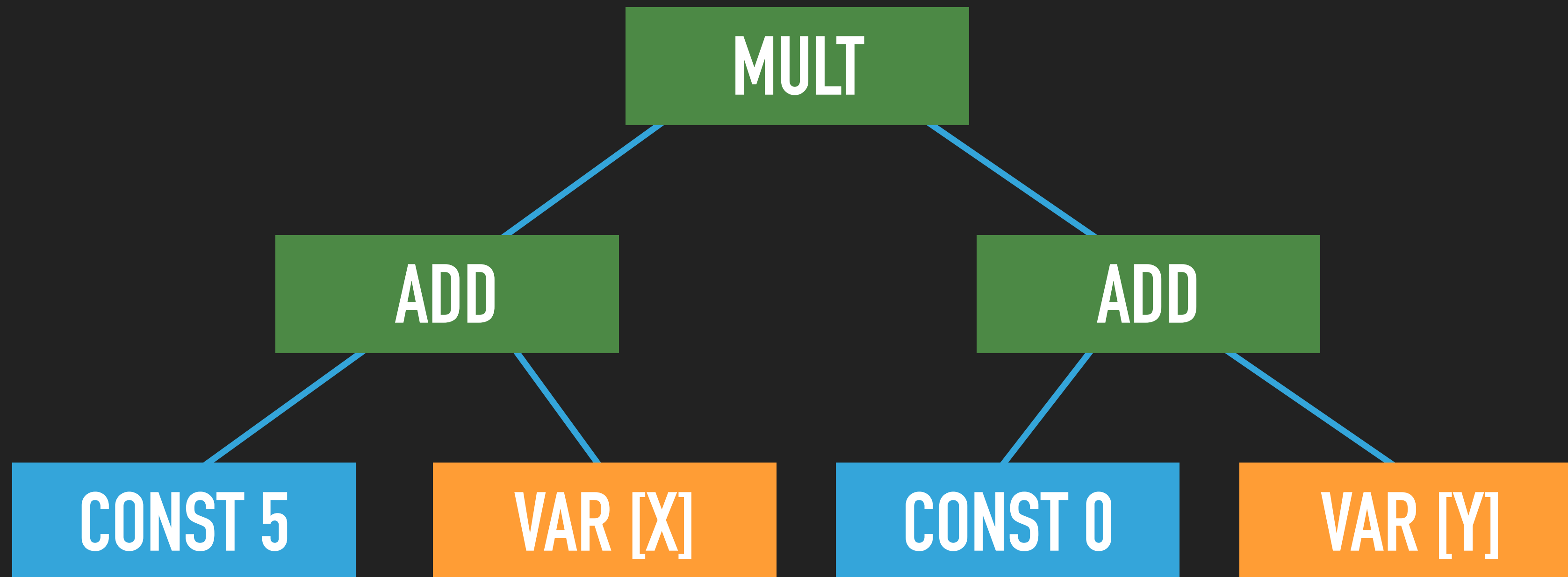
DEPENDENCIES



DEPENDENCIES

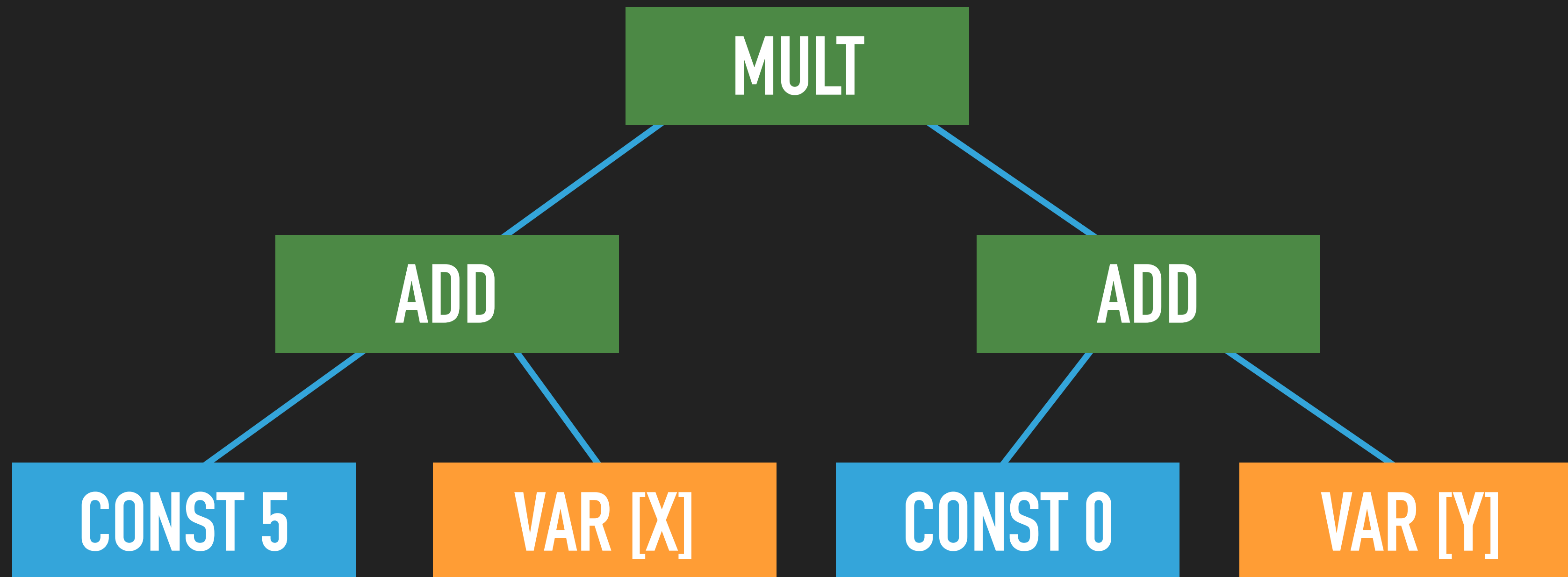


DEPENDENCIES

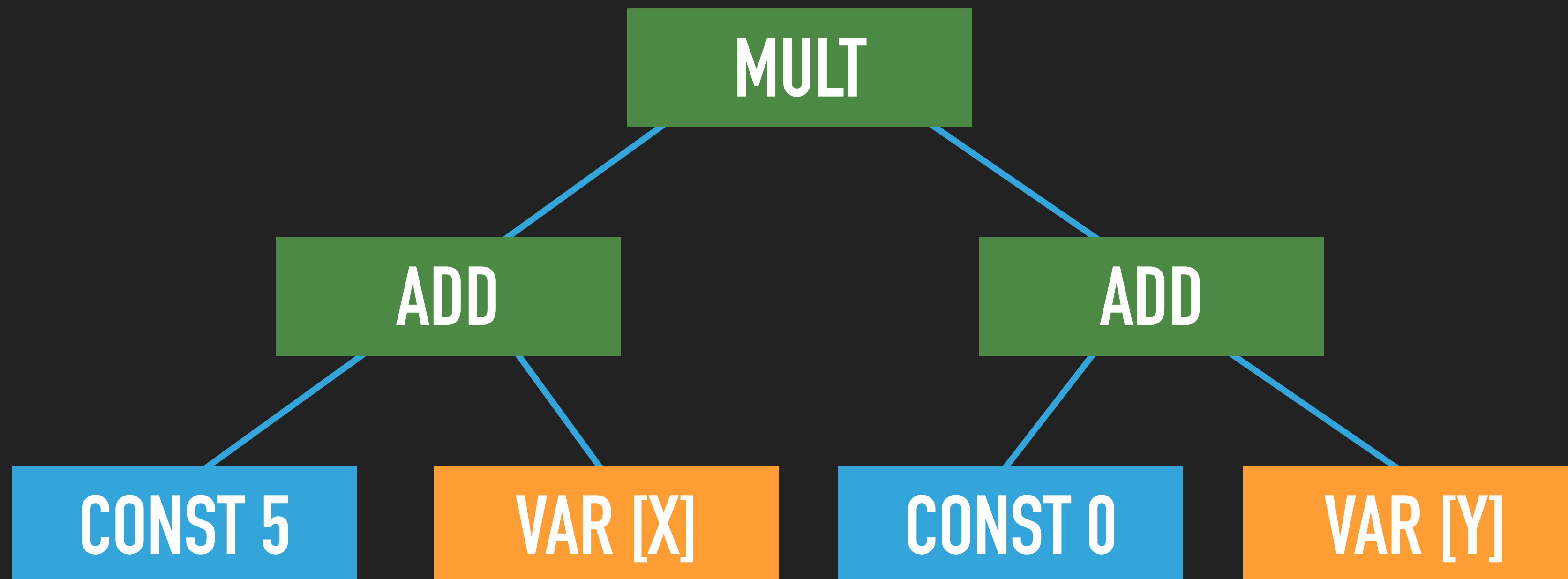


$\# \{ X, Y \}$

EVALUATE

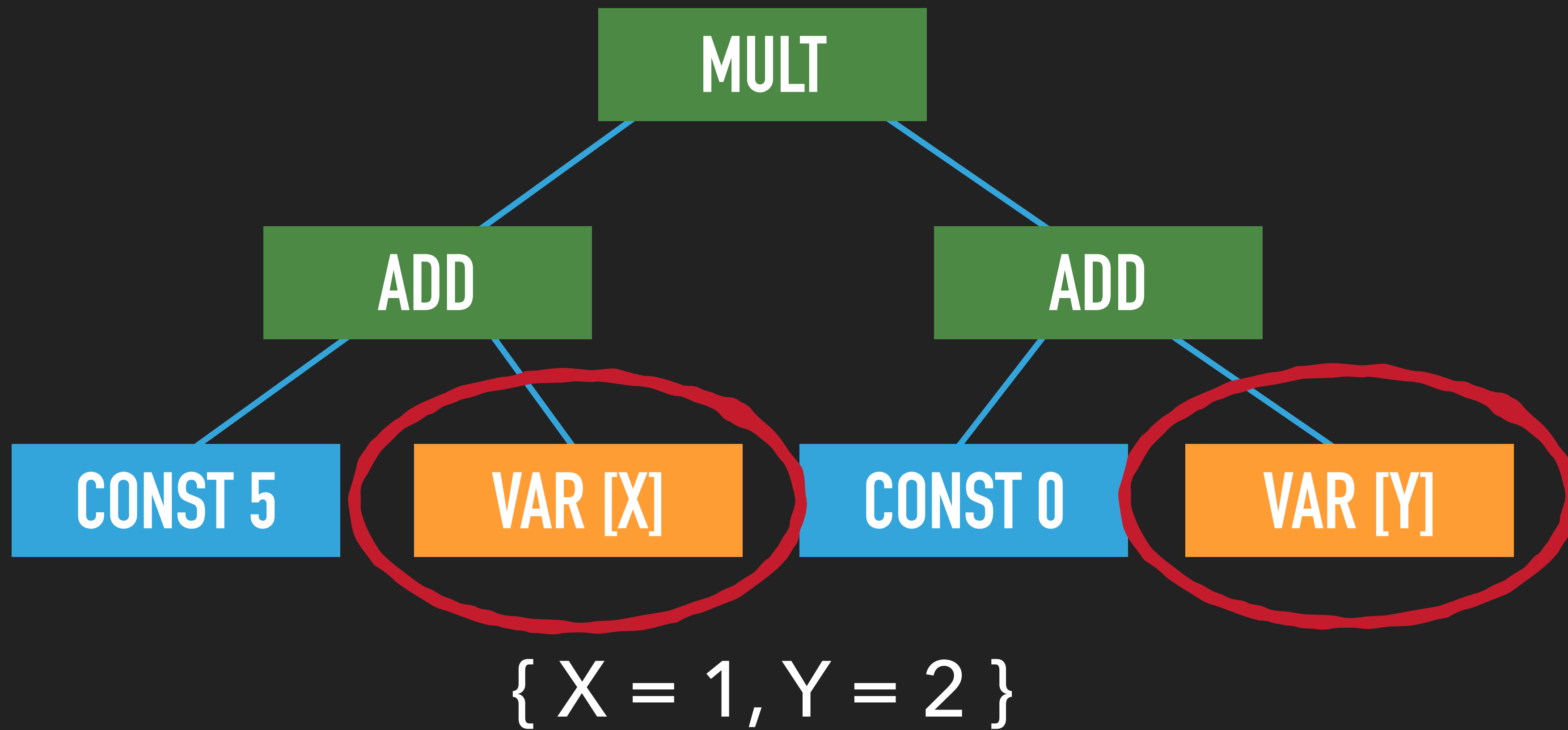


EVALUATE

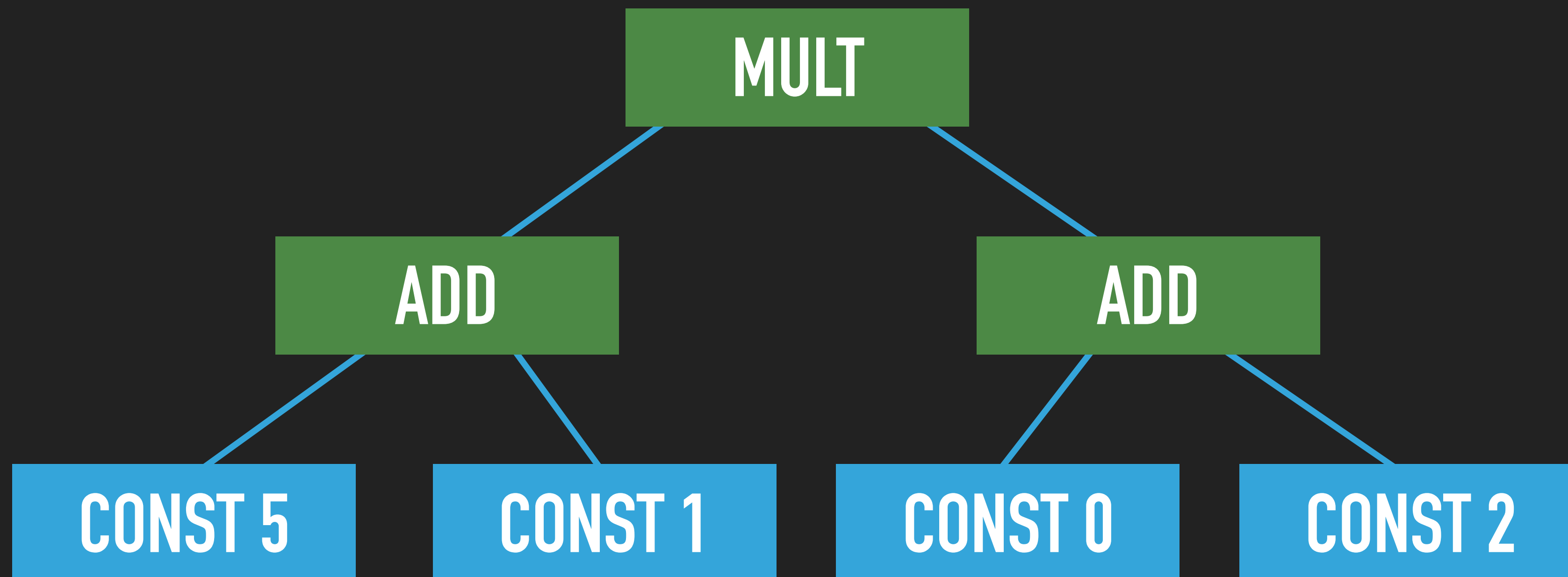


$\{ X = 1, Y = 2 \}$

EVALUATE

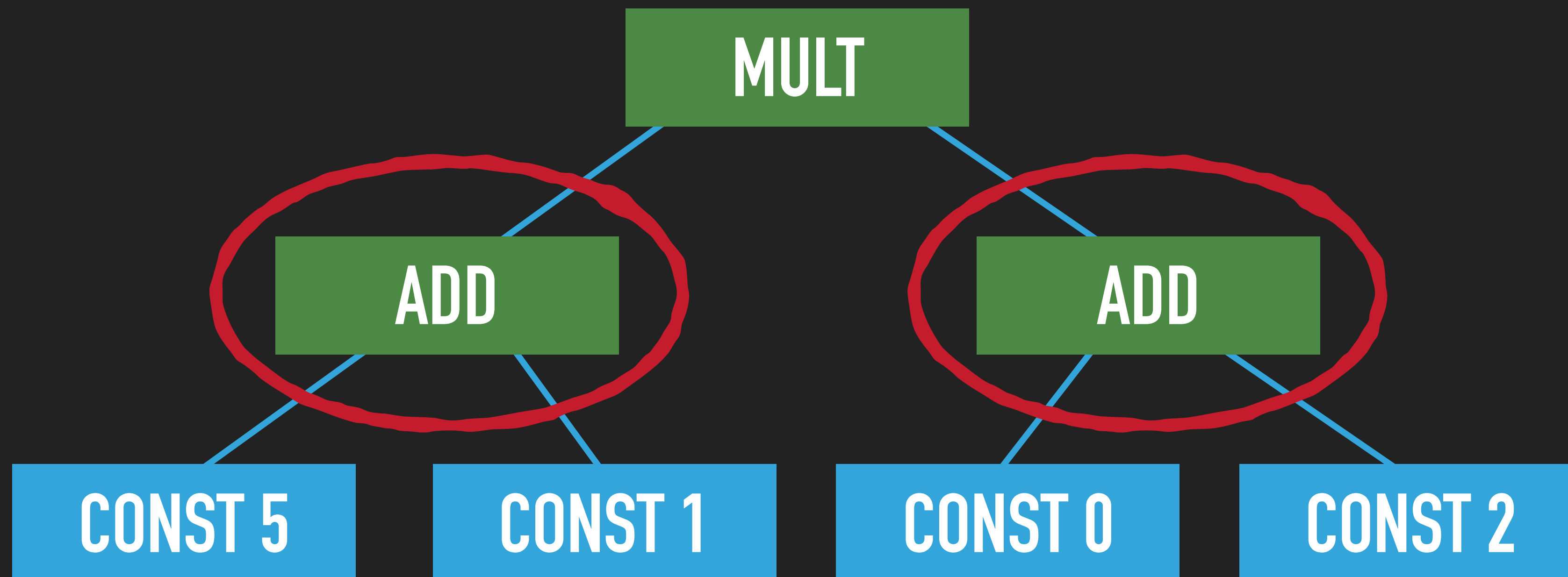


EVALUATE



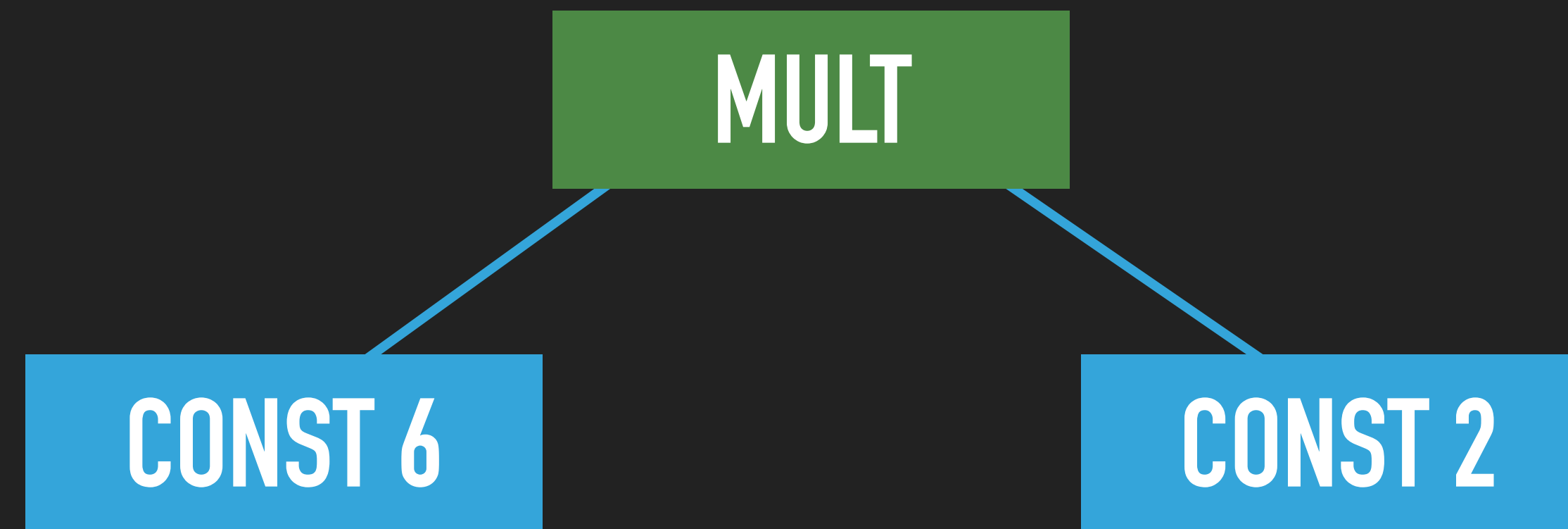
$\{ X = 1, Y = 2 \}$

EVALUATE



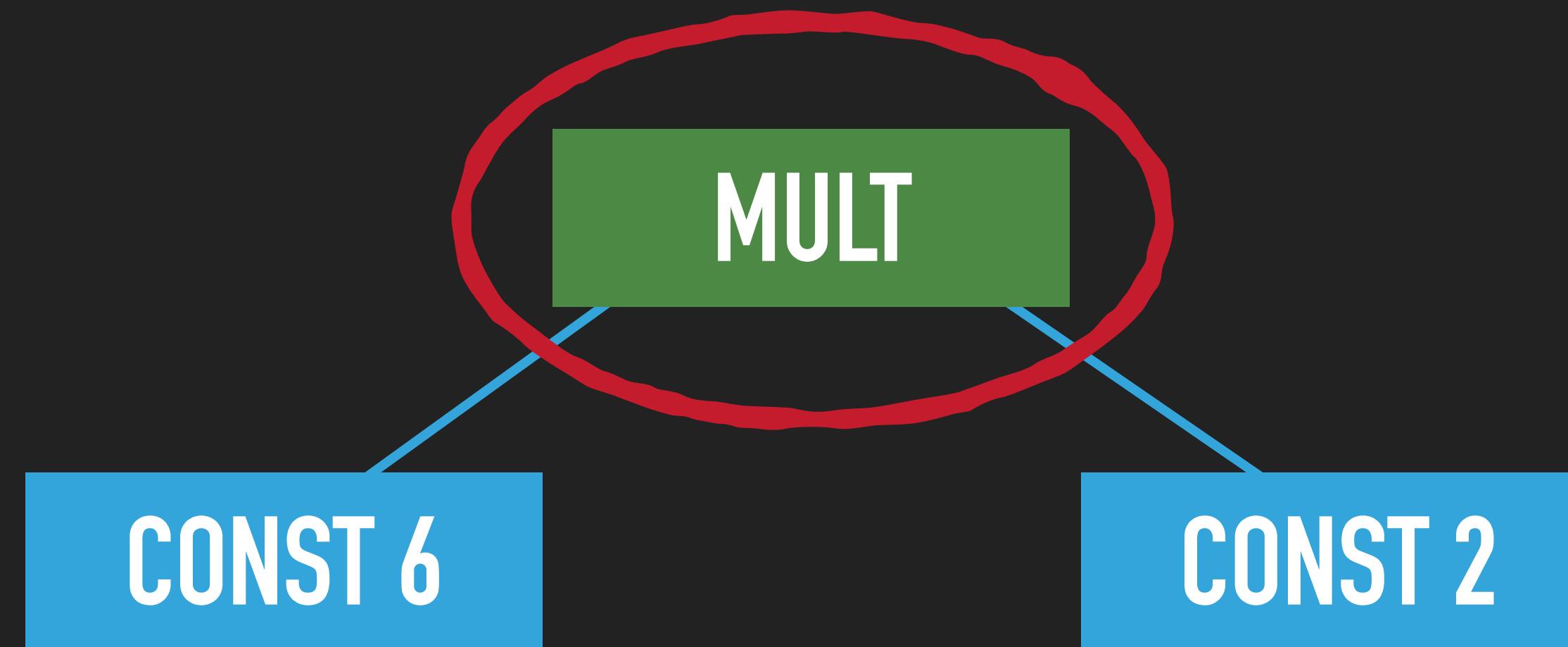
$\{ X = 1, Y = 2 \}$

EVALUATE



$\{ X = 1, Y = 2 \}$

EVALUATE



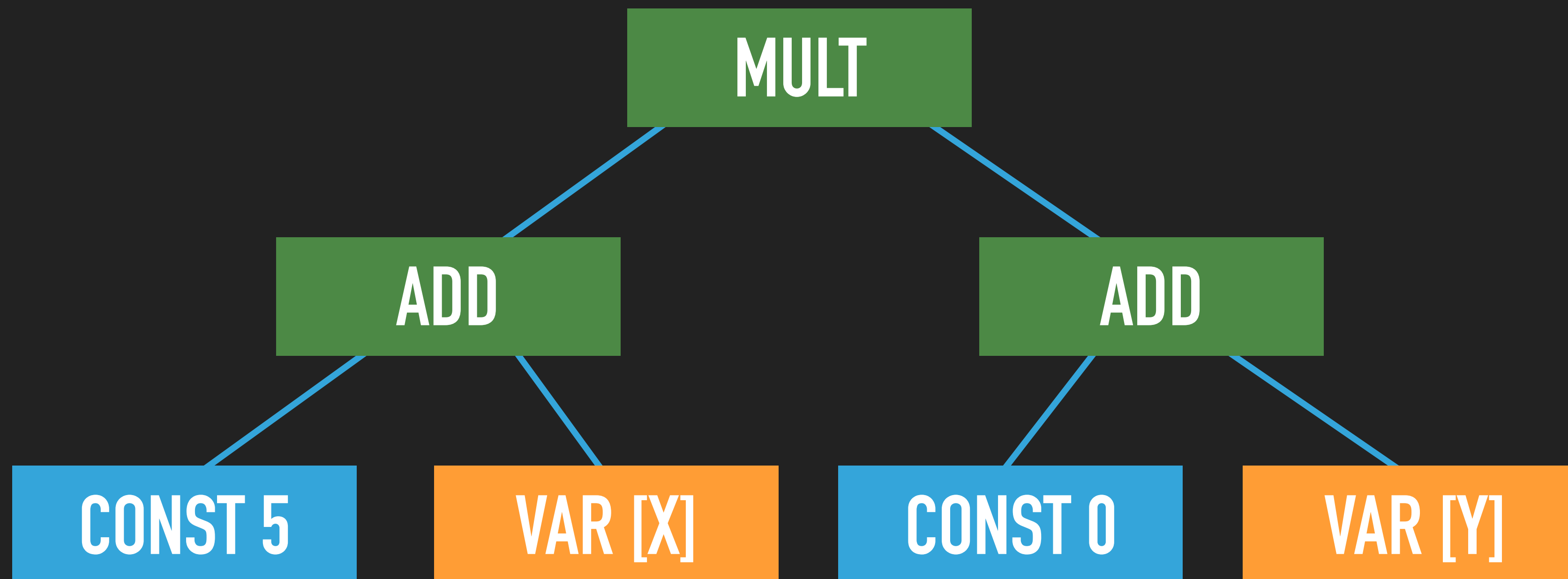
$\{ X = 1, Y = 2 \}$

EVALUATE

CONST 12

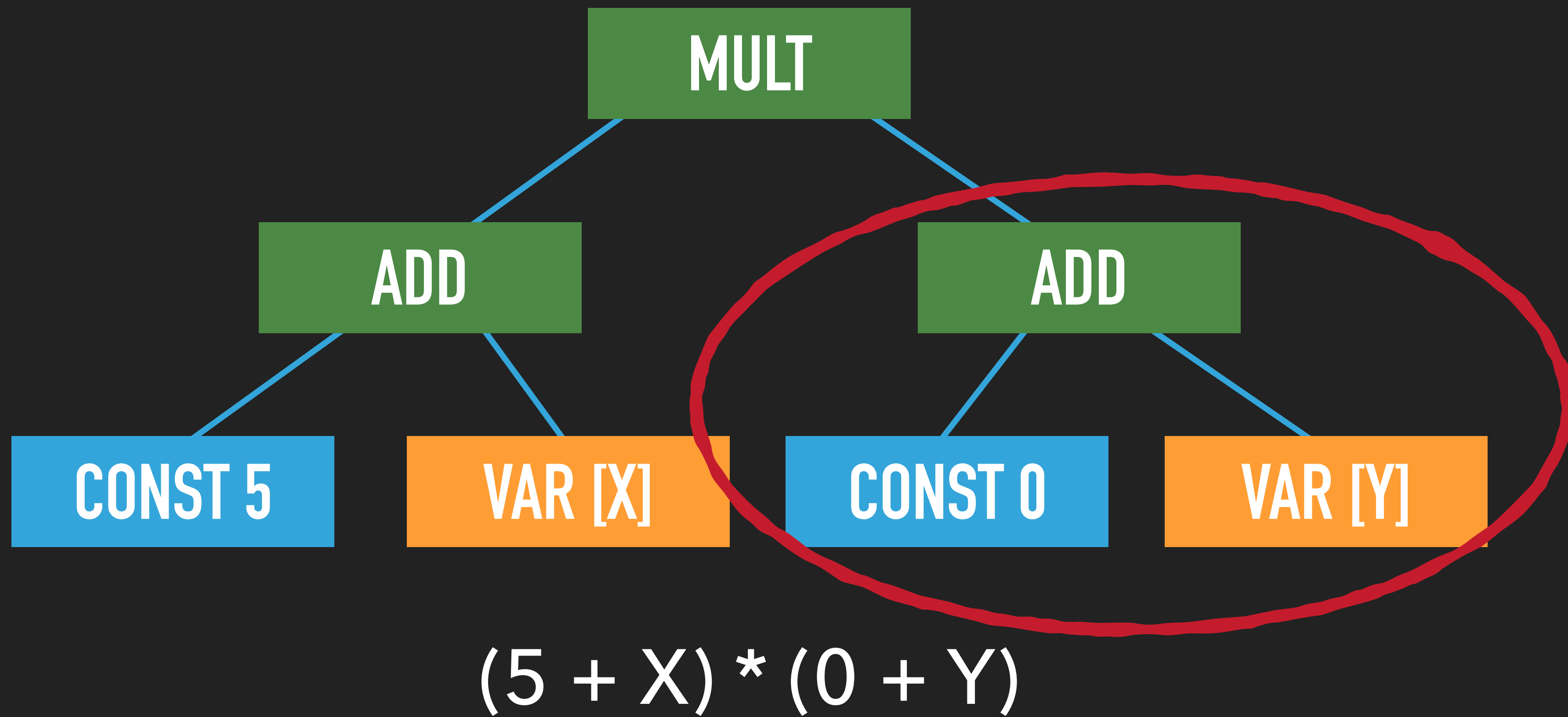
$\{ X = 1, Y = 2 \}$

OPTIMIZE

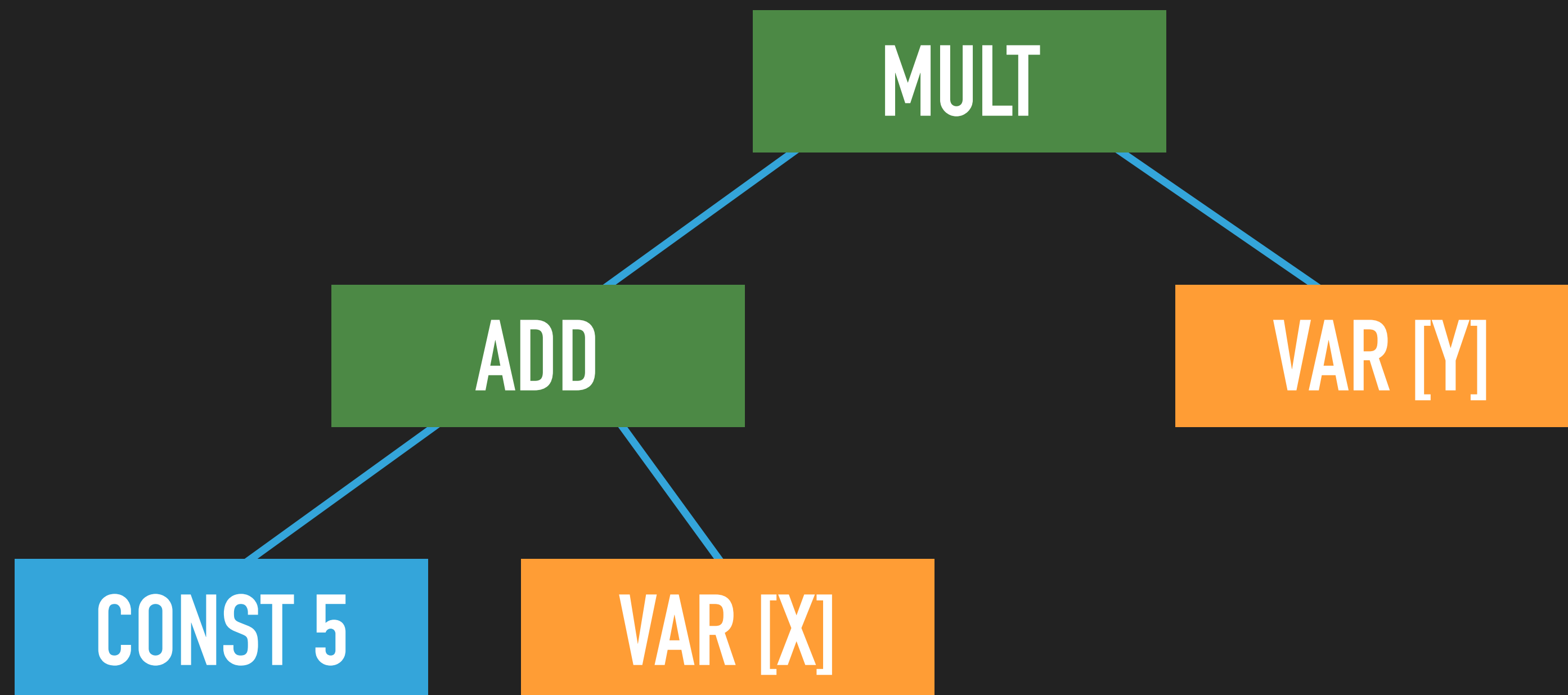


$(5 + X) * (0 + Y)$

OPTIMIZE



OPTIMIZE



$(5 + X) * Y$

HOW CAN I TEST THAT MESS?

TESTING INTERPRETERS

TEST RELATIONS, NOT INDIVIDUAL FUNCTIONS

- ▶ Given a random expression E
 - ▶ Extract its dependencies D
 - ▶ Assign value to each var of D
- ▶ evaluate(E, D) should be a constant

TEST RELATIONS, NOT INDIVIDUAL FUNCTIONS

`prop_deps_allow_eval :: Expr -> Property`

`prop_deps_allow_eval e =`

`forAll (makeEnvWith (dependencies e) $`

`\env -> isCst (eval env e)`

FOCUS ON EQUIVALENCE CLASS

- ▶ Given a random expression E
 - ▶ With no dependencies
- ▶ optimize(E) should be a constant
- ▶ evaluate($E, \{\}$) == optimize(E)

FOCUS ON EQUIVALENCE CLASS

`prop_optimize_constant :: Expr -> Property`

`prop_optimize_constant e =`

`Set.null (dependencies e)`

`==> isCst (optimize e)`

FOCUS ON EQUIVALENCE CLASS

*** Gave up! Passed only 41 tests.

FOCUS ON EQUIVALENCE CLASS

`prop_optimize_constant :: Property`

`prop_optimize_constant =`

`forAll (sized genCstExpr)`

`(isCst . optimize)`

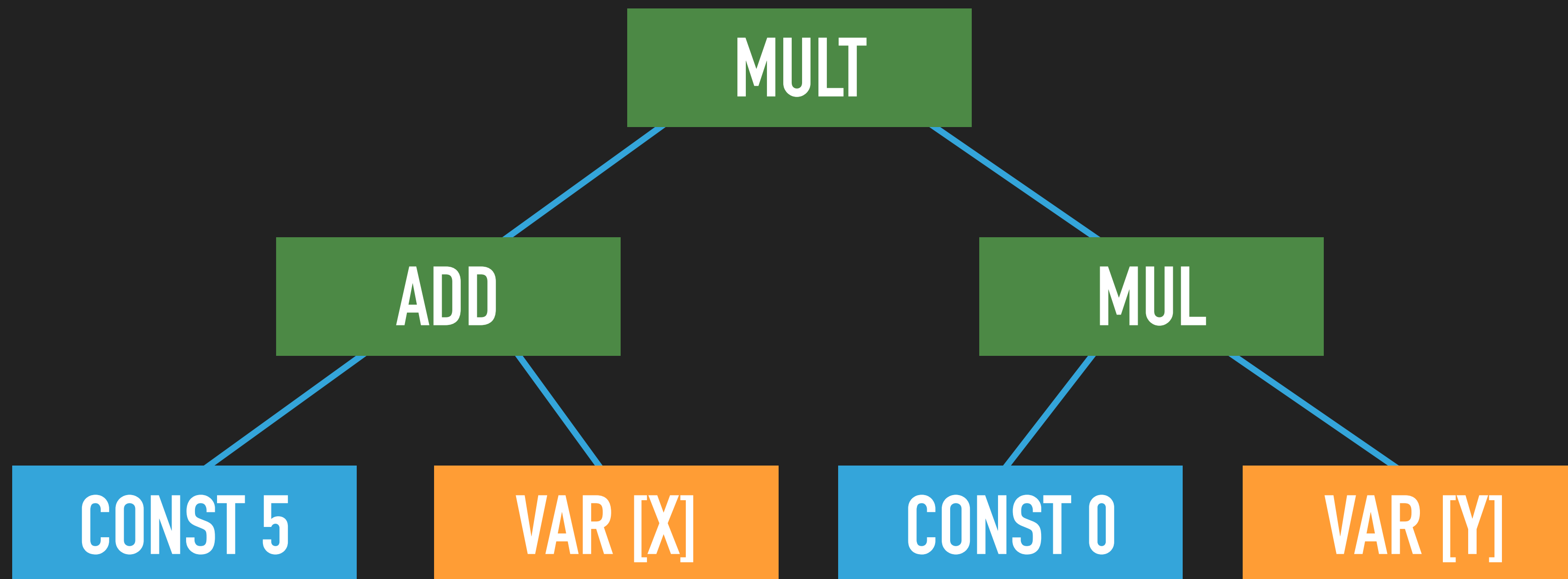
QUICK CHECK BY EXAMPLE

WHAT IF MY PROPERTY FAILS?

CONTRADICTION

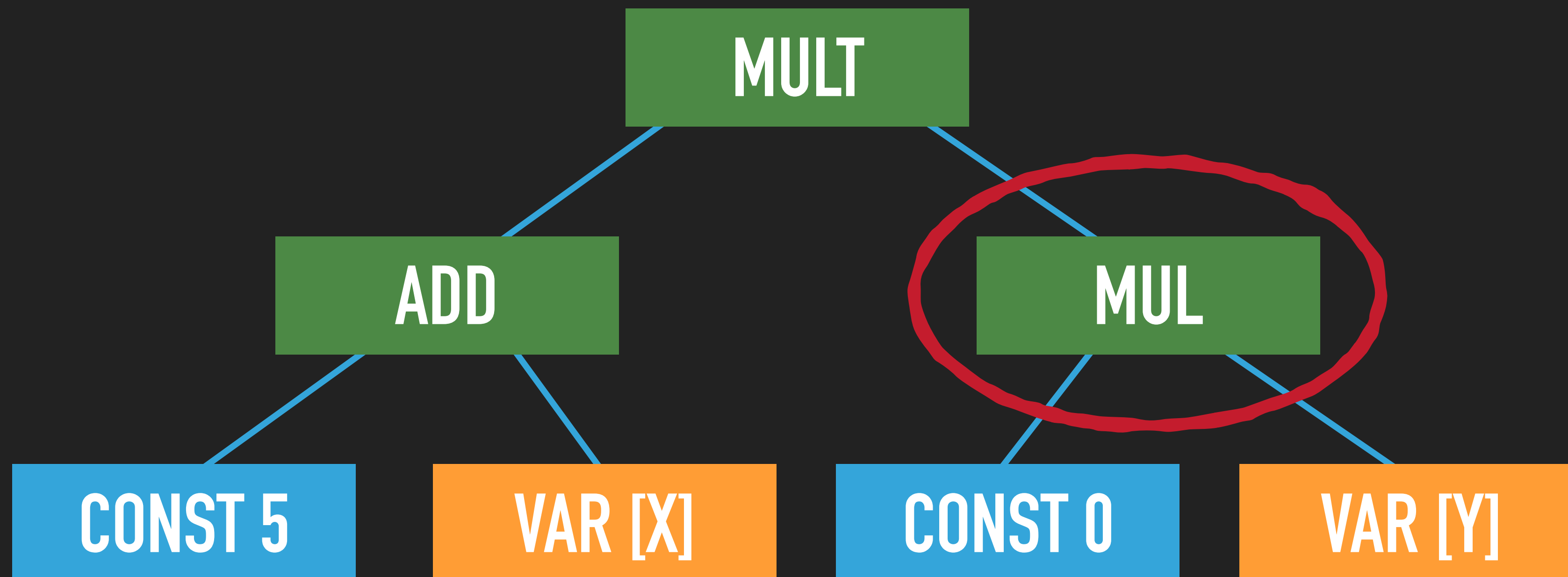
- ▶ Given a random expression E
 - ▶ Extract its dependencies D
 - ▶ Extract dependencies D' of optimize(E)
- ▶ We expect $D == D'$

CONTRADICTION



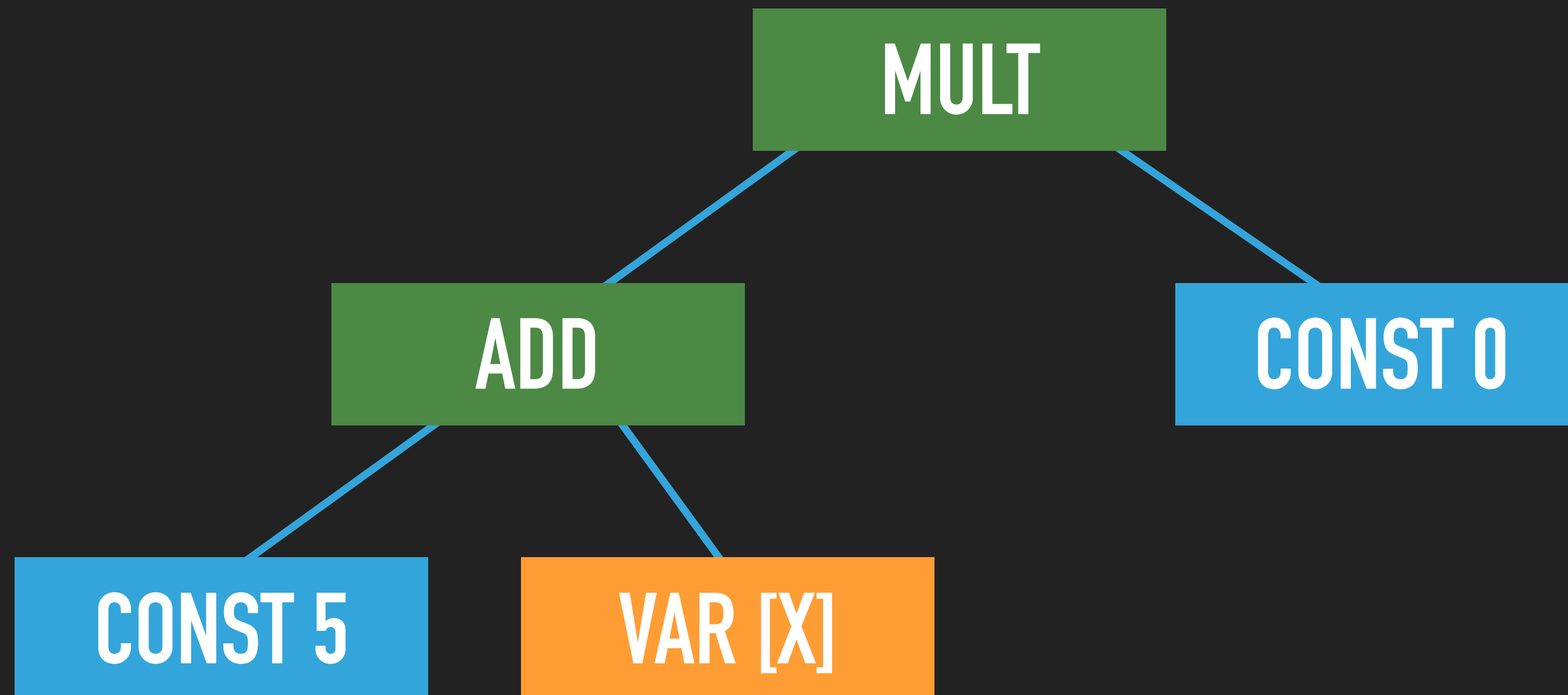
$(5 + X) * (0 * Y)$

CONTRADICTION



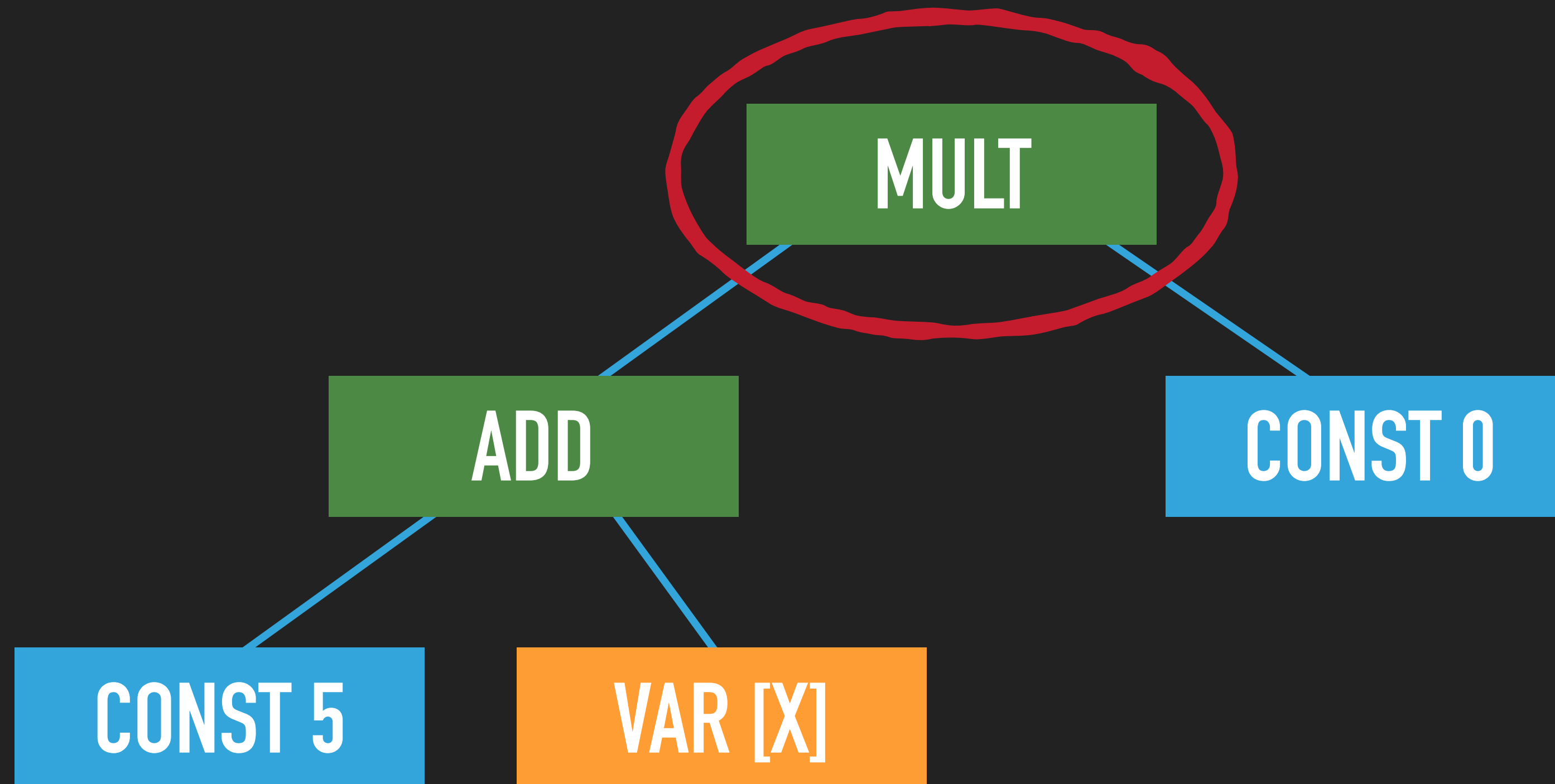
$(5 + X) * (0 * Y)$

CONTRADICTION



$$(5 + X) * 0$$

CONTRADICTION



$$(5 + X) * 0$$

CONTRADICTION

CONST 0

$\#\{ X, Y \} \neq \#\{ \}$

CONTRADICTION (VARIATION)

- ▶ Given a random expression E
 - ▶ A sub-set D of its dependencies
 - ▶ Assign to each variable of D a value
- ▶ $\text{evaluate}(E, D) == \text{evaluate}(\text{optimise}(E, D))$

EMBRACE CONTRADICTION

- ▶ Contradiction increases domain knowledge
 - ▶ Keep negative answers
 - ▶ Transmit them to domain experts
- ▶ You can keep them with expectFailure

QUICK CHECK BY EXAMPLE

CONCLUSION & LINKS

FINDING PROPERTIES

- ▶ Think **relations** not just individual results
- ▶ Use **equivalence** classes to find properties
- ▶ Ask **questions** and use **contradictions**

RESOURCES

- ▶ <http://www.cs.tufts.edu/~nr/cs257/archive/john-hughes/quick.pdf>
- ▶ <https://www.youtube.com/watch?v=zi0rHwfiX1Q>
- ▶ <https://deque.blog/2017/02/14/quickcheck-in-action/>
- ▶ <https://deque.blog/2017/02/17/quickcheck-is-fun-deal-with-it/>