HASKELL MEETUP - 27/04/2017

QUICKCHECK BY EXAMPLE

GOALS FOR TODAY

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

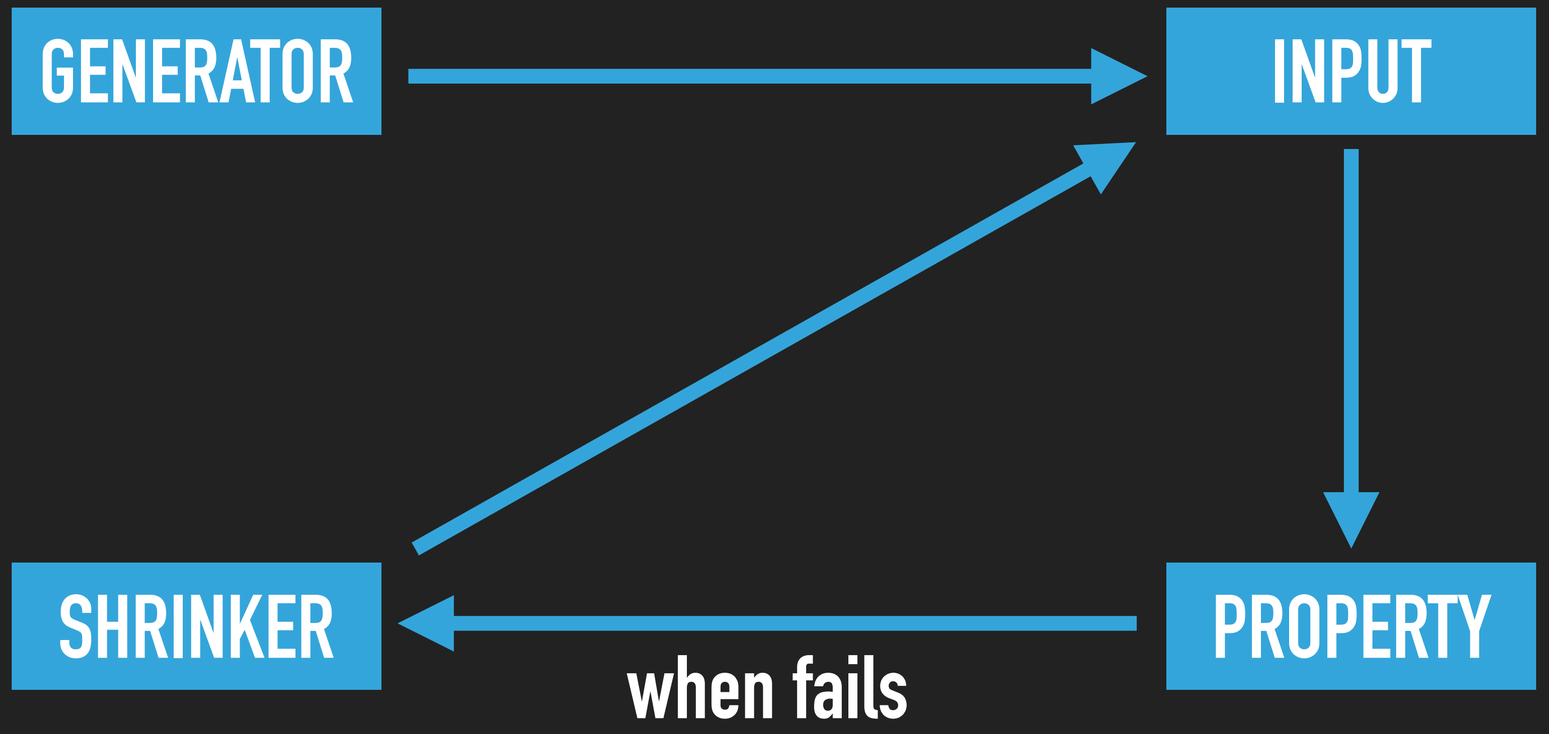
QUICK RECAP

WHAT IS QUICKCHECK?

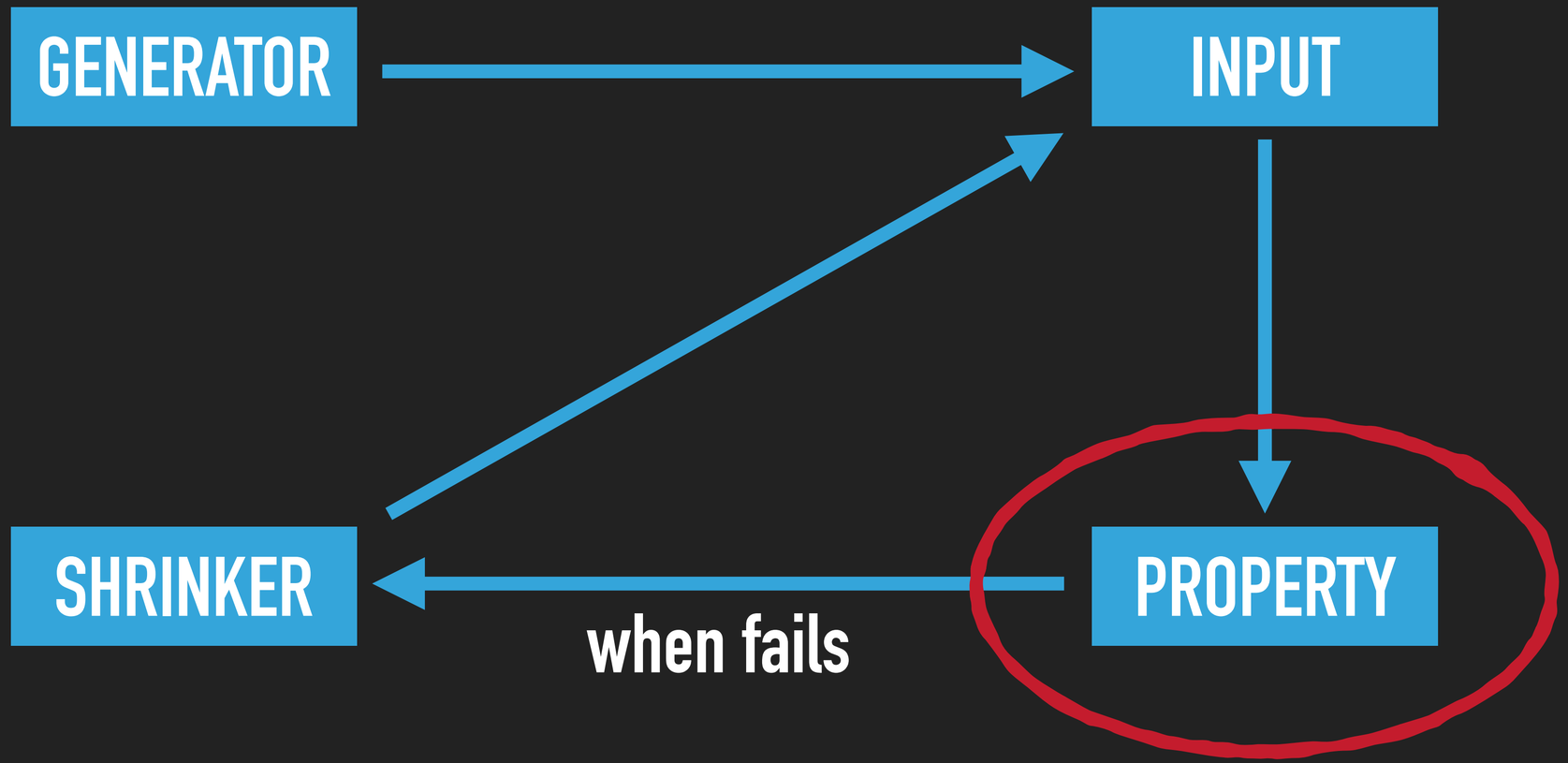
PROPERTY BASED TESTING – 101

- 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: do + undo = identity
  - Parse & Serialize

FINDING PROPERTIES

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

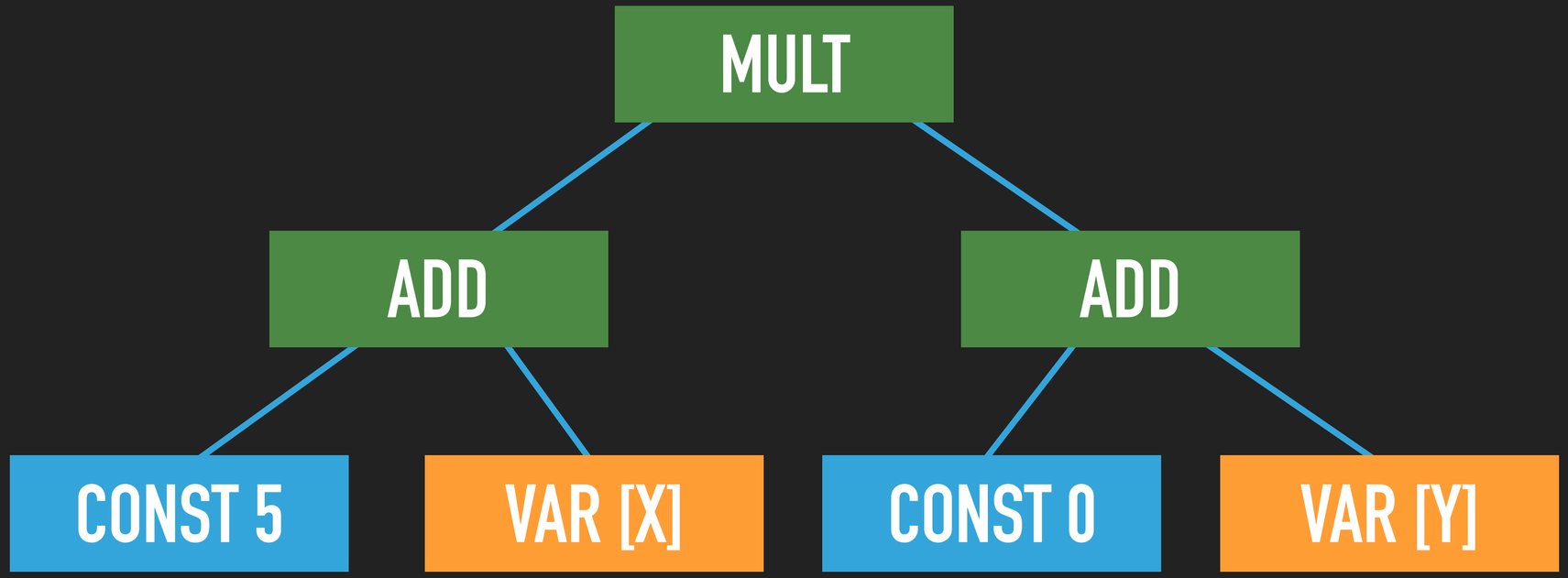
IN PRACTICE…

QUICKCHECK BY EXAMPLE

ARITHMETIC DSL

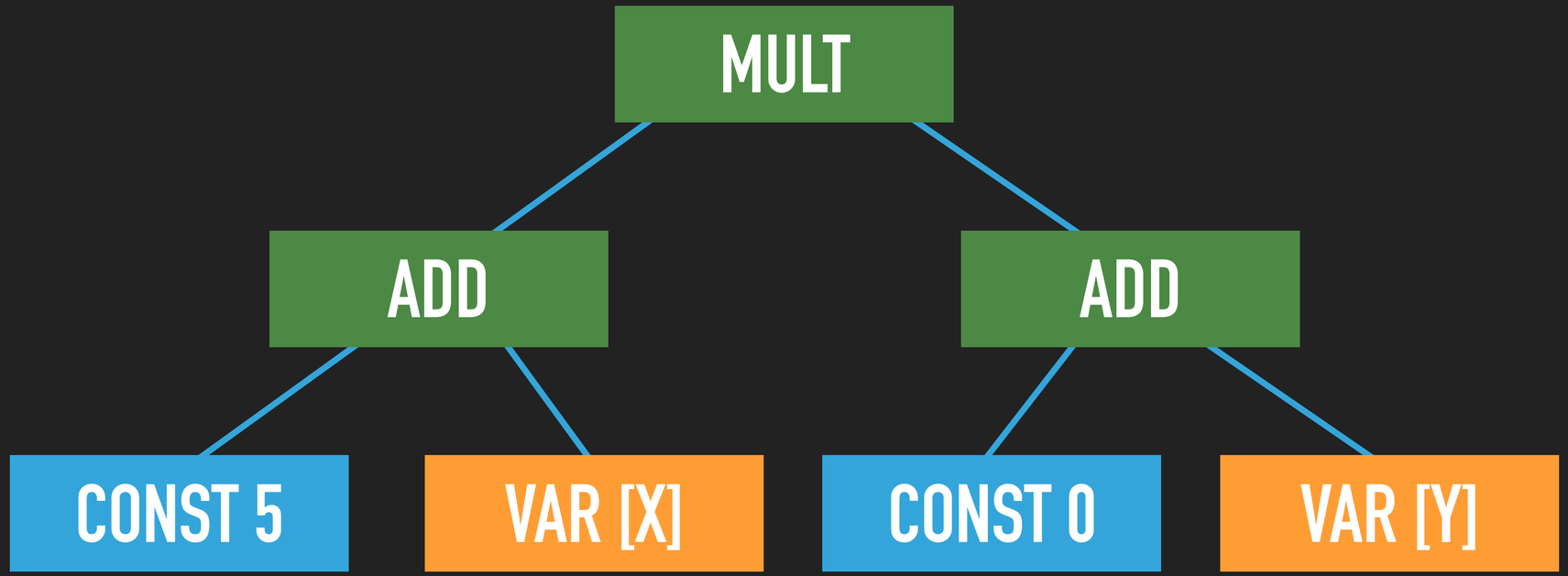
- Integer, Variable, (+) and (*)
- Goal: testing interpreters
  - Dependency
  - Evaluation
  - Optimization

ARITHMETIC EXPRESSION

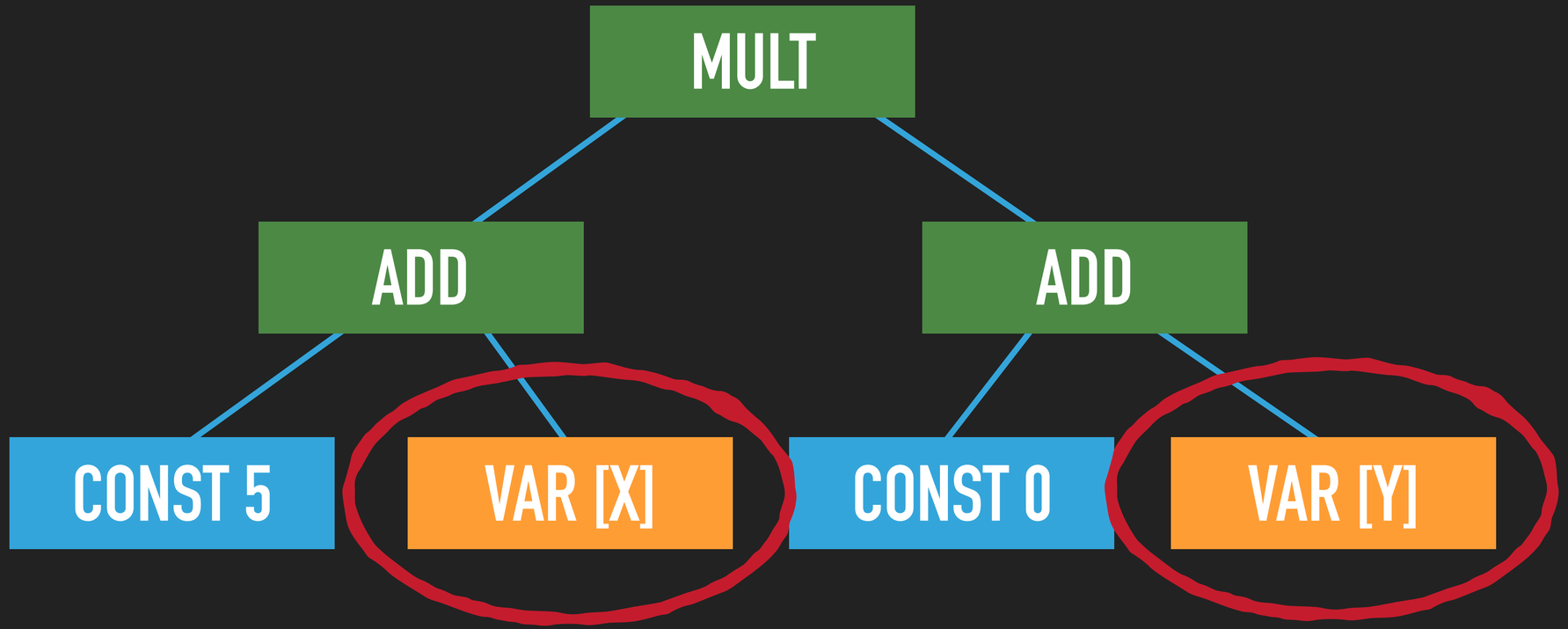


(5 + X) * (0 + Y)

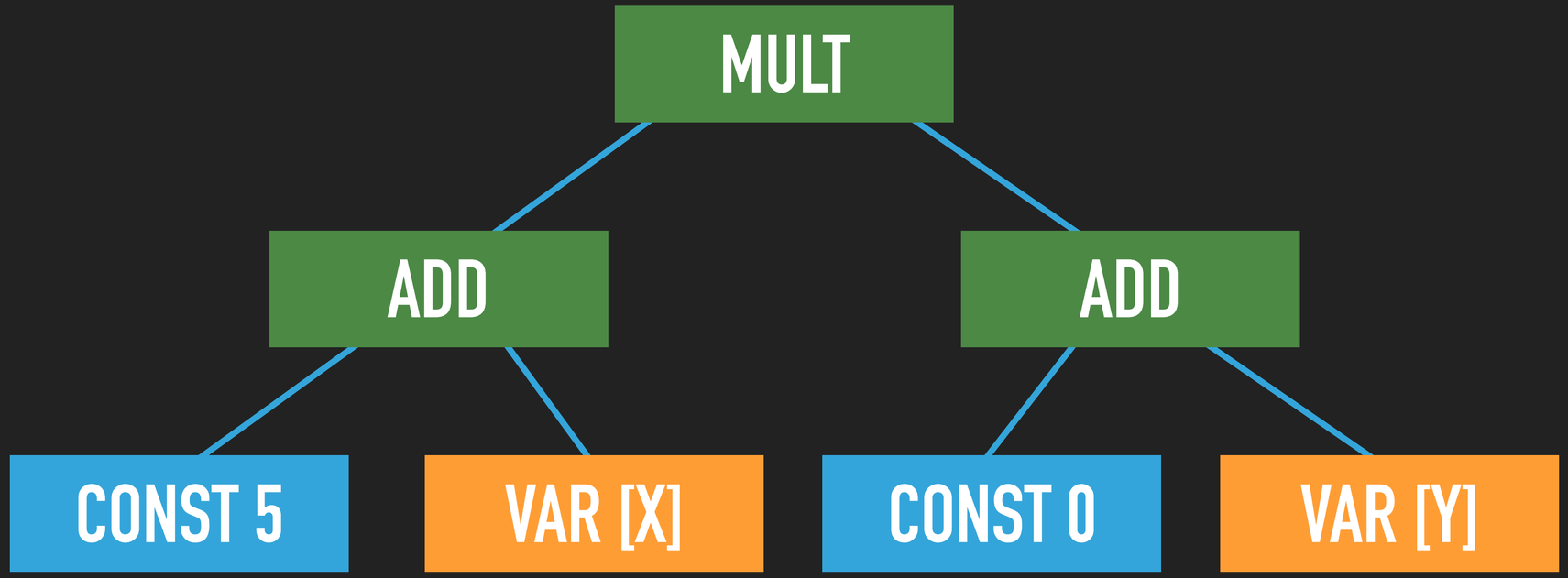
DEPENDENCIES



DEPENDENCIES

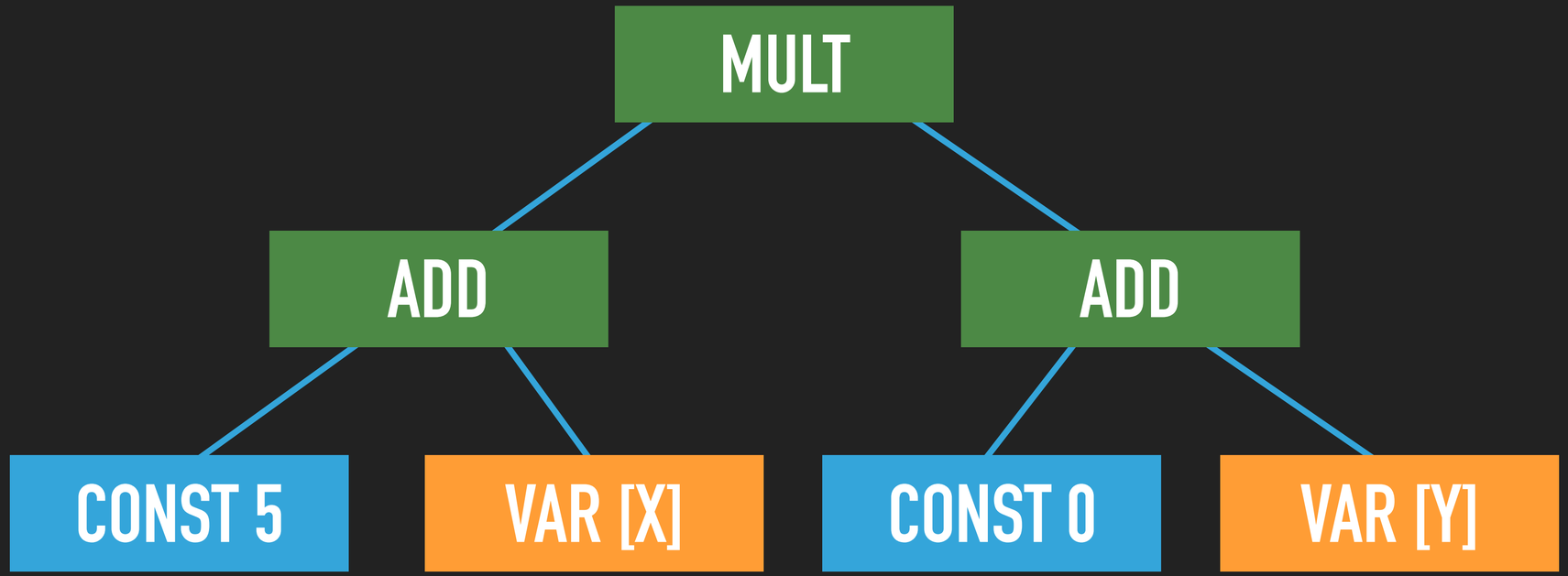


DEPENDENCIES

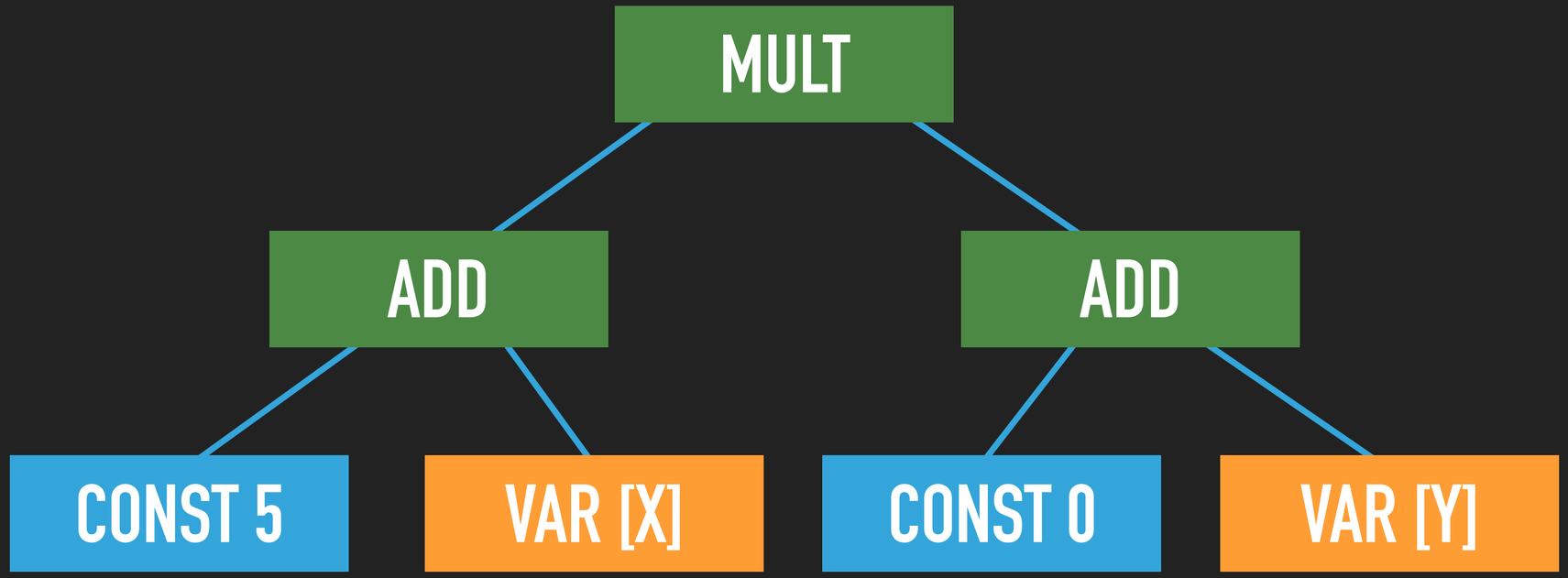


#{ X, Y }

EVALUATE

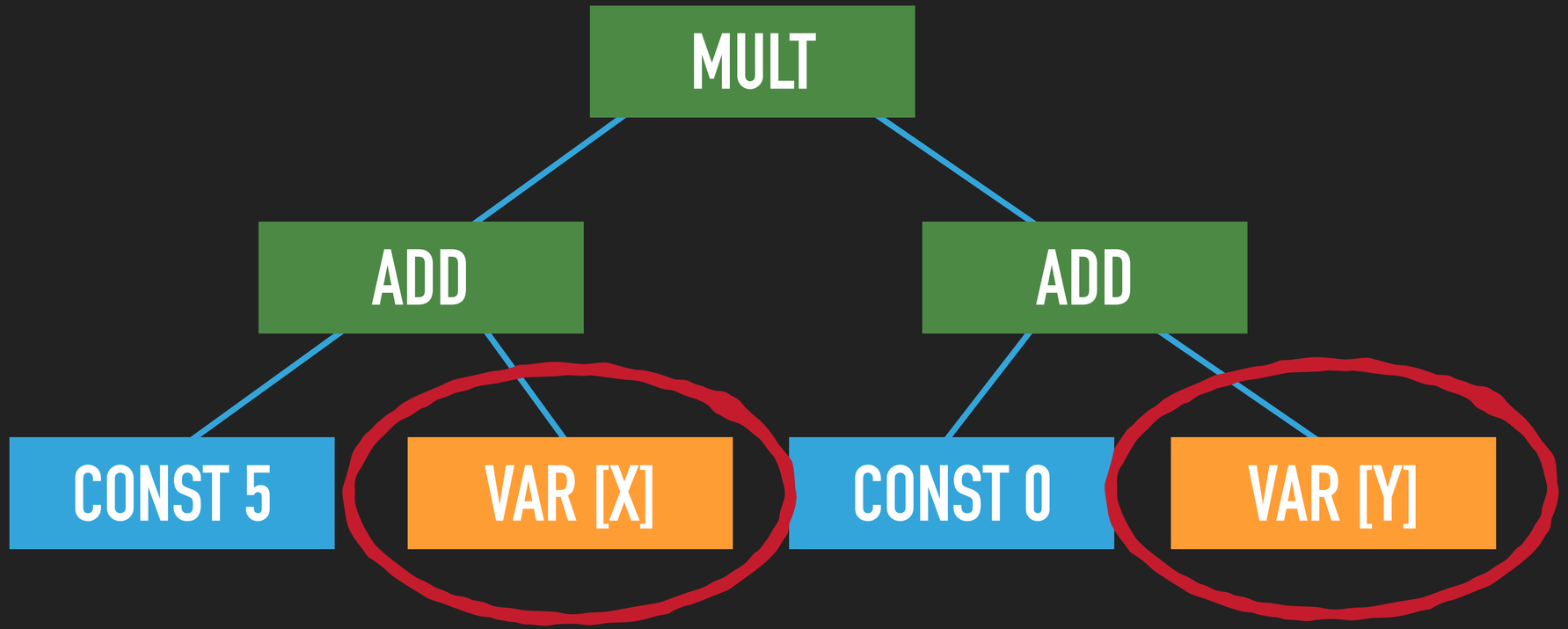


EVALUATE



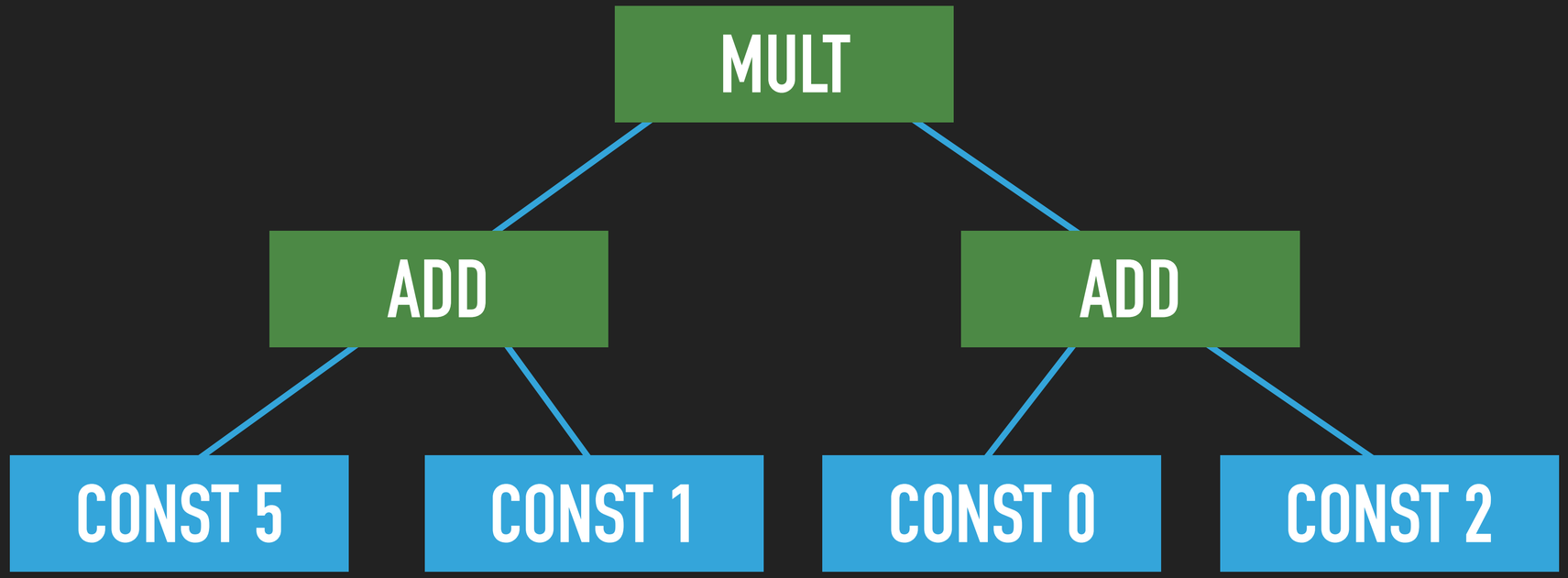
{ X = 1, Y = 2 }

EVALUATE



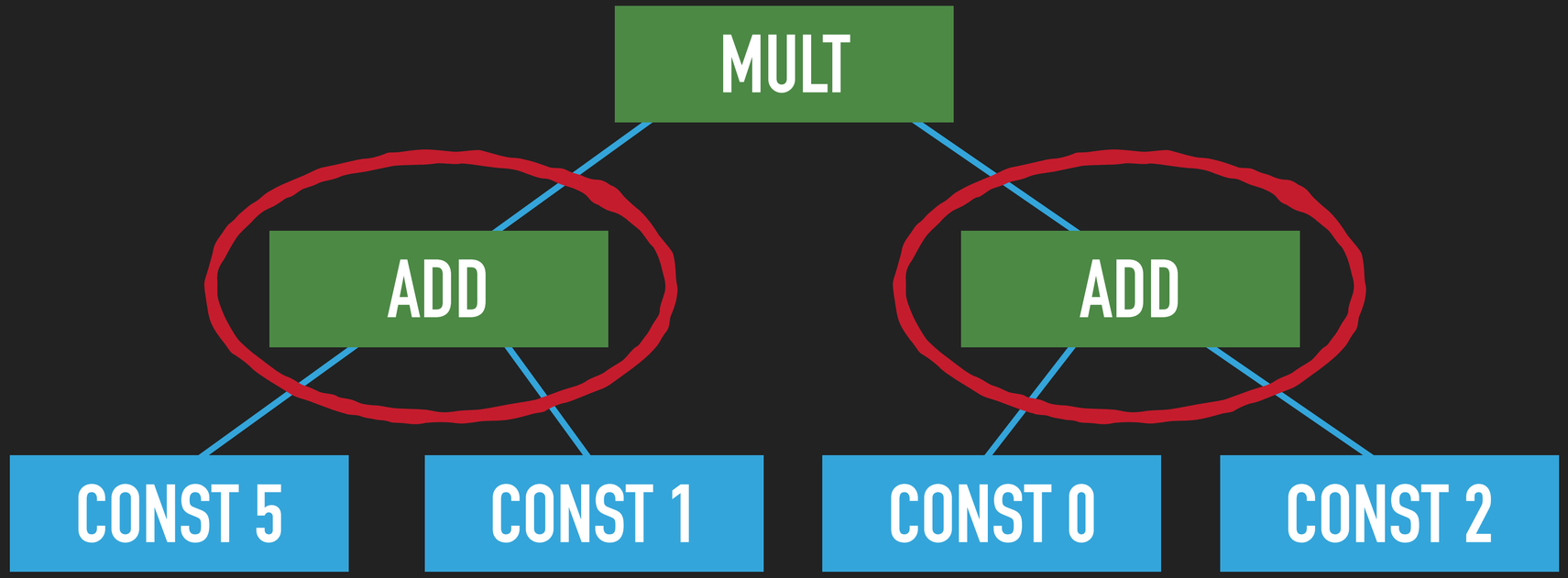
{ X = 1, Y = 2 }

EVALUATE



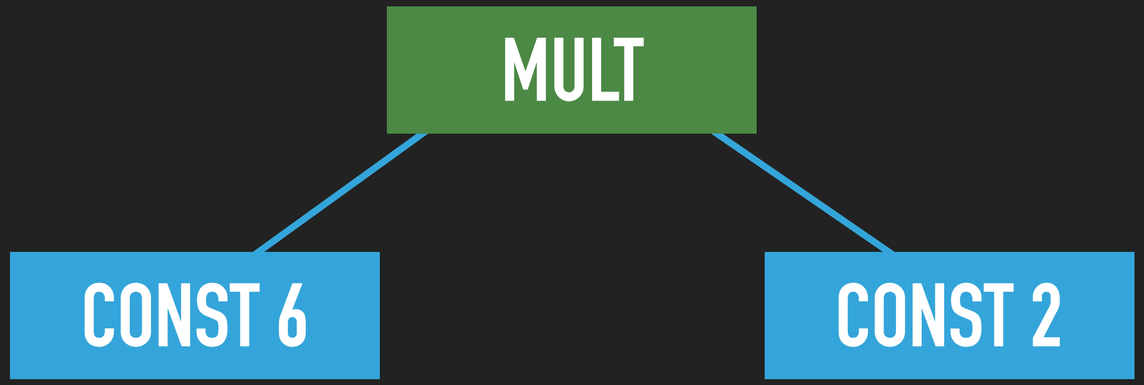
{ X = 1, Y = 2 }

EVALUATE



{ X = 1, Y = 2 }

EVALUATE



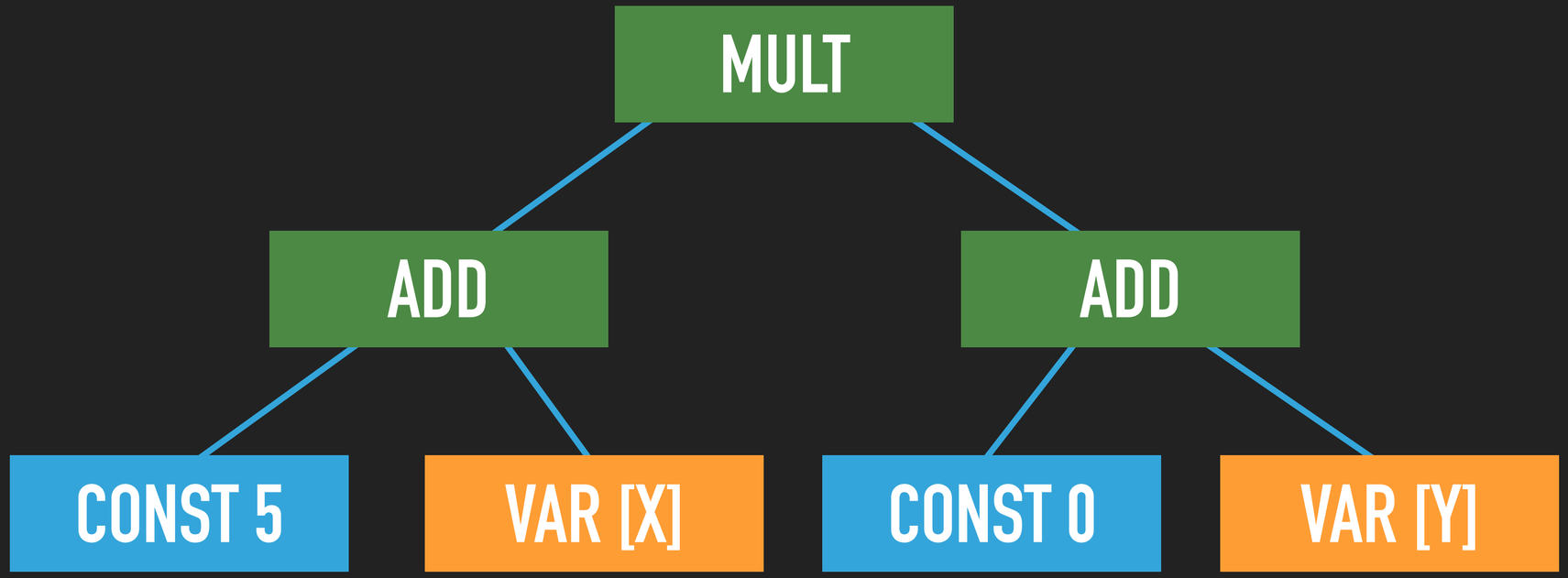
{ X = 1, Y = 2 }

EVALUATE

CONST 12

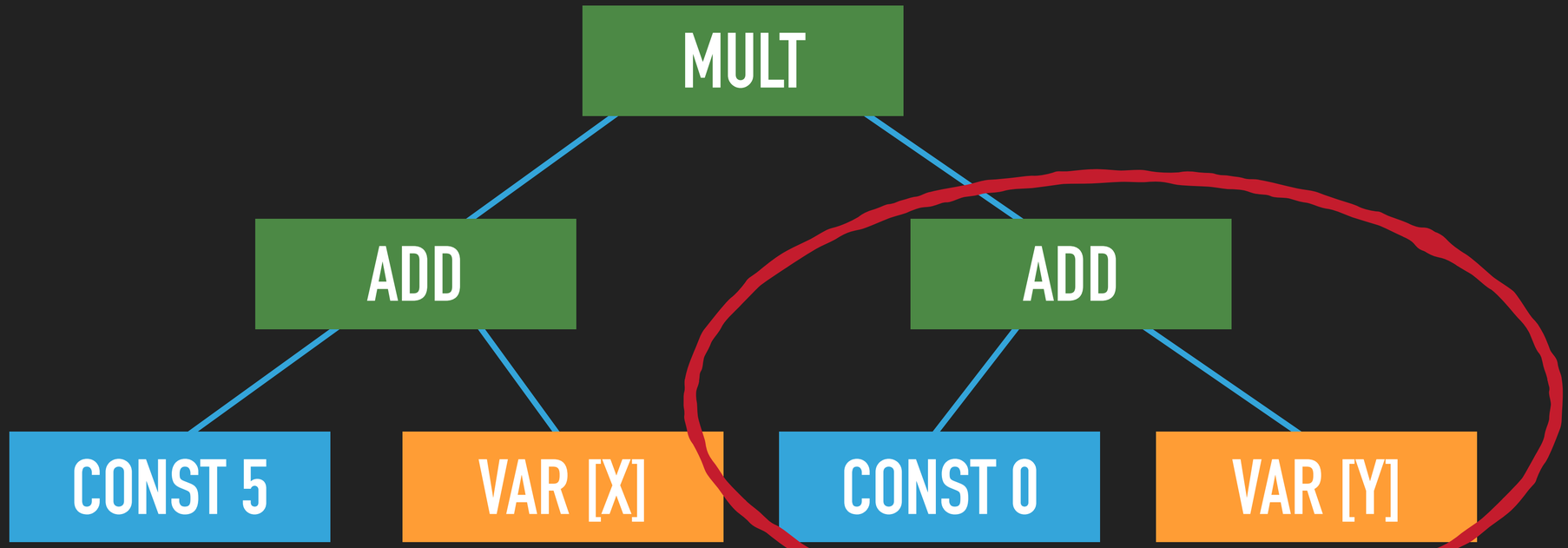
{ X = 1, Y = 2 }

OPTIMIZE



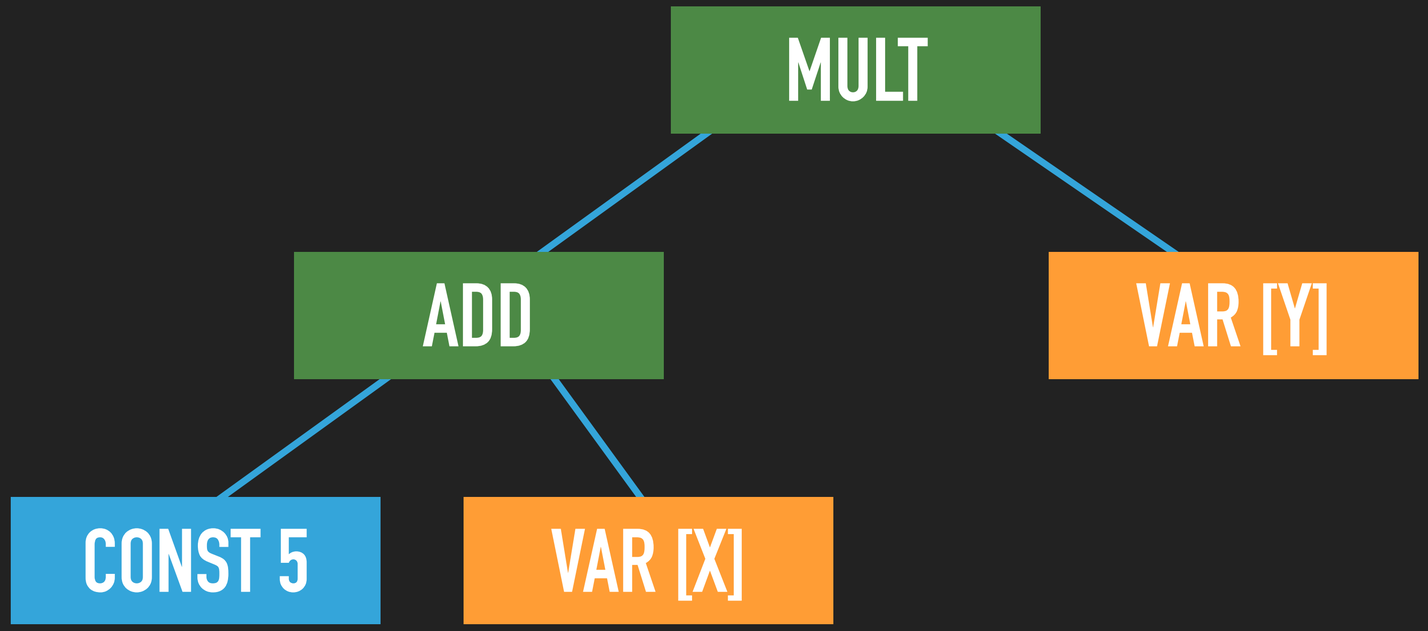
(5 + X) * (0 + Y)

OPTIMIZE



(5 + X) * (0 + Y)

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_full_eval_with_all_deps =
  forAll genExpr $ \e ->
    forAll (makeEnvWith (dependencies e)) $ \d ->
      isCst (eval d e)
```

FOCUS ON EQUIVALENCE CLASS

- Given a random expression E
  - With no dependencies

- optimize(E) should be a constant
- evaluate(E, {}) == optimize(E)

NARROW EQUIVALENCE CLASS HAVE MORE PROPERTIES

- Write your own custom generator:

```
prop_optimize_constant :: Property
prop_optimize_constant =
  forAll (sized genCstExpr) (isCst . optimize)
```

QUICK CHECK BY EXAMPLE

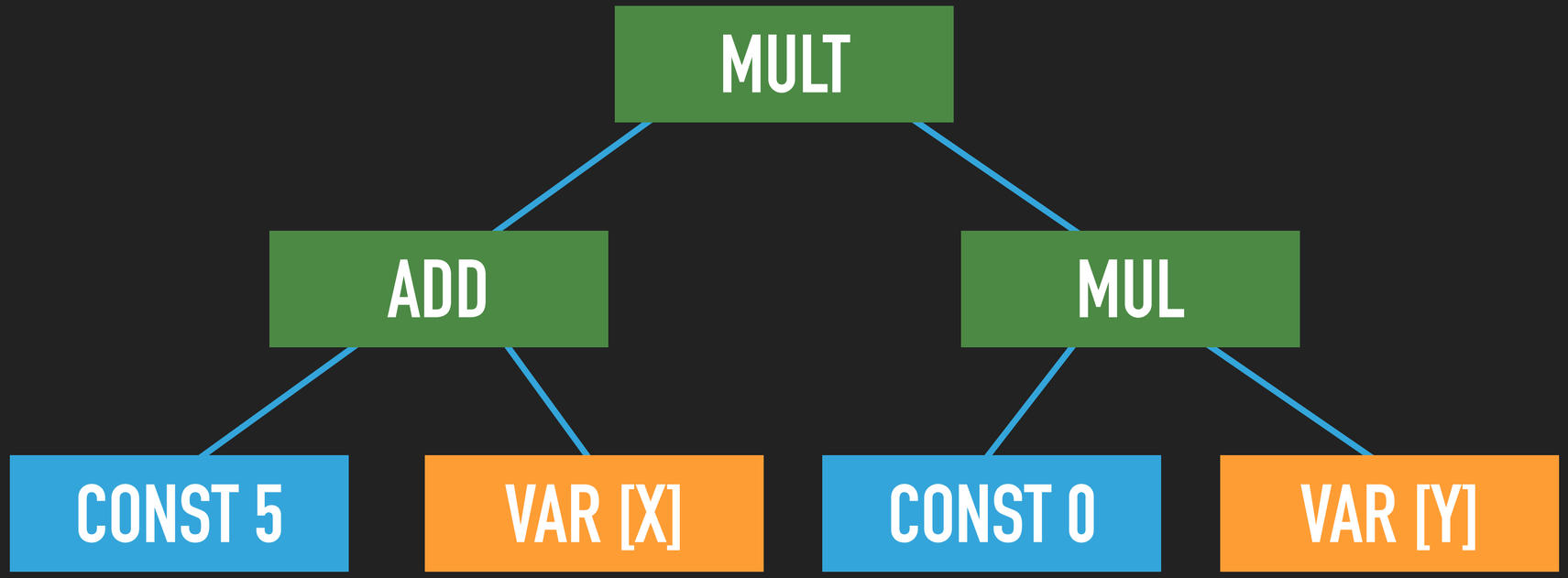
WHAT IF MY PROPERTY FAILS?

CONTRADICTION

- Given a random expression E
  - Extract its dependencies D
  - Extract dependencies D′ of optimize(E)

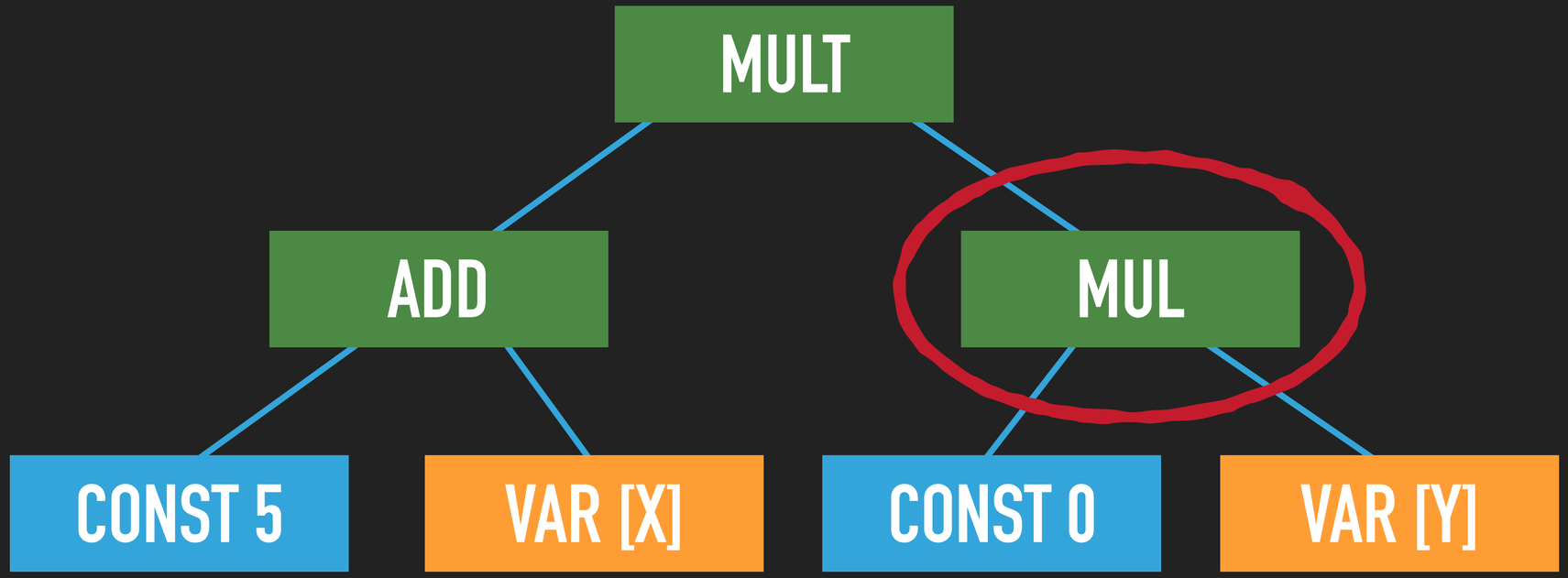
- We expect D == D′

CONTRADICTION



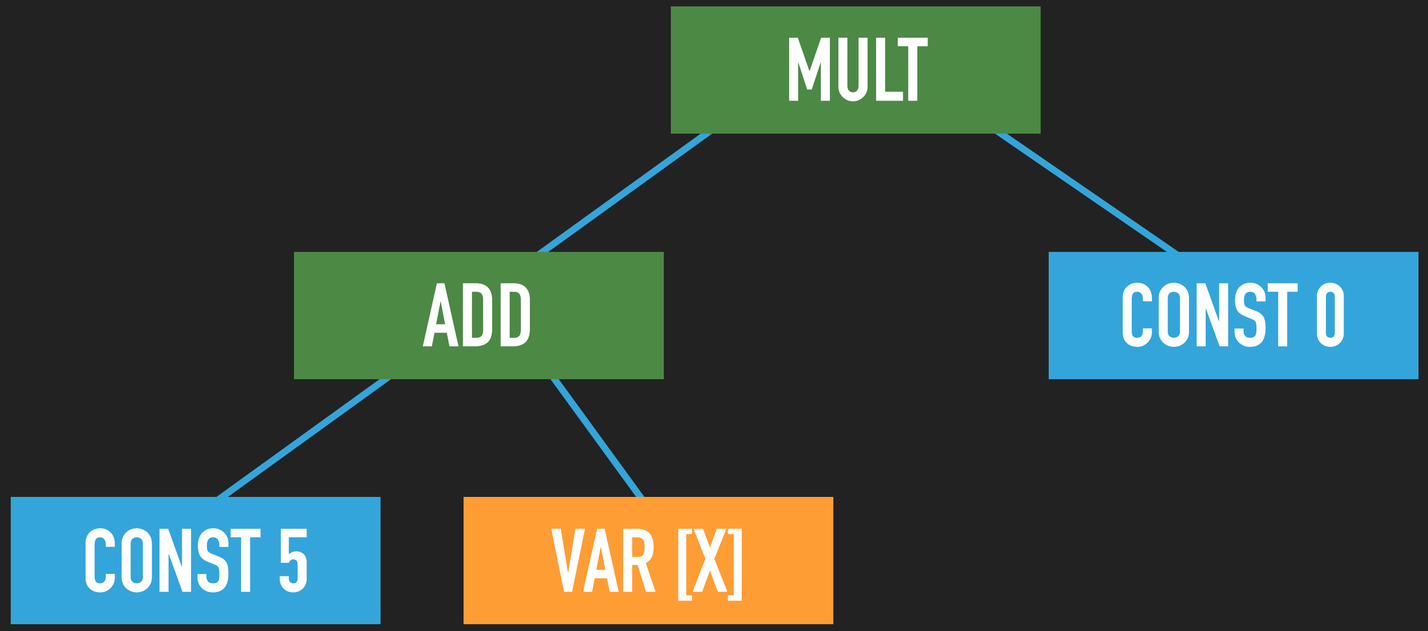
(5 + X) * (0 * Y)

CONTRADICTION



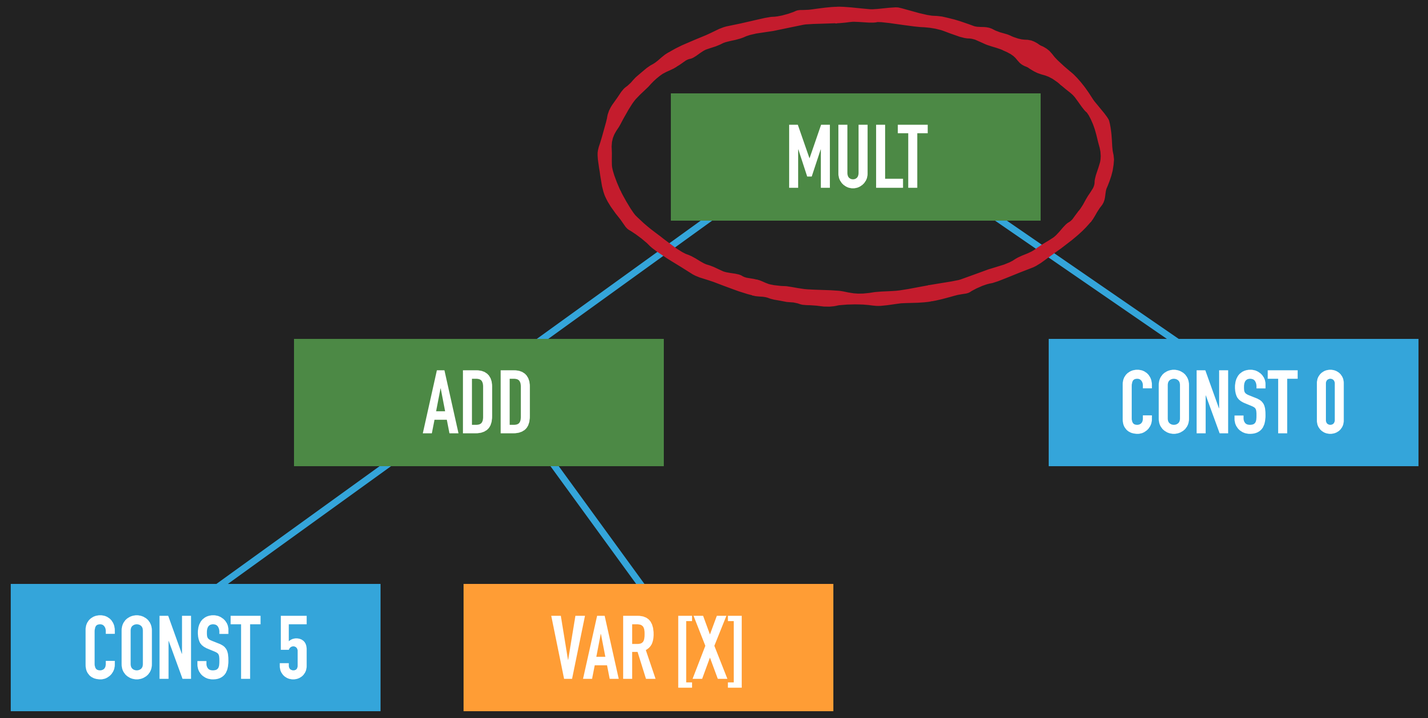
(5 + X) * (0 * Y)

CONTRADICTION



(5 + X) * 0

CONTRADICTION



(5 + X) * 0

CONTRADICTION

CONST 0

#{ X, Y } /= #{}

CONTRADICTION (VARIATION)

- Given a random expression E
  - A sub-set D of its dependencies
  - Assign to each variable of D a value

- evaluate(E,D) == evaluate(optimise(E,D))

EMBRACE CONTRADICTION

- Contradiction increases domain knowledge
  - Not unusual, even for simple code
  - Keep negative answers to questions

- You can keep them with expectFailure

QUICK CHECK BY EXAMPLE

CONCLUSION AND OPENING

FINDING PROPERTIES

- Think relations not just individual results
- Use equivalence classes to find properties
- Use contradictions as learning tools

RESOURCES

- http://www.cs.tufts.edu/~nr/cs257/archive/john-hughes/quick.pdf
- https://hackage.haskell.org/package/QuickCheck-2.9.2/docs/Test-QuickCheck.html
- https://deque.blog/2017/02/14/quickcheck-in-action/
- https://deque.blog/2017/02/17/quickcheck-is-fun-deal-with-it/