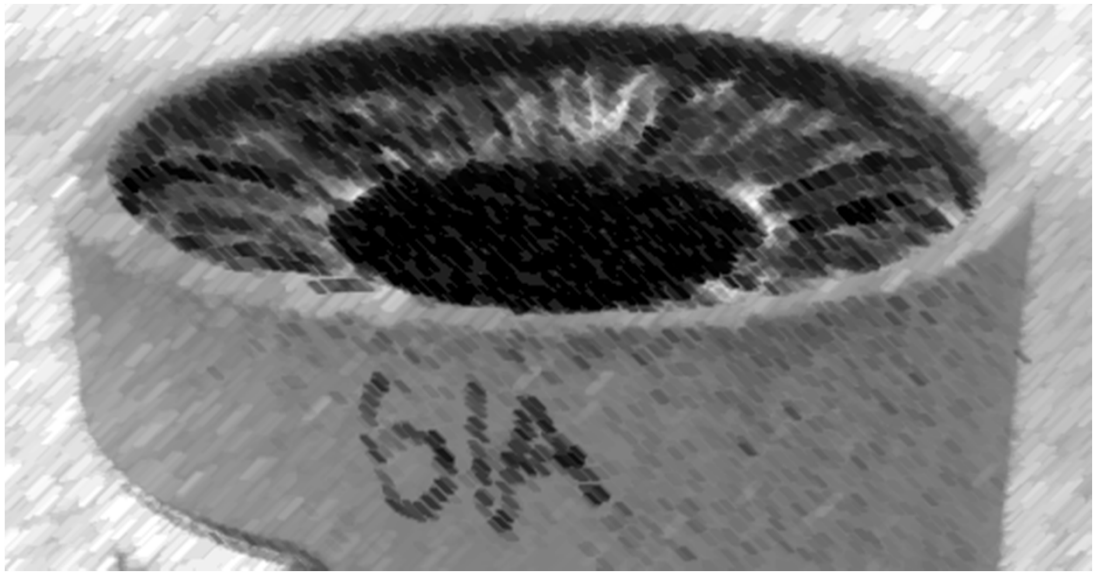


CS61A Lecture 1

Amir Kamil
UC Berkeley
January 23, 2013

Welcome to CS61A!

The Course Staff

Amir Kamil

I've been at Berkeley a long time, and took CS61A a while back. Read the course info to find out when!

TAs essentially run the course

Hamilton Nguyen

Email: cs61a-ta

Joy Jeng

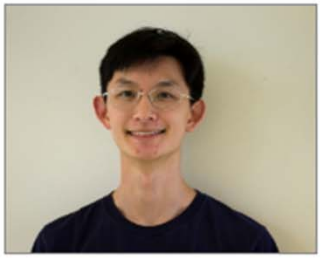
Email: cs61a-ti

Keegan Mann

Email: cs61a-tk

Stephen Martinis

Email: cs61a-tc



Albert Wu

Email: cs61a-tg

Julia Oh

Email: cs61a-te

Robert Huang

Email: cs61a-th

Mark Miyashita

Email: cs61a-td

Sharad Vikram

Email: cs61a-tj

Soumya Basu

Email: cs61a-tf

Richard Hwang

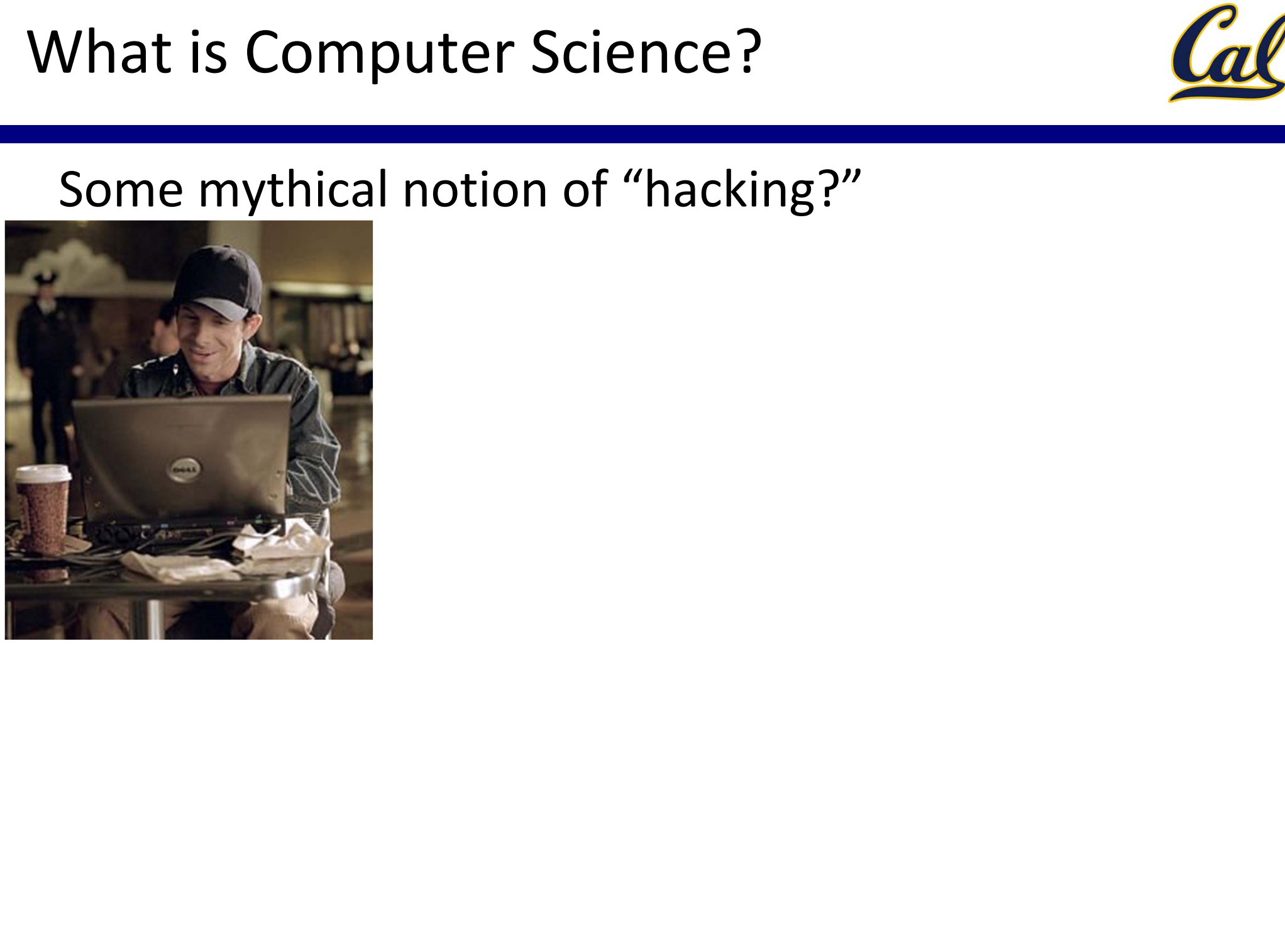
Email: cs61a-tb

Readers, lab assistants help you learn the material

What is Computer Science?

Some mythical notion of “hacking?”

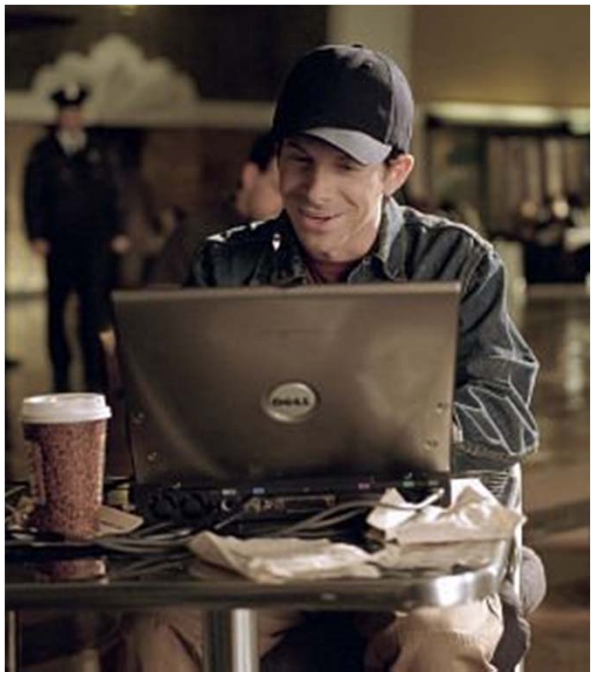
What is Computer Science?



Some mythical notion of “hacking?”

What is Computer Science?

Some mythical notion of “hacking?”



What is Computer Science?

Some mythical notion of “hacking?”

What is Computer Science?

Some mythical notion of “hacking?”

What is Computer Science?

"Computer science deals with the theoretical foundations of information and computation, together with practical techniques for the implementation and application of these foundations"

- Wikipedia

Computer Science is Everywhere

Computer Science is Everywhere

Phones

Cars

Politics

Games

Movies

Music

Sports

Anything connected to the Internet

…

Computer Science is Everywhere

Phones

Cars

Politics

Games

Movies

Music

Sports

Anything connected to the Internet

…

Systems

Programming Languages

Graphics

Artificial Intelligence

Databases

Theory

Security

Parallel Computing

Quantum Computing

…

What is CS61A?

- An introduction to the "big ideas" in programming
  - Functions, data structures, recursion, interpretation, parallelism, …
- We use Python as our programming vehicle in this course, but the ideas apply to any language
- General focus: how to manage complexity
  - Primary tool is *abstraction*

What is Abstraction?

What is Abstraction?

- Abstraction is exposing the *what* of something while hiding the *how*

What is Abstraction?

- Abstraction is exposing the *what* of something while hiding the *how*

- Many layers of abstraction in a typical system

What is Abstraction?

- Abstraction is exposing the *what* of something while hiding the *how*

- Many layers of abstraction in a typical system

Application
Libraries (Graphics, Physics)
Operating System
Hardware (CPU, RAM, etc.)
Logic Gates

What is Abstraction?

- Abstraction is exposing the *what* of something while hiding the *how*

- Many layers of abstraction in a typical system

Application
Libraries (Graphics, Physics)
Operating System
Hardware (CPU, RAM, etc.)
Logic Gates

- This course will teach you how to build and use abstractions

Course Policies

The purpose of this course is to help you learn

The staff is here to make you successful

All the details are on the website:

http://inst.eecs.berkeley.edu/~cs61a/sp13/about.html

Ask questions on Piazza

https://piazza.com/class#spring2013/cs61a

Course Organization

- **Readings** cover the material; read before lecture
- **Lectures** summarize material, present in new way
- **Labs** introduce new topics or practical skills
- **Discussions** provide practice on the material
- **Homeworks** are deeper exercises that require more thought than labs
  - Graded on effort, generally due Wed. at 11:59pm
- **Projects** are larger assignments designed to teach you how use and combine ideas from the course in interesting ways

Collaboration

- Discuss everything with each other
- EPA: Effort, participation, and altruism
- Homework may be completed with a partner
- Projects should be completed with a partner
- Find a project partner in your section!

The limits of collaboration

- Never share code
- Copying projects is a serious offense, and we **will** find out if you do

FAQ

- Both lectures are the same; you may attend either, space permitting
- Lectures are webcast; link will be online soon
- Midterms are on 2/13 and 3/21
- Final exam is 5/14 for both lectures
  - Let us know ASAP if you have a conflict with any exam
- See the Course Info for enrollment issues
- If you are on the waitlist, still complete assignments!

Announcements

- Make sure you have an account form and register
  - You will need one to submit homework and projects
  - Get one in discussion or office hours if you don't have one
- Office hours start tomorrow
  - See website schedule
- Study session Wed. 9:30-11:30am in the Woz
  - Mega office hours with multiple staff members present
  - Opportunities for collaboration and EPA
- Homework 0 due Fri. at 7pm
- Homework 1 due Wed. at 11:59pm

Data, Functions, and Interpreters

Data: the things that programs fiddle with

Data, Functions, and Interpreters

Data: the things that programs fiddle with

"Super Bowl XLVII"

2

Shakespeare's 37 plays

Mike Krzyzewski

Data, Functions, and Interpreters

Data: the things that programs fiddle with

"Super Bowl XLVII"

2

Shakespeare's 37 plays

Mike Krzyzewski

Functions: rules for manipulating data

Data, Functions, and Interpreters

Data: the things that programs fiddle with

"Super Bowl XLVII"

2

Shakespeare's 37 plays

Mike Krzyzewski

Functions: rules for manipulating data

Count the words in a line of text

Add up numbers

Pronounce someone's name

Data, Functions, and Interpreters

Data: the things that programs fiddle with

"Super Bowl XLVII"

2

Shakespeare's 37 plays

Mike Krzyzewski

Functions: rules for manipulating data

Count the words in a line of text

Add up numbers

Pronounce someone's name

Interpreter: an implementation of the procedure for evaluation

Primitive Values and Expressions

- Primitive values are the simplest type of data

Primitive Values and Expressions

- Primitive values are the simplest type of data
  - Integers: 2, 3, 2013, -837592010
  - Floating point (decimal) values: -4.5, 98.6
  - Strings: “It was a dark and stormy night”
  - Booleans: True, False

Primitive Values and Expressions

- Primitive values are the simplest type of data
  - Integers: 2, 3, 2013, -837592010
  - Floating point (decimal) values: -4.5, 98.6
  - Strings: “It was a dark and stormy night”
  - Booleans: True, False

- An *expression* is something that produces a value

Primitive Values and Expressions

- Primitive values are the simplest type of data

  Integers: 2, 3, 2013, -837592010

  Floating point (decimal) values: -4.5, 98.6

  Strings: “It was a dark and stormy night”

  Booleans: True, False

- An *expression* is something that produces a value

  2 + 3

  sqrt(2401)

  abs(-128 + 42 * 3)

Call Expressions in Python

- All expressions can use function call notation

Call Expressions in Python

- All expressions can use function call notation

2 + 3	add(2, 3)
sqrt(2401)	sqrt(2401)
abs(-128 + 42 * 3)	abs(add(-128, mul(42, 3)))

Call Expressions in Python

- All expressions can use function call notation

2 + 3	add(2, 3)
sqrt(2401)	sqrt(2401)
abs(-128 + 42 * 3)	abs(add(-128, mul(42, 3)))

- Infix operator notation is *syntactic sugar* for function calls

Call Expressions in Python

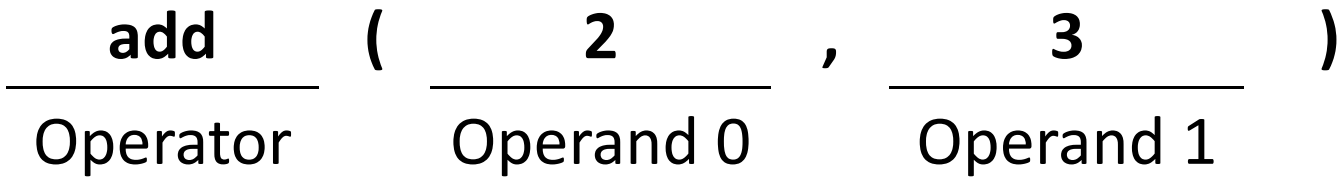
- All expressions can use function call notation

2 + 3	add(2, 3)
sqrt(2401)	sqrt(2401)
abs(-128 + 42 * 3)	abs(add(-128, mul(42, 3)))

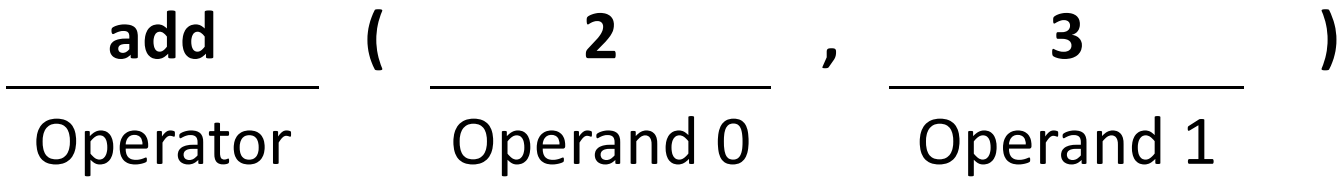
- Infix operator notation is *syntactic sugar* for function calls

- Mathematical operators obey usual precedence rules

Anatomy of a Call Expression

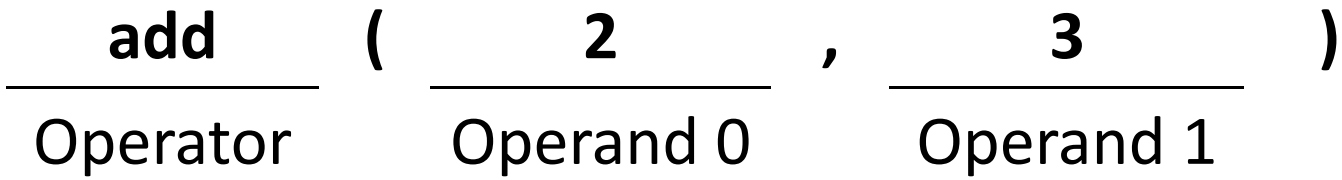


Anatomy of a Call Expression



Operators and operands are expressions, so they evaluate to values

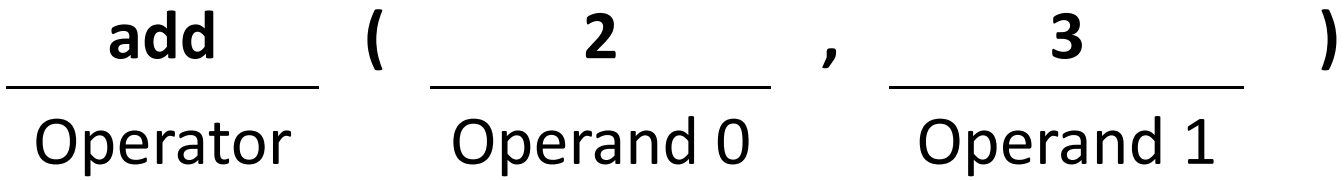
Anatomy of a Call Expression



Operators and operands are expressions, so they evaluate to values

Evaluation procedure for call expressions:

Anatomy of a Call Expression

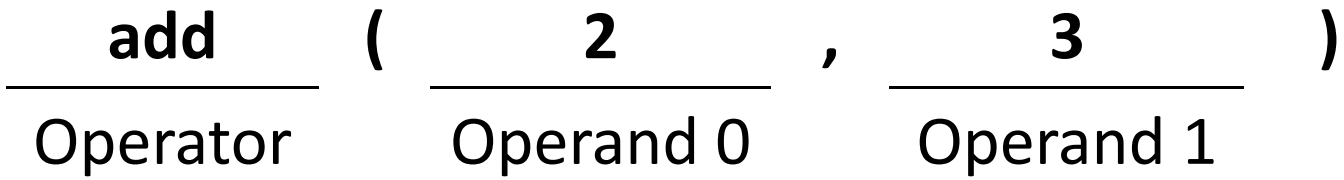


Operators and operands are expressions, so they evaluate to values

Evaluation procedure for call expressions:

1. Evaluate the operator and operand subexpressions in order from left to right.

Anatomy of a Call Expression

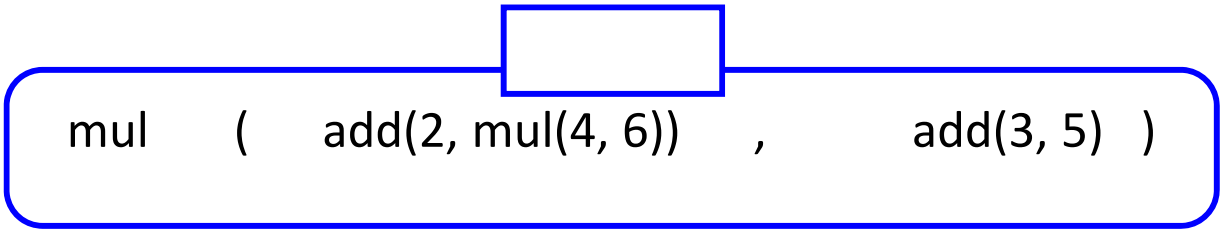


Operators and operands are expressions, so they evaluate to values

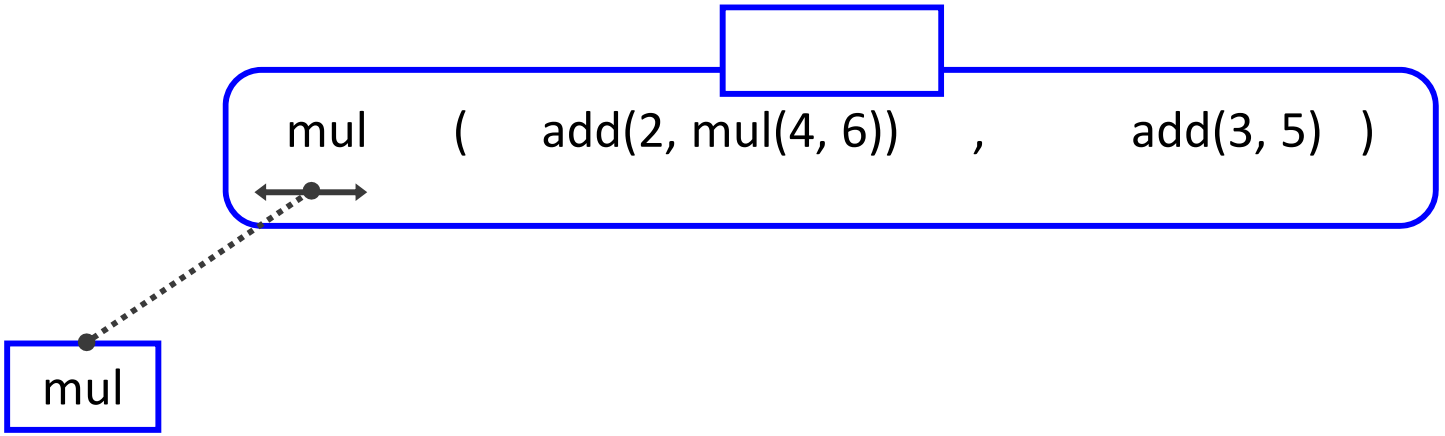
Evaluation procedure for call expressions:

1. Evaluate the operator and operand subexpressions in order from left to right.
2. Apply the function that is the value of the operator subexpression to the arguments that are the values of the operand subexpressions

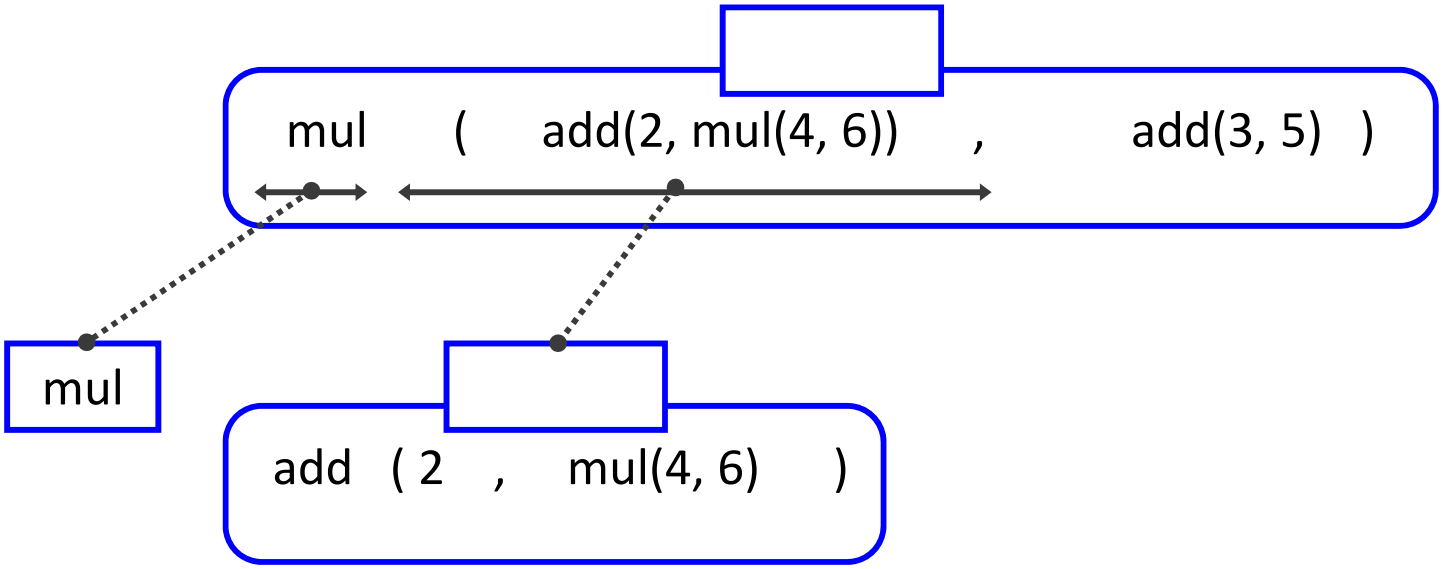
Evaluating Nested Expressions



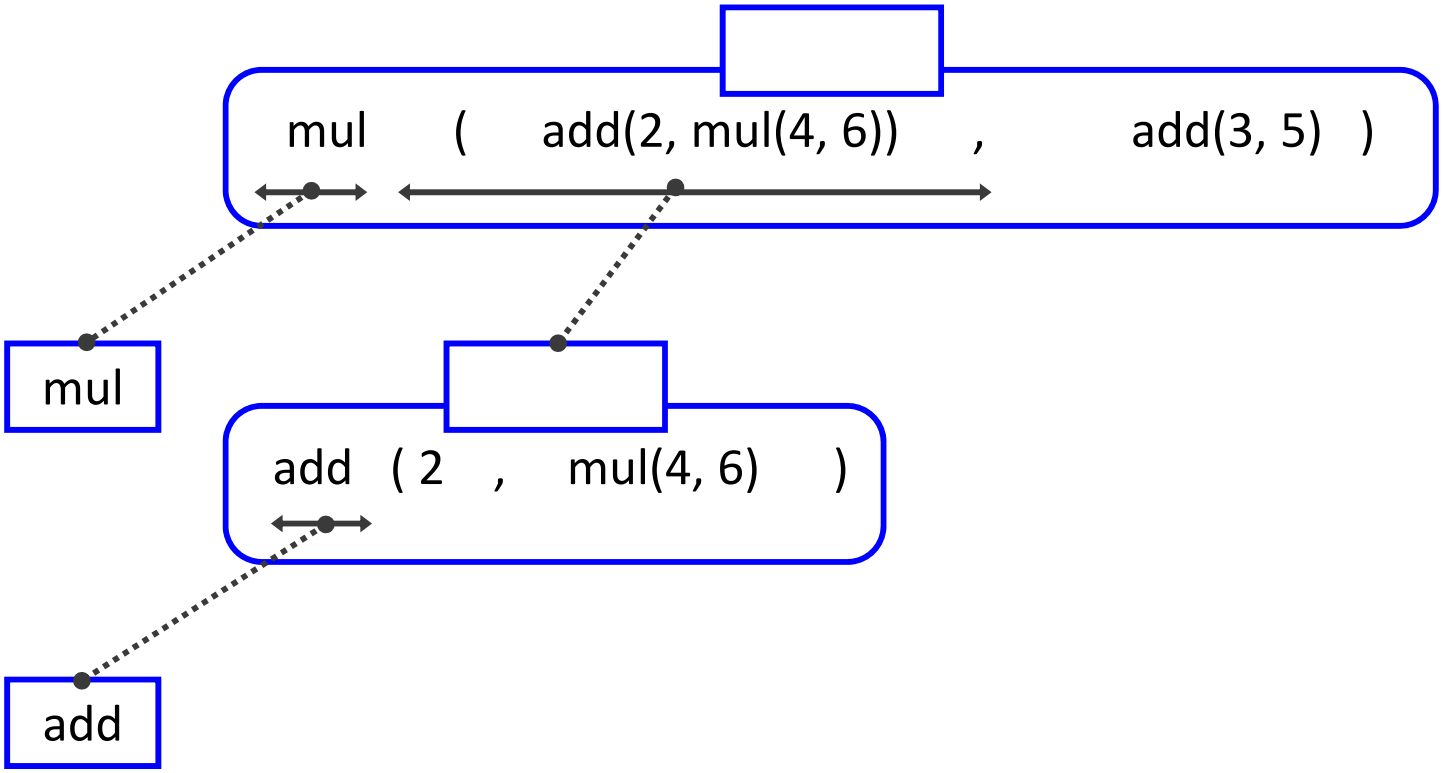
Evaluating Nested Expressions



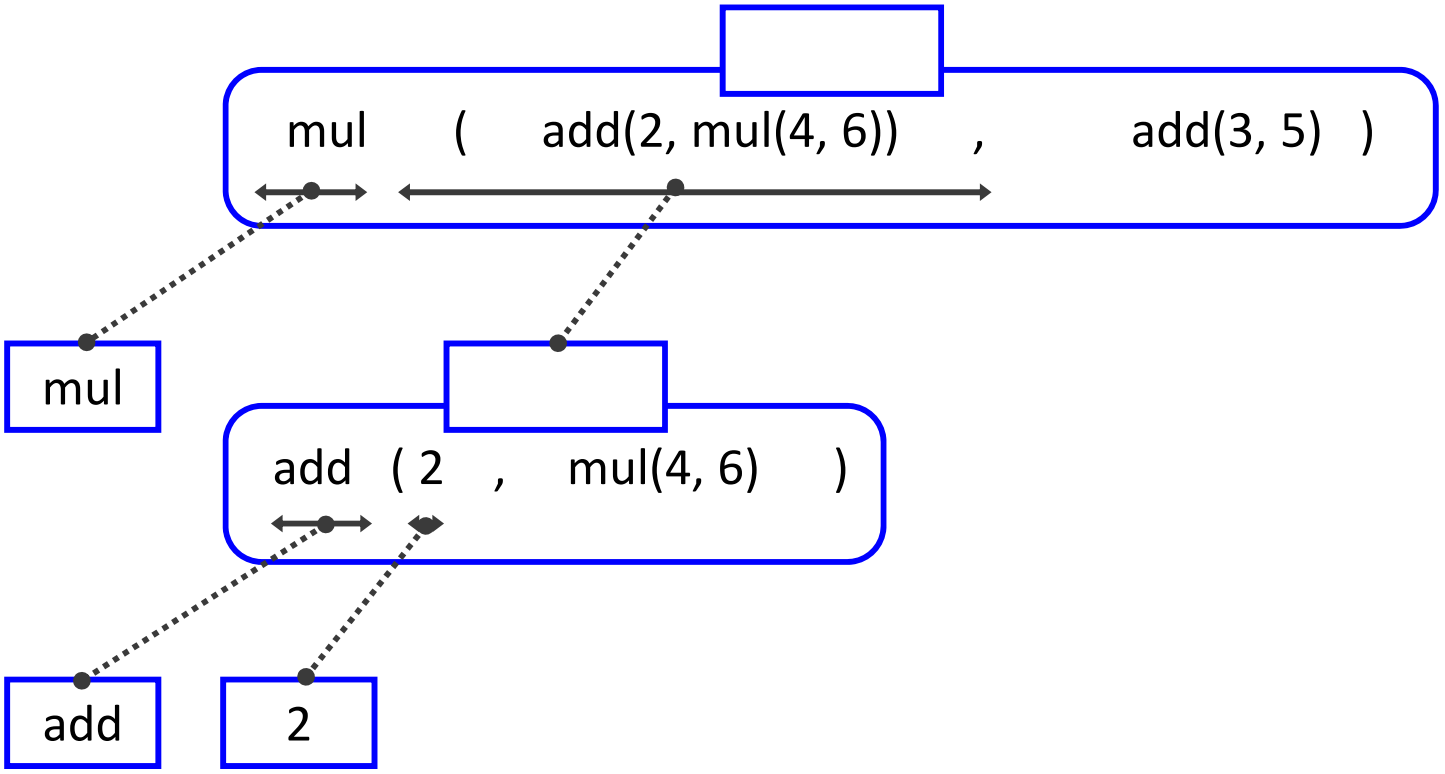
Evaluating Nested Expressions



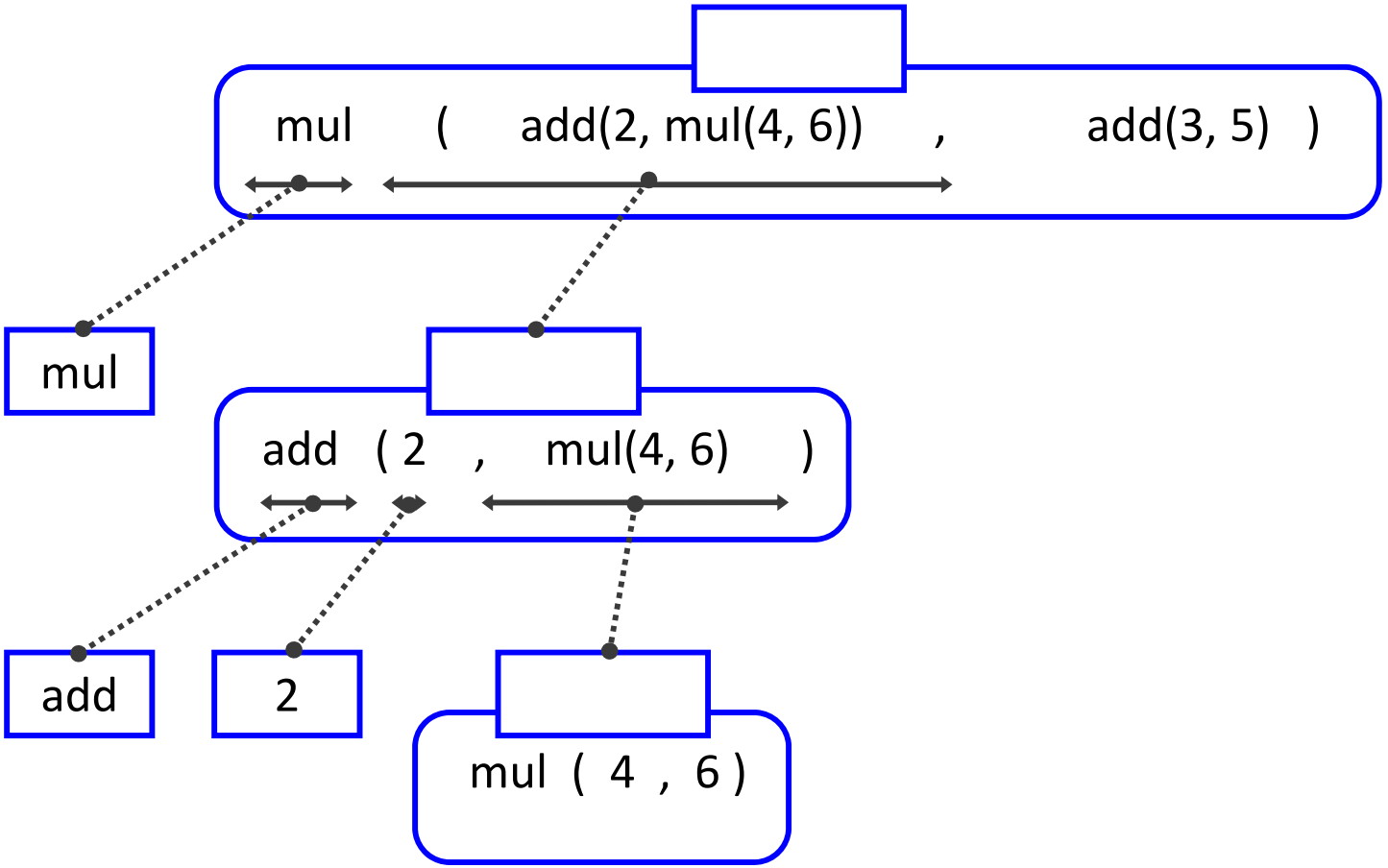
Evaluating Nested Expressions



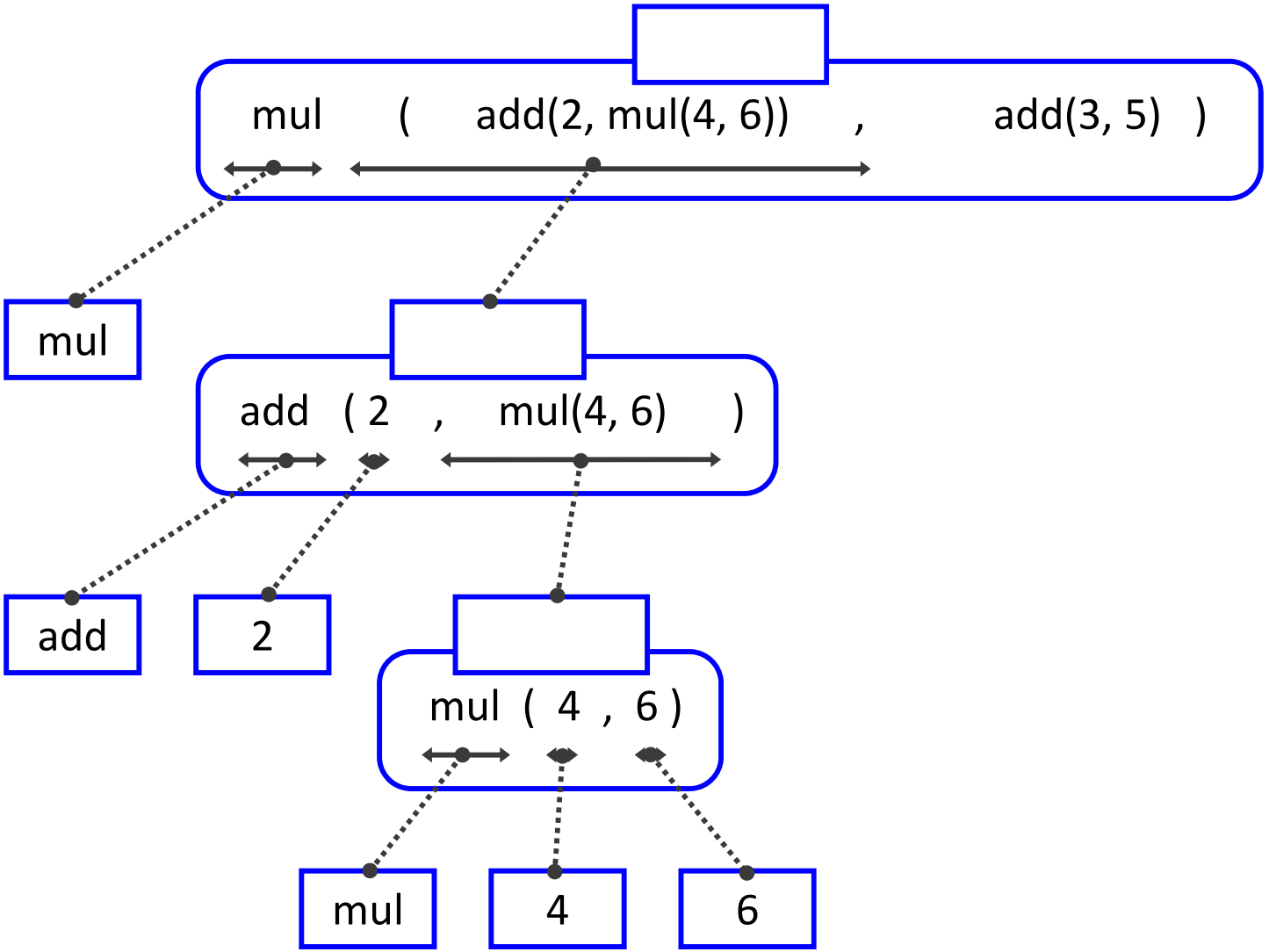
Evaluating Nested Expressions



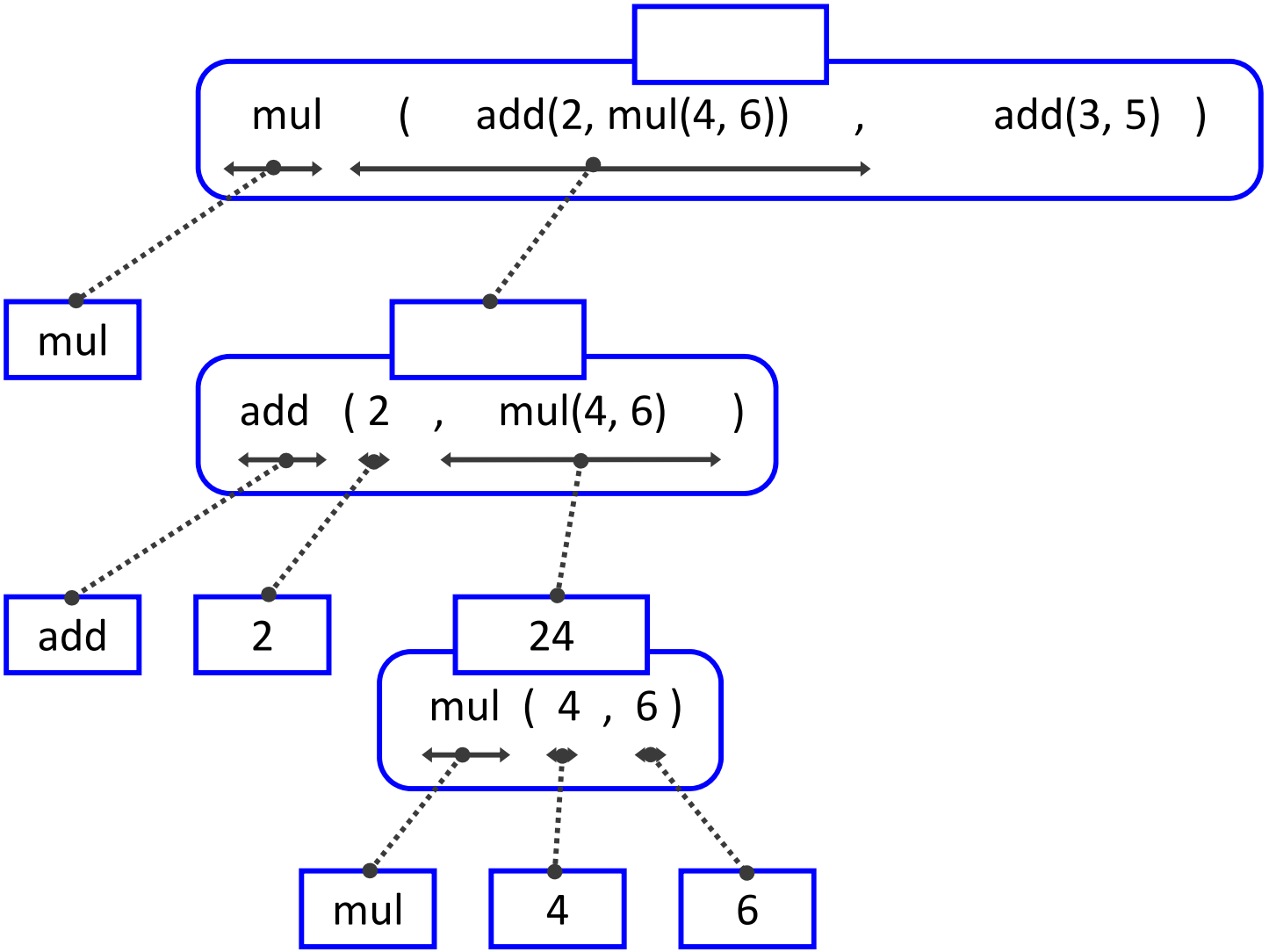
Evaluating Nested Expressions



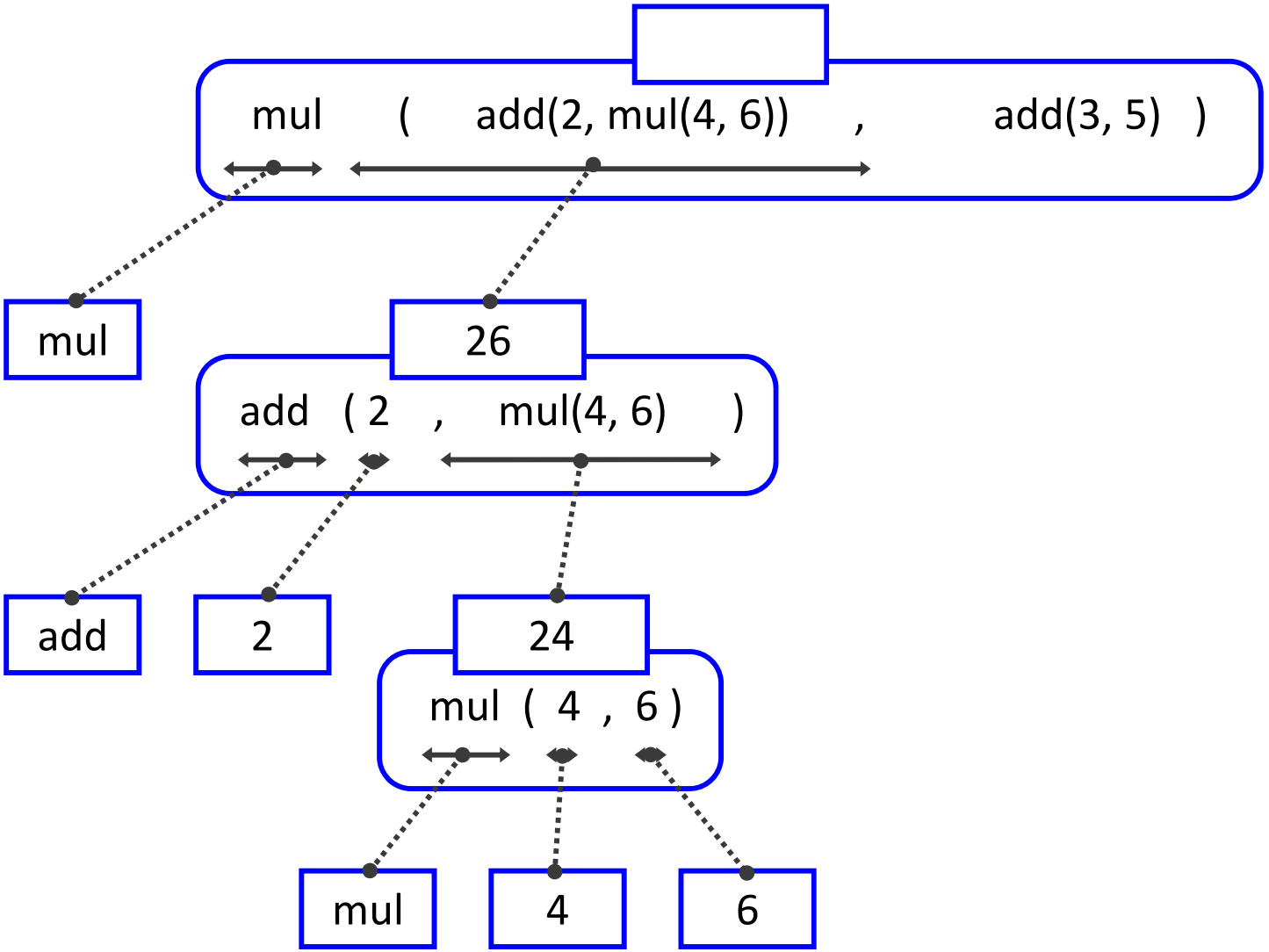
Evaluating Nested Expressions



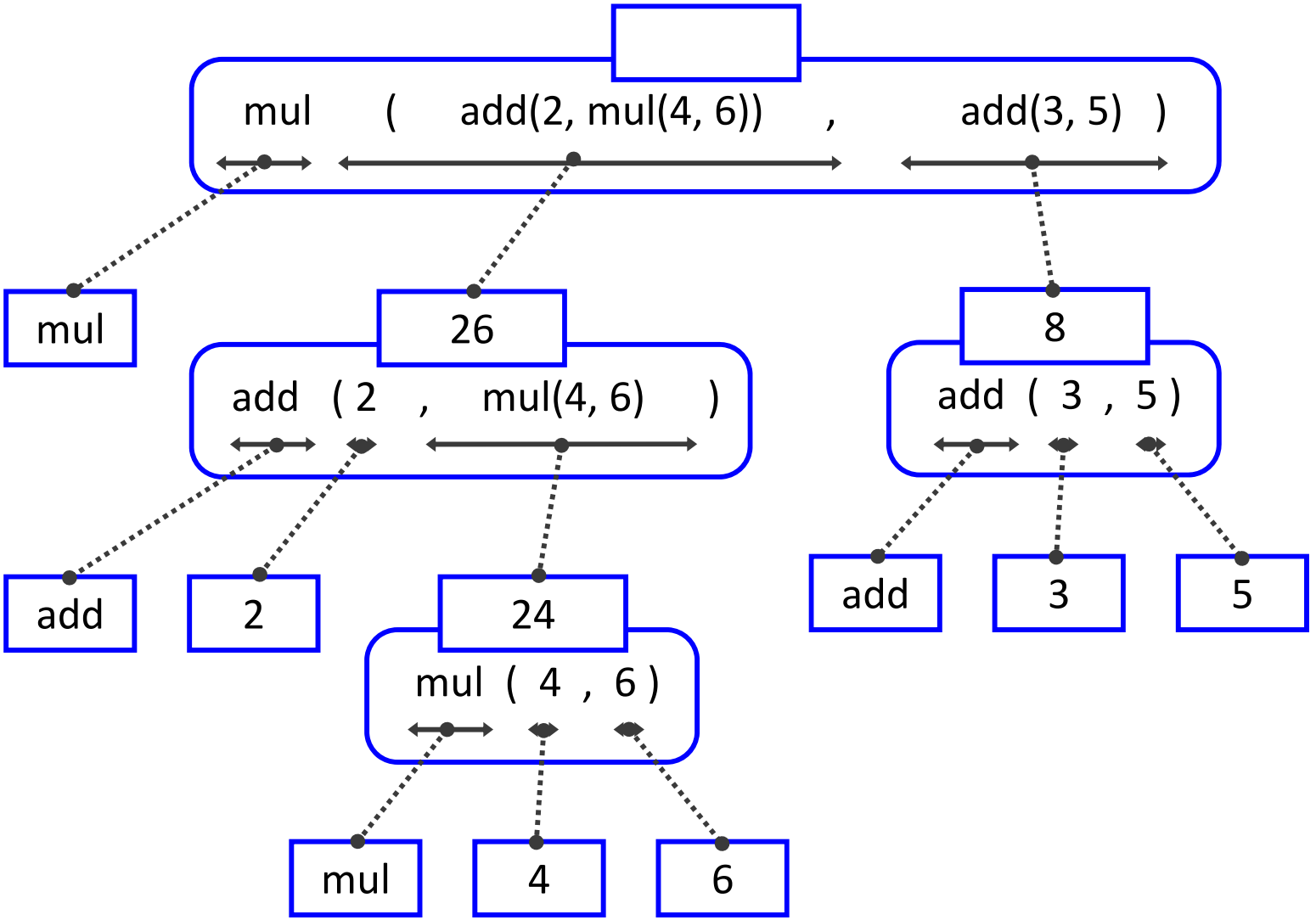
Evaluating Nested Expressions



Evaluating Nested Expressions



Evaluating Nested Expressions



Evaluating Nested Expressions