COMP 410 Lecture 1

Kyle Dewey

About Me

- My research

  - Automated program testing + CS education

  - Programming language design (with JPL and ARCS)

- My dissertation used logic programming extensively

- This is my sixth time teaching this class

About this Class

- See something wrong? Want something improved? Email me about it! (kyle.dewey@csun.edu)

- I generally operate based on feedback

Bad Feedback

- This guy sucks.

- This class is boring.

- This material is useless.

-I can't do anything in response to this

Good Feedback

- This guy sucks, *I can't read his writing.*

- This class is boring, *it's way too slow.*

- This material is useless, *I don't see how it relates to anything in reality.*

- I can't fix anything if I don't know what's wrong

-I can actually do something about this!

What is Logic Programming?

-Major programming paradigm – a way of thinking about problems
-Emphases thinking about exactly _what_ the problem is, as opposed to exactly _how_ to solve it. This is called declarative programming.
-For example: it's generally easier to say what constraints must hold for a valid Sudoku solution, as opposed to directly finding a valid Sudoku solution.
-Somewhat related to functional programming – we generally lack mutable state
-Unlike any other major paradigm, the distinction between inputs and outputs is intentionally blurred. You can take advantage of this.
-Basis in formal logic. It's the only major paradigm where "=" has the same meaning as it does in math.

What is Logic Programming?

- <u>What</u>, not <u>how</u>

-Major programming paradigm – a way of thinking about problems
-Emphases thinking about exactly _what_ the problem is, as opposed to exactly _how_ to solve it. This is called declarative programming.
-For example: it's generally easier to say what constraints must hold for a valid Sudoku solution, as opposed to directly finding a valid Sudoku solution.
-Somewhat related to functional programming – we generally lack mutable state
-Unlike any other major paradigm, the distinction between inputs and outputs is intentionally blurred. You can take advantage of this.
-Basis in formal logic. It's the only major paradigm where "=" has the same meaning as it does in math.

What is Logic Programming?

- <u>What</u>, not <u>how</u>

- No mutable state

-Major programming paradigm – a way of thinking about problems
-Emphases thinking about exactly _what_ the problem is, as opposed to exactly _how_ to solve it. This is called declarative programming.
-For example: it's generally easier to say what constraints must hold for a valid Sudoku solution, as opposed to directly finding a valid Sudoku solution.
-Somewhat related to functional programming – we generally lack mutable state
-Unlike any other major paradigm, the distinction between inputs and outputs is intentionally blurred. You can take advantage of this.
-Basis in formal logic. It's the only major paradigm where "=" has the same meaning as it does in math.

What is Logic Programming?

- <u>What</u>, not <u>how</u>

- No mutable state

- Basis in formal logic

    - = means =

-Major programming paradigm – a way of thinking about problems
-Emphases thinking about exactly _what_ the problem is, as opposed to exactly _how_ to solve it. This is called declarative programming.
-For example: it's generally easier to say what constraints must hold for a valid Sudoku solution, as opposed to directly finding a valid Sudoku solution.
-Somewhat related to functional programming – we generally lack mutable state
-Unlike any other major paradigm, the distinction between inputs and outputs is intentionally blurred. You can take advantage of this.
-Basis in formal logic. It's the only major paradigm where "=" has the same meaning as it does in math.

What is Logic Programming?

- <u>What</u>, not <u>how</u>

- No mutable state

- Basis in formal logic

    - = means =

- Line between input/output is blurry

-Major programming paradigm – a way of thinking about problems
-Emphases thinking about exactly _what_ the problem is, as opposed to exactly _how_ to solve it. This is called declarative programming.
-For example: it's generally easier to say what constraints must hold for a valid Sudoku solution, as opposed to directly finding a valid Sudoku solution.
-Somewhat related to functional programming – we generally lack mutable state
-Unlike any other major paradigm, the distinction between inputs and outputs is intentionally blurred. You can take advantage of this.
-Basis in formal logic. It's the only major paradigm where "=" has the same meaning as it does in math.

What is this Course?

-Strong emphasis on programming and using logic programming languages
-I want you to think in this paradigm, not merely force Java into it
-The ideas can be applied in non-logical languages, and your first assignment will force you to write in a logical way outside of a logic programming language (though you won't realize that's what you're doing yet)
-Little bit of theory

What is this Course?

- Programming, programming, programming

-Strong emphasis on programming and using logic programming languages
-I want you to think in this paradigm, not merely force Java into it
-The ideas can be applied in non-logical languages, and your first assignment will force you to write in a logical way outside of a logic programming language (though you won't realize that's what you're doing yet)
-Little bit of theory

What is this Course?

- Programming, programming, programming

- Thinking in a logic programming way

-Strong emphasis on programming and using logic programming languages
-I want you to think in this paradigm, not merely force Java into it
-The ideas can be applied in non-logical languages, and your first assignment will force you to write in a logical way outside of a logic programming language (though you won't realize that's what you're doing yet)
-Little bit of theory

What is this Course?

- Programming, programming, programming

- Thinking in a logic programming way

- Applying logic programming without a logic programming language

-Strong emphasis on programming and using logic programming languages
-I want you to think in this paradigm, not merely force Java into it
-The ideas can be applied in non-logical languages, and your first assignment will force you to write in a logical way outside of a logic programming language (though you won't realize that's what you're doing yet)
-Little bit of theory

What this course **isn't**

What this course **isn't**

- Artificial intelligence

-"Artificial intelligence" used to refer to search techniques, which is relevant to logic programming. Now the term largely refers to machine learning. What it means is a moving target.
-Machine learning (we won't do any sort of statistics)
-You can spend a career on the theory behind this stuff. I know some, but it's not my speciality.

What this course **isn't**

- Artificial intelligence
- Machine learning

-"Artificial intelligence" used to refer to search techniques, which is relevant to logic programming. Now the term largely refers to machine learning. What it means is a moving target.
-Machine learning (we won't do any sort of statistics)
-You can spend a career on the theory behind this stuff. I know some, but it's not my speciality.

What this course **isn't**

- Artificial intelligence

- Machine learning

- Theoretical

–"Artificial intelligence" used to refer to search techniques, which is relevant to logic programming. Now the term largely refers to machine learning. What it means is a moving target.
–Machine learning (we won't do any sort of statistics)
–You can spend a career on the theory behind this stuff. I know some, but it's not my speciality.

Syllabus

Outline

- Abstract Syntax Trees and evaluation

- SAT and Semantic Tableau

Abstract Syntax Trees
and
Evaluation

Abstract Syntax Tree

- Abbreviation: AST

- Unambiguous tree-based representation of a sentence in a language

- Very commonly used in compilers, interpreters, and related software

–Generally we work with ASTs instead of Strings or any other code representation

```
(1 + 2) - 3 * 4
```

-Key parts: we need parentheses to direct that 1 + 2 happens first. We know that the 3 * 4 should happen after the part in parentheses from PEMDAS rules
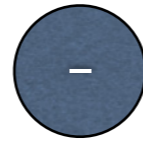
–Lowest priority thing ends up in the top of the tree
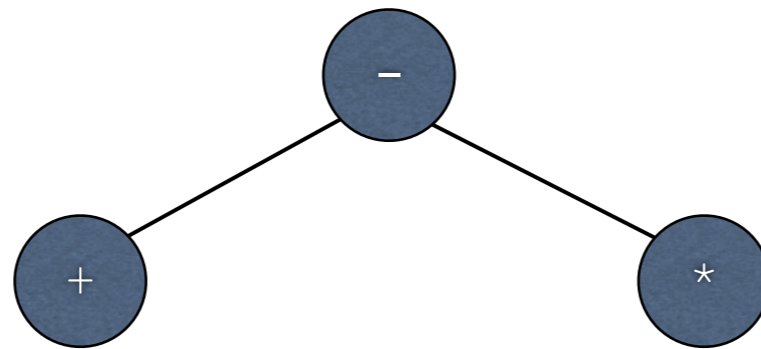
−Next level of priority

$(1 + 2) - 3 * 4$
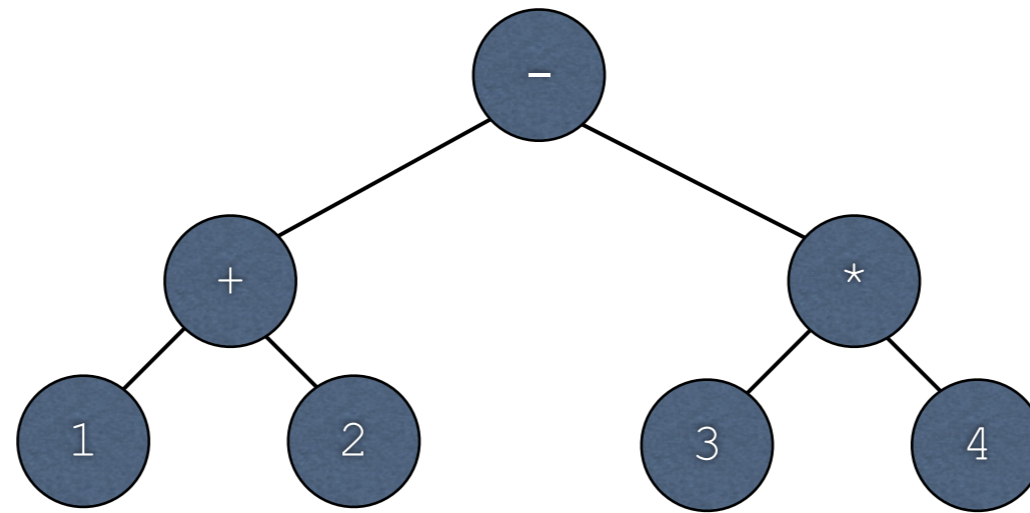
–Next level of priority

Exercise: First Side of AST/Evaluation Sheet

Evaluation
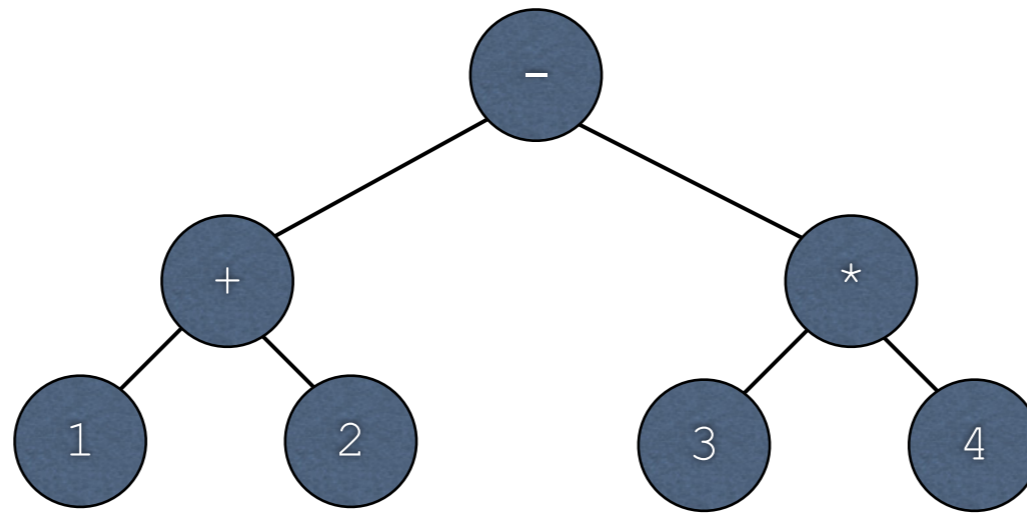
-Key point: bubble-up values from the leaves
-This can be implemented in code via a recursive function starting from the root (code in a bit later)

Evaluation
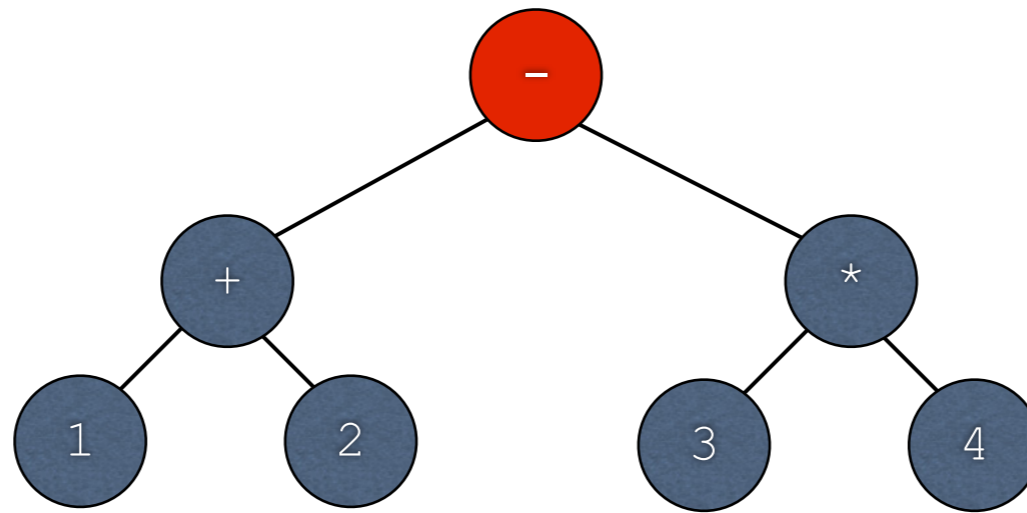
–We start evaluation from the root...

Evaluation
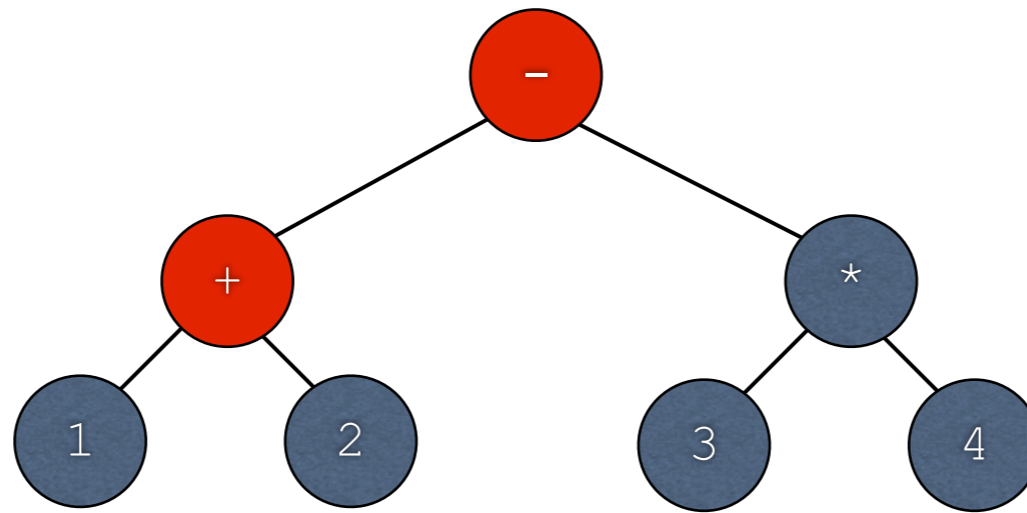
-In order to evaluate the root, we need to evaluate the left subtree of the root (+)

Evaluation
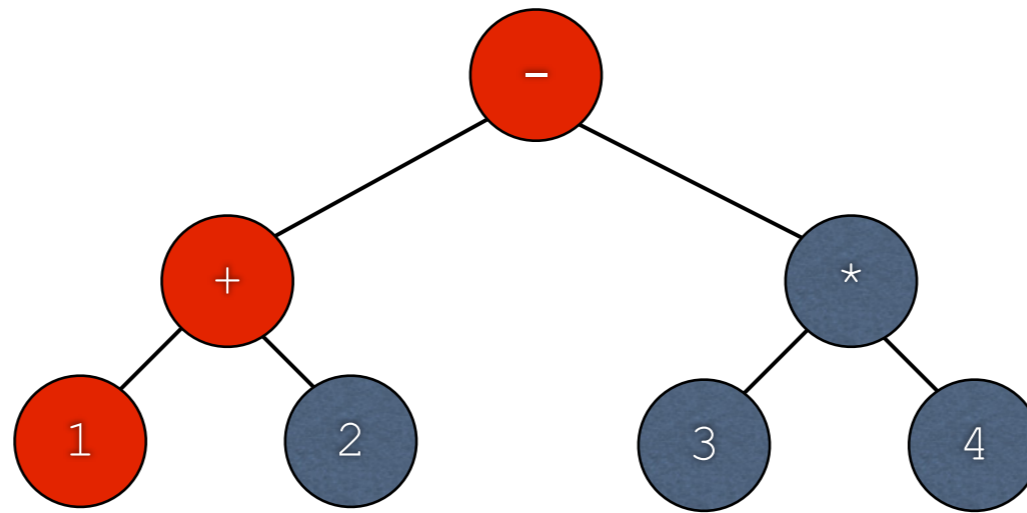
-In order to evaluate +, we need to evaluate the left subtree (as with the root)

Evaluation
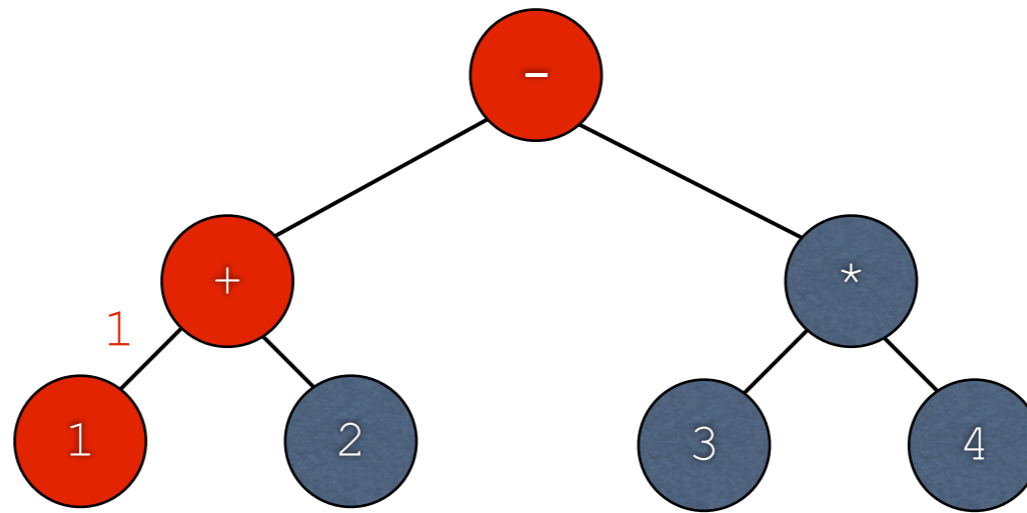
-For arithmetic, leaves are simply numbers
-Evaluating a leaf returns the number held within

Evaluation
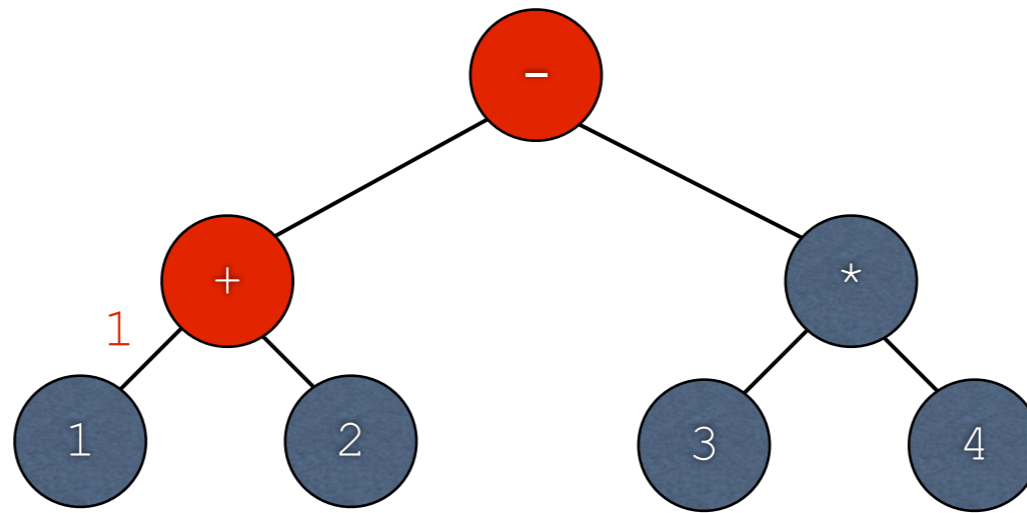
-The left subtree of + has now been evaluated
-Now + needs the value of the right subtree
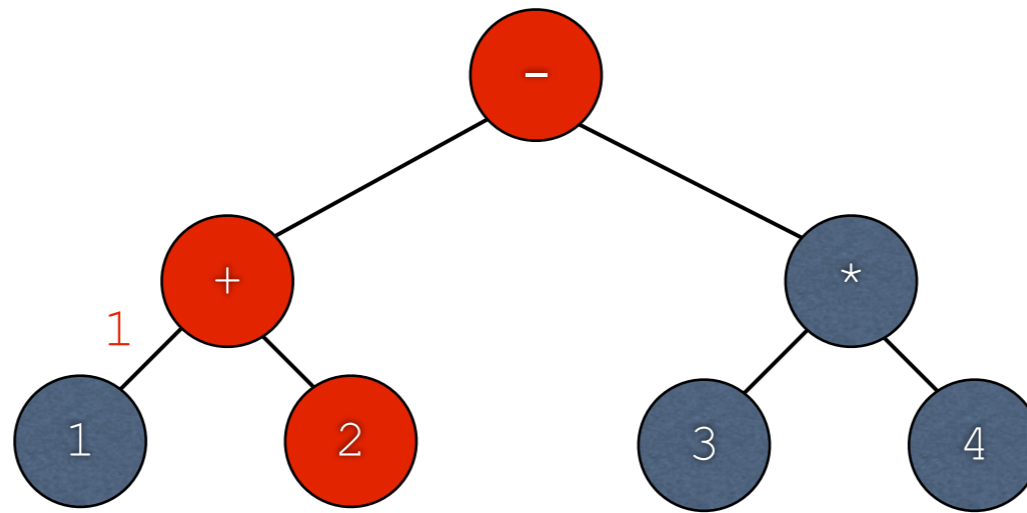
Evaluation

–The left subtree of + has now been evaluated
–Now + needs the value of the right subtree
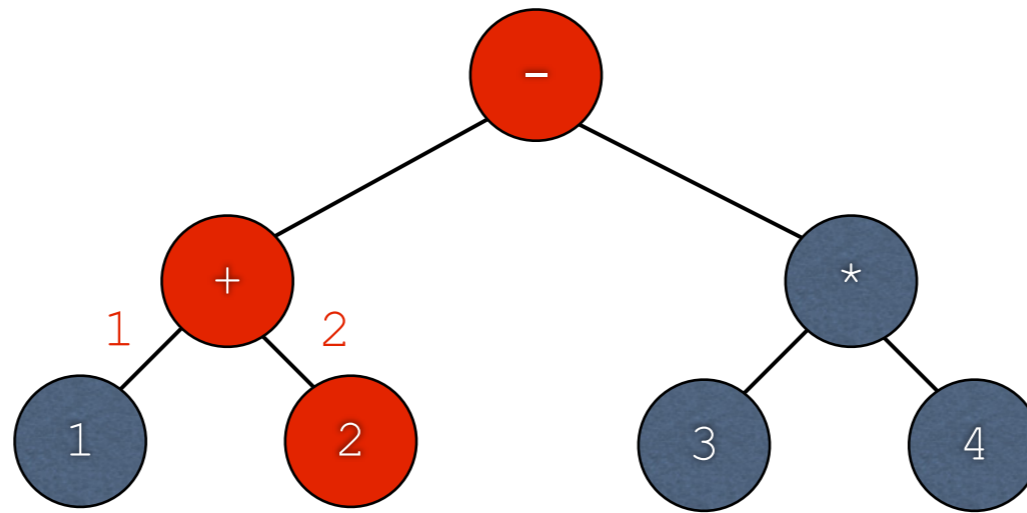
Evaluation

–As before, leaves just return the value held within
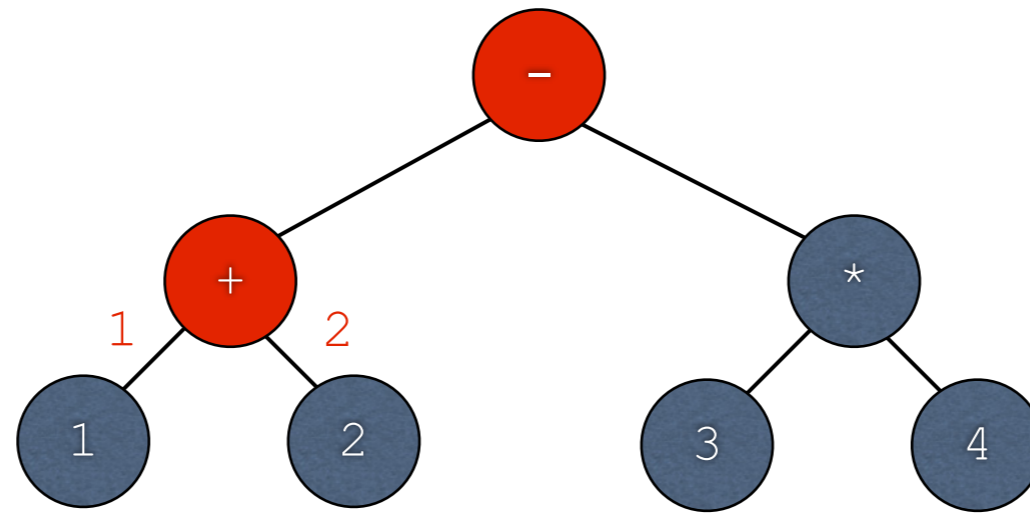
Evaluation

-Subtrees of + are now taken care of
-Now + has two values that it needs to work with...

Evaluation
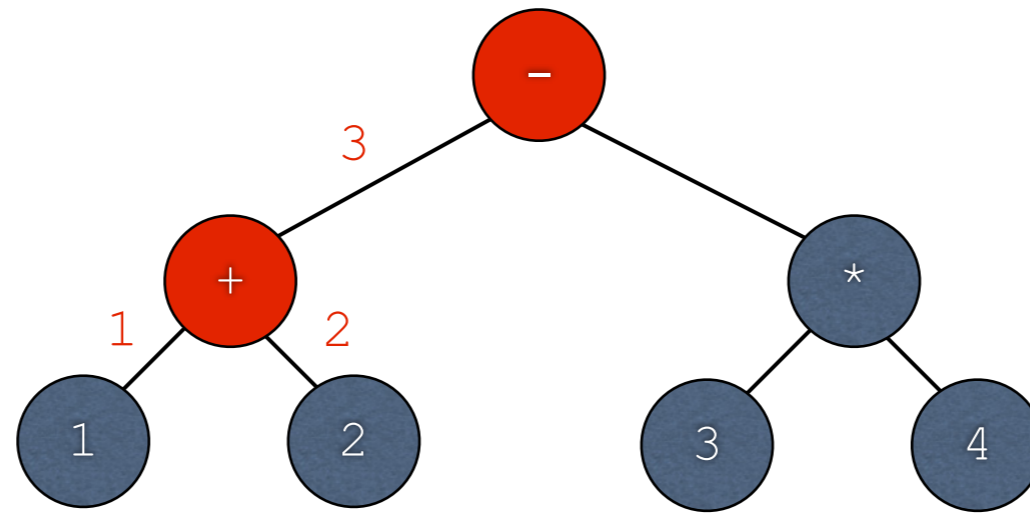
-+ performs the actual addition
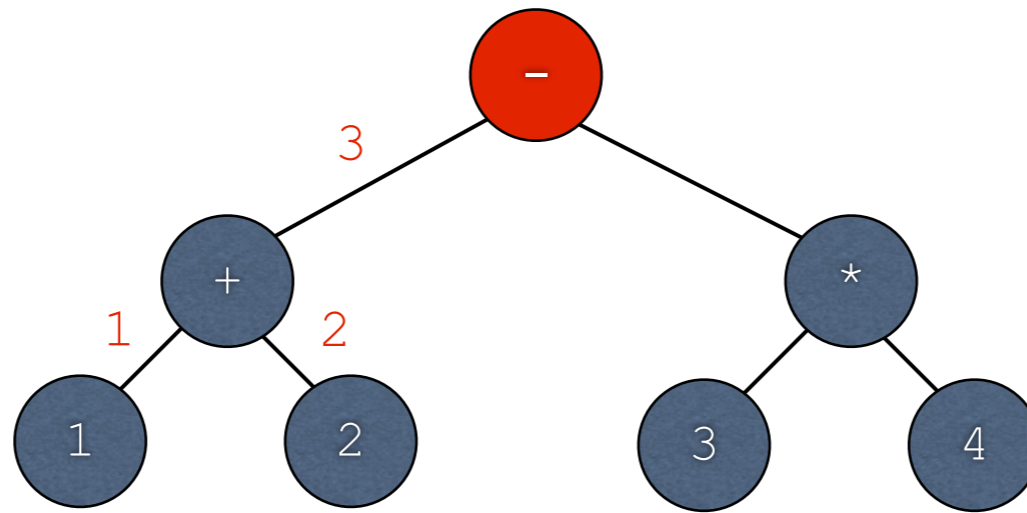
–Now + is taken care of
–Going back to –, – now has the value of the left subtree, and it needs the value of the right subtree
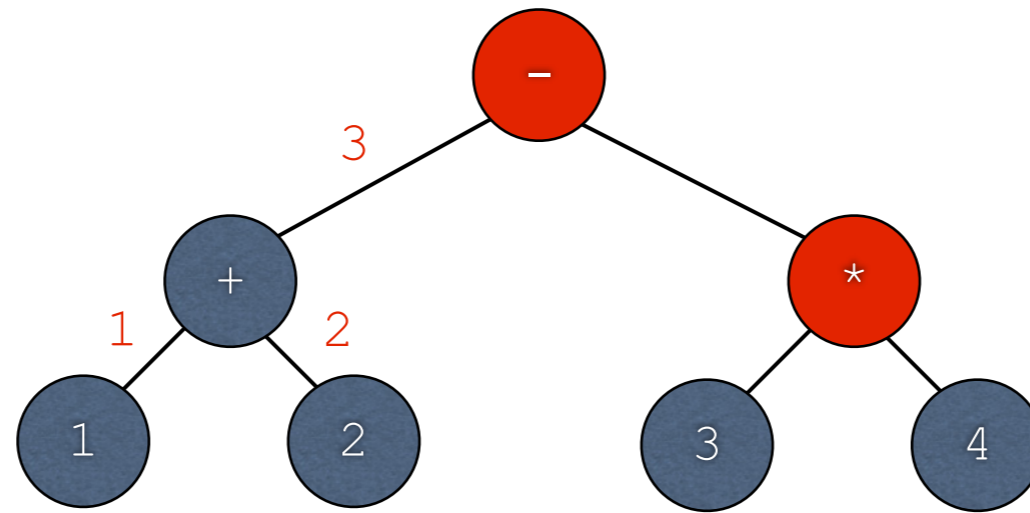
Evaluation

–Now we're on *, which needs the value of the left subtree…
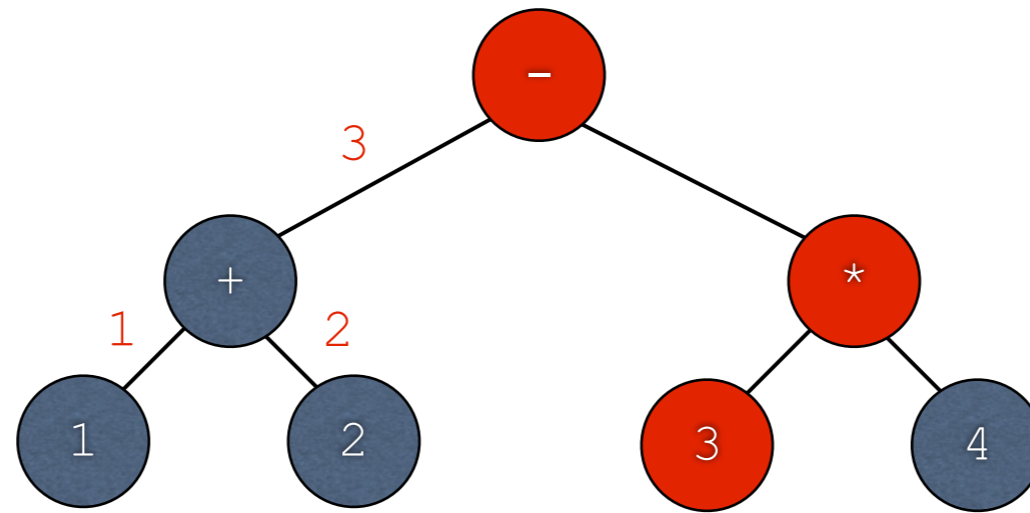
Evaluation

–Now we're on *, which needs the value of the left subtree...
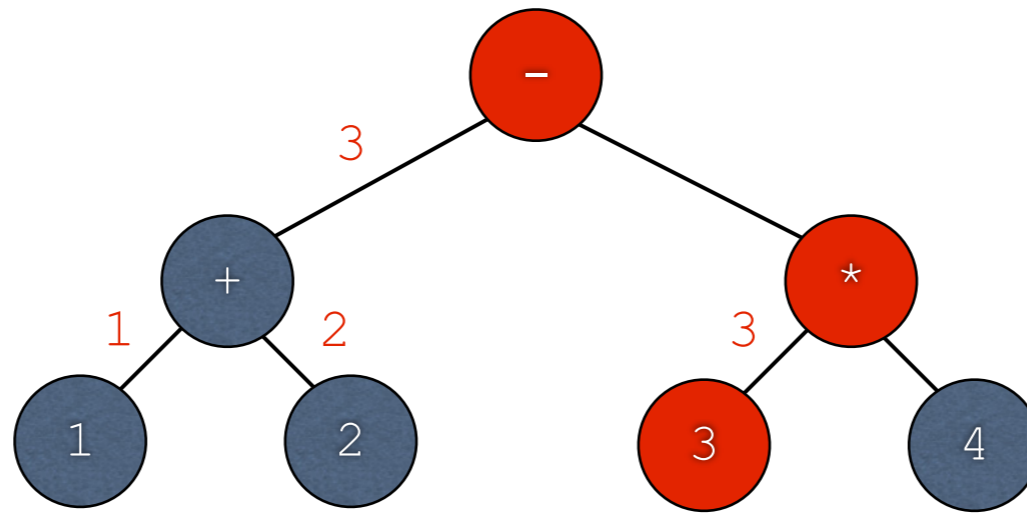
Evaluation

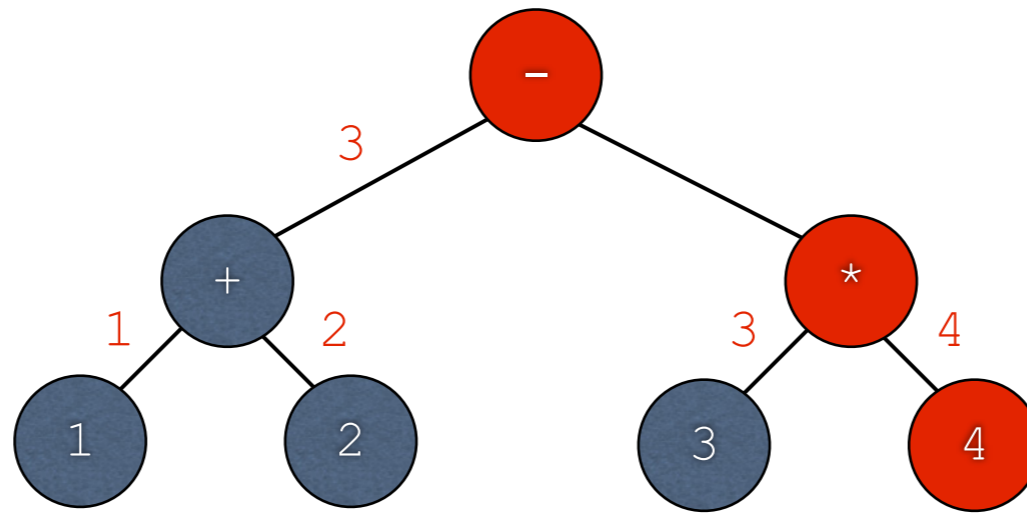-Leaves again return the values held within...

Evaluation
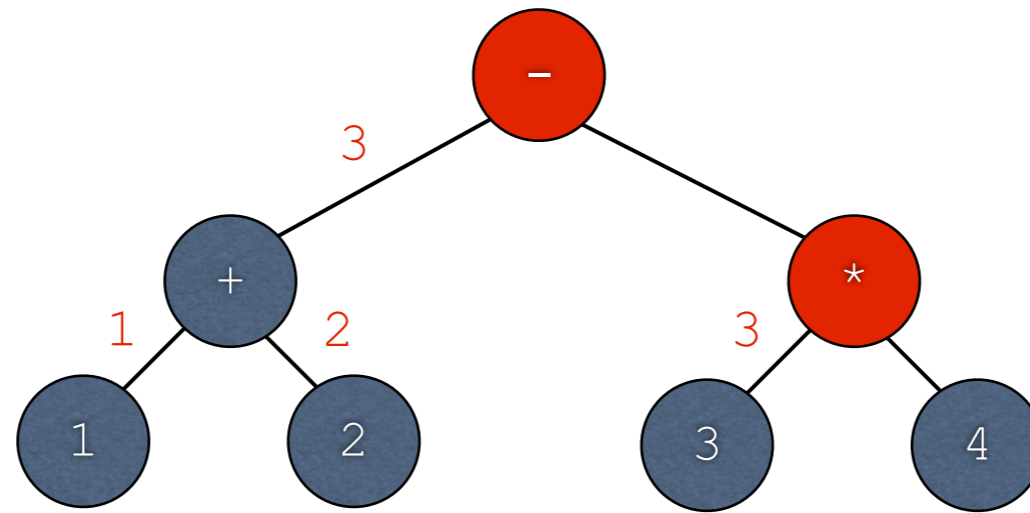
–Left subtree done; * now needs the value of the right subtree...
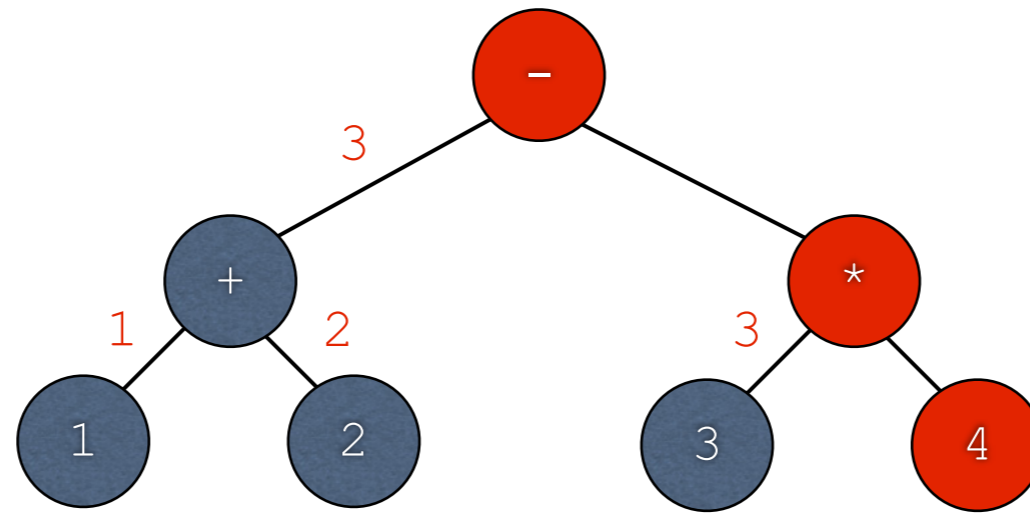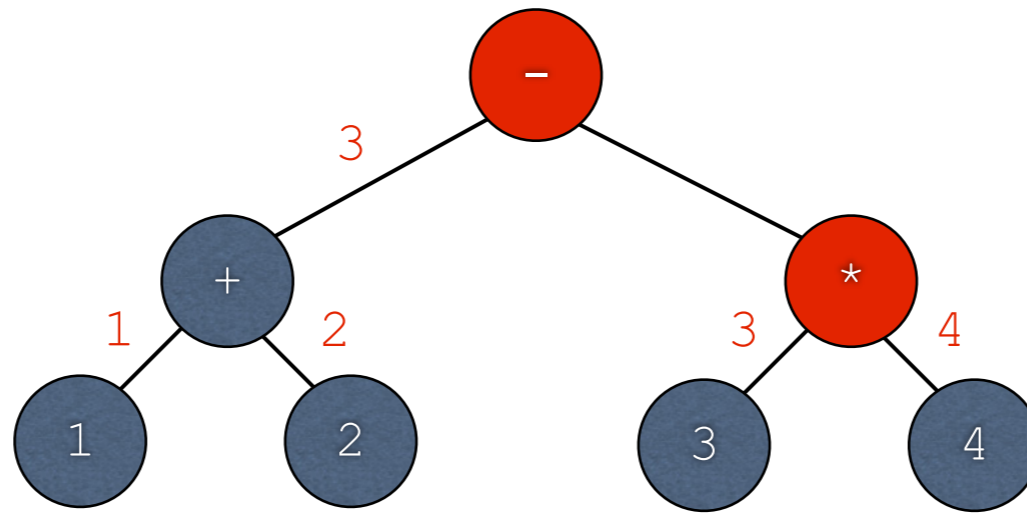
Evaluation

–Left subtree done; * now needs the value of the right subtree…
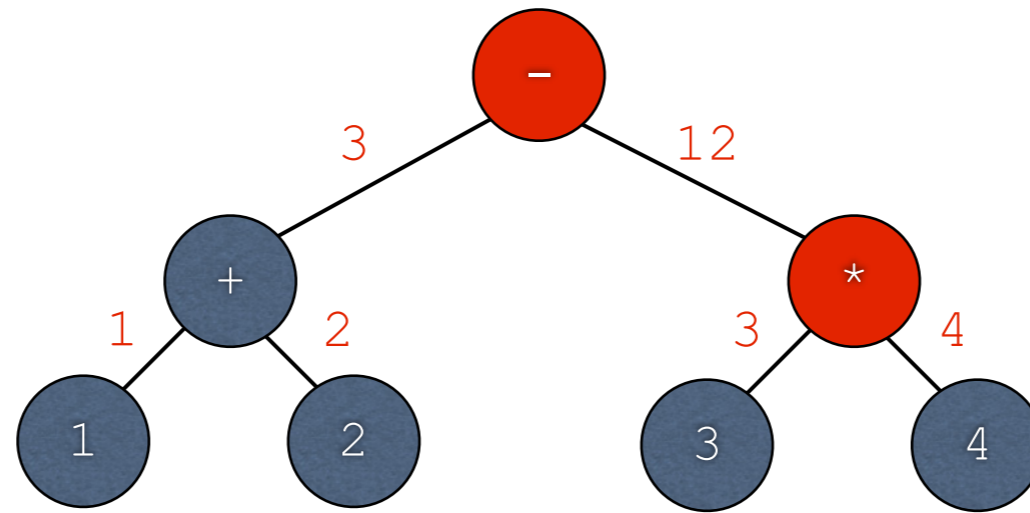
Evaluation

–Leaf returns value held within
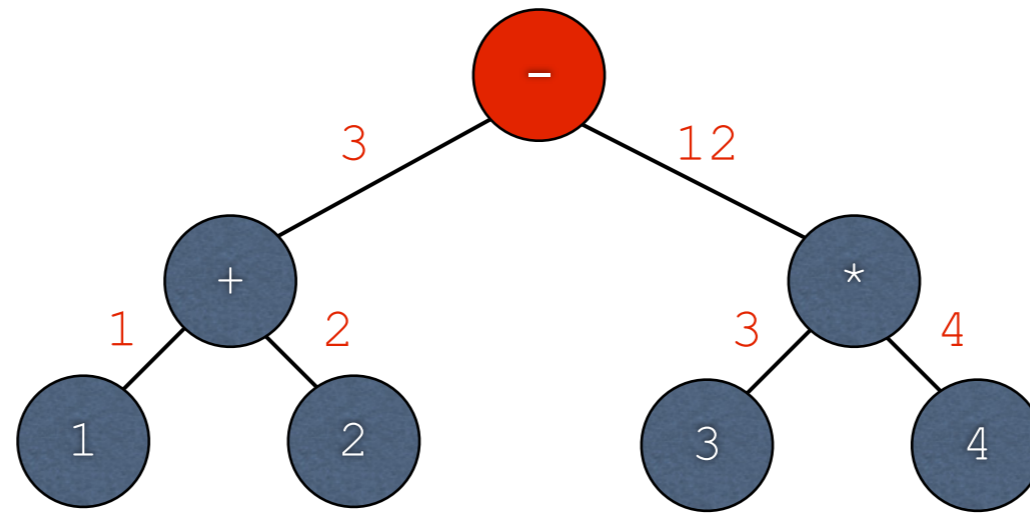
Evaluation

–Leaf is done. * now has both operands it needs...
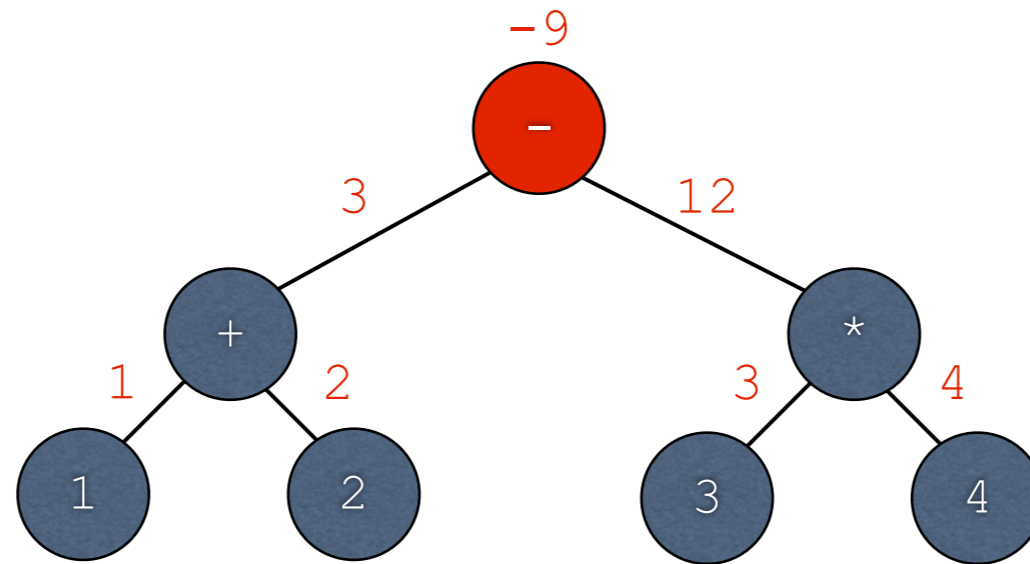
Evaluation

-* performs the multiplication and returns the value

Evaluation

–The root – node now has both operands...

Evaluation

−...and it returns the result of the subtraction

Exercise: Second Side of AST/Evaluation Sheet

Evaluator Example:
`arithmetic_evaluator.py`

-Complete example online; we'll live-code this in class