COMP 410 Lecture 1

Kyle Dewey

About Me

- I research automated testing techniques and their intersection with CS education
- My dissertation used logic programming extensively
- This is my third 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.

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

What is Logic Programming?

What is Logic Programming?

- What, not how

What is Logic Programming?

- What, not how
- No mutable state

What is Logic Programming?

- What, not how
- No mutable state
- Basis in formal logic
  - = means =

What is Logic Programming?

- What, not how
- No mutable state
- Basis in formal logic
  - = means =
- Line between input/output is blurry

What is this Course?

What is this Course?

- Programming, programming, programming

What is this Course?

- Programming, programming, programming
- Thinking in a logic programming way

What is this Course?

- Programming, programming, programming
- Thinking in a logic programming way
- Applying logic programming without a logic programming language

What this course **isn't**

What this course **isn't**

- Artificial intelligence

What this course **isn't**

- Artificial intelligence
- Machine learning

What this course **isn't**

- Artificial intelligence
- Machine learning
- Theoretical

Syllabus

Outline

- Abstract Syntax Trees and evaluation
- SAT and Semantic Tableau

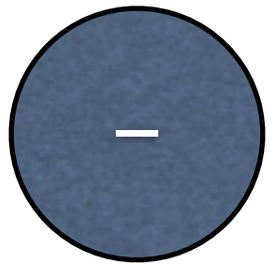
Abstract Syntax Trees and Evaluation

Abstract Syntax Tree

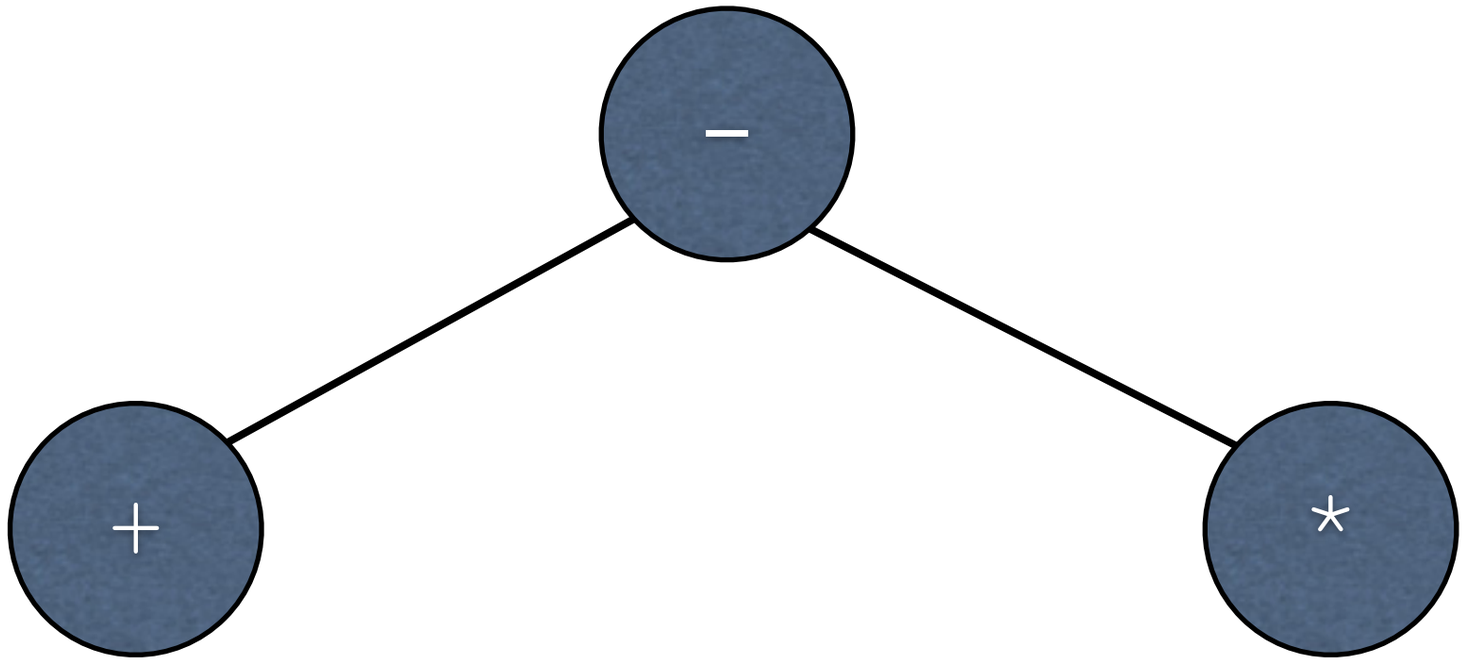
- Abbreviation: AST
- Unambiguous tree-based representation of a sentence in a language
- Very commonly used in compilers, interpreters, and related software

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

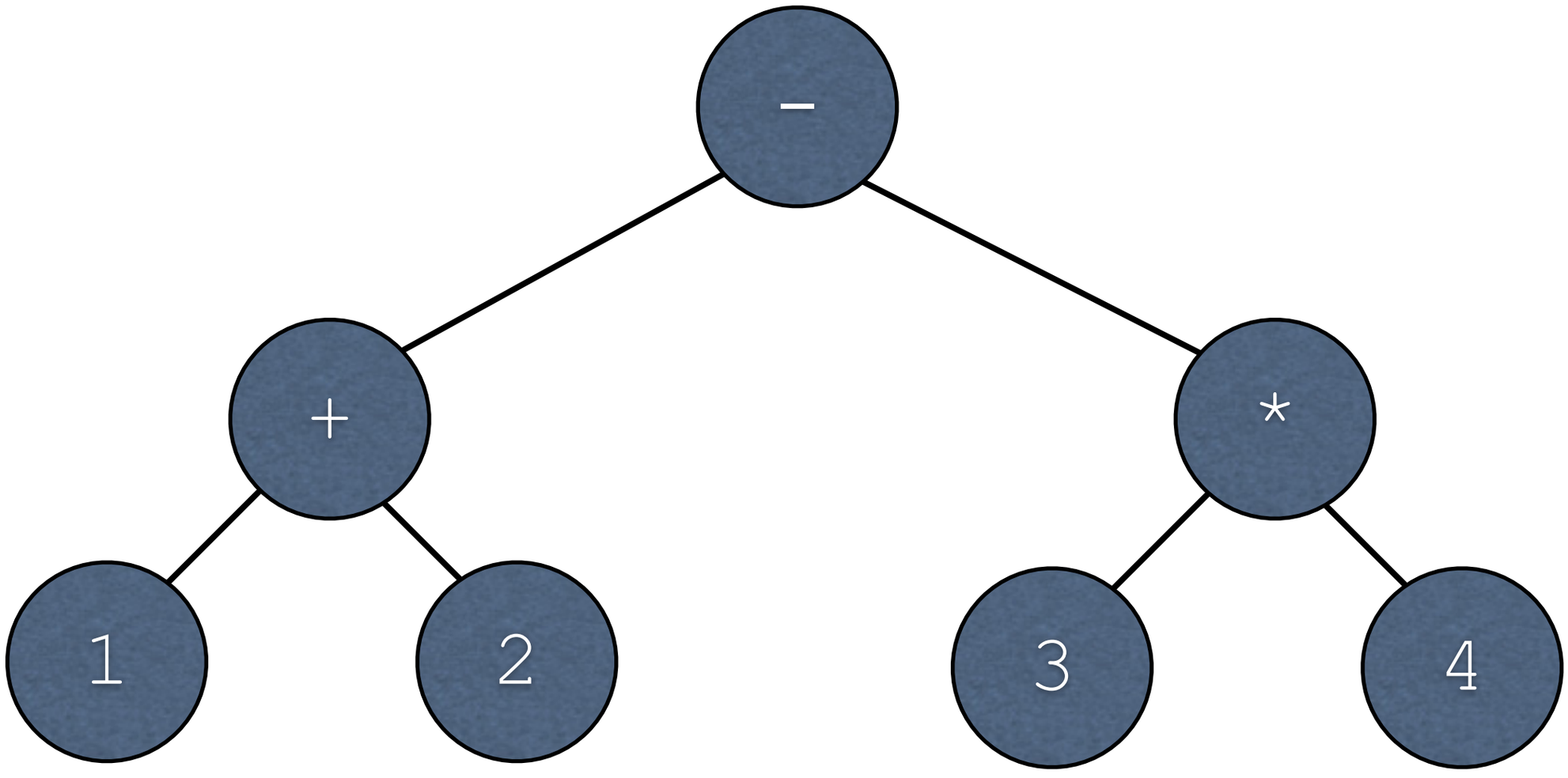
(1 + 2) − 3 * 4



(1 + 2) - 3 * 4

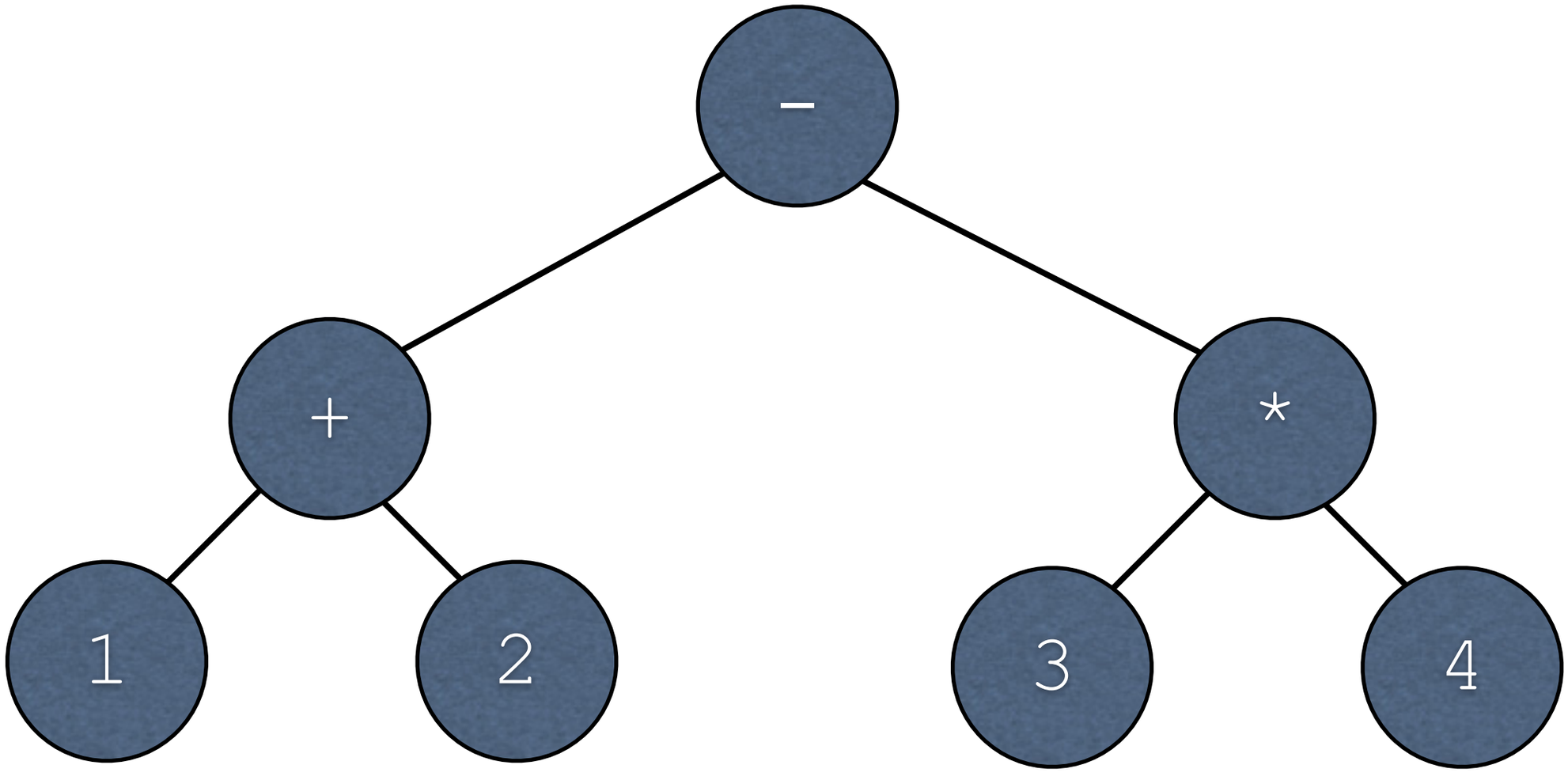


(1 + 2) - 3 * 4

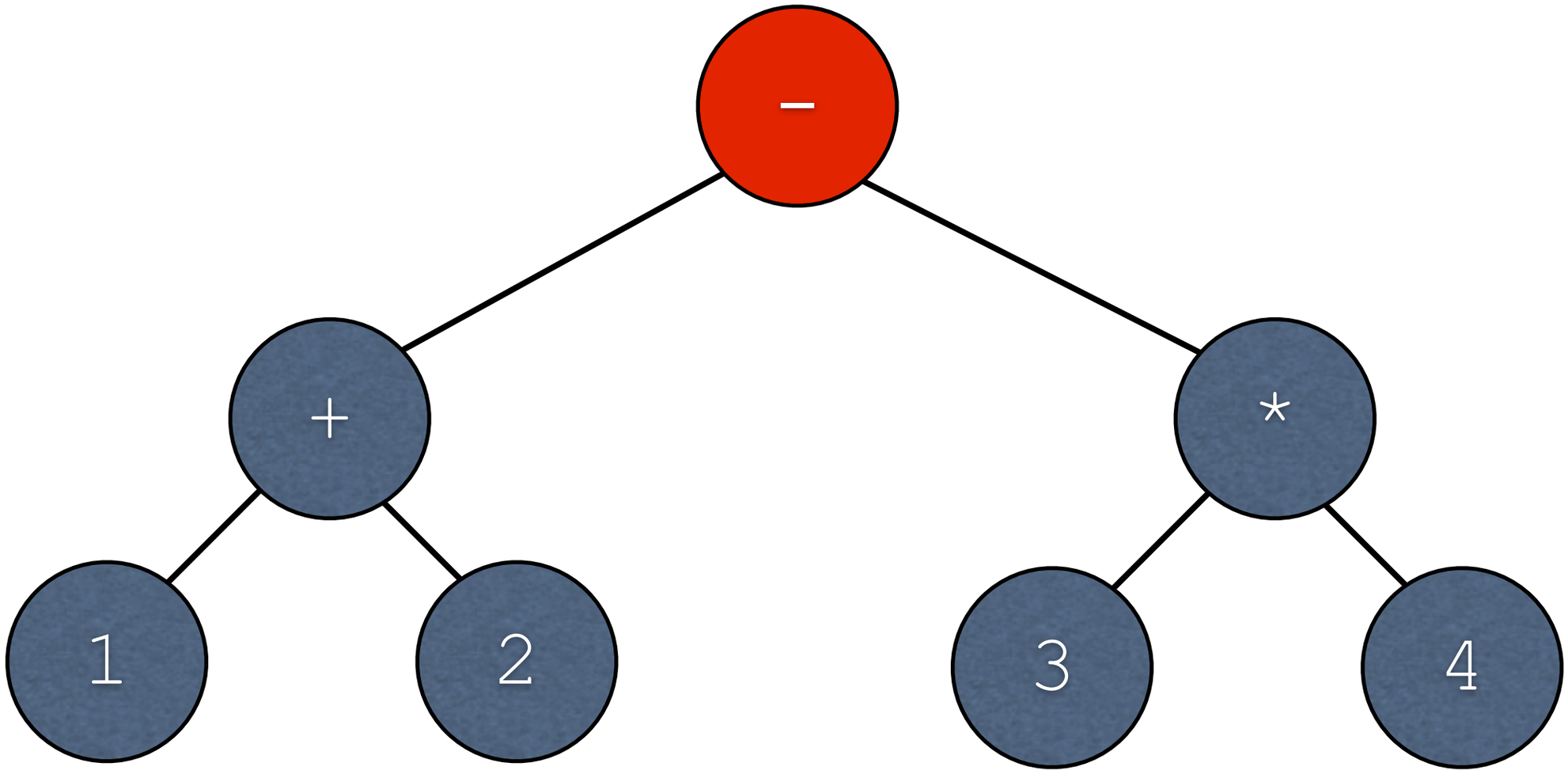


Exercise: First Side of AST/Evaluation Sheet

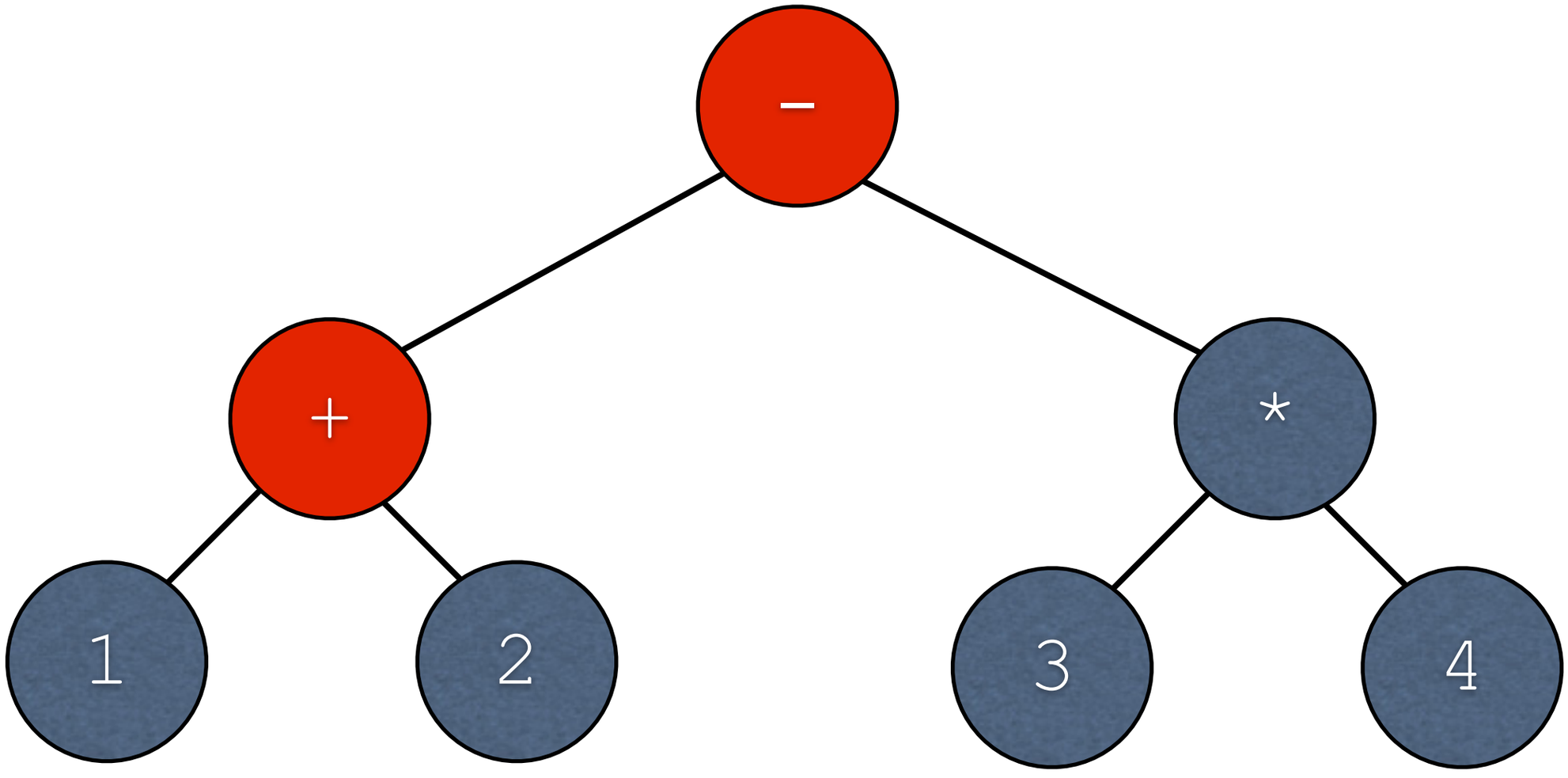
Evaluation



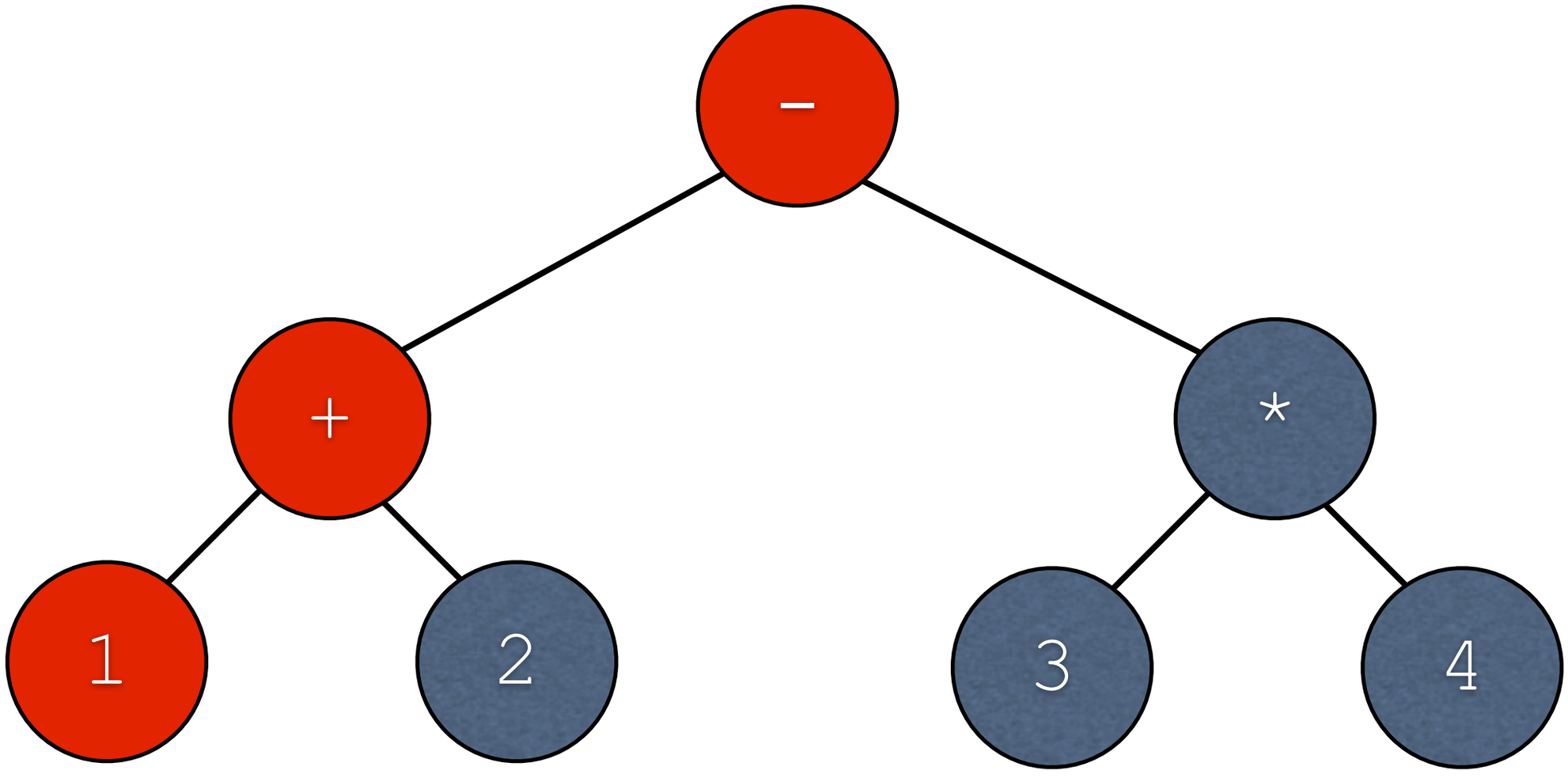
Evaluation



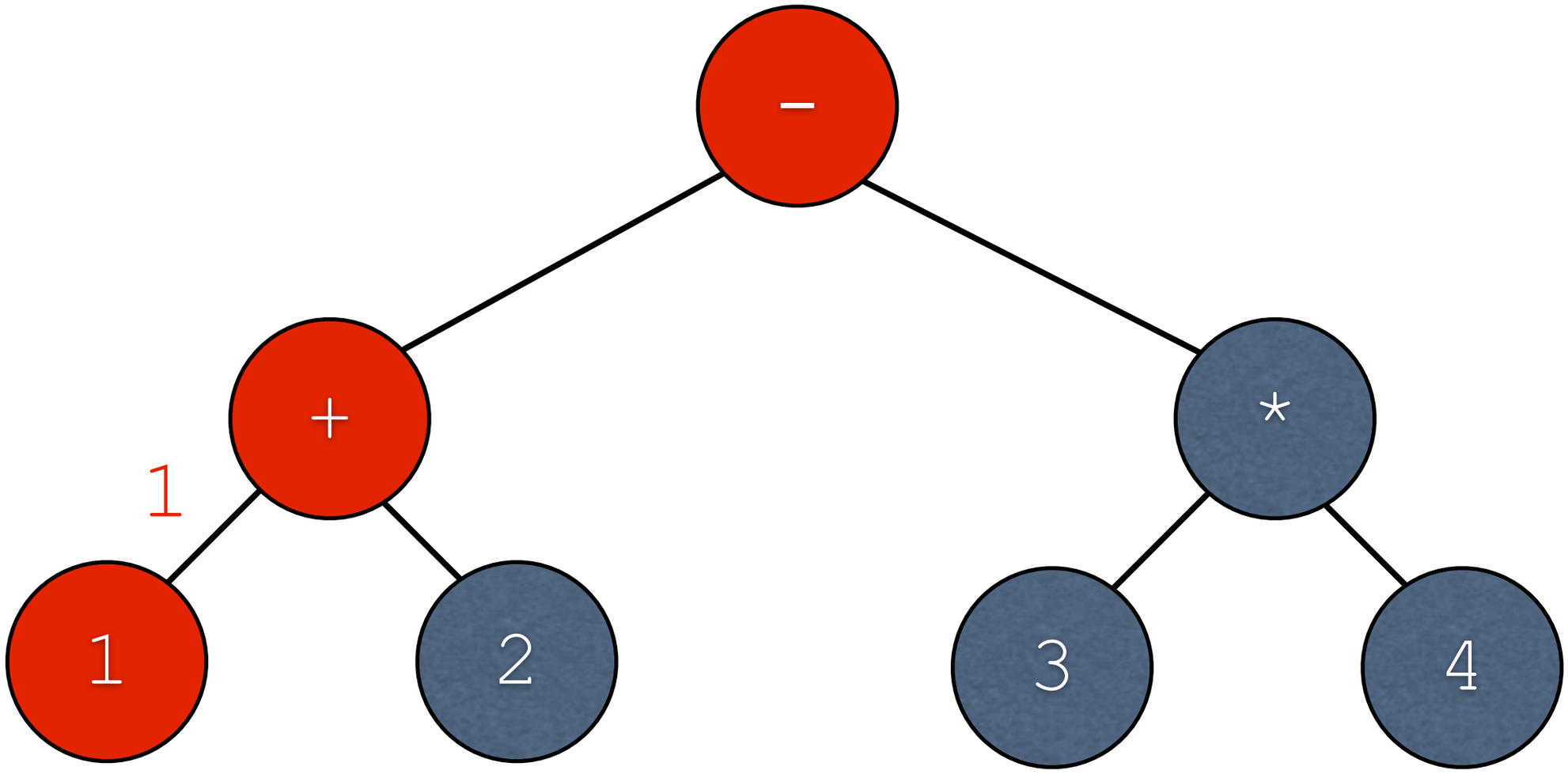
Evaluation



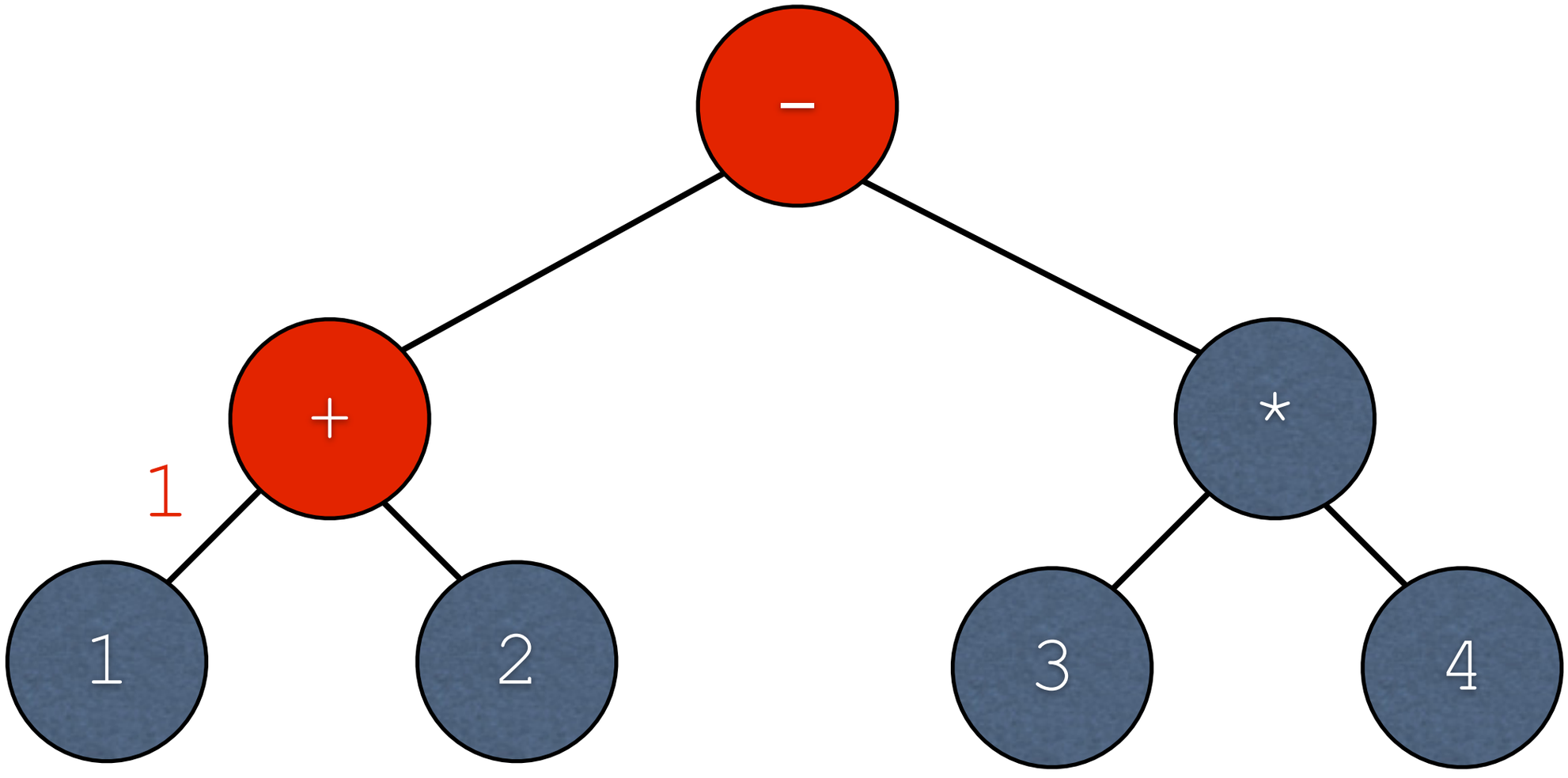
Evaluation



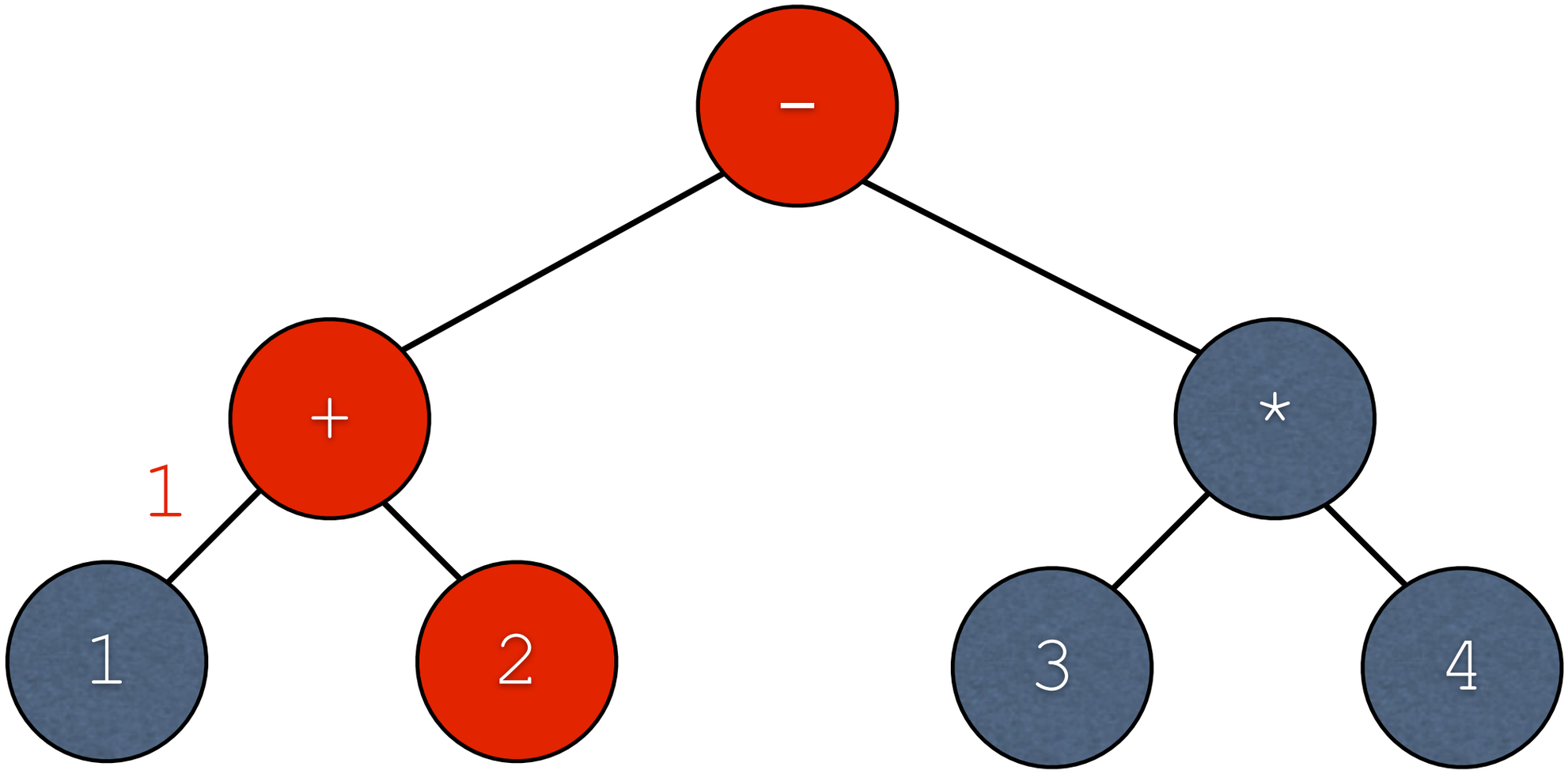
Evaluation



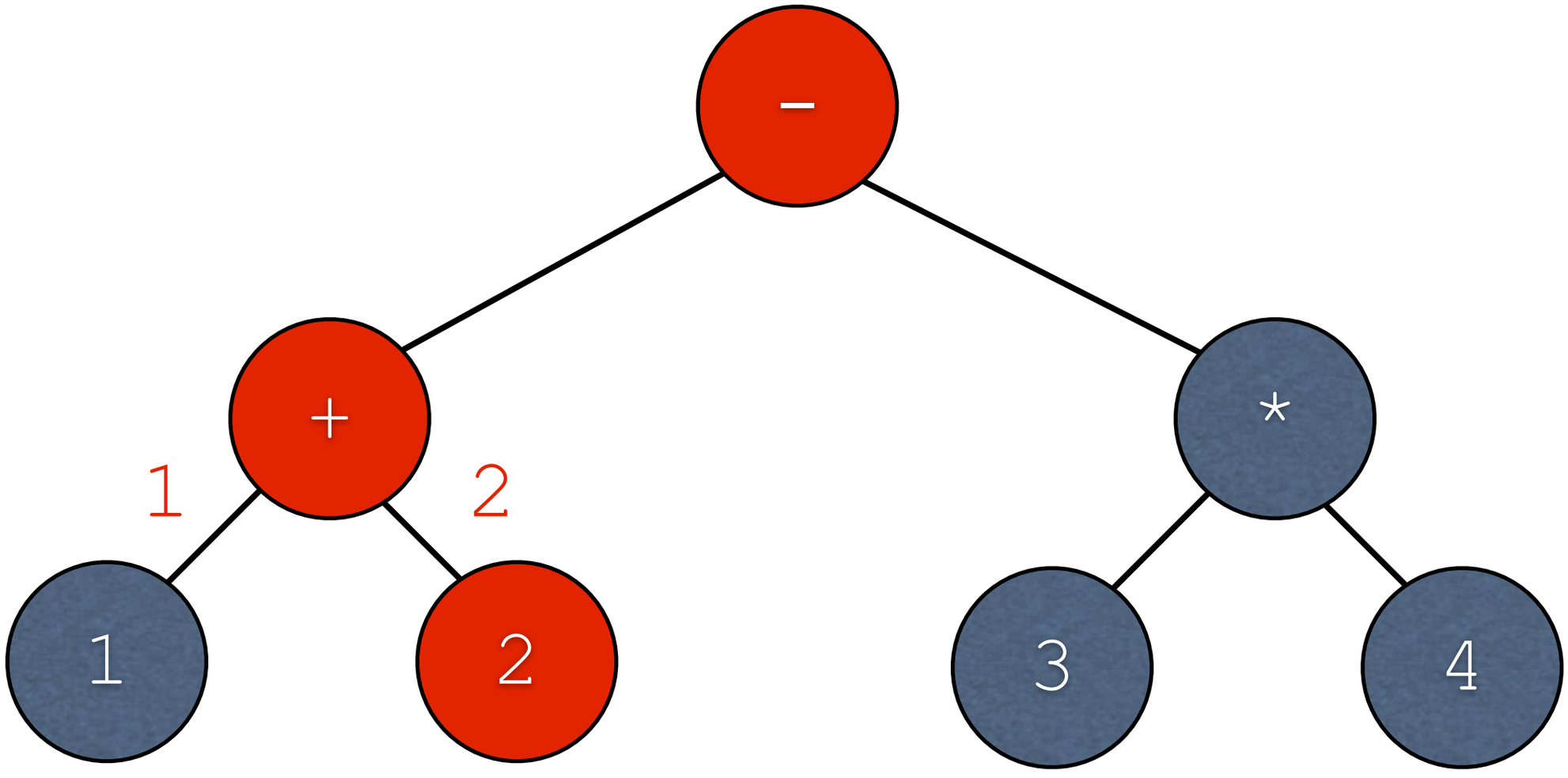
Evaluation



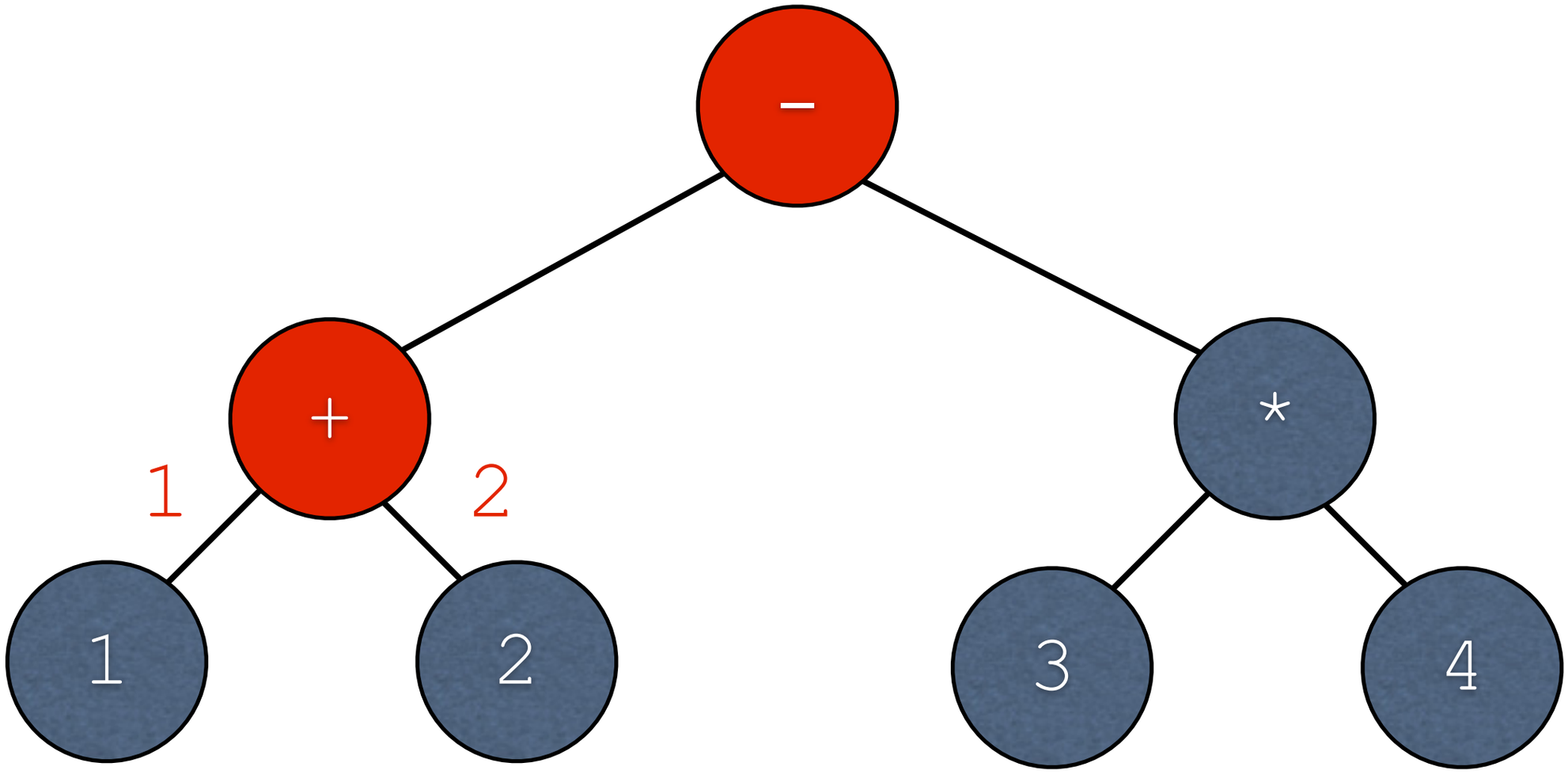
Evaluation



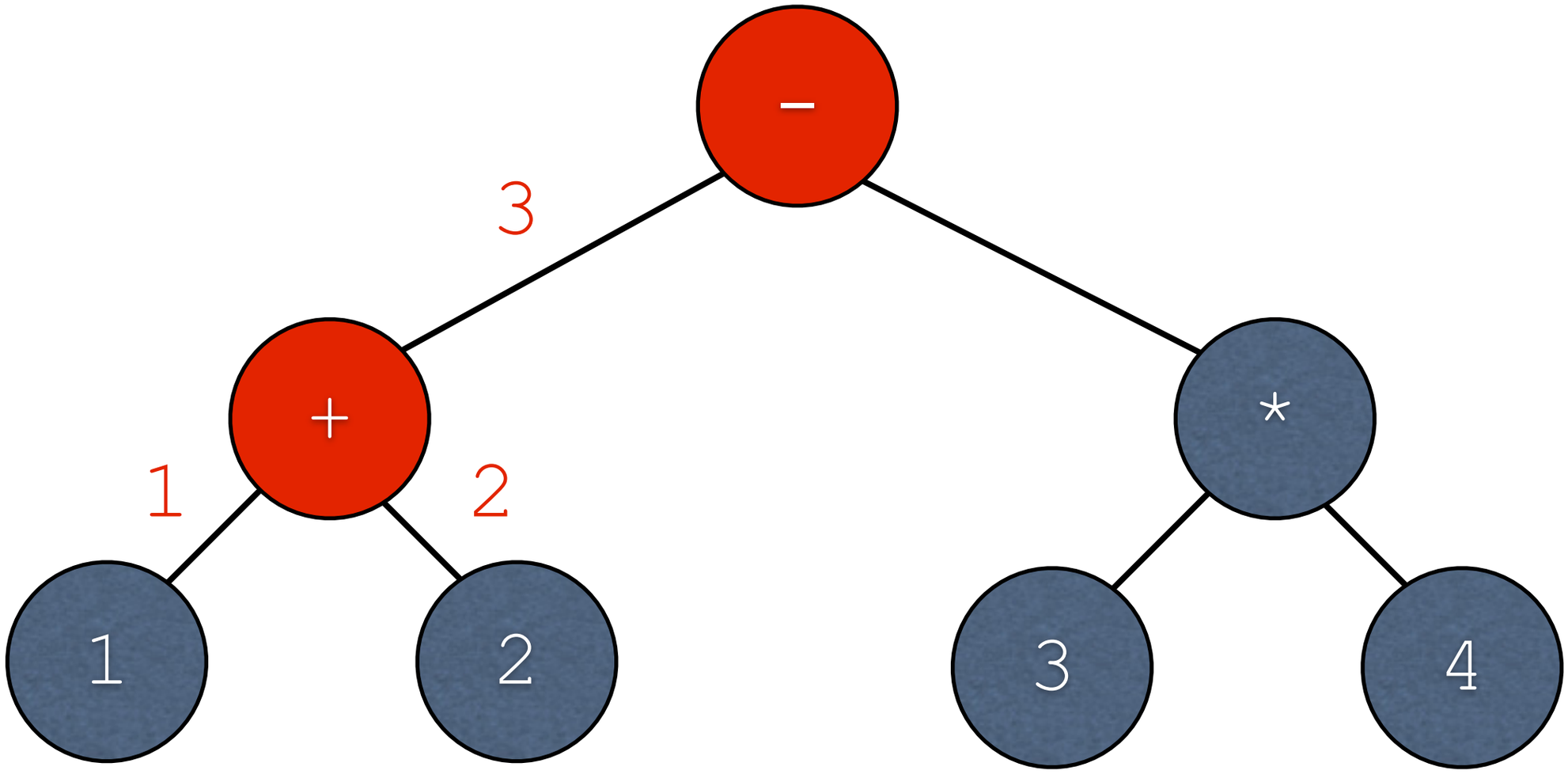
Evaluation



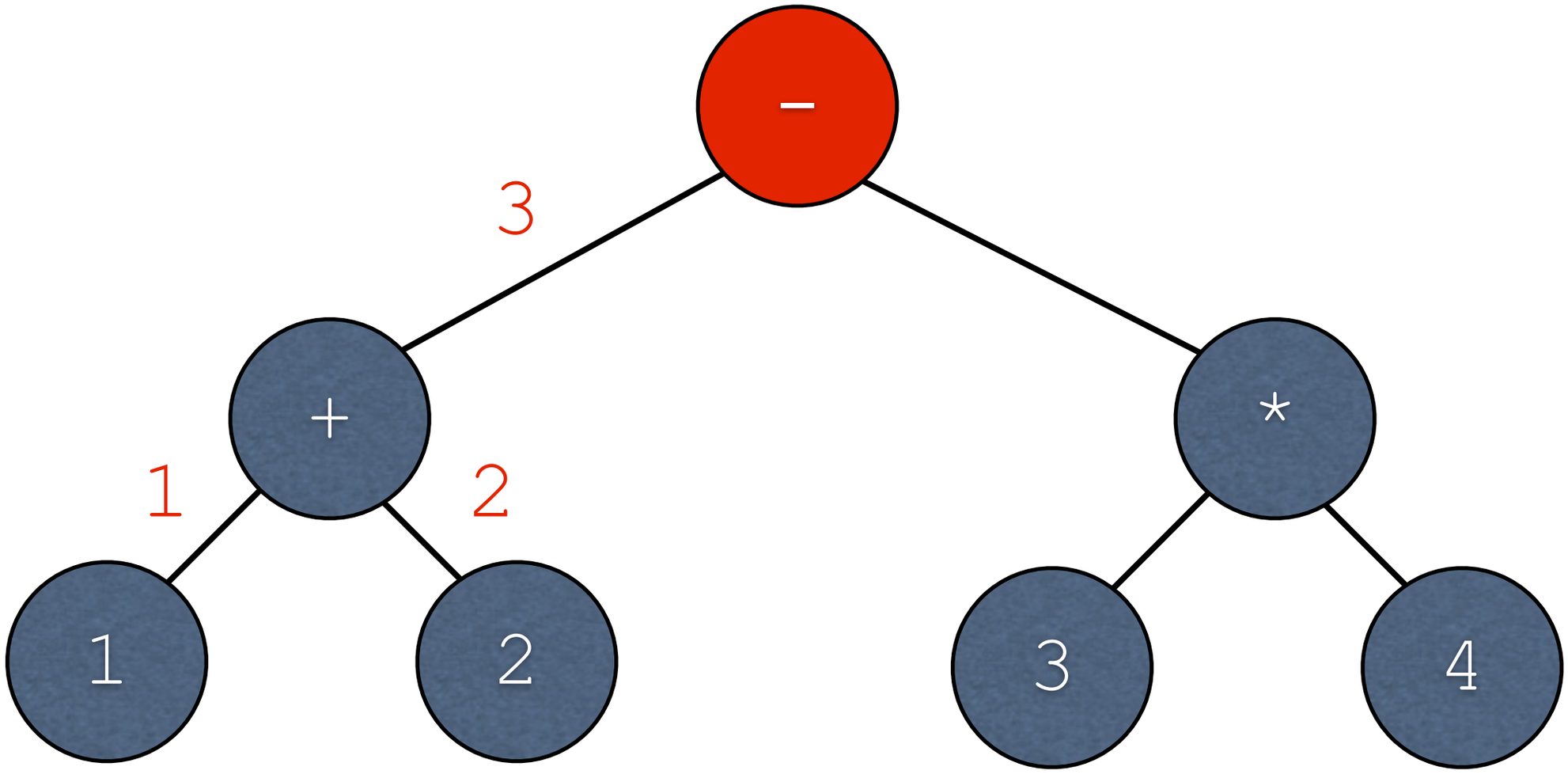
Evaluation



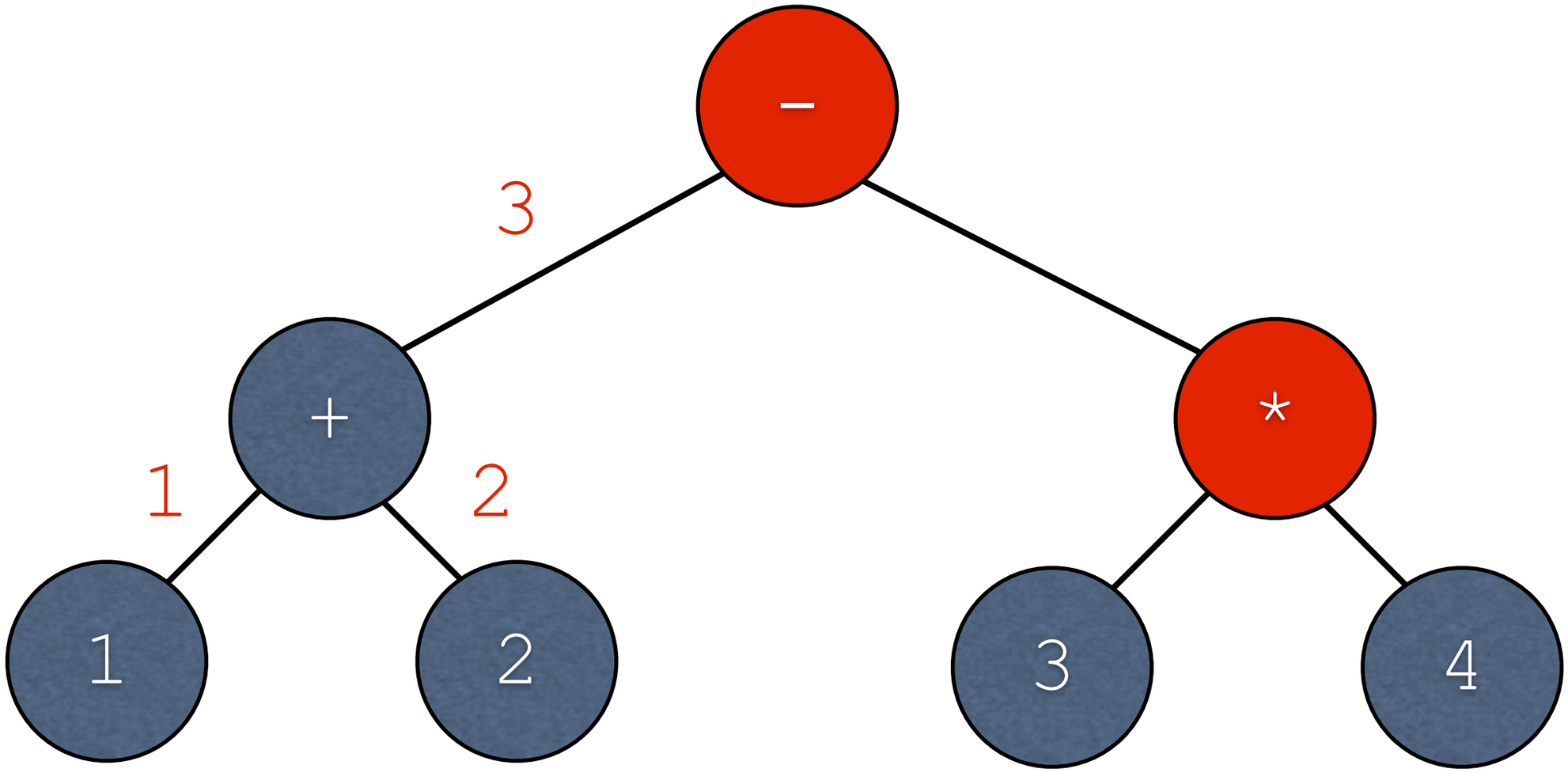
Evaluation



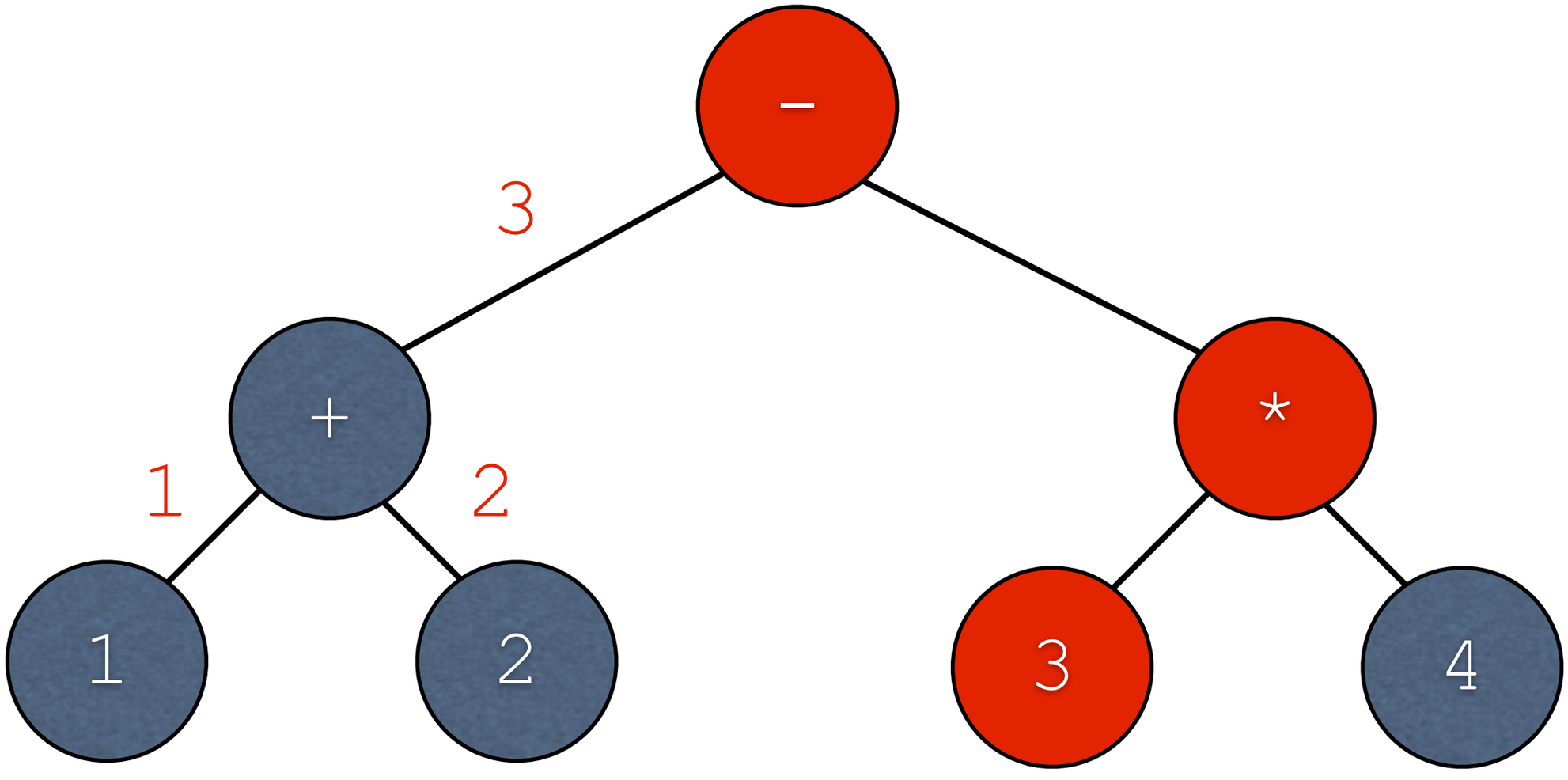
Evaluation



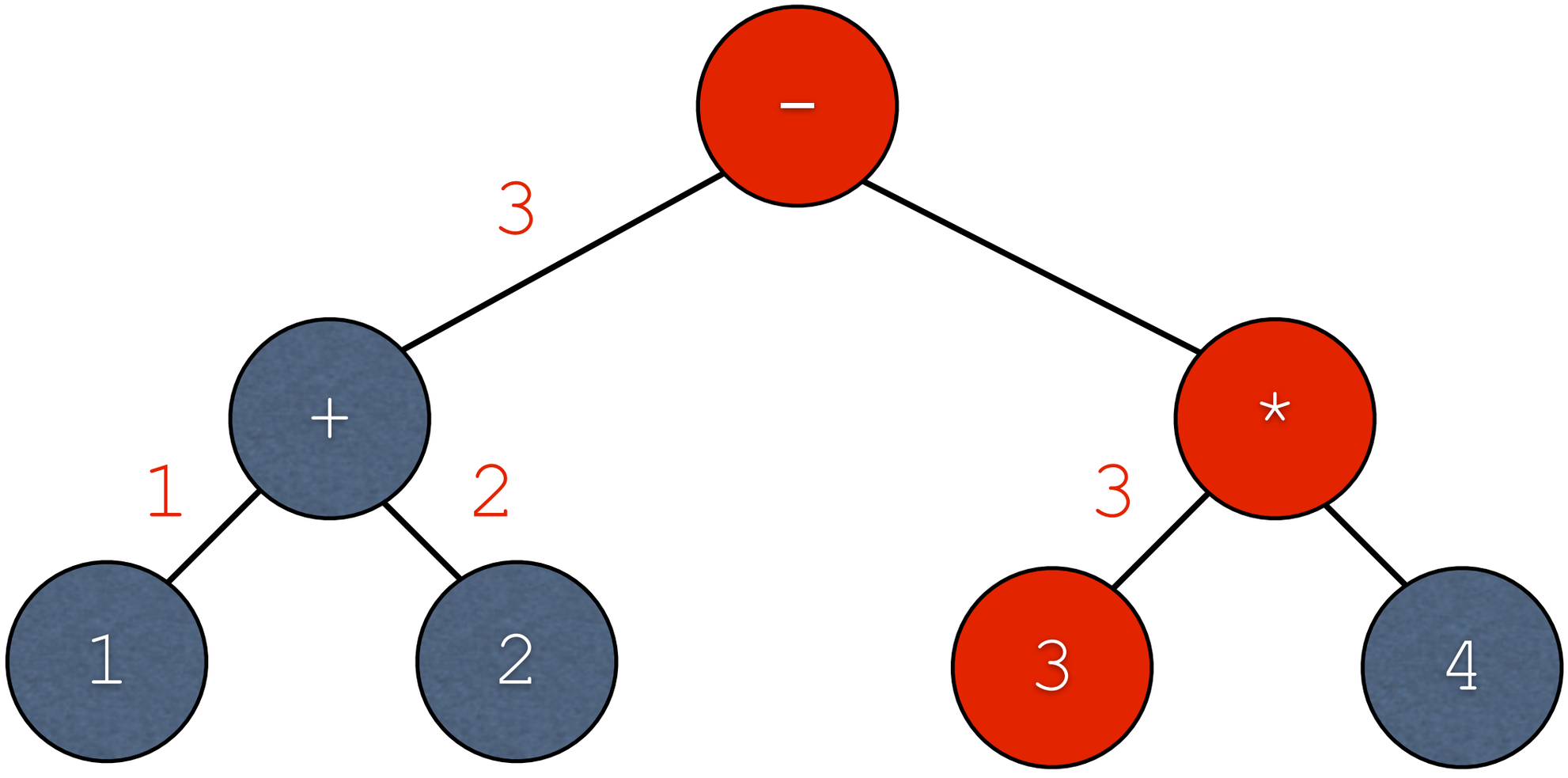
Evaluation



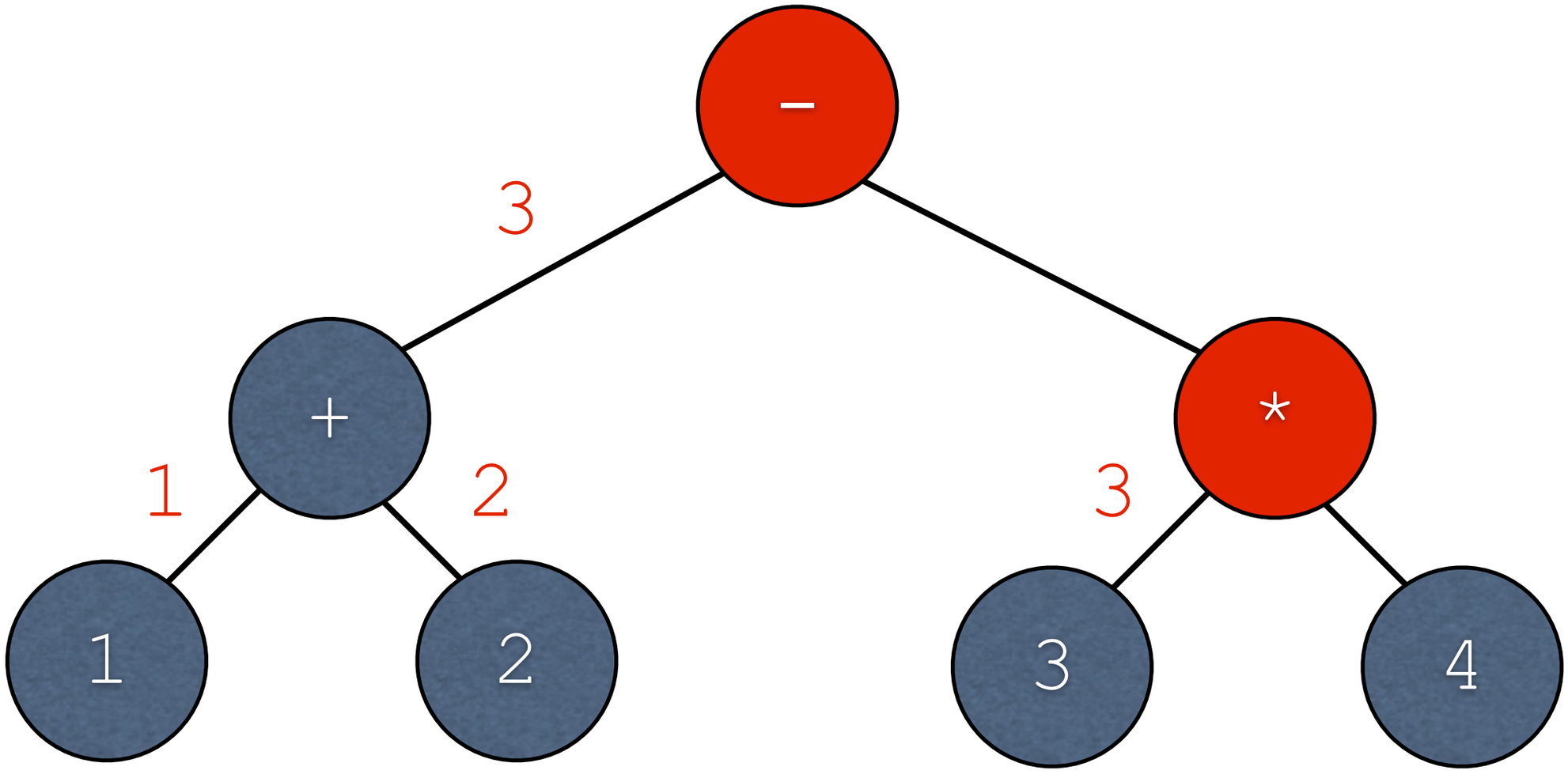
Evaluation



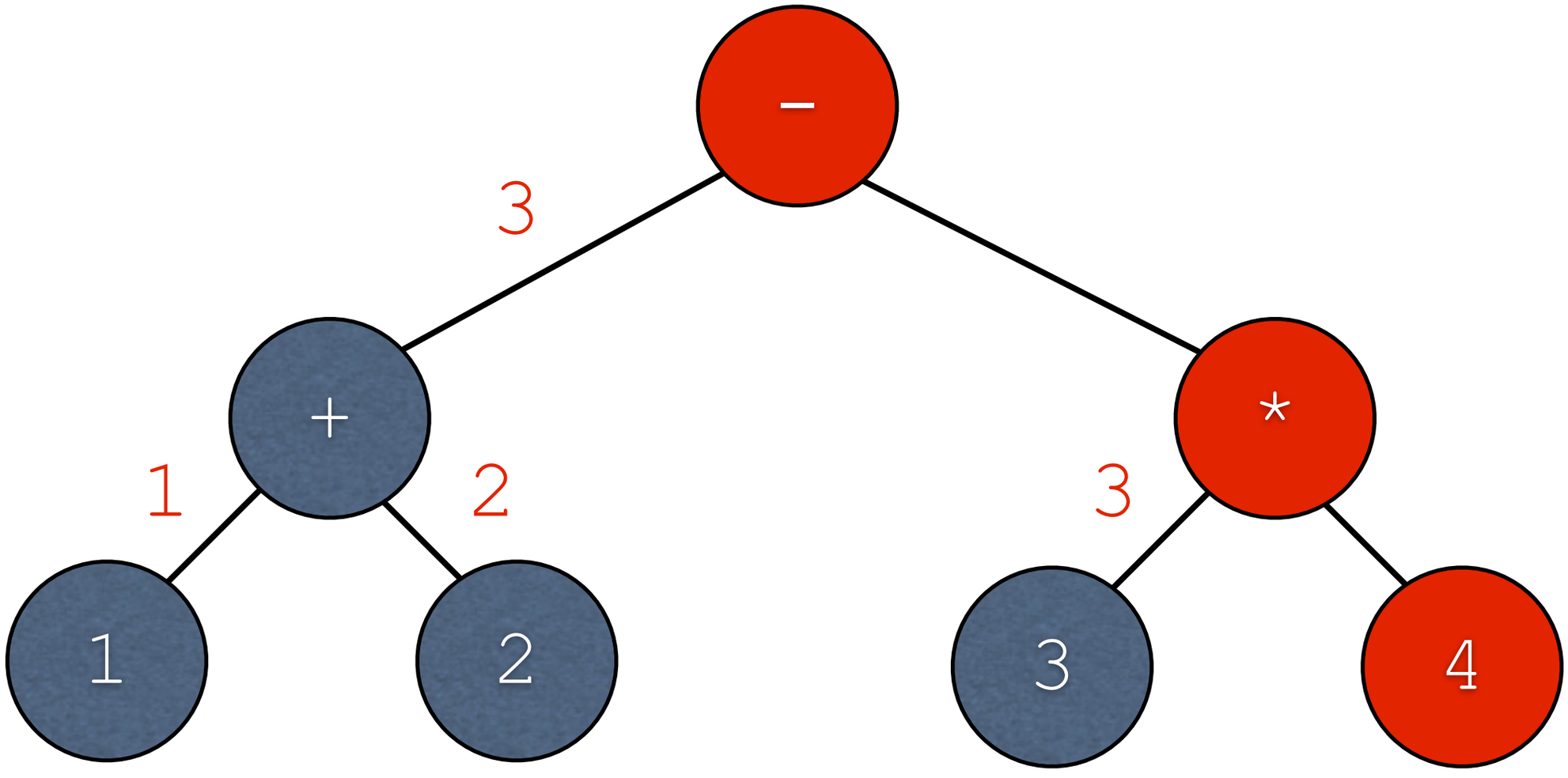
Evaluation



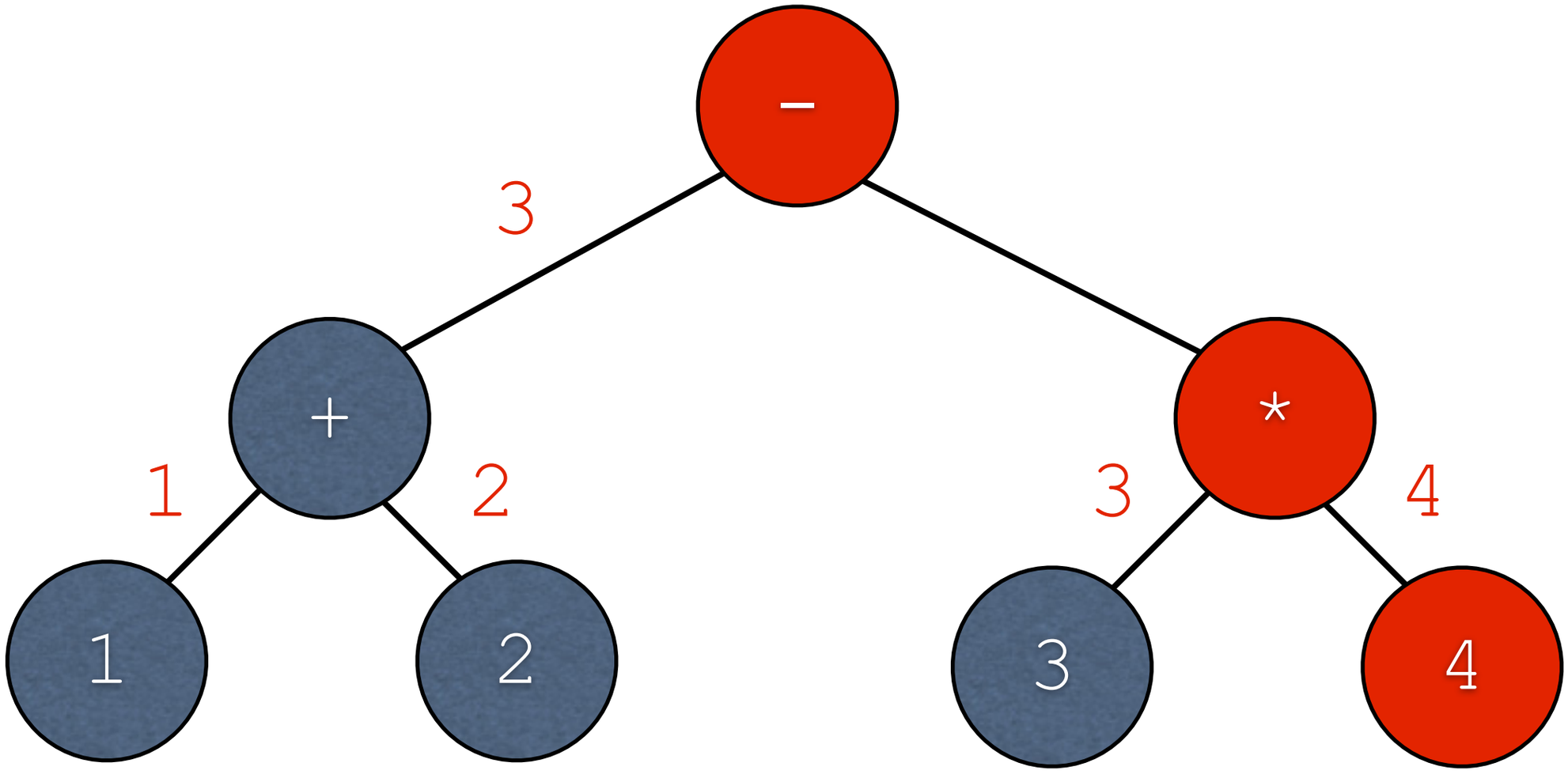
Evaluation



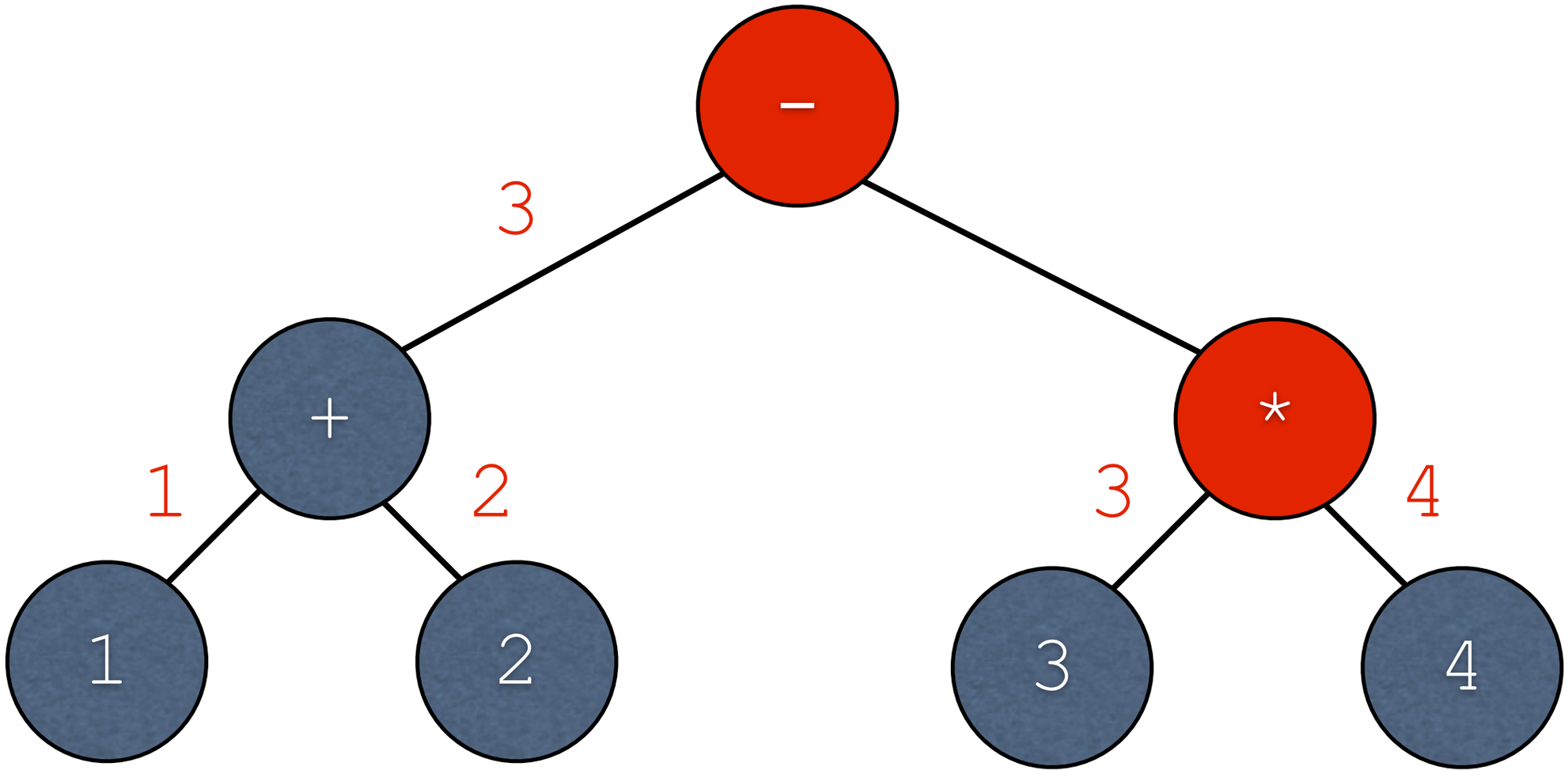
Evaluation



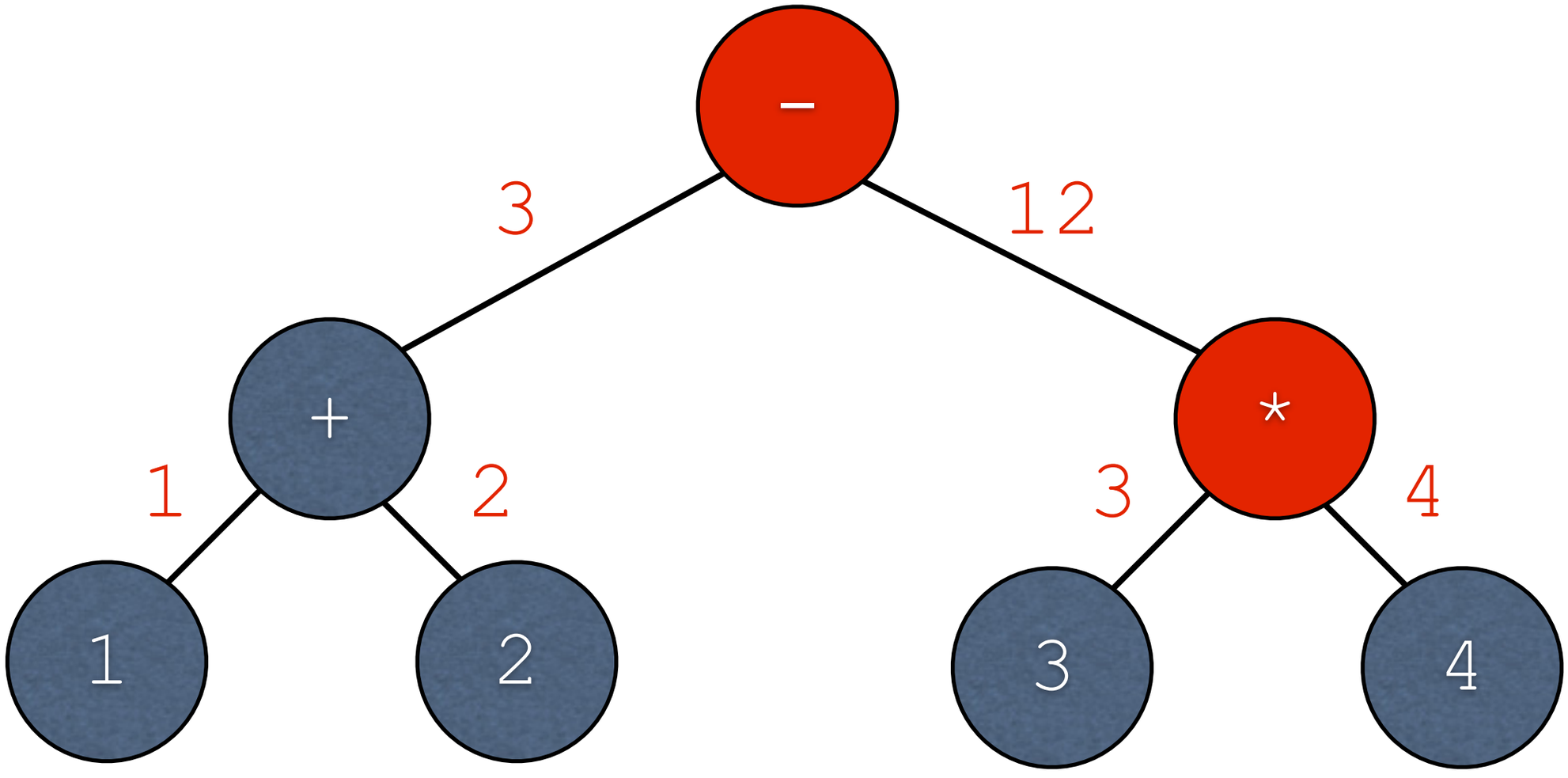
Evaluation



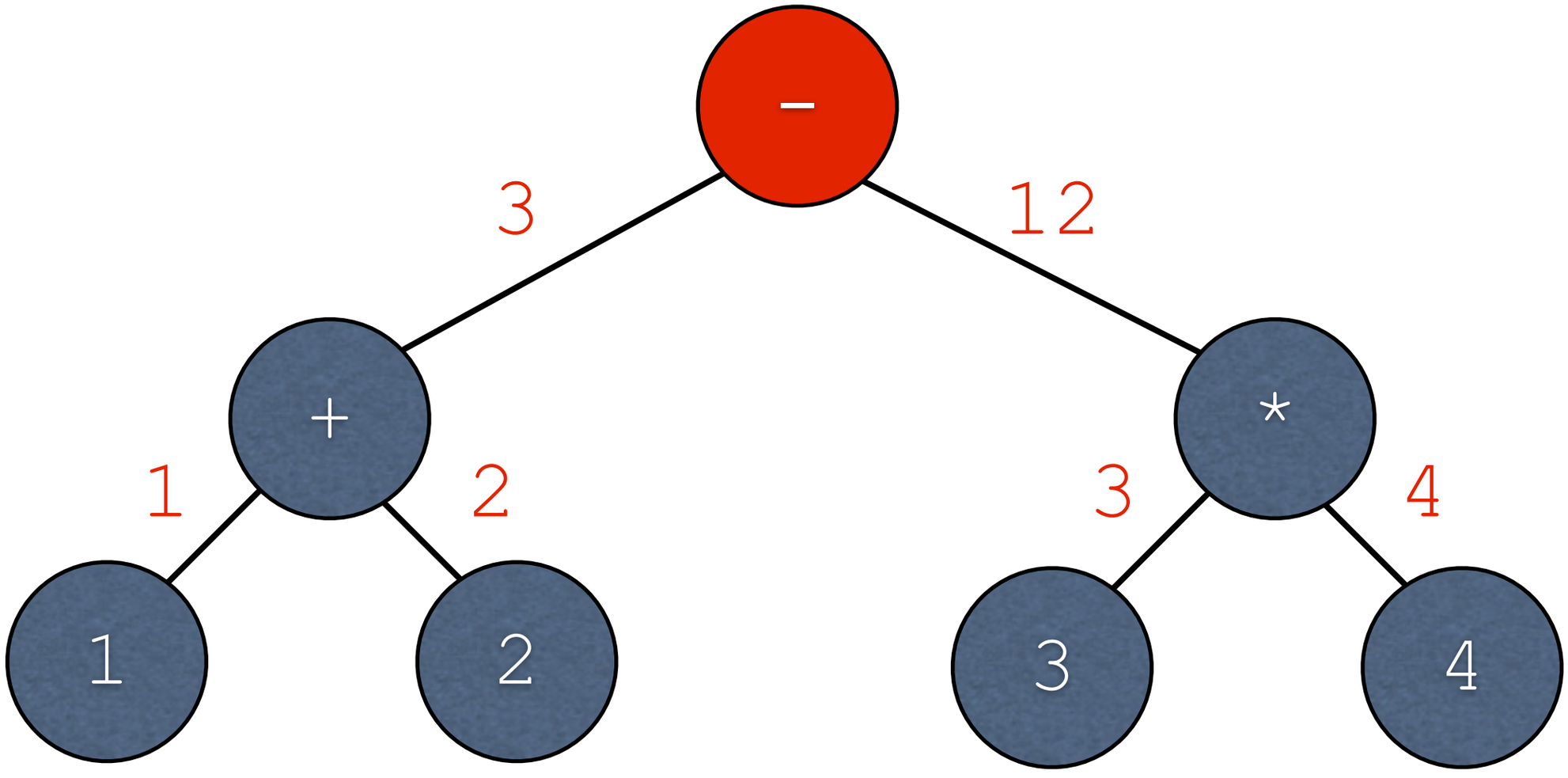
Evaluation



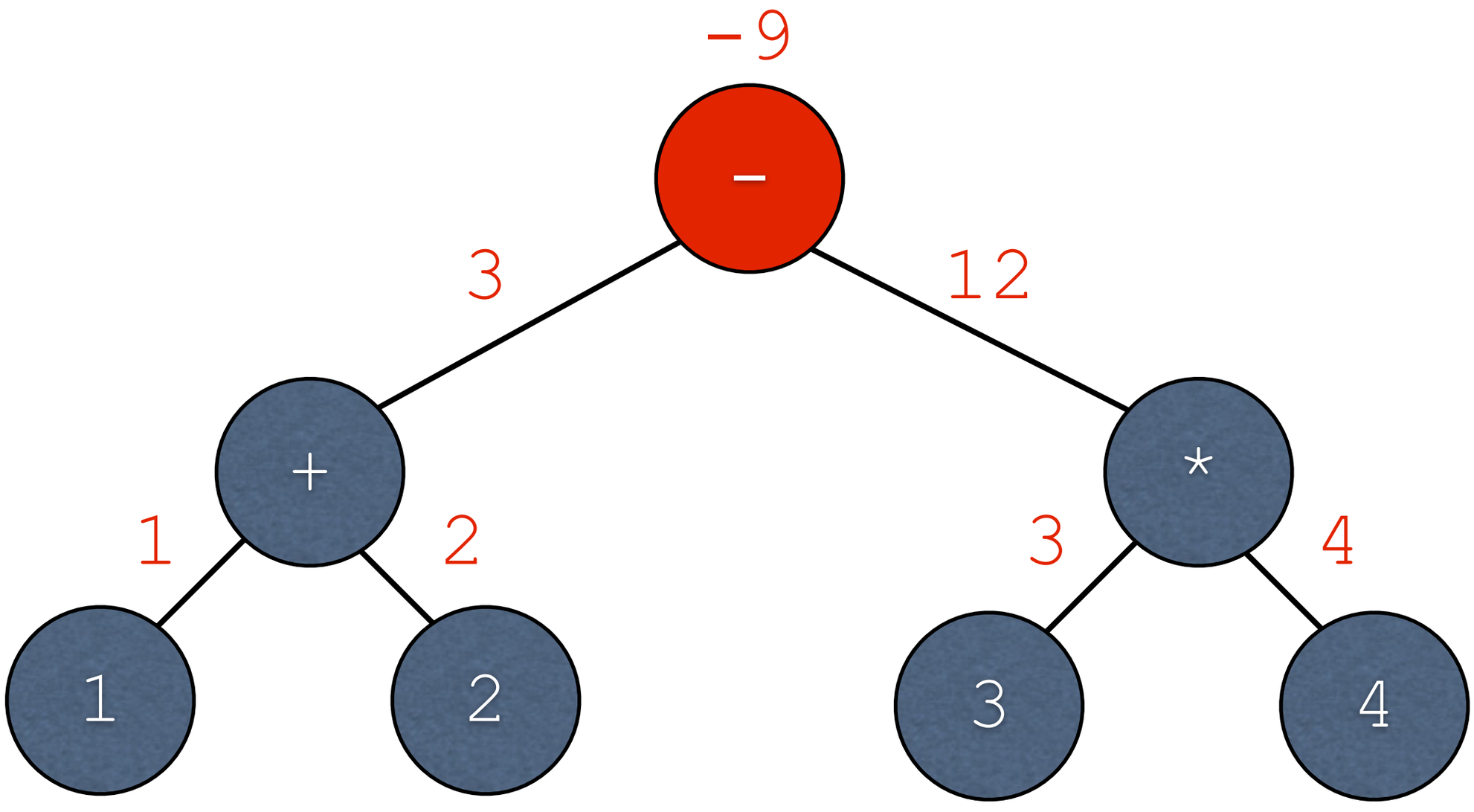
Evaluation



Evaluation



Evaluation



Exercise: Second Side of AST/Evaluation Sheet

Evaluator Example:

arithmetic_evaluator.py

SAT and Semantic Tableau

SAT Background

SAT

- Short for the Boolean satisfiability problem
- Given a Boolean formula with variables, is there an assignment of true/false to the variables which makes the formula true?

SAT

- Short for the Boolean satisfiability problem
- Given a Boolean formula with variables, is there an assignment of true/false to the variables which makes the formula true?

$$(x \lor \neg y) \land (\neg x \lor z)$$

SAT

- Short for the Boolean satisfiability problem
- Given a Boolean formula with variables, is there an assignment of true/false to the variables which makes the formula true?

$(x \lor \neg y) \land (\neg x \lor z)$

Yes: x is true, z is true

SAT

- Short for the Boolean satisfiability problem
- Given a Boolean formula with variables, is there an assignment of true/false to the variables which makes the formula true?

$(x \lor \neg y) \land (\neg x \lor z)$

Yes: x is true, z is true

$(x \land \neg x)$

SAT

- Short for the Boolean satisfiability problem
- Given a Boolean formula with variables, is there an assignment of true/false to the variables which makes the formula true?

$(x \lor \neg y) \land (\neg x \lor z)$

Yes: x is true, z is true

$(x \land \neg x)$

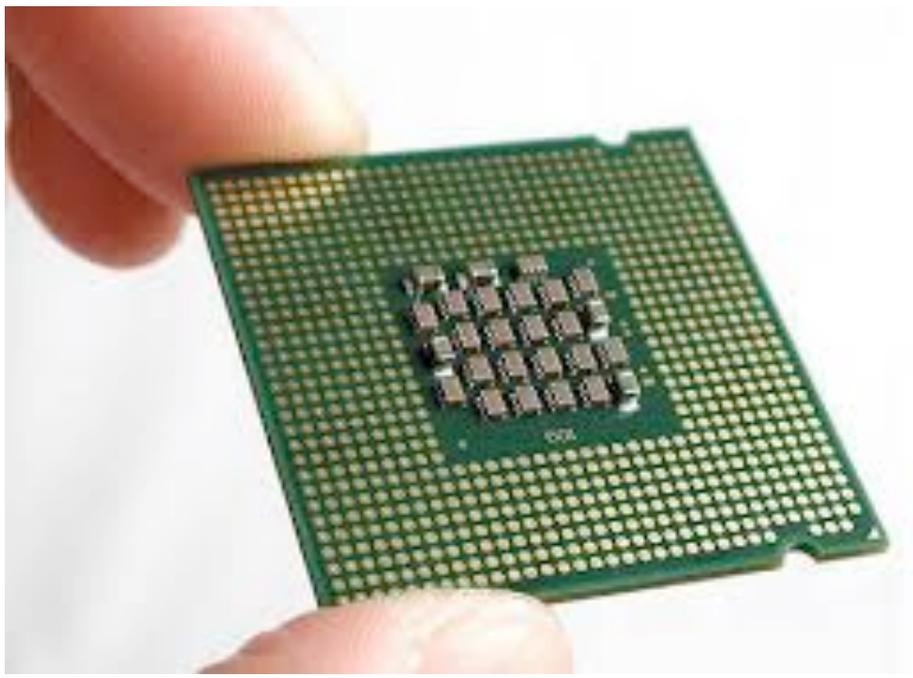
No

Relevance

Widespread usage in hardware and software verification

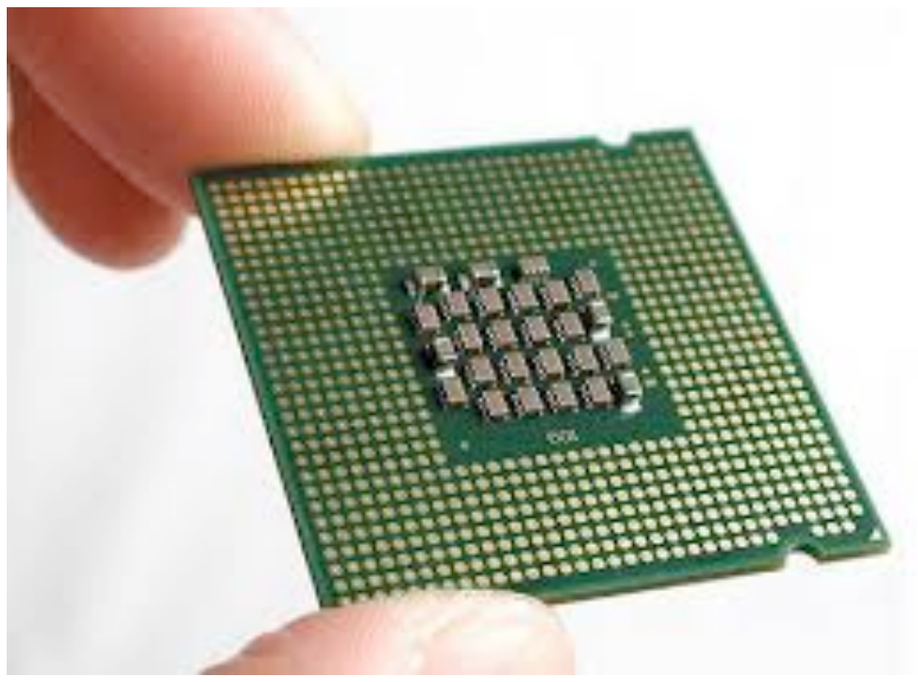
Relevance

Widespread usage in hardware and software verification



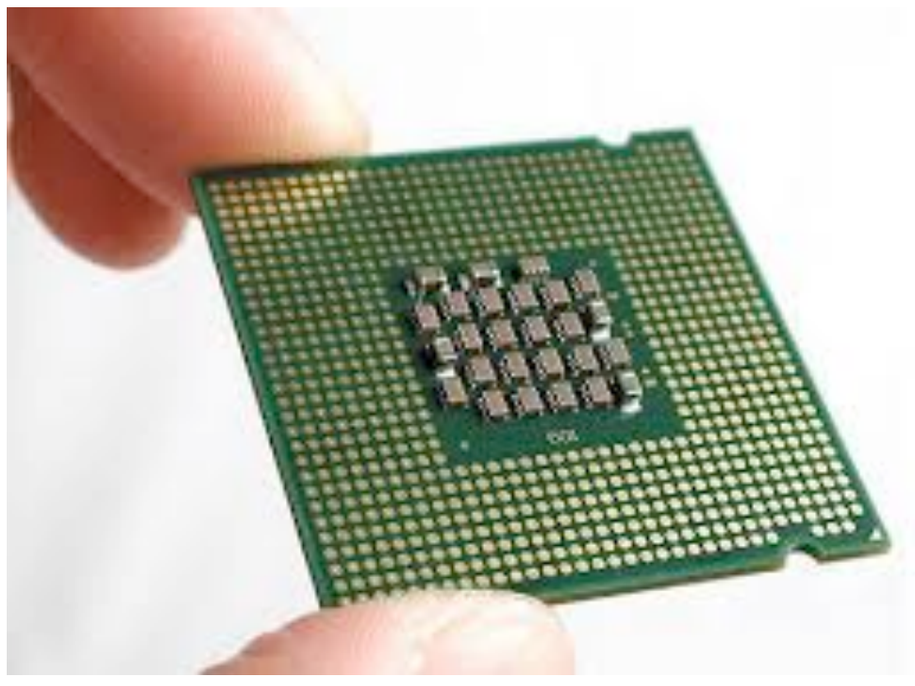
Relevance

Widespread usage in hardware and software verification



Relevance

Widespread usage in hardware and software verification



Relevance to Logic Programming

- Methods for solving SAT can be used to execute logic programs
- Logic programming can be phrased as SAT with some additional stuff

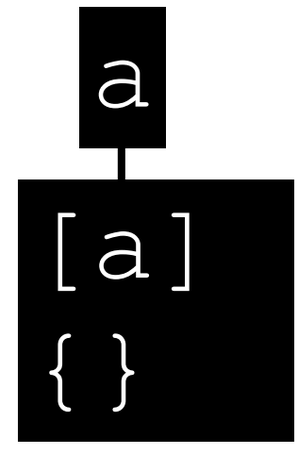
Semantic Tableau

- One method for solving SAT instances
- Basic idea: iterate over the formula
  - Maintain subformulas that must be true
  - Learn which variables must be true/false
  - Stop at conflicts (unsatisfiable), or when no subformulas remain (have solution)

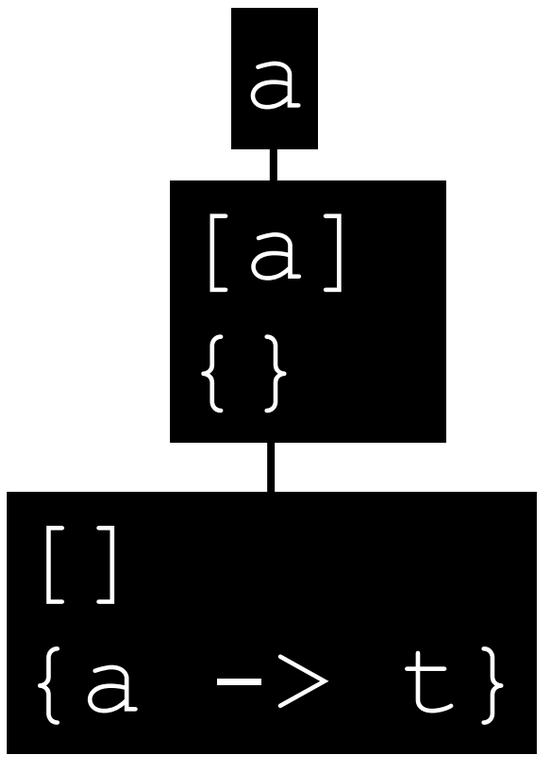
Positive Literals

a

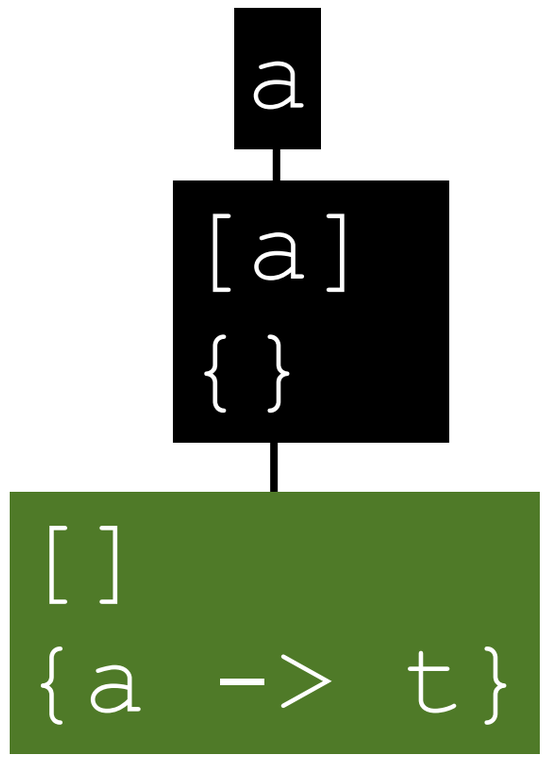
Positive Literals



Positive Literals



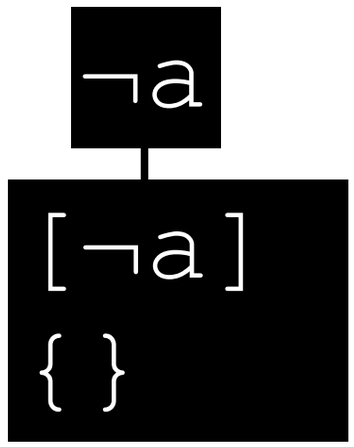
Positive Literals



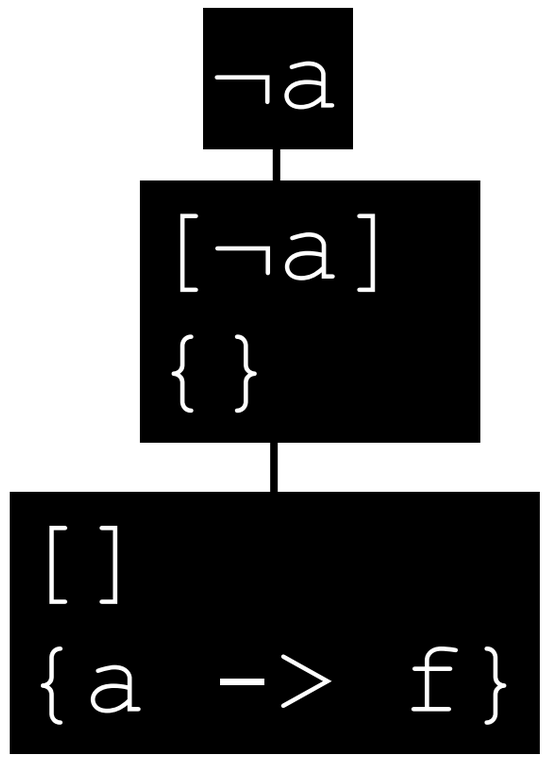
Negative Literals

$\neg a$

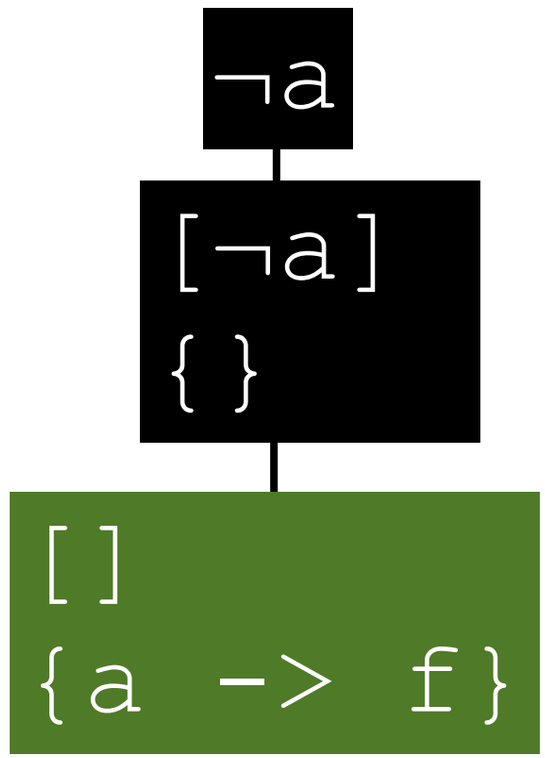
Negative Literals



Negative Literals



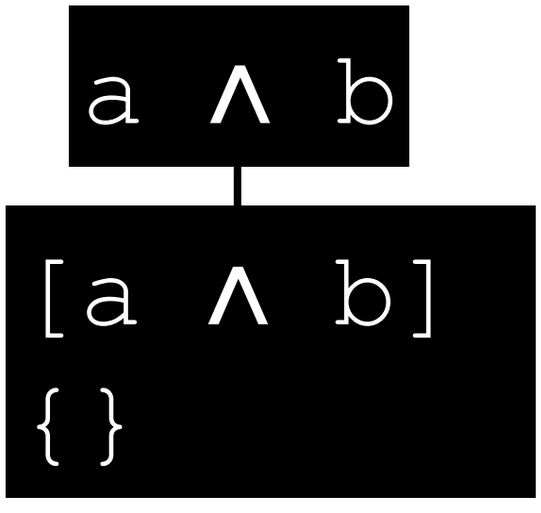
Negative Literals



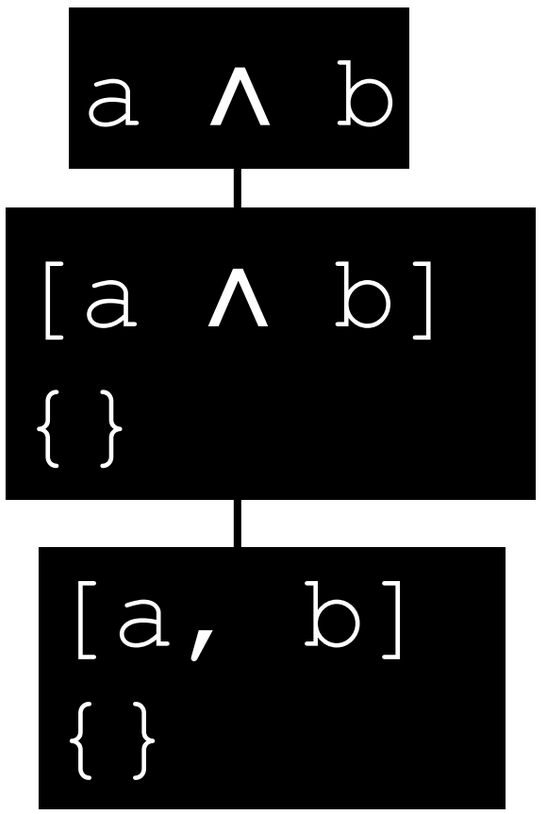
Logical And

$$a \wedge b$$

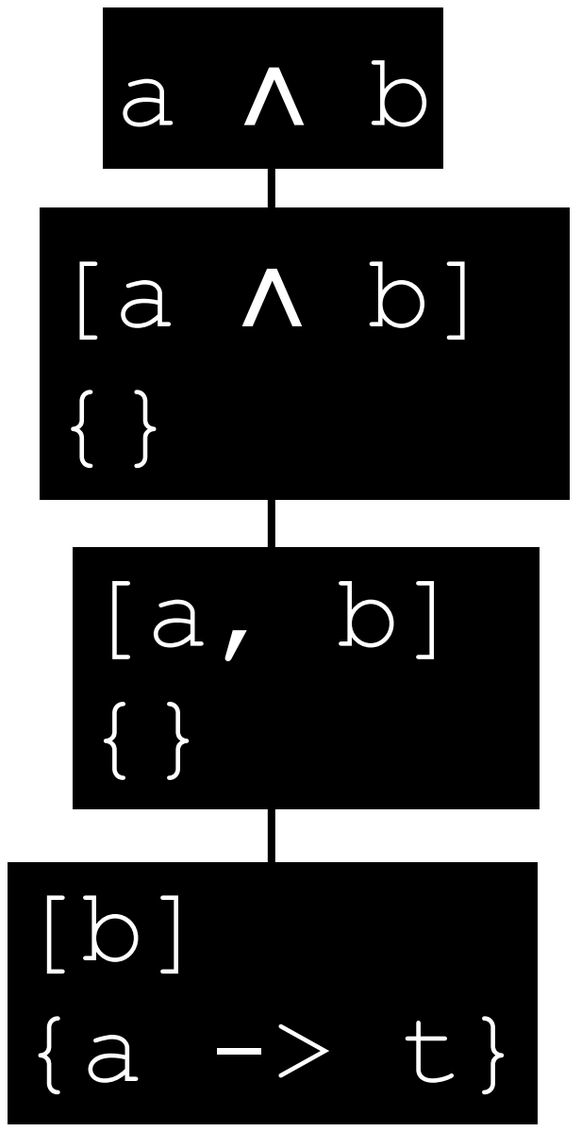
Logical And



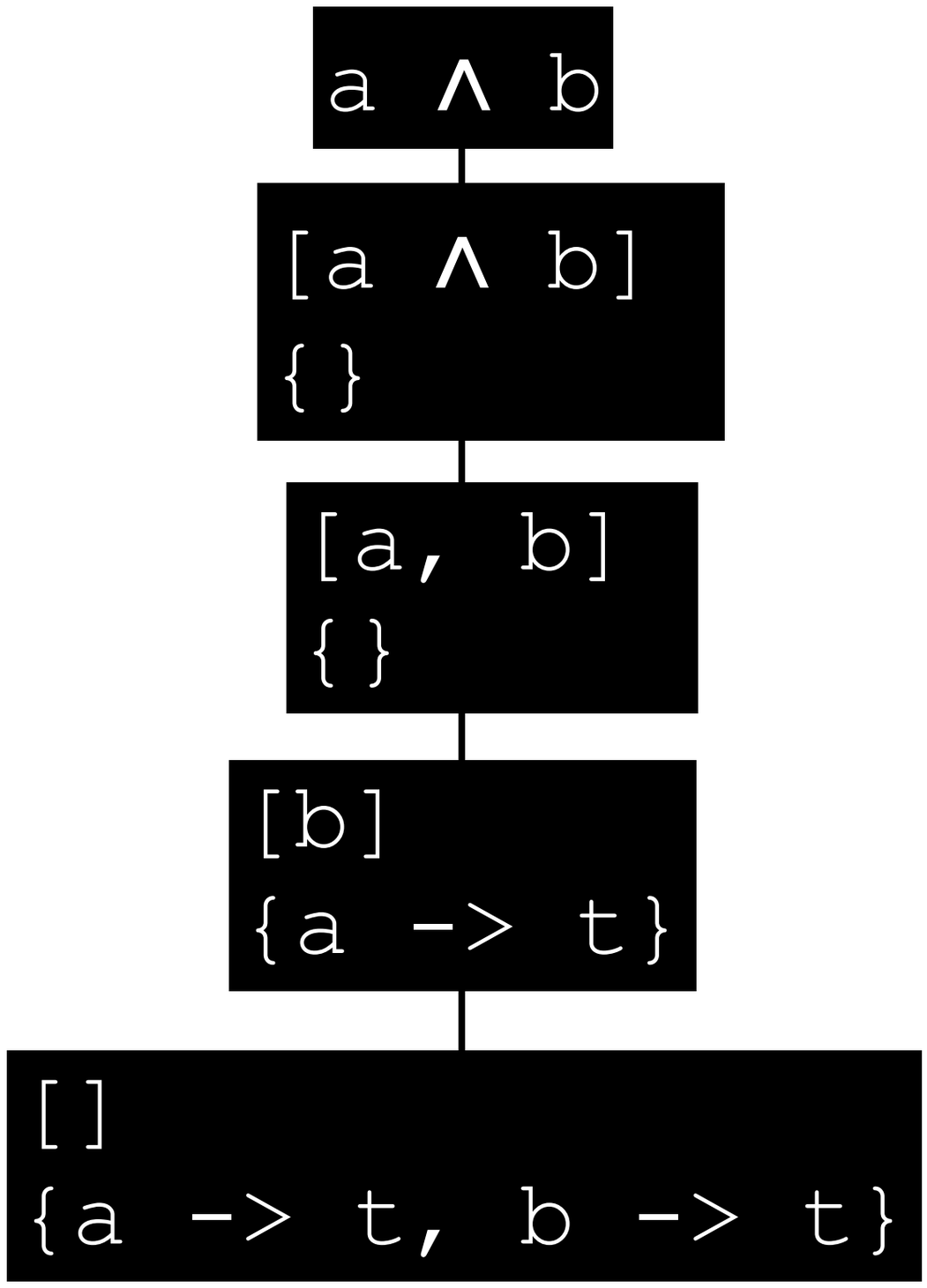
Logical And



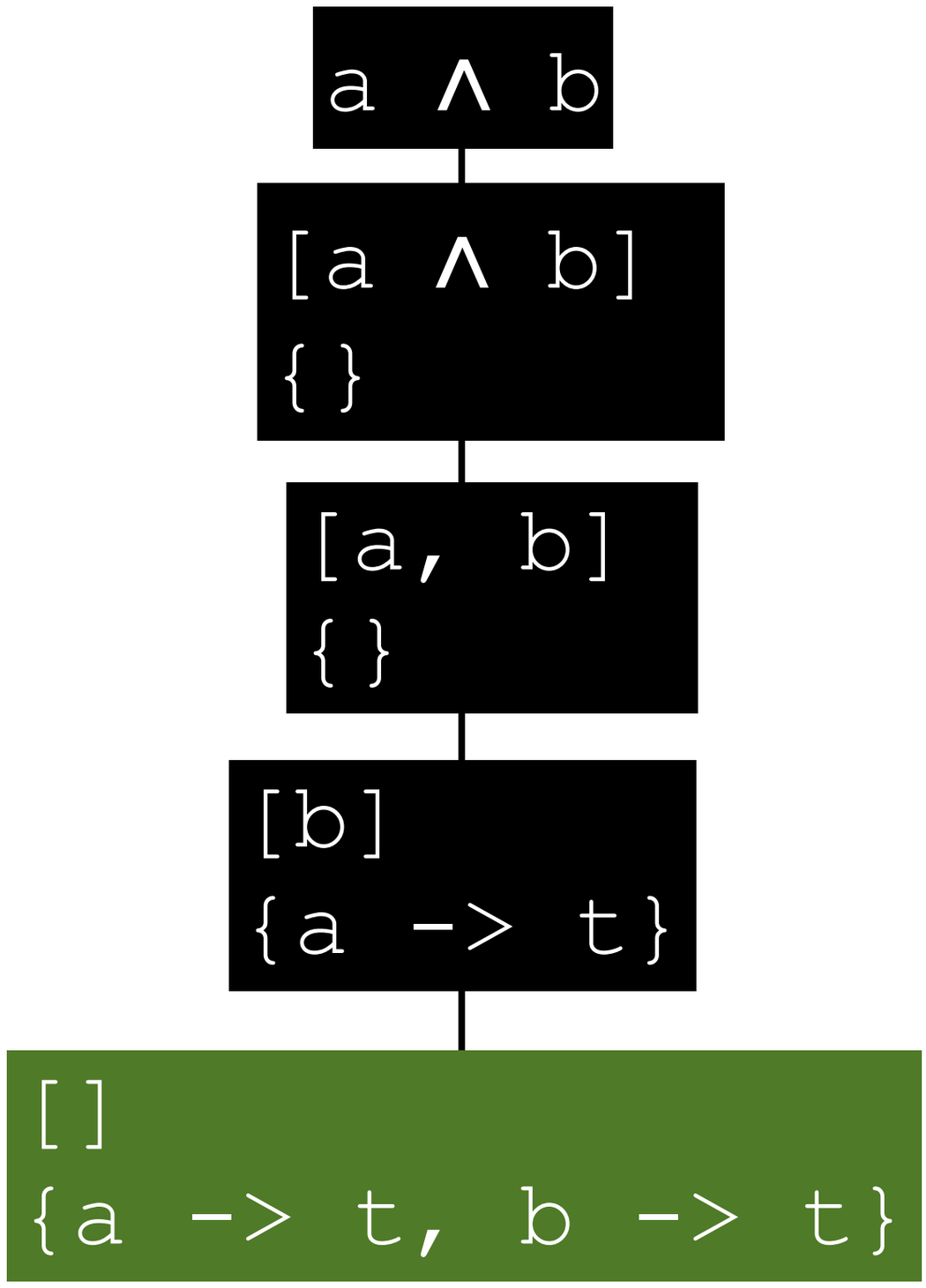
Logical And



Logical And



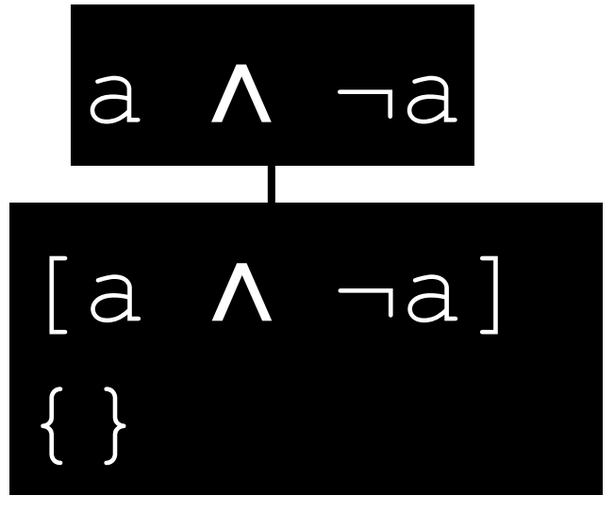
Logical And



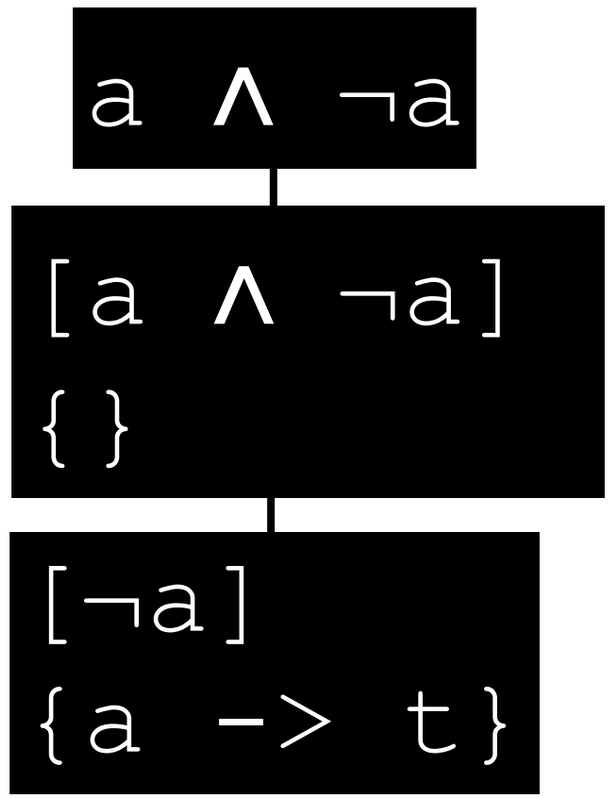
Logical And

$$a \wedge \neg a$$

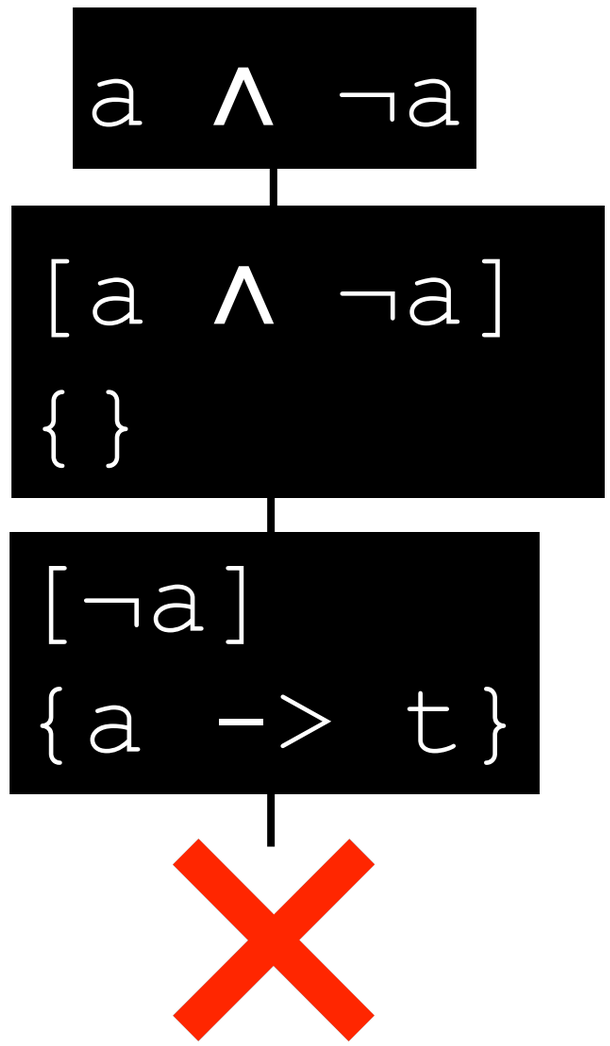
Logical And



Logical And



Logical And

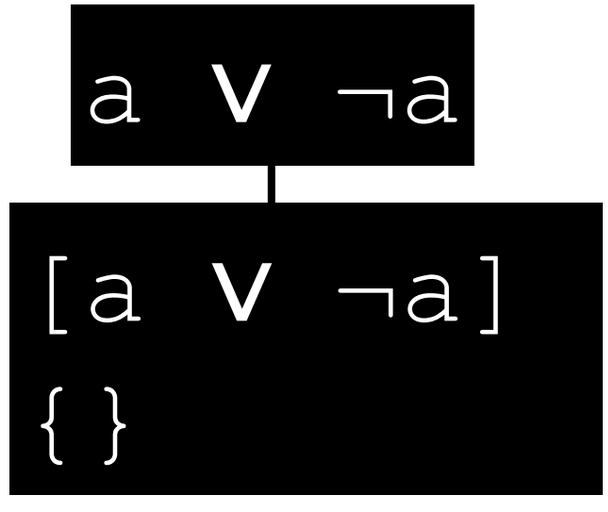


Exercise: First Side of SAT Sheet

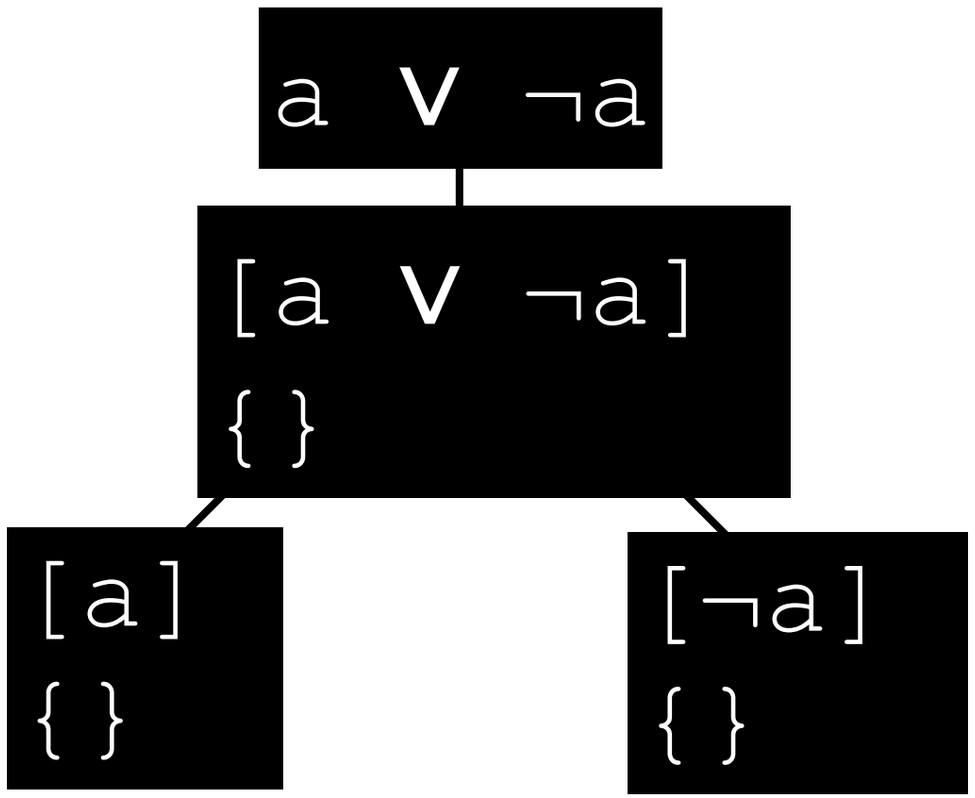
Logical Or

$$a \vee \neg a$$

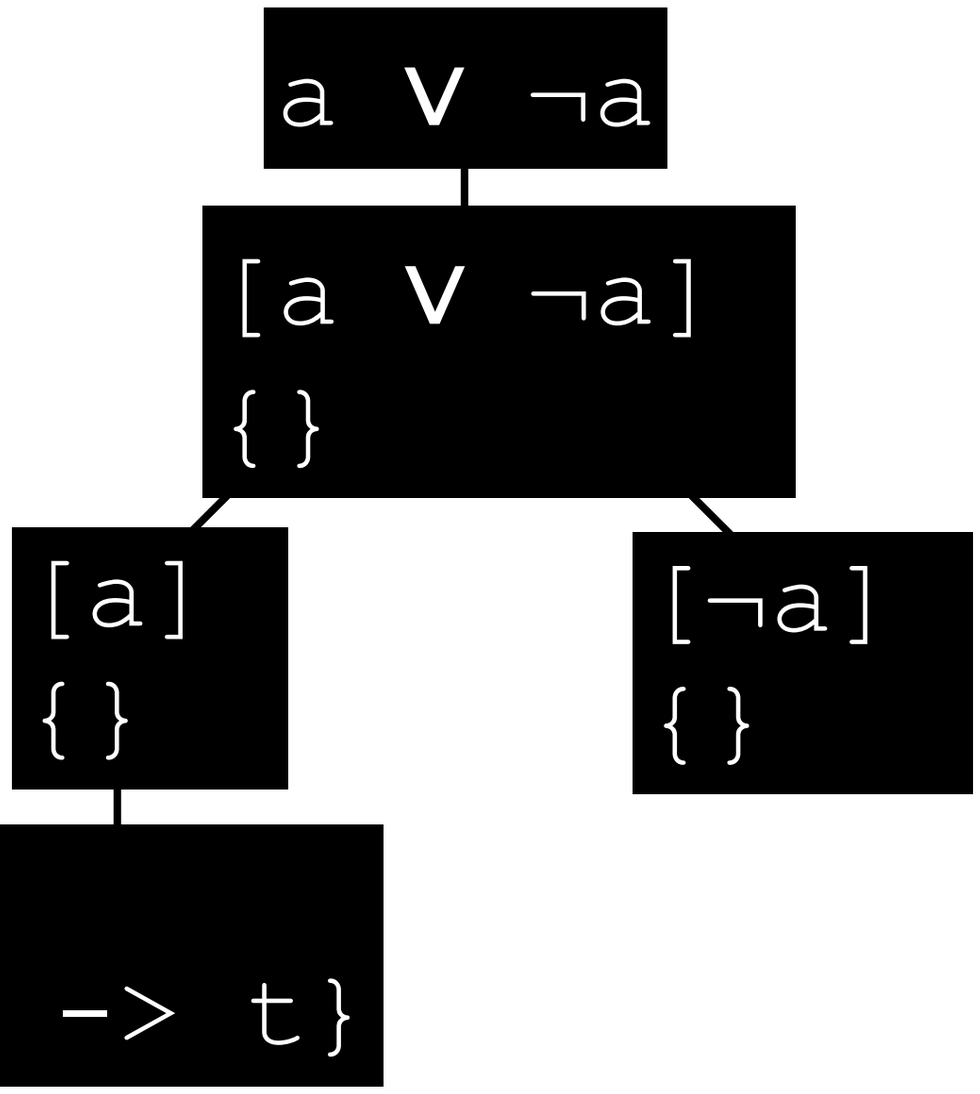
Logical Or



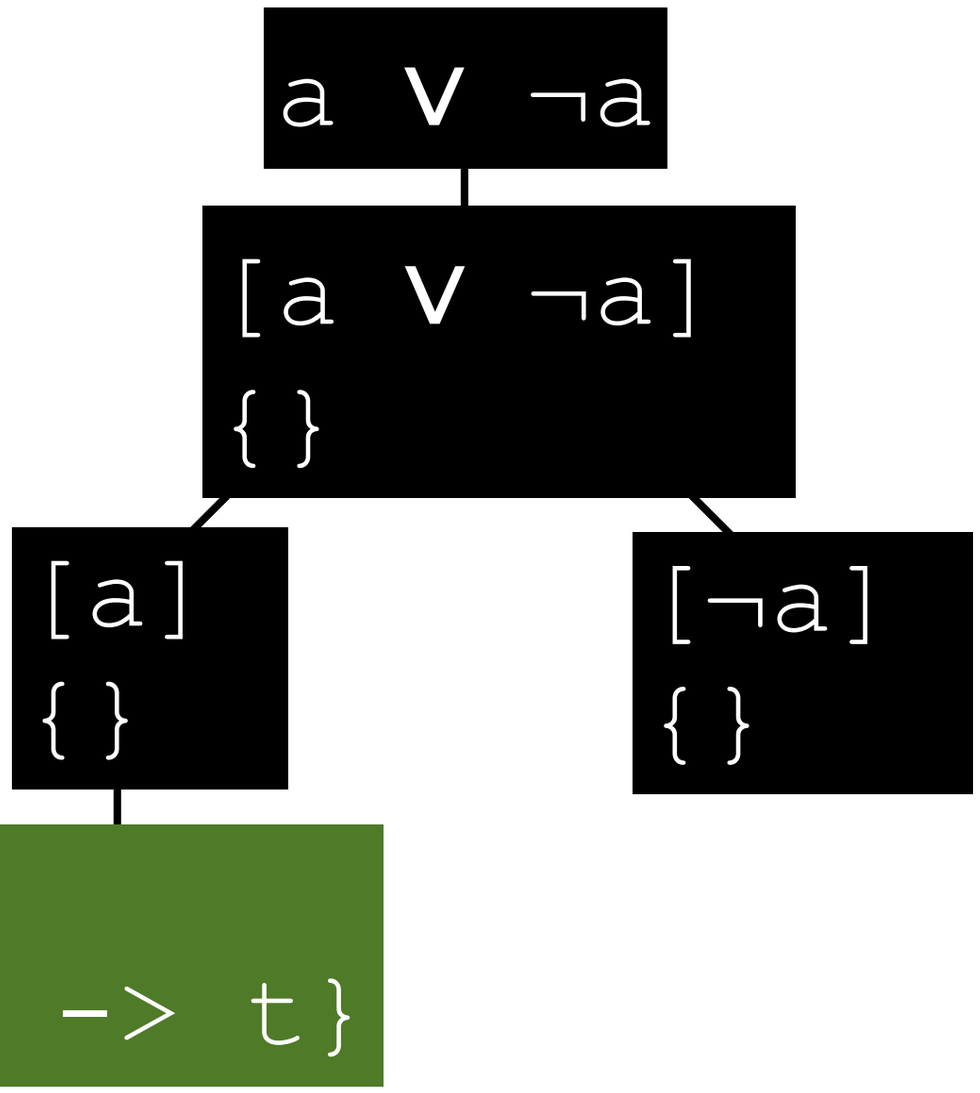
Logical Or



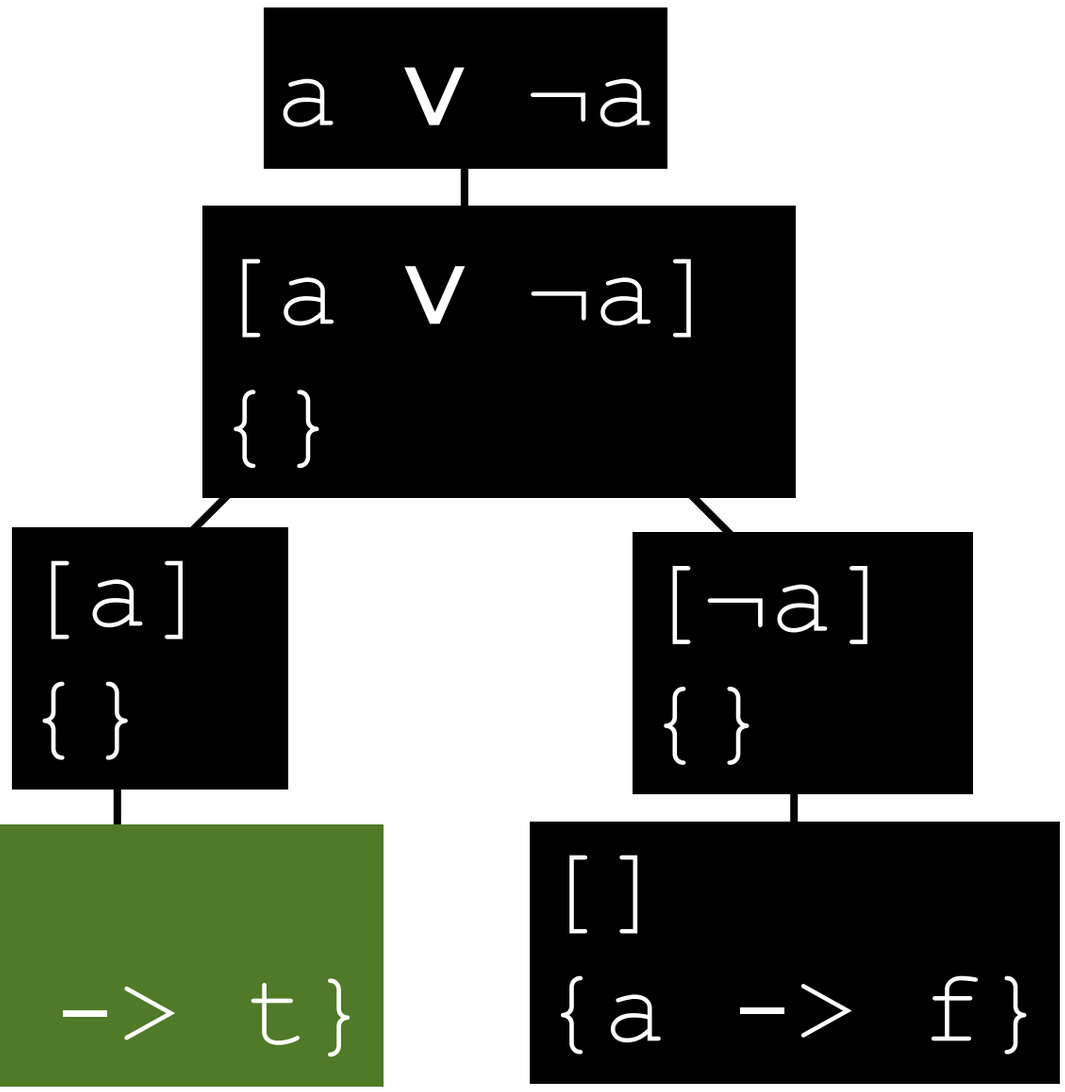
Logical Or



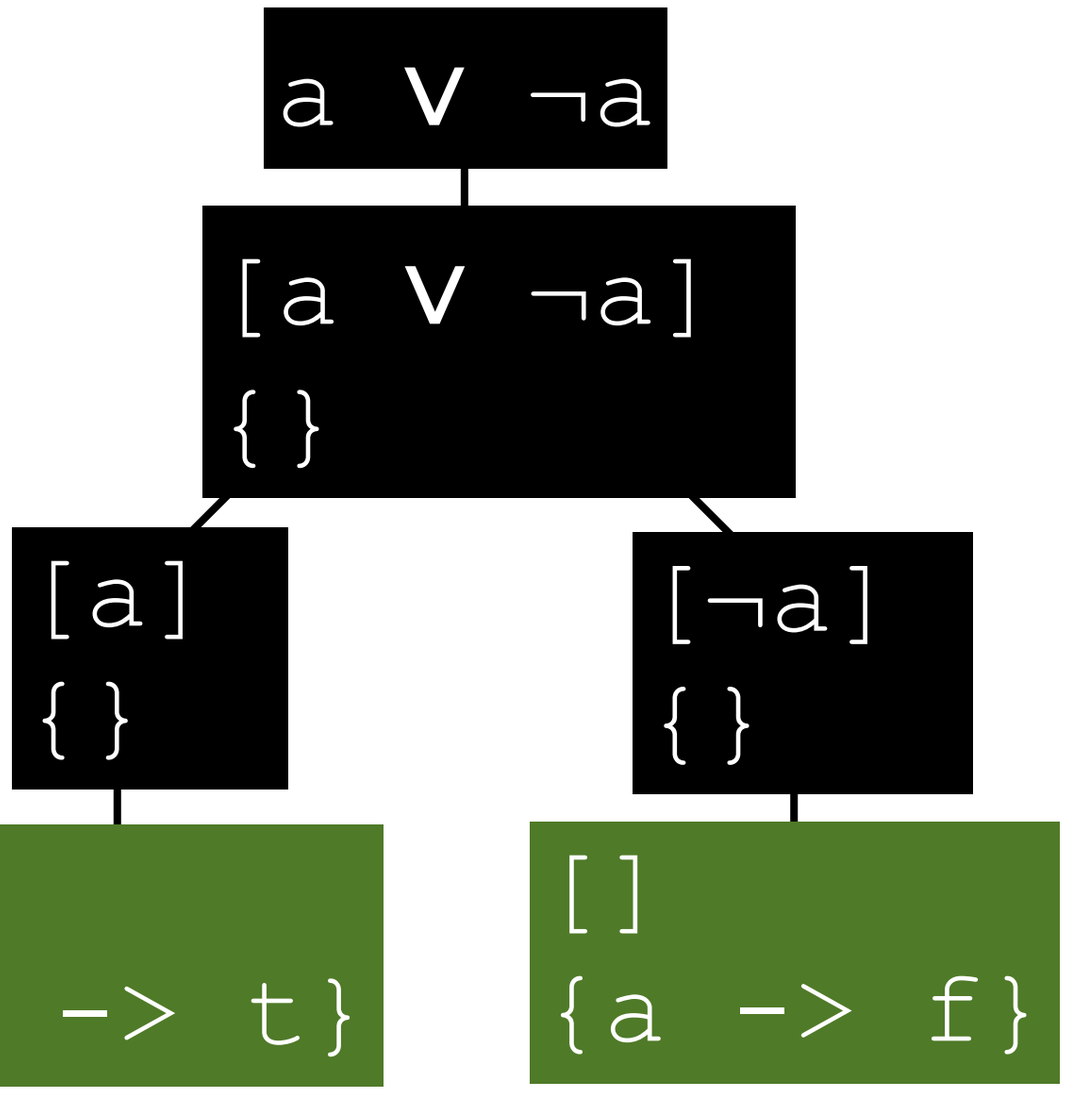
Logical Or



Logical Or



Logical Or

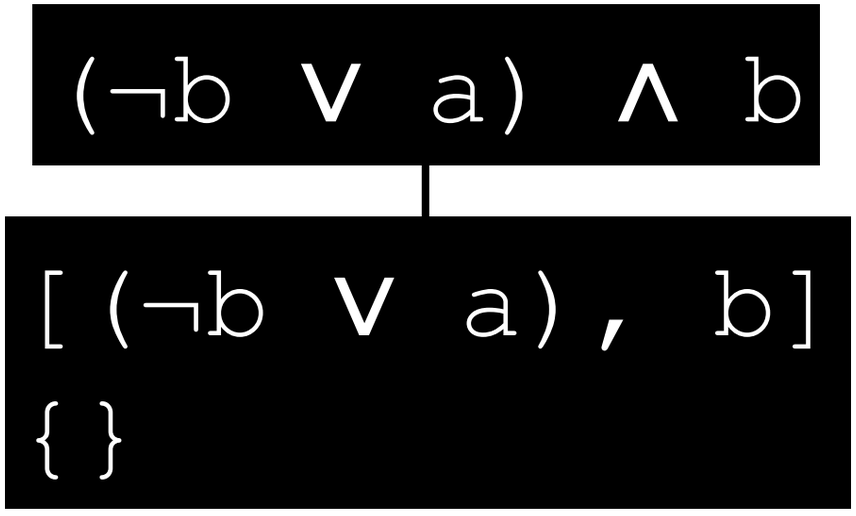


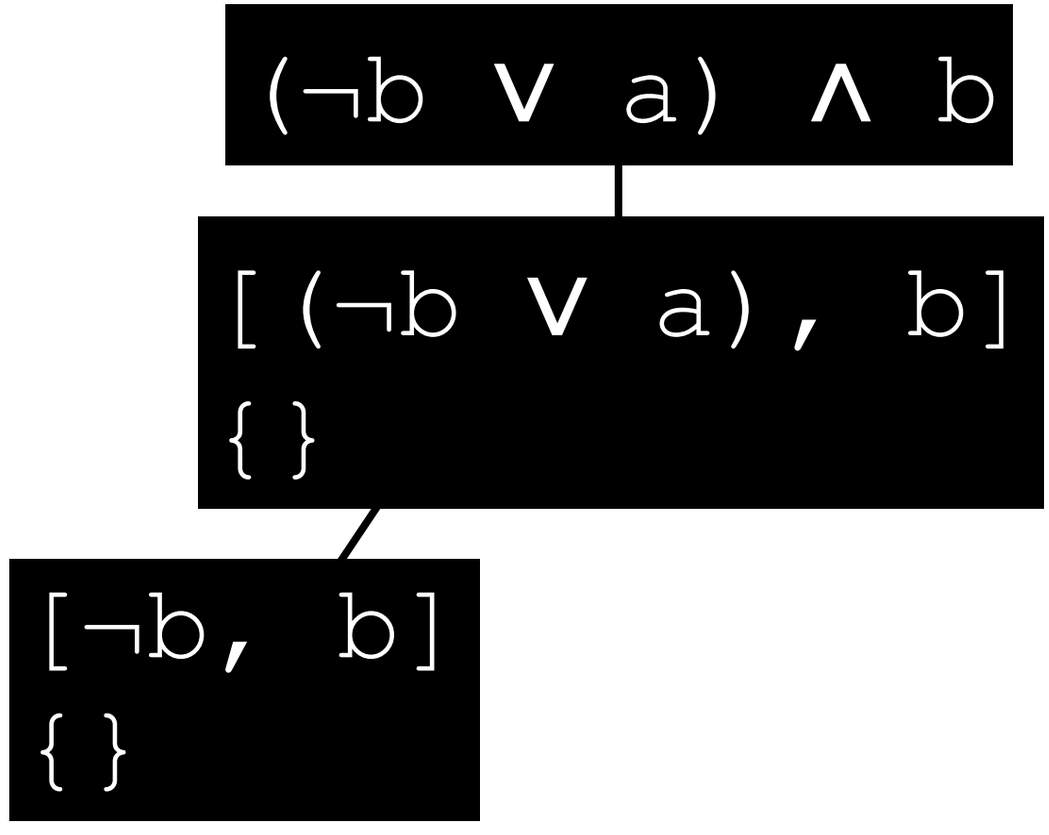
Examples

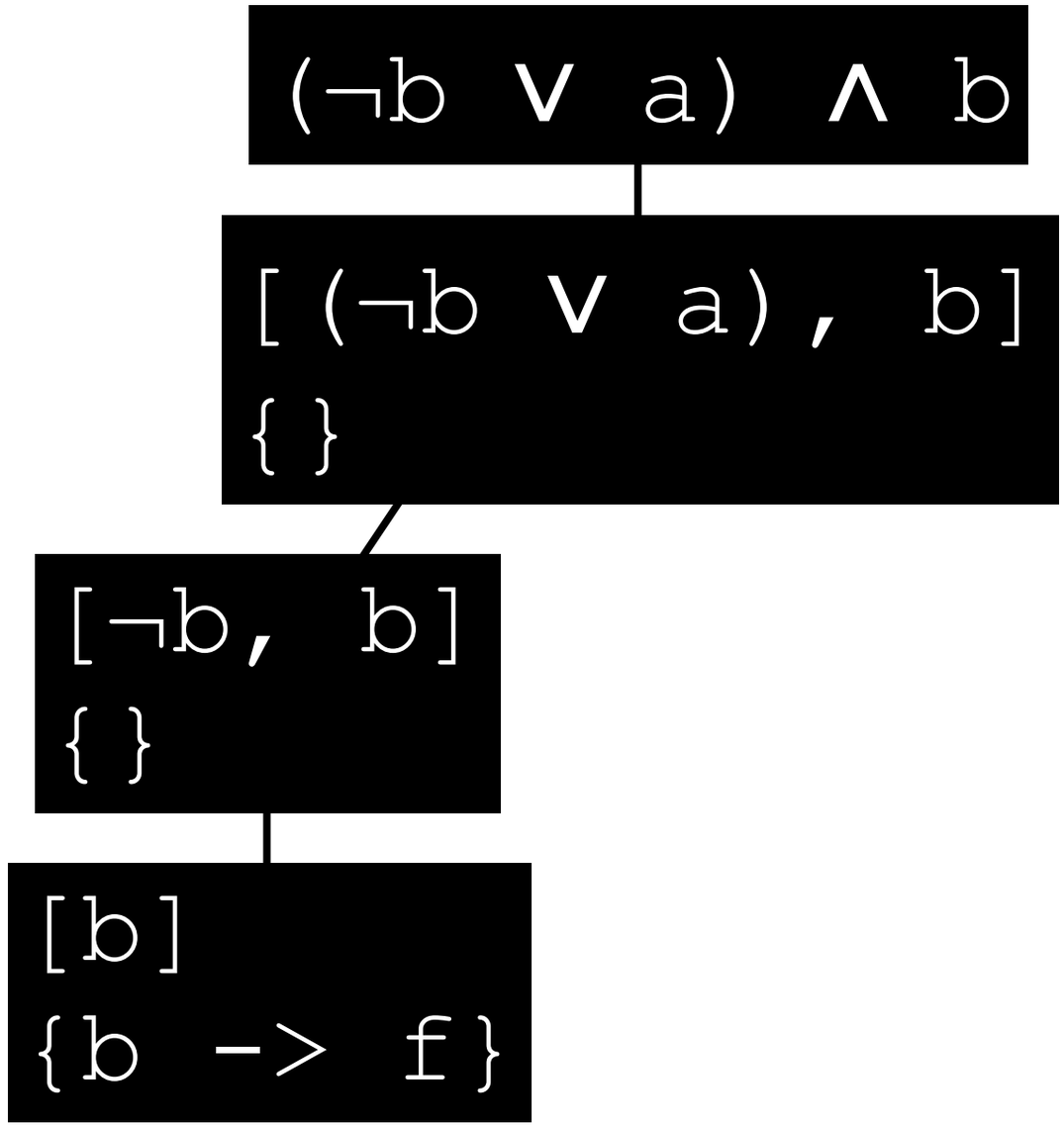
Example 1:

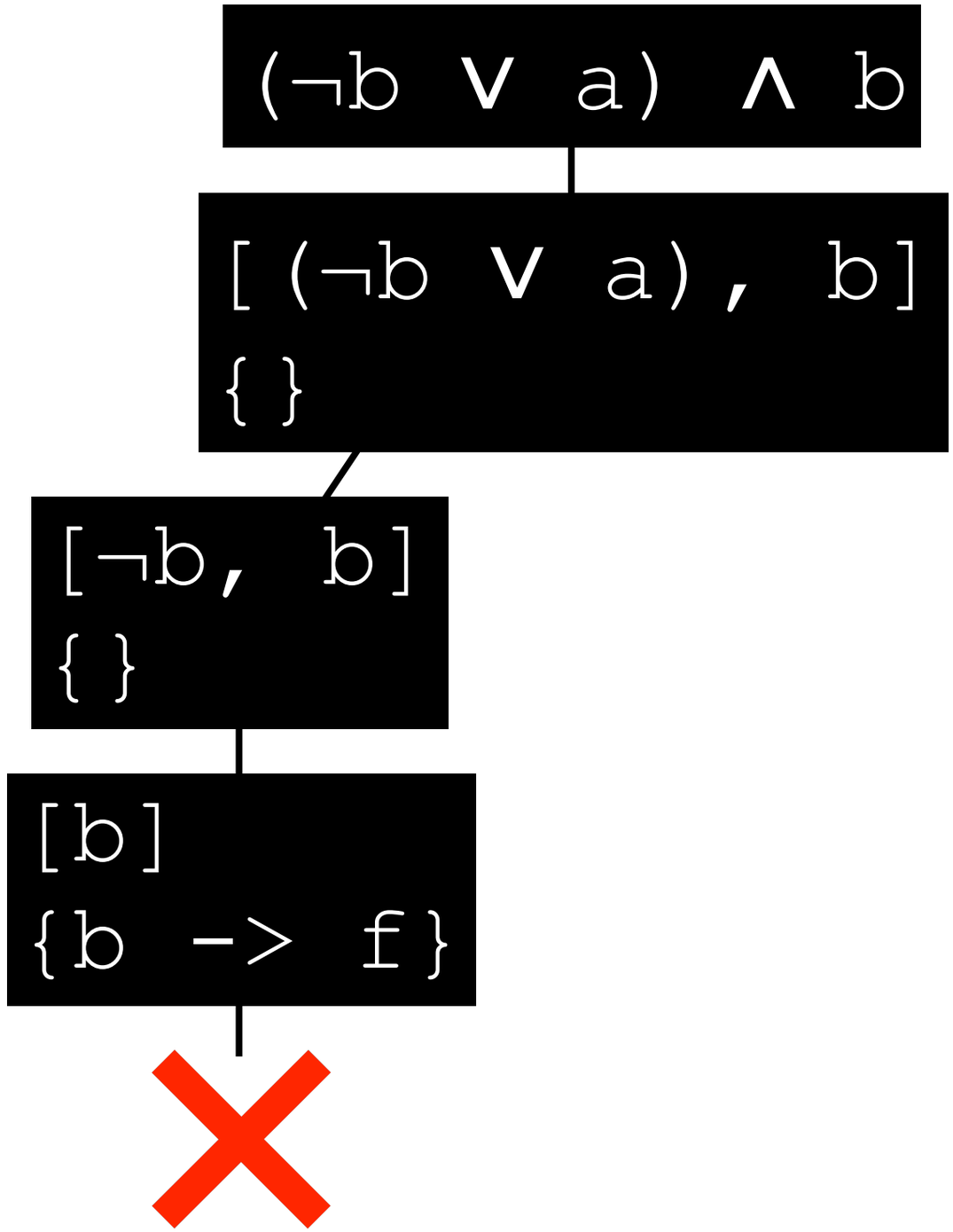
$(\neg b \lor a) \land b$

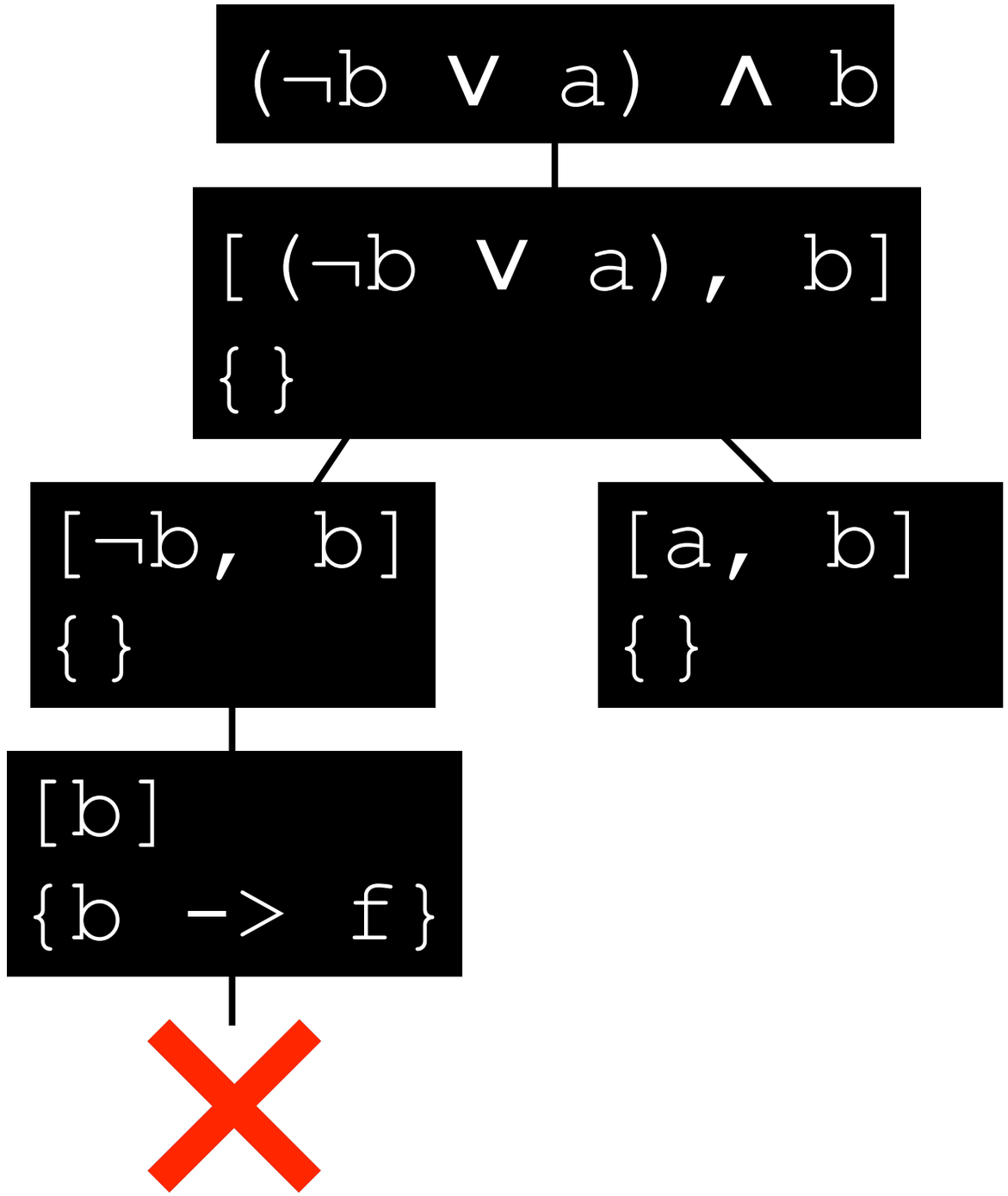
$$(\neg b \lor a) \land b$$

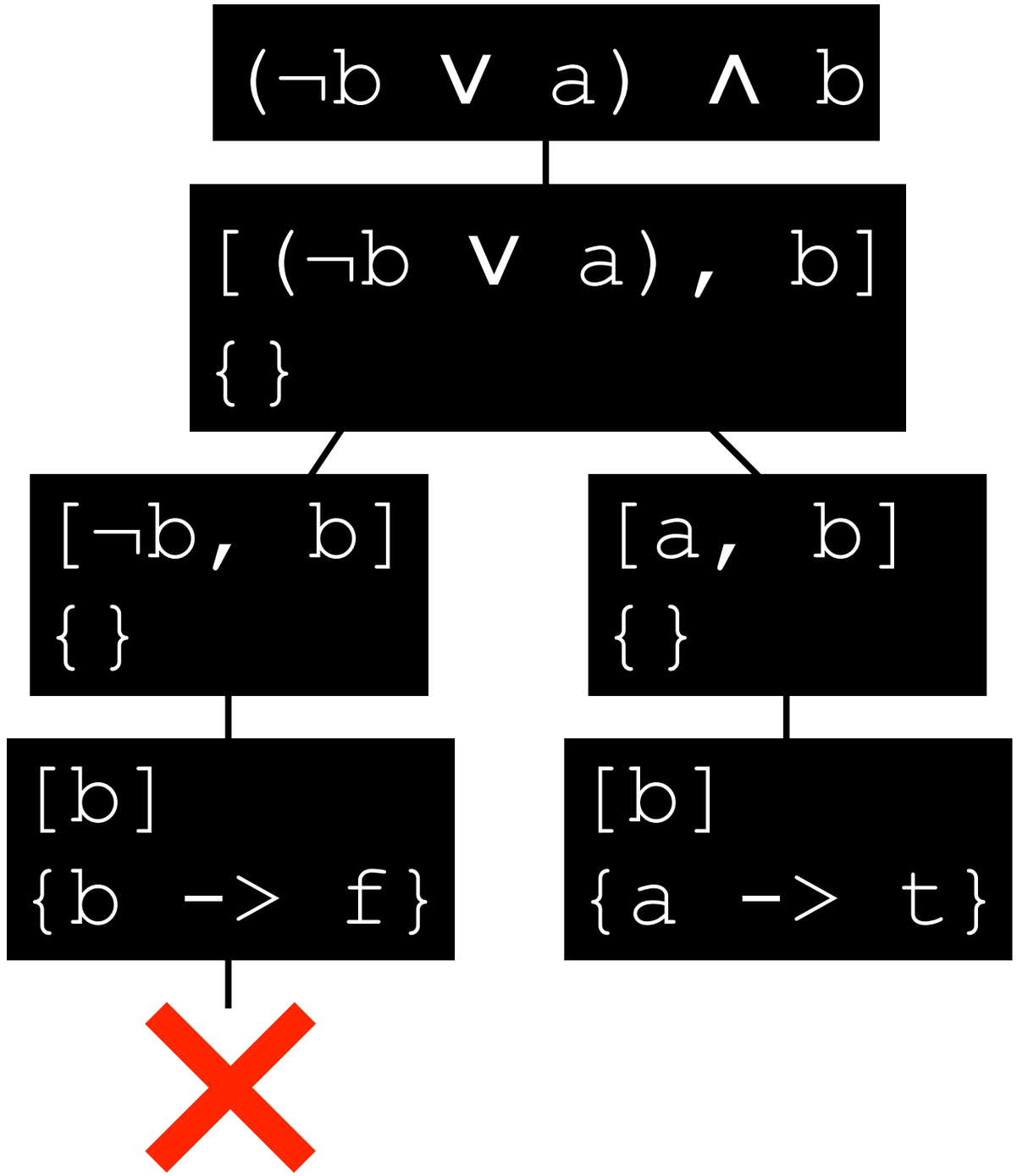

(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}

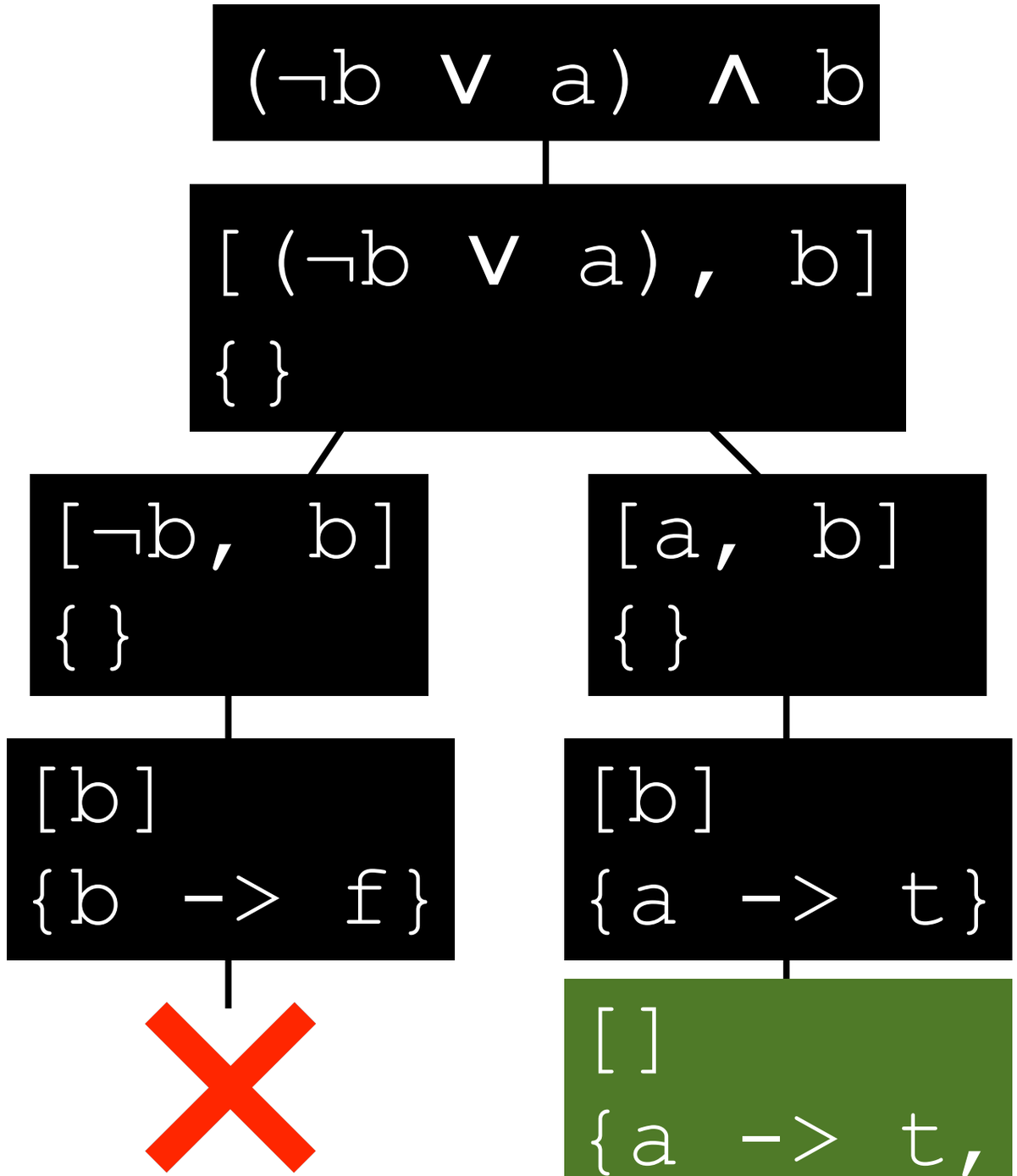

(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}


(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}
[b]
{b -> f}


(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}
[b]
{b -> f}


(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}
[a, b]
{}
[b]
{b -> f}

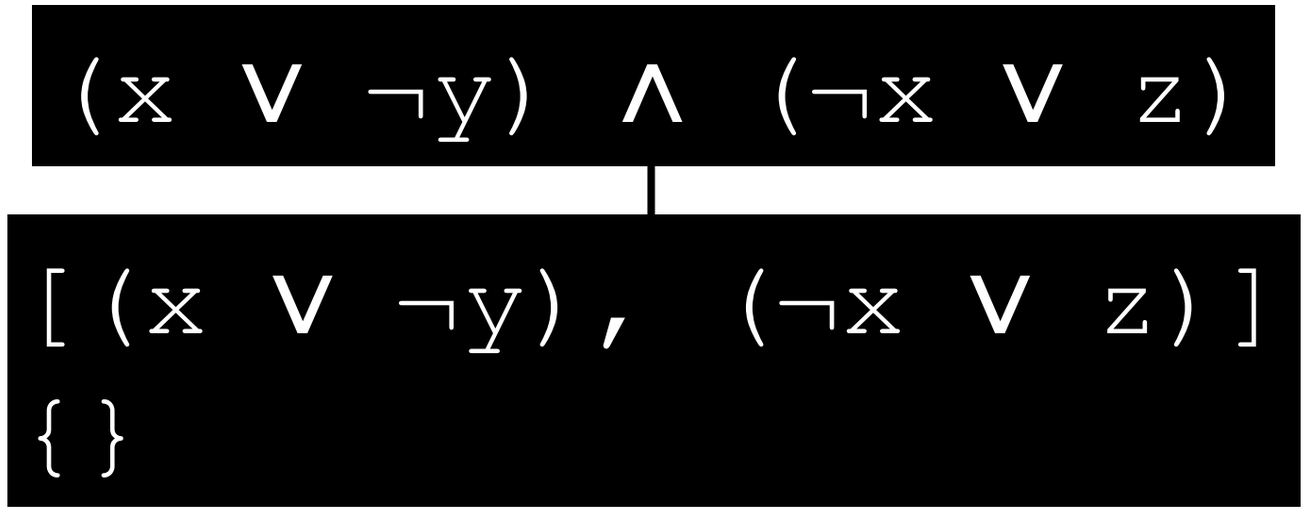

(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}
[a, b]
{}
[b]
{b -> f}
[b]
{a -> t}

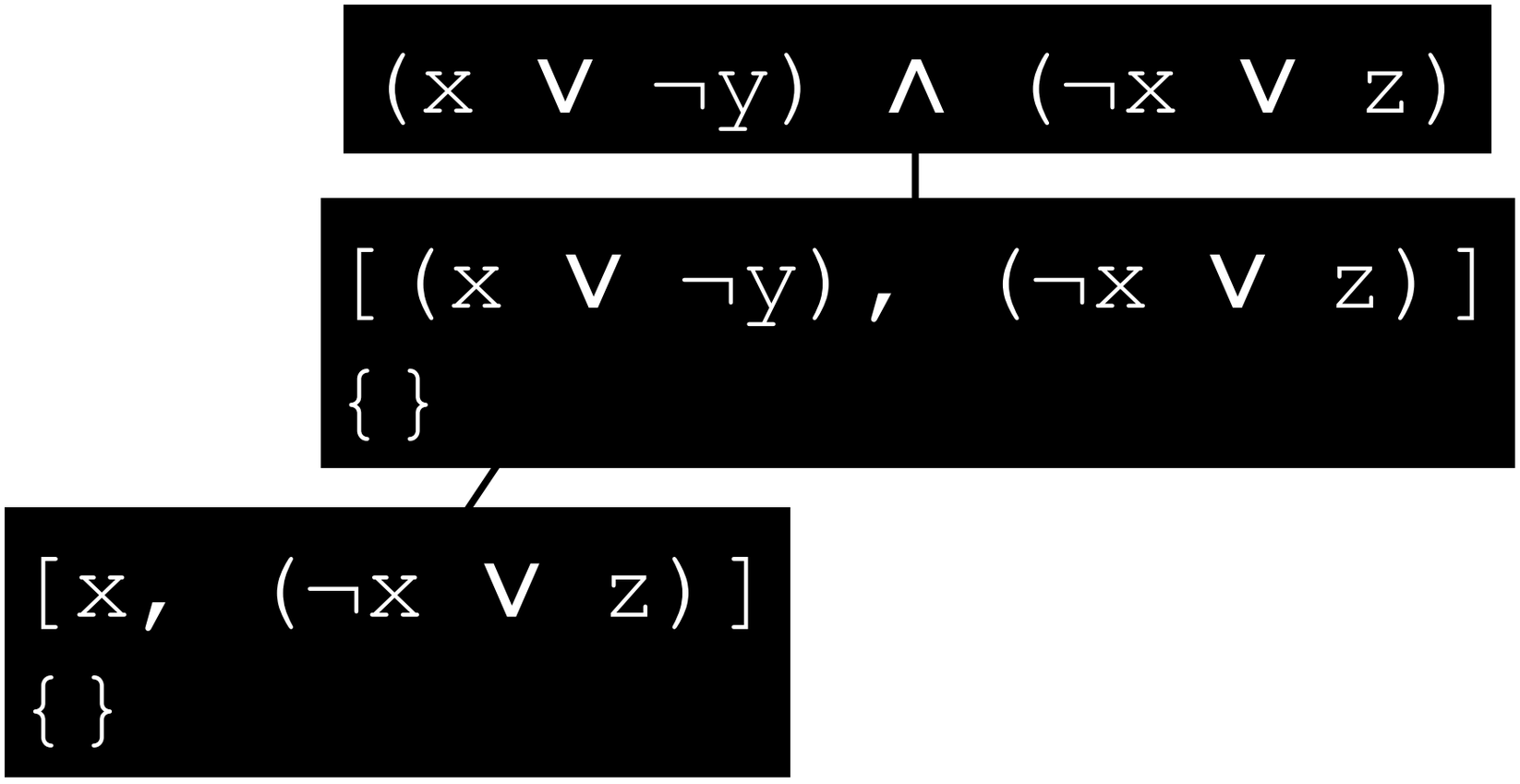

(¬b ∨ a) ∧ b
[(¬b ∨ a), b]
{}
[¬b, b]
{}
[a, b]
{}
[b]
{b -> f}
[b]
{a -> t}
[]
{a -> t,
b -> t}

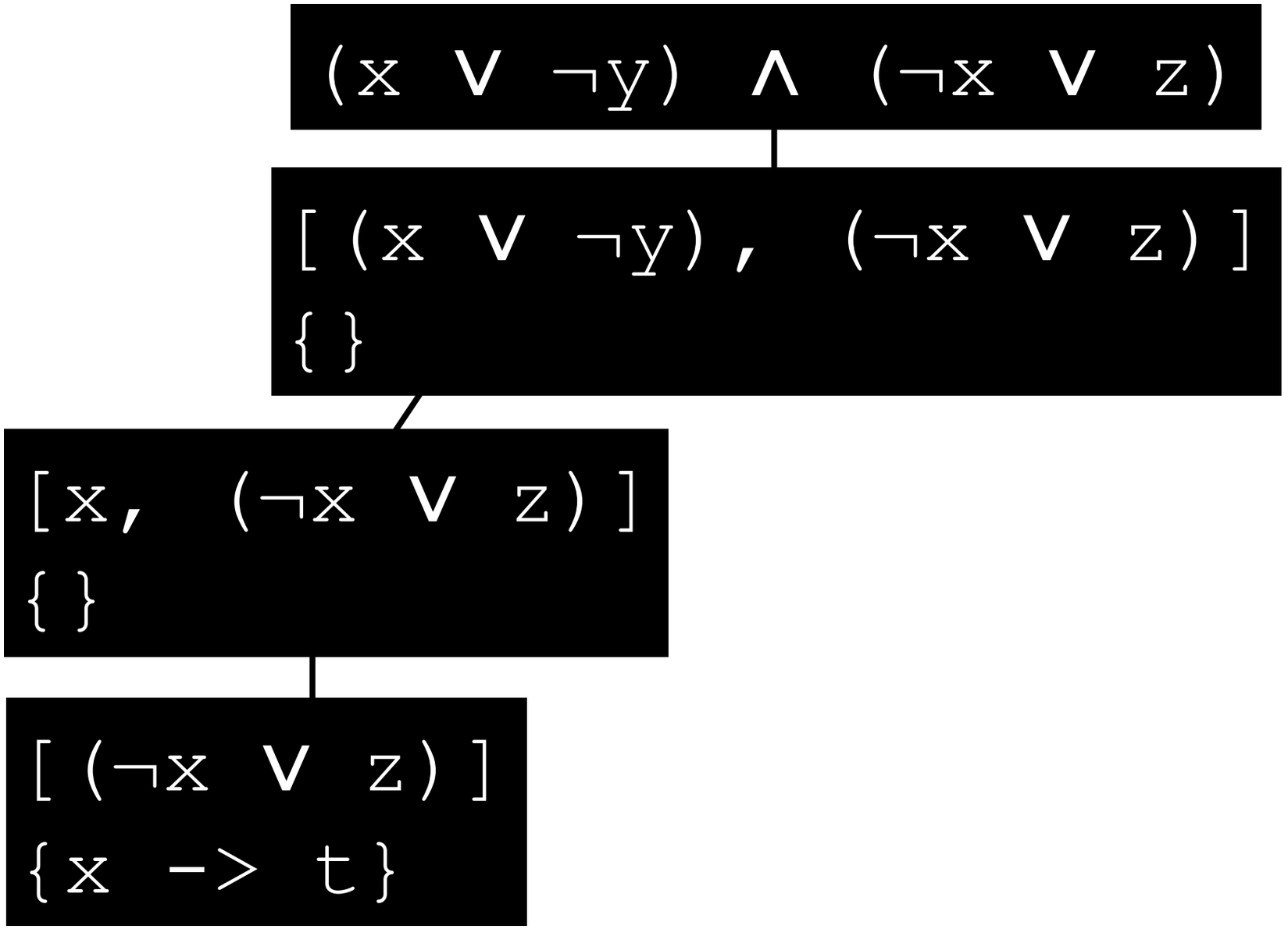
Example 2:

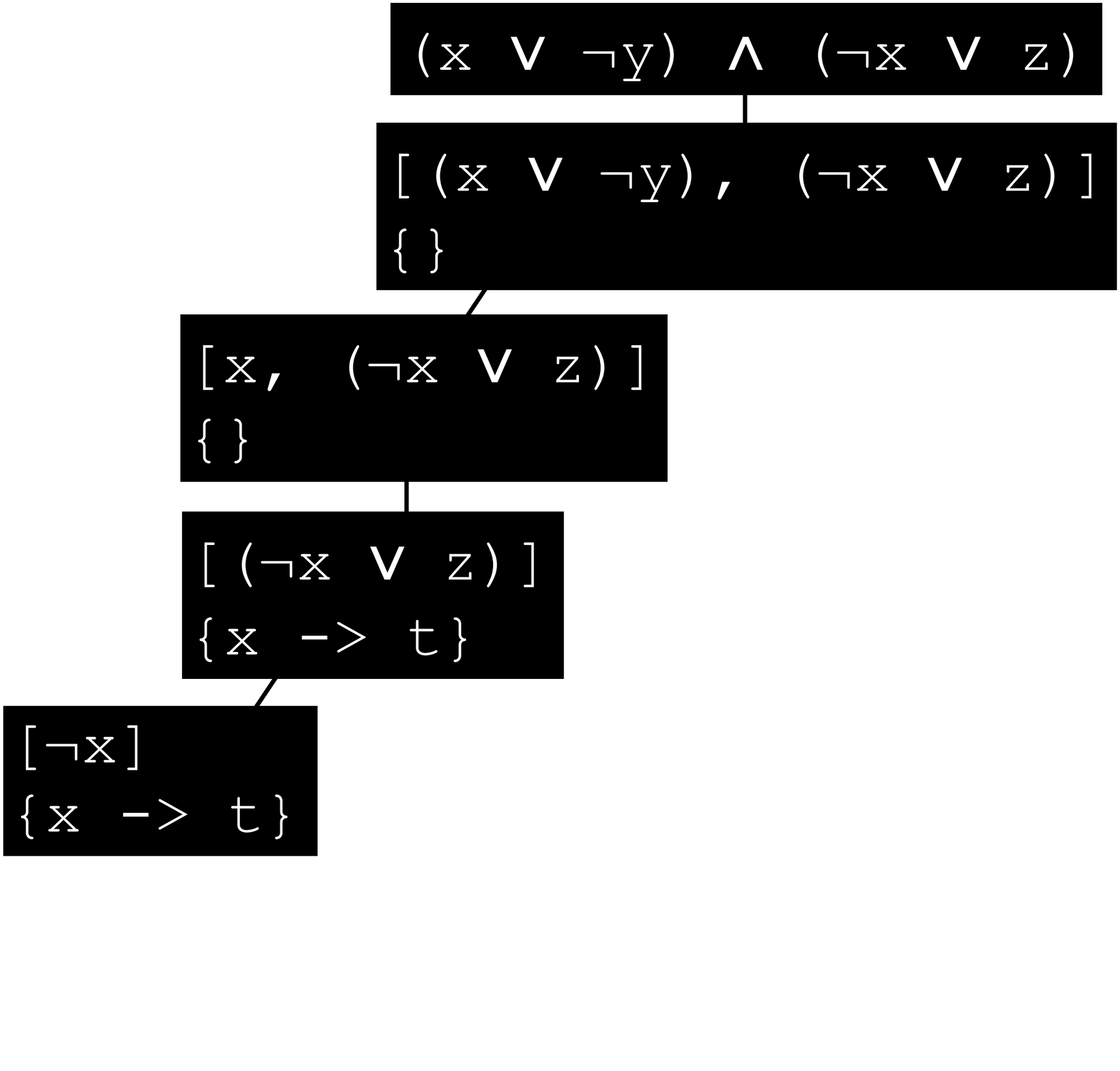
$$(x \lor \neg y) \land (\neg x \lor z)$$

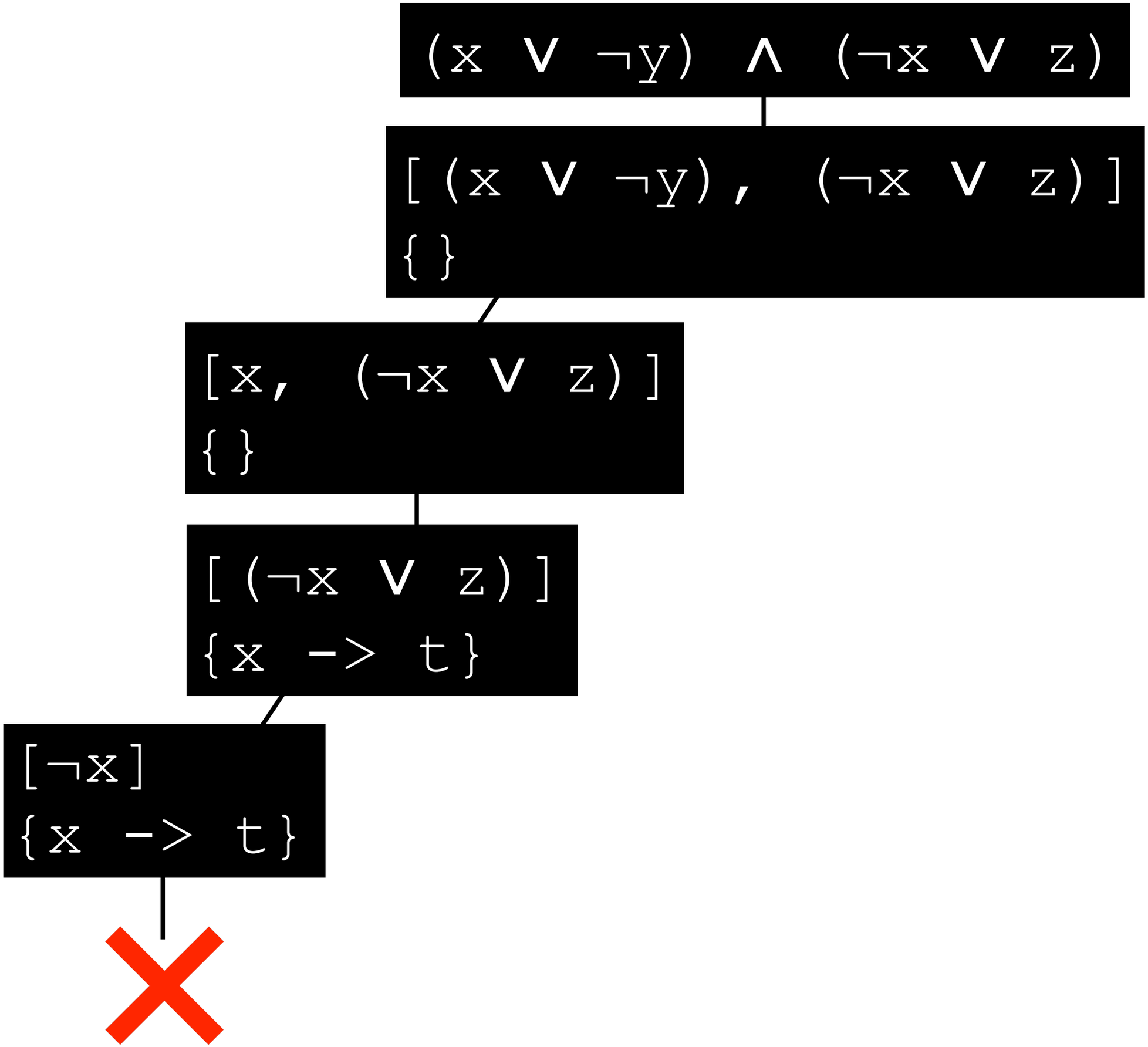
$$(x \lor \neg y) \land (\neg x \lor z)$$

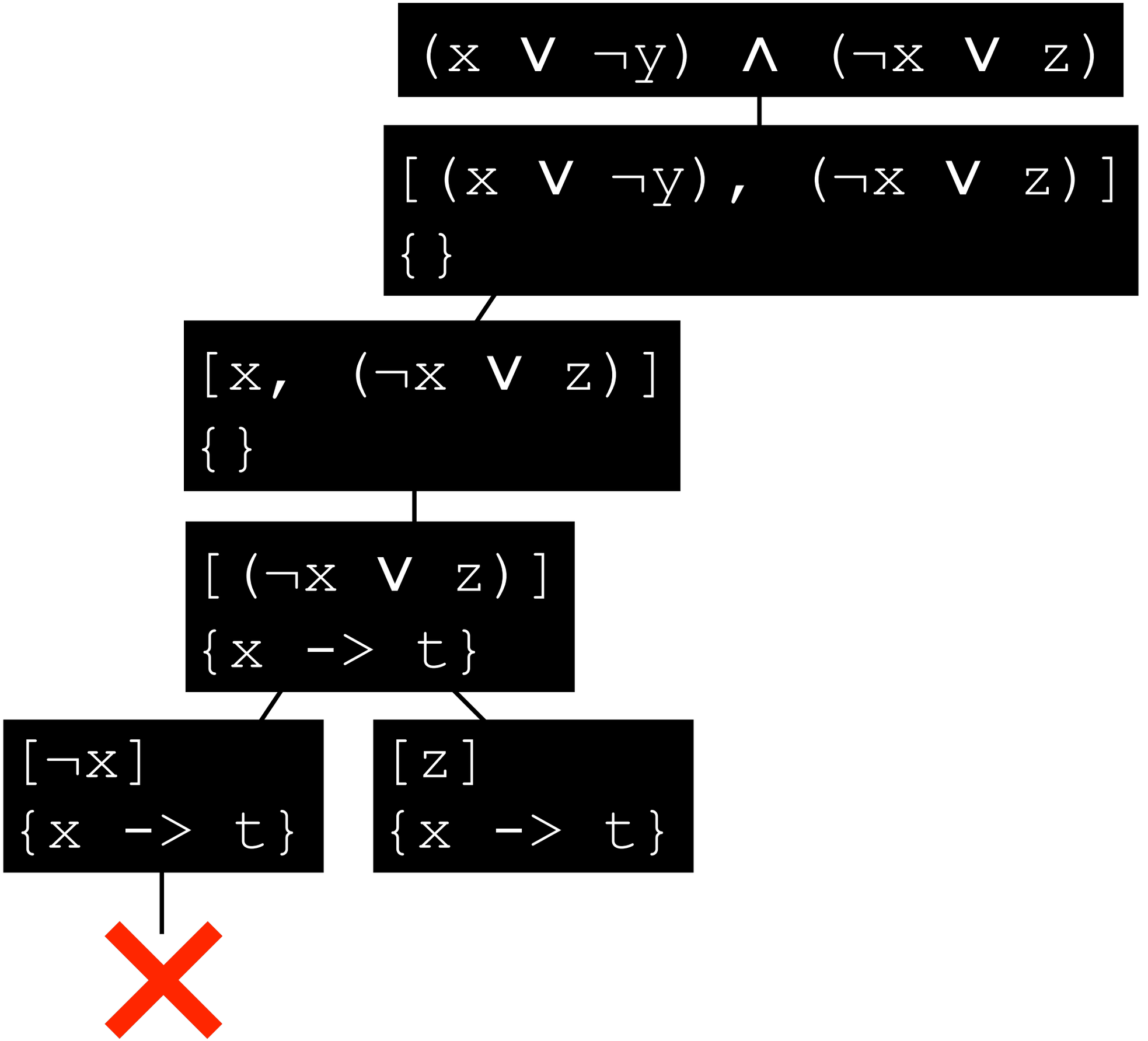

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}

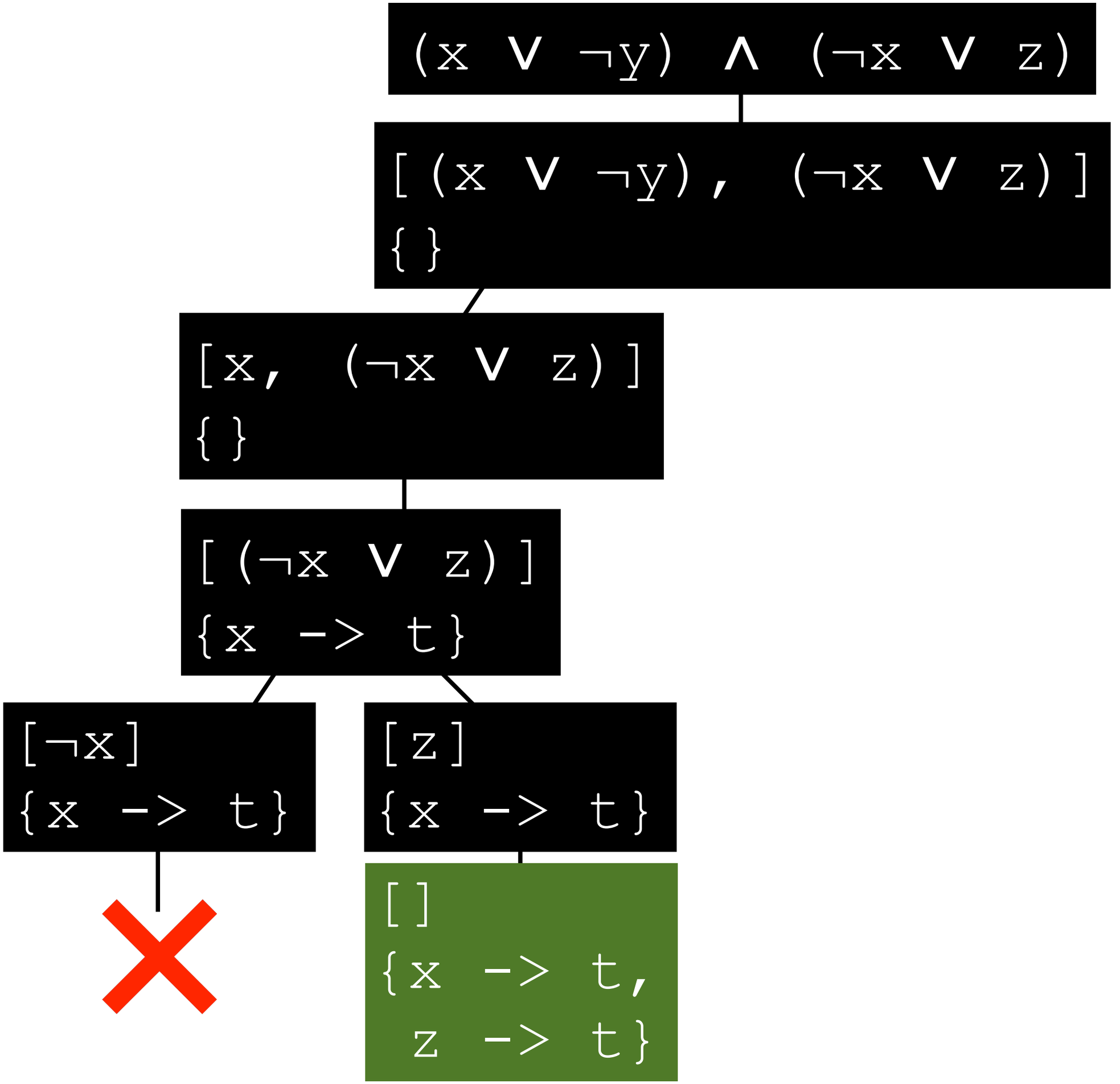

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}

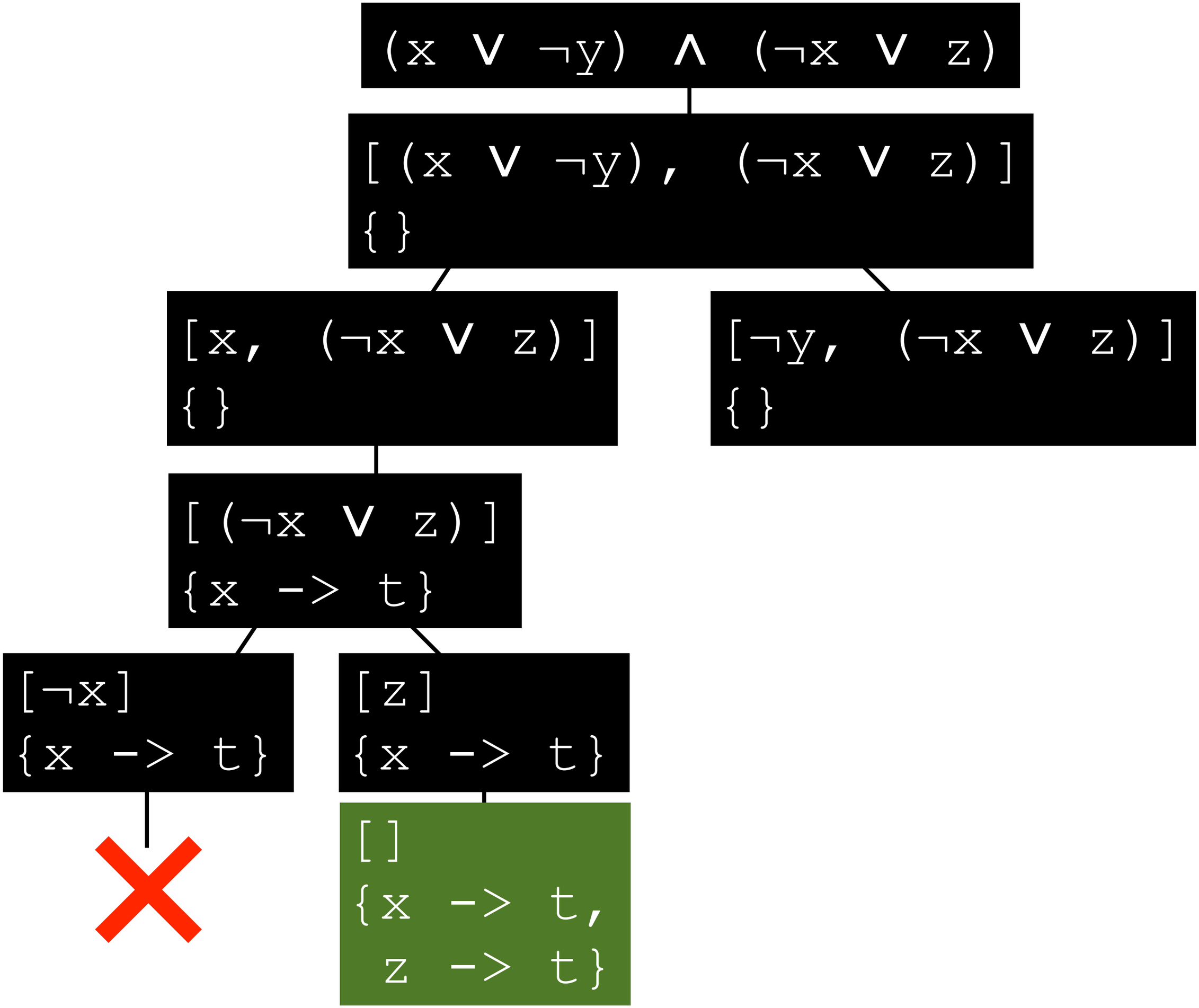

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}

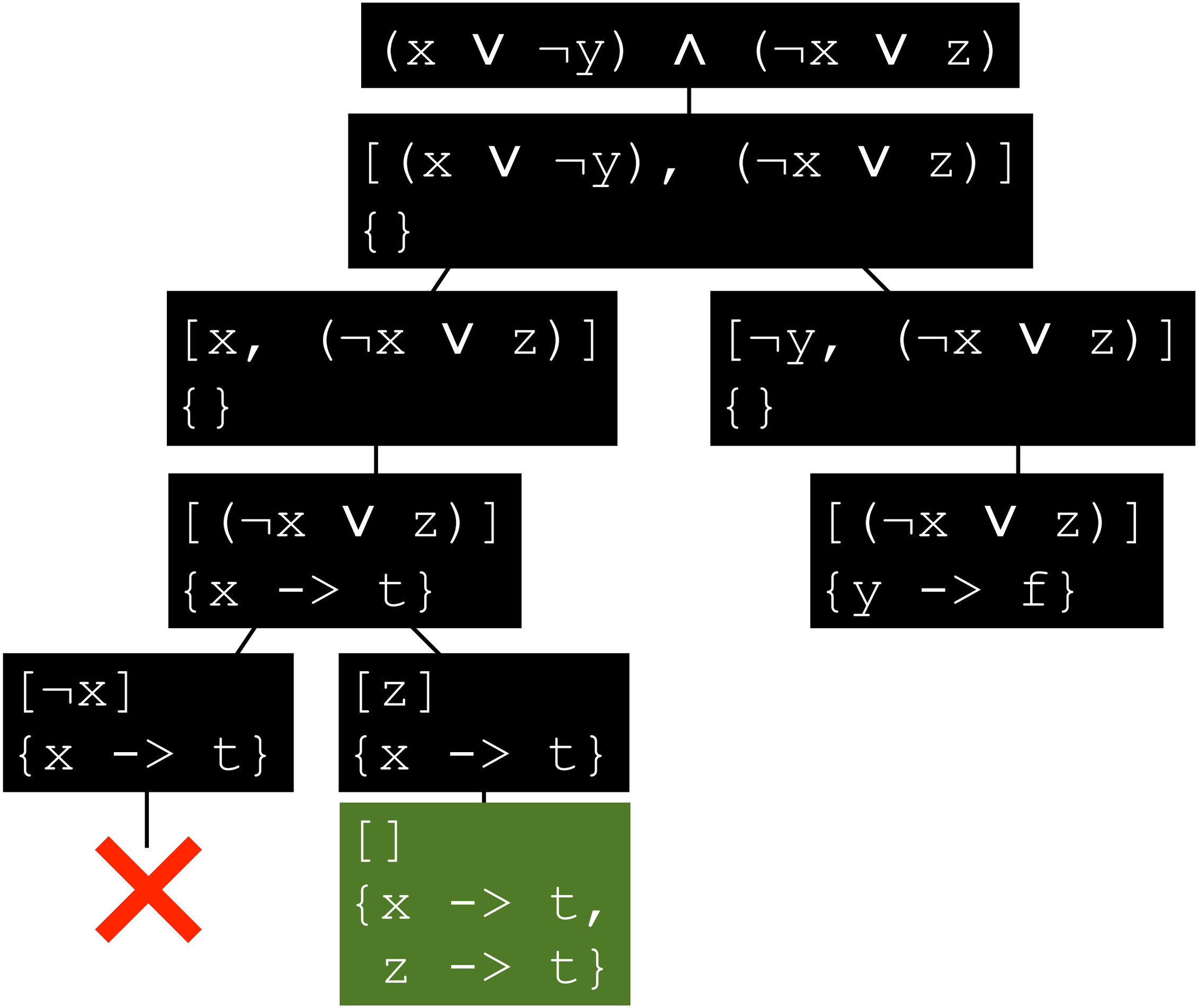

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[¬x]
{x -> t}

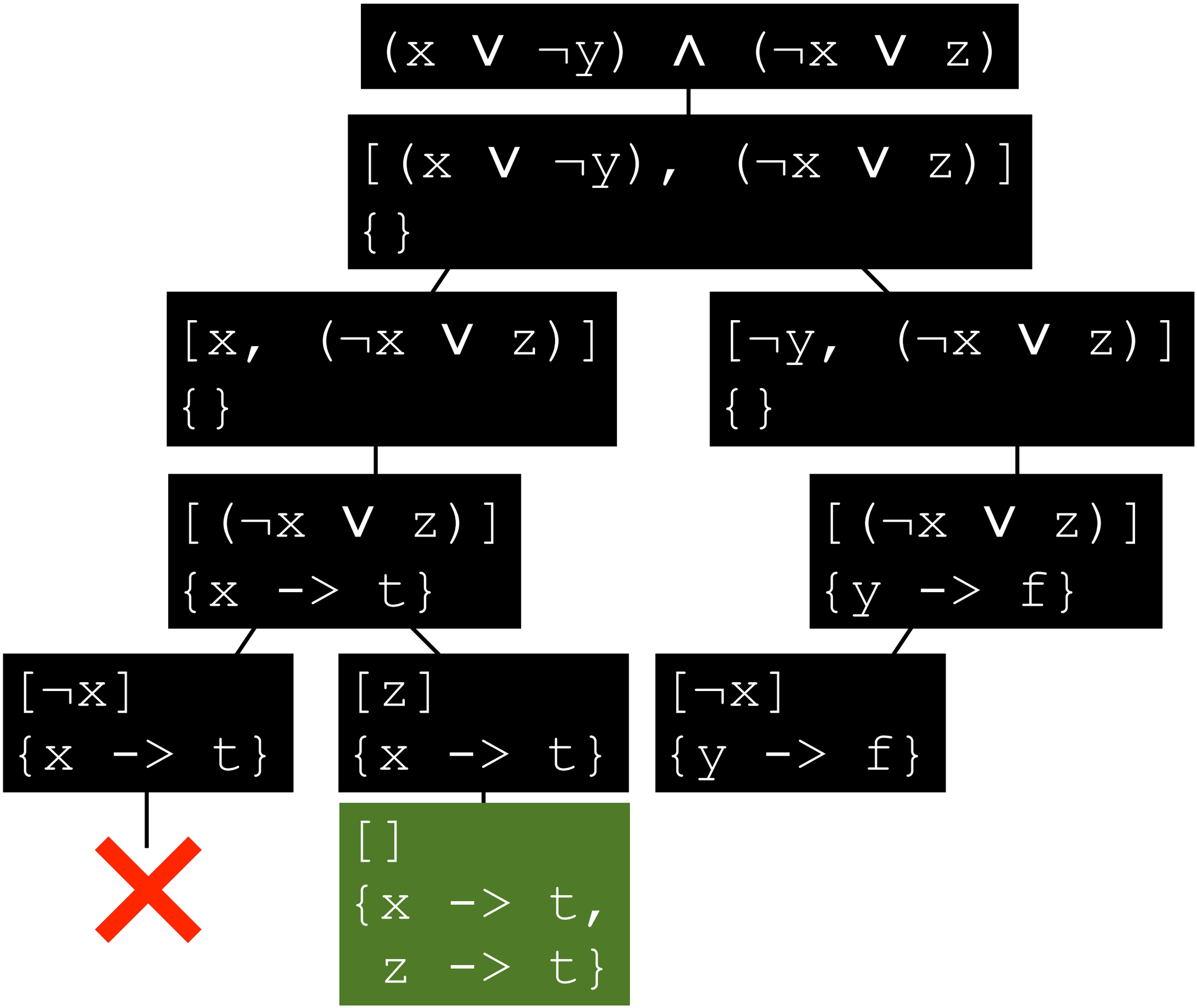

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[¬x]
{x -> t}

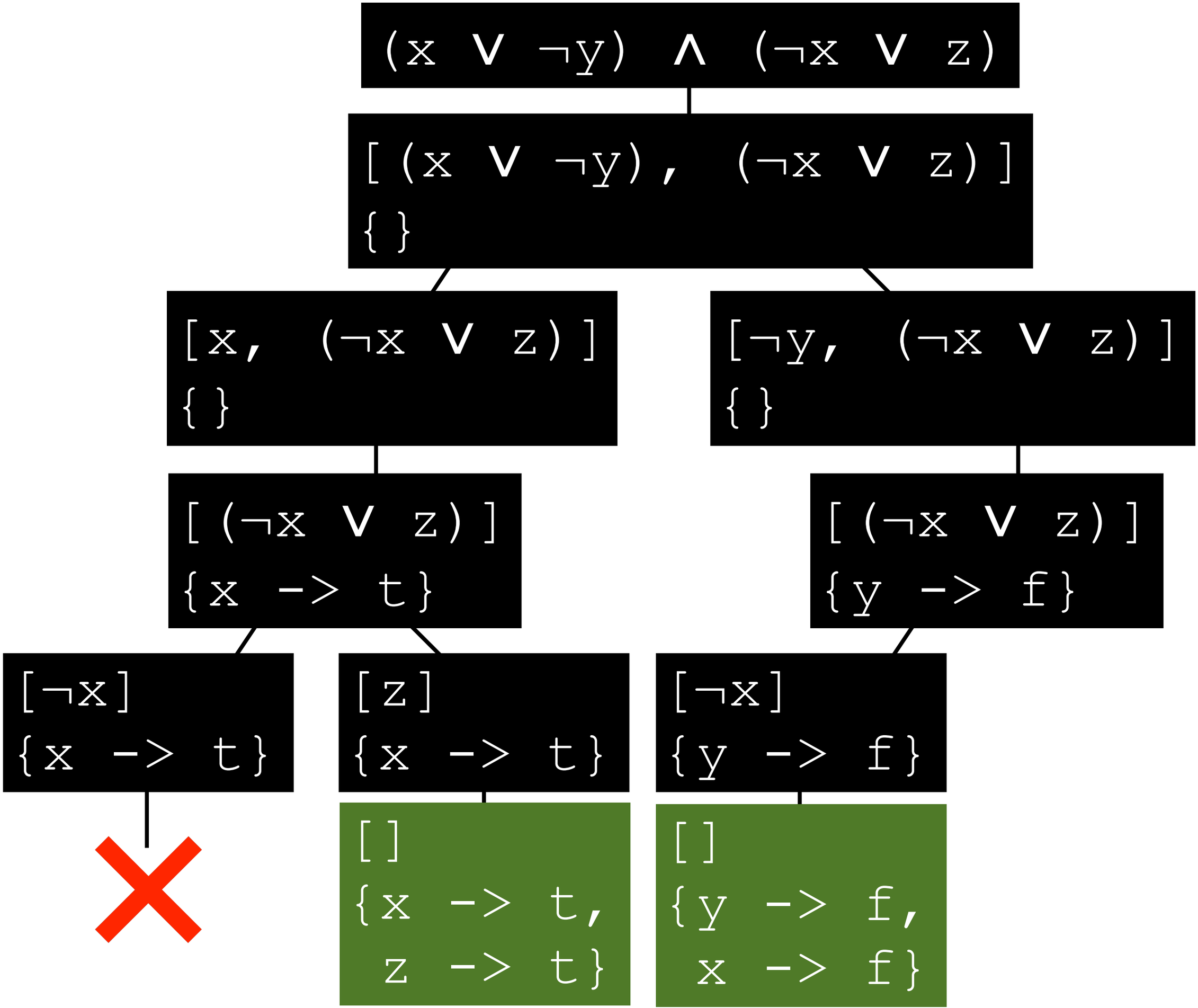

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[¬x]
{x -> t}
[z]
{x -> t}

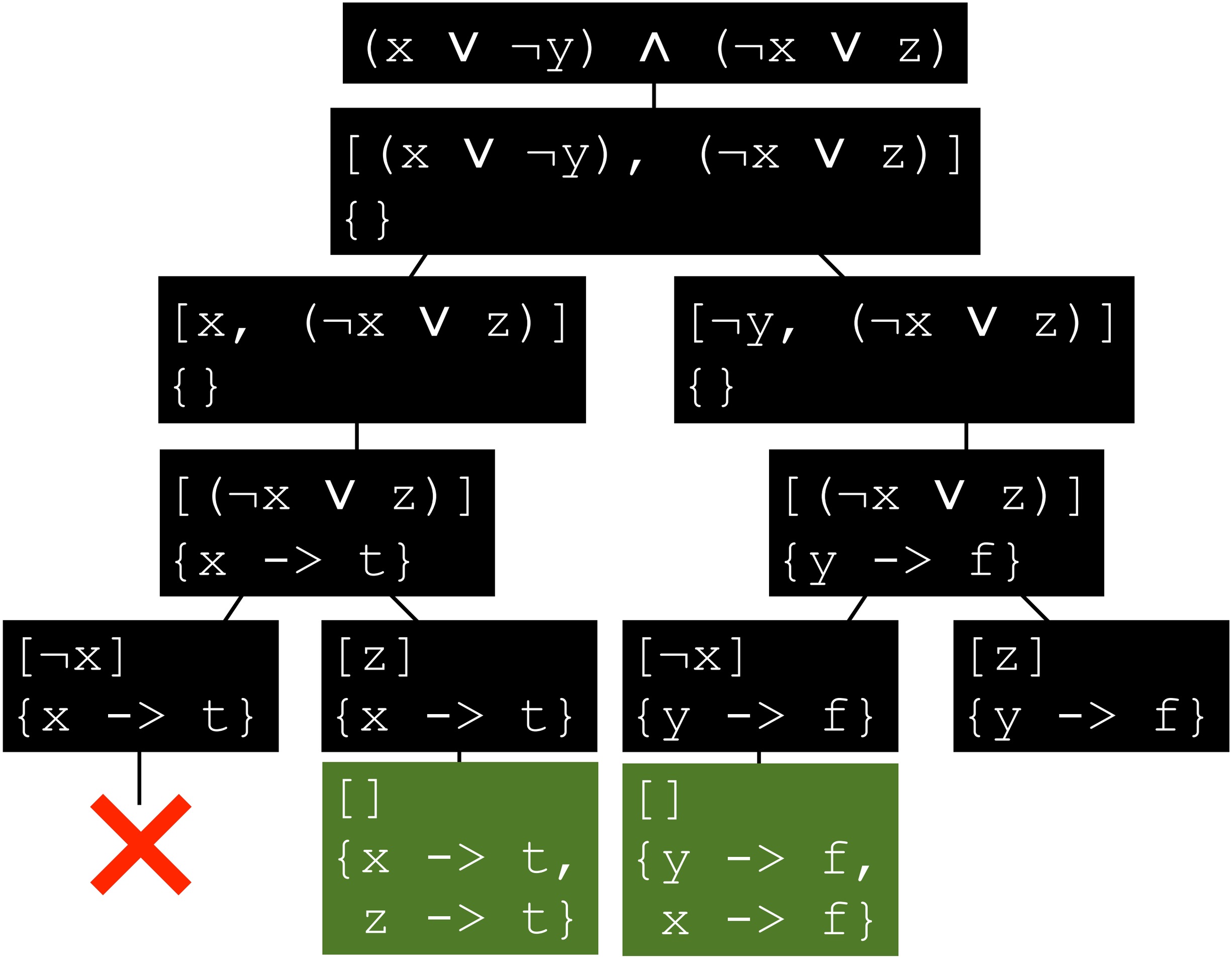

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[¬x]
{x -> t}
[z]
{x -> t}
[]
{x -> t,
z -> t}

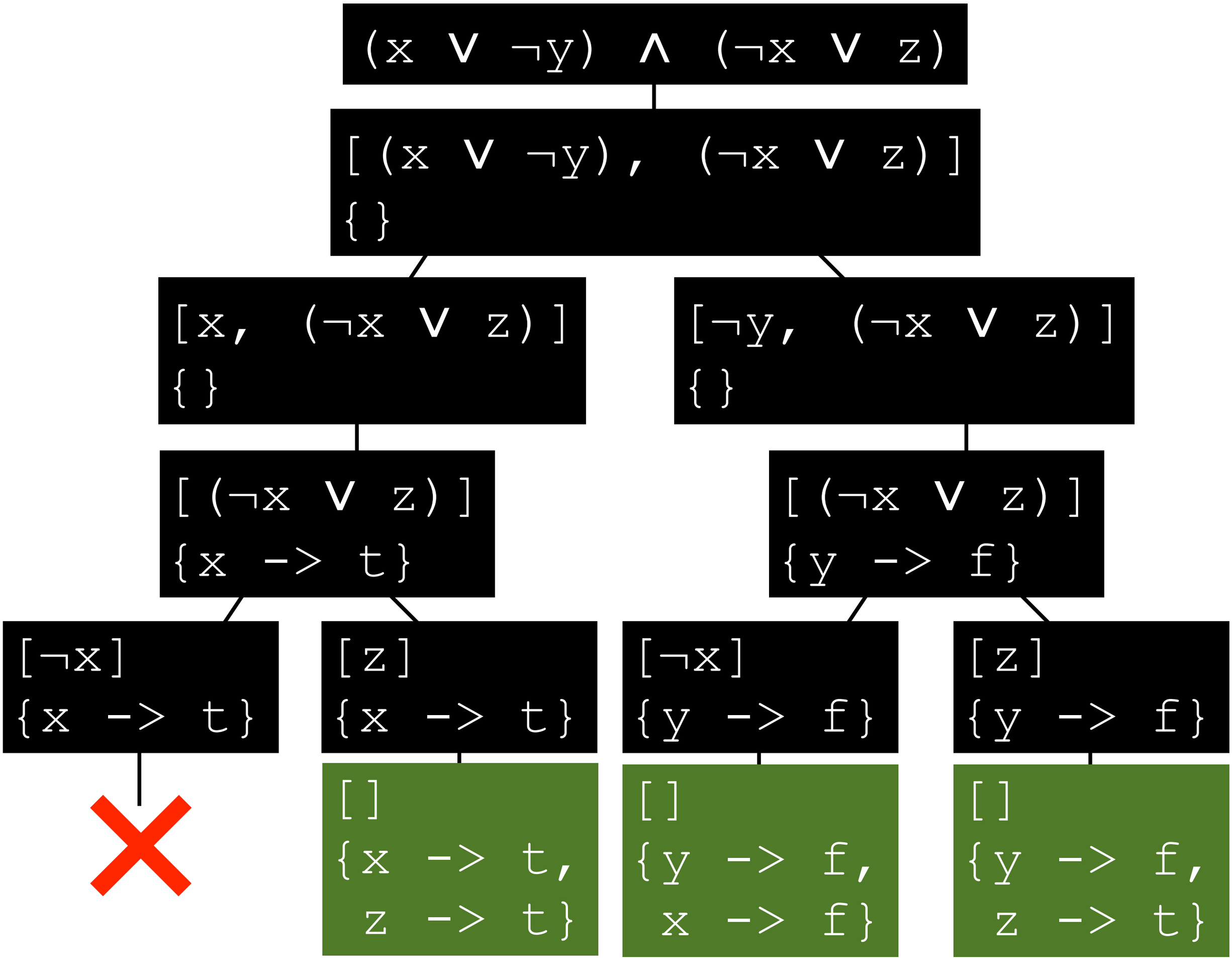

(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[¬x]
{x -> t}
[z]
{x -> t}
[]
{x -> t,
z -> t}


(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[(¬x ∨ z)]
{y -> f}
[¬x]
{x -> t}
[z]
{x -> t}
[]
{x -> t,
z -> t}


(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[(¬x ∨ z)]
{y -> f}
[¬x]
{x -> t}
[z]
{x -> t}
[¬x]
{y -> f}
[]
{x -> t,
z -> t}


(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[(¬x ∨ z)]
{y -> f}
[¬x]
{x -> t}
[z]
{x -> t}
[¬x]
{y -> f}
[]
{x -> t,
z -> t}
[]
{y -> f,
x -> f}


(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[(¬x ∨ z)]
{y -> f}
[¬x]
{x -> t}
[z]
{x -> t}
[¬x]
{y -> f}
[z]
{y -> f}
[]
{x -> t,
z -> t}
[]
{y -> f,
x -> f}


(x ∨ ¬y) ∧ (¬x ∨ z)
[(x ∨ ¬y), (¬x ∨ z)]
{}
[x, (¬x ∨ z)]
{}
[¬y, (¬x ∨ z)]
{}
[(¬x ∨ z)]
{x -> t}
[(¬x ∨ z)]
{y -> f}
[¬x]
{x -> t}
[z]
{x -> t}
[¬x]
{y -> f}
[z]
{y -> f}
[]
{x -> t,
z -> t}
[]
{y -> f,
x -> f}
[]
{y -> f,
z -> t}

Exercise: Second Side of SAT Sheet