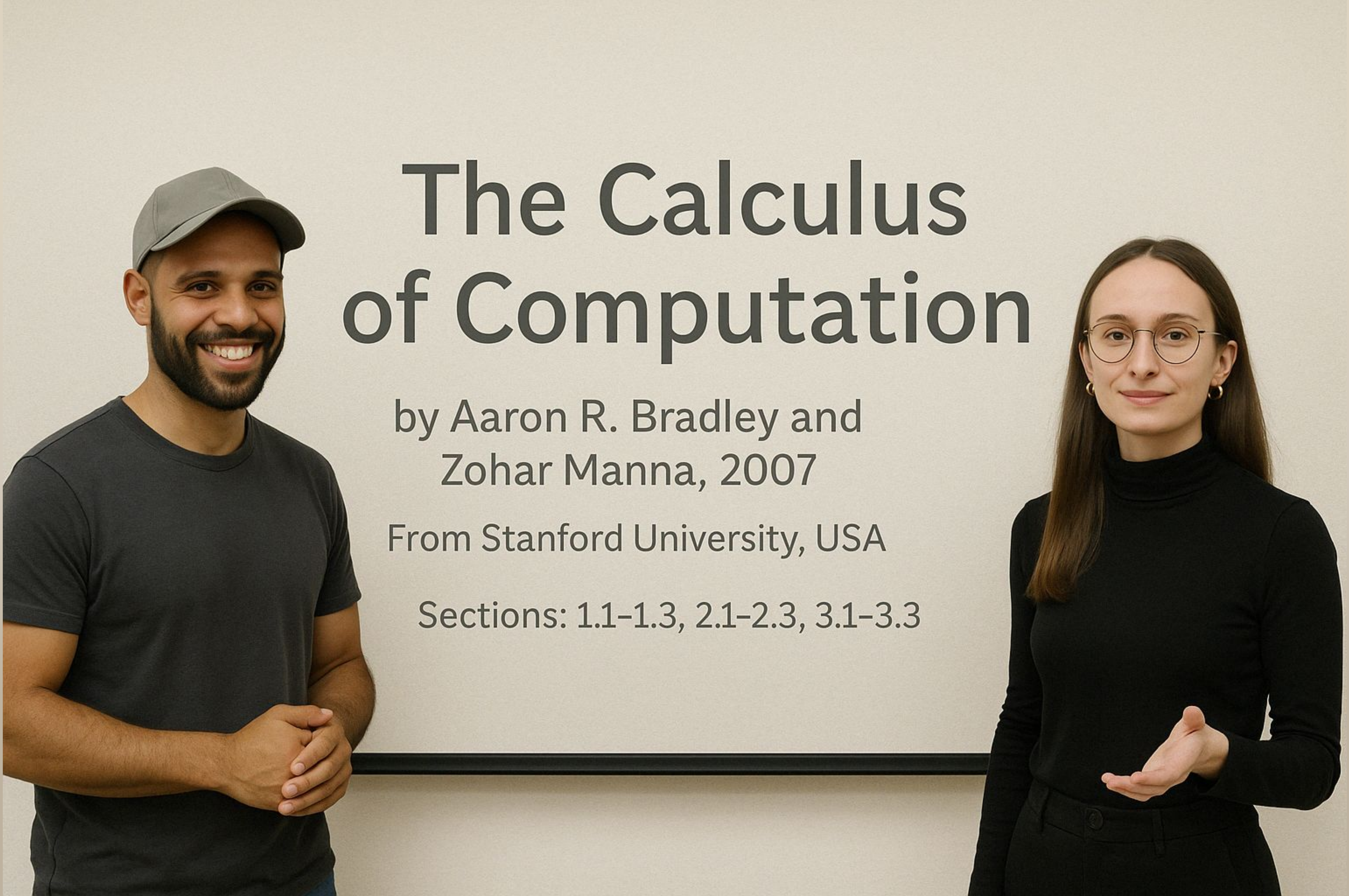
THE CALCULUS OF COMPUTATION

Sharon Tartakovsky

Nadav Mor


The Calculus
of Computation
by Aaron R. Bradley and
Zohar Manna, 2007
From Stanford University, USA
Sections: 1.1–1.3, 2.1–2.3, 3.1–3.3

MOTIVATION

```python
def foo(a, b):
    if not a and not b:
        h()
    else:
        if not a:
            g()
        else:
            f()
```

```python
def goo(a, b):
    if a:
        f()
    else:
        if b:
            g()
        else:
            h()
```

$$\varphi_{foo}: (\neg a \land \neg b \land h) \lor (\neg(\neg a \land \neg b) \land ((\neg a \land g) \lor (\neg(\neg a) \land f)$$

$$\varphi_{goo}: (a \land f) \lor (\neg a \land ((b \land g) \lor (\neg b \land h)))$$

$$\varphi_{goo} \leftrightarrow \varphi_{foo}$$

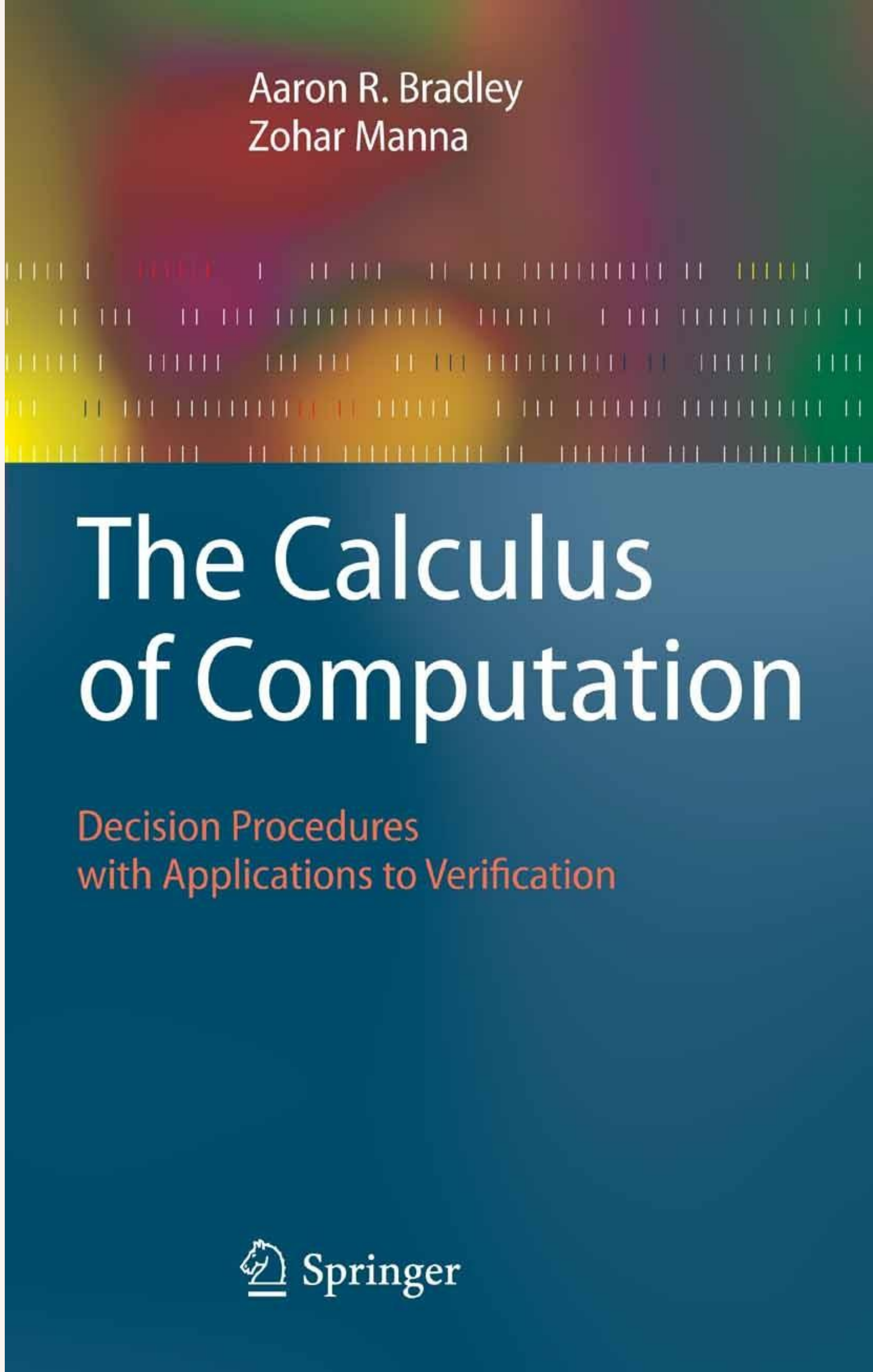
THE CALCULUS OF COMPUTATION

Part I

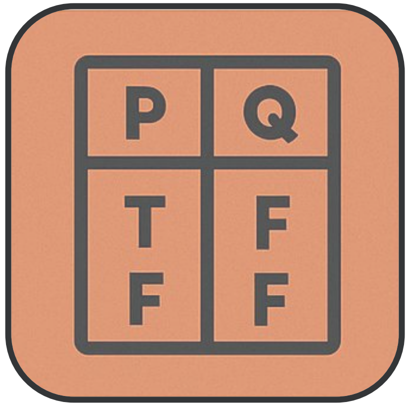
provides the mathematical foundations for precise engineering.

Part II

ivestigate algorithmic aspects of applying these foundations.



LECTURE OUTLINE



Propositional Logic (PL)

The logic of statements that are either true or false.

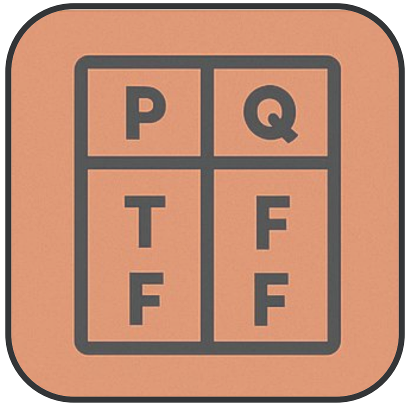
First Order Logic (FOL)

Logic with predicates, functions, and quantifiers.

First Order Theories

Logical frameworks for reasoning about numbers, arrays, and data.

LECTURE OUTLINE



Propositional Logic (PL)

The logic of statements that are either true or false.

First Order Logic (FOL)

Logic with predicates, functions, and quantifiers.

First Order Theories

Logical frameworks for reasoning about numbers, arrays, and data.

PROPOSITIONAL LOGIC (PL)

INTRODUCTION

WHAT IS A CALCULUS?

Calculus = A Formal System

- A set of symbols and rules for manipulating those symbols.
- A meaningful calculus applies to a real-world domain.

C.S NEEDS A DIFFERENT CALCULUS

- Domain: **Computation.**
- Based on **state** — assignment of values to variables.
- **Computation** = Sequence of transitions between states.
- A **program** defines:
    - State structure
    - Allowed transitions
    - All possible computations

PURPOSE OF A LOGICAL CALCULUS IN CS

We want to **check how programs behave**:

- Does the program **sort** the array?
- Does it **use memorysafely**?
- Does it **always halt**?

Key Concepts

- **State**: Assignment of values (Booleans, integers, etc.) to variables.
- **Transitions**: Pairs of states in computation.
- **Computation**: Sequence of states.
- A program's **set of computations** defines it as precisely as its source code.

<u>**Goal**</u>: Develop a **logical calculus** that lets us reason **formally** about correctness,

just like differential calculus lets us reason about physical quantities.

PROPOSITIONAL LOGIC (PL)

SYNTAX

How formulas are built

SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

PROPOSITIONAL LOGIC (PL)

SYNTAX

How formulas are built

SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SYNTAX

The **syntax** of a logical language = its **symbols** + **rules** for combining them.

SYMBOLS (⊤, ⊥, P, Q, ETC.)

- Truth symbols: ⊤ (true), ⊥ (false)
- Propositional variables:
  P, Q, R, P_1, P_2, …

LOGICAL CONNECTIVES

Connective	Meaning
$\neg F$	negation (“*not*)
$F_1 \wedge F_2$	conjunction (“*and*)
$F_1 \vee F_2$	disjunction (“*or*)
$F_1 \rightarrow F_2$	implication (“impplies”)
$F_1 \leftrightarrow F_2$	biconditional (if and

SYNTAX

- **Atom**: truth symbol ($\top$, $\bot$) or propositional variable (P).
- **Literal**: An atom or its negation (e.g., P or ¬P).
- **Formula**: literal or An application of a logical connective to other formulas

$\neg F$	"not" (negation)
$F1 \wedge F2$	"and" (conjunction)
$F1 \vee F2$	"or" (disjunction)
$F1 \rightarrow F2$	"implies" (implication)
$F1 \leftrightarrow F2$	"if and only if" (iff)

$$(P \wedge Q) \rightarrow R$$

$$\neg(P \wedge Q) \leftrightarrow (\neg P \vee \neg Q)$$

$$(\neg P \vee Q) \leftrightarrow (P \rightarrow Q)$$

SUBFORMULAS

- Formula **G** is a **subformula** of **F** if it occurs within **F.**
- Rules:
  - Subformula of P is P
  - Subformulae of ¬F: ¬F and subformulae of F
  - Subformulae of $F_1 \circ F_2$ ($\circ$ = **logical connectives**): the full formula and the subformulae of F_1 and F_2

- **Strict subformulae**: all subformulae **except** the formula itself

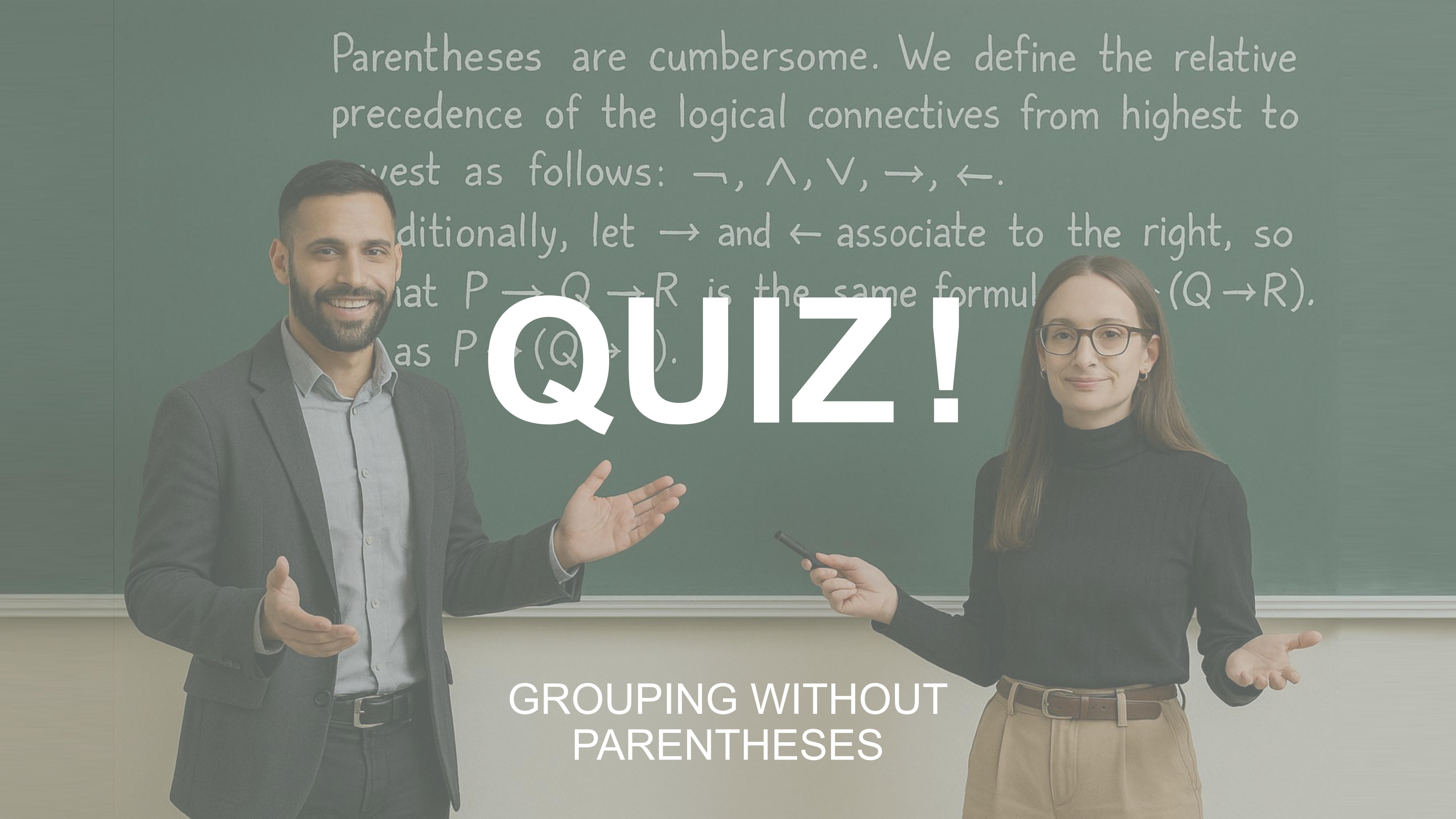
SUBFORMULAS - EXAMPLE

$$F: \; (P \land Q) \; \to \; (P \lor \neg Q)$$

F	$P \land Q$	$P \lor \neg Q$	P	Q	$\neg Q$

PRECEDENCE & ASSOCIATIVITY

- Formula **G** is a **subformula** of **F** if it occurs within **F.**
- Precedence Order (Highest to Lowest):
  - $\neg$ (not)
  - $\land$ (and)
  - $\lor$ (or)
  - $\to$ (implies)
  - $\leftrightarrow$ (if and only if)
- Associativity Rules:
  - $\to$ and $\leftrightarrow$ are right-associative.
  - Example: $P \to Q \to R$ is interpreted as $P \to (Q \to R)$.



QUIZ !

GROUPING WITHOUT PARENTHESES

Which of the following is logically equivalent to:

$$P_1 \wedge \neg P_2 \wedge \top \vee \neg P_1 \wedge P_2$$

a $(P_1 \wedge (\neg P_2 \wedge \top)) \vee (\neg P_1 \wedge P_2)$

b $P_1 \wedge \neg(P_2 \wedge (\top \vee \neg P_1)) \wedge P_2$

c $((P_1 \wedge \neg P_2) \wedge \top) \vee \neg(P_1 \wedge P_2)$

d $((P_1 \wedge \neg P_2 \wedge \top \vee \neg P_1) \wedge P_2)$

Which of the following is logically equivalent to:

$$P_1 \land \neg P_2 \land \top \lor \neg P_1 \land P_2$$

a $(P_1 \land (\neg P_2 \land \top)) \lor (\neg P_1 \land P_2)$

b $P_1 \land \neg(P_2 \land (\top \lor \neg P_1)) \land P_2$

c $((P_1 \land \neg P_2) \land \top) \lor \neg(P_1 \land P_2)$

d $((P_1 \land \neg P_2 \land \top \lor \neg P_1) \land P_2)$

SYNTAX - EXAMPLE

- **Formula:** $F : (P \wedge Q) \rightarrow (T \vee \neg Q)$
- **atoms:** P , Q , T
- **literal:** $\neg Q$
- **subformulas:** $\mathrm{P} \wedge Q$, $T \vee \neg Q$

$$F : \overbrace{(P \wedge Q)}^{\text{subformula}} \rightarrow (\overbrace{T}^{\text{atom}} \vee \overbrace{\neg Q}^{\text{literal}})$$

PROPOSITIONAL LOGIC (PL)

SYNTAX

How formulas are built

SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SEMANTICS

What is Semantics?

- **Semantics** defines the **meaning** of logical formulas.
- In PL, the meaning is based on **truth values**: true or false

INTERPRETATION

Interpretation (I) assigns a truth value to each propositional variable.

- Example: $I: \{P \mapsto \text{true},\ Q \mapsto \text{false},\ \ldots\}$

TRUTH TABLES

Negation (¬):

F	$\neg F$
0	1
1	0

Connectives Table:

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

TRUTH TABLES - EXAMPLE

Formula: $F: \; P \wedge Q \; \rightarrow \; P \vee \neg Q$

Interpretation: $I: \; \{P \mapsto \text{true}, \; Q \mapsto \text{false}\}$

Step-by-step Table:

P	Q	$\neg Q$	$P \wedge Q$	$P \vee \neg Q$	F
1	0	1	0	1	1

F evaluates to true under interpretation I.

TRUTH TABLES

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$	$F_1 \leftrightarrow F_2$
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Is truth table a efficient method

to evalute formulas in PL?

INDUCTIVE DEFINITION OF PL'S SEMANTICS

$I \models F$ if F evaluates to true under I
$I \not\models F$ false

<u>Base Case</u>:

$I \models \top$
$I \not\models \bot$

$I \models P$ iff $I[P] = \text{true}$
$I \not\models P$ iff $I[P] = \text{false}$

<u>Inductive Case</u>:

$I \models \neg F$ iff $I \not\models F$
$I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$
$I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$
$I \models F_1 \rightarrow F_2$ iff, if $I \models F_1$ then $I \models F_2$
$I \models F_1 \leftrightarrow F_2$ iff, $I \models F_1$ and $I \models F_2$,
or $I \not\models F_1$ and $I \not\models F_2$

<u>Note</u>:

$I \not\models F_1 \rightarrow F_2$ iff $I \models F_1$ and $I \not\models F_2$

INDUCTIVE DEFINITION - EXAMPLE

Formula: $F: \; P \wedge Q \; \rightarrow \; P \vee \neg Q$

Interpretation: $I: \; \{P \mapsto \text{true}, \; Q \mapsto \text{false}\}$

$$
\begin{array}{llll}
1. & I & \models & P & \text{since } I[P] = \text{true} \\
2. & I & \not\models & Q & \text{since } I[Q] = \text{false} \\
3. & I & \models & \neg Q & \text{by 2 and semantics of } \neg \\
4. & I & \not\models & P \wedge Q & \text{by 2 and semantics of } \wedge \\
5. & I & \models & P \vee \neg Q & \text{by 1 and semantics of } \vee \\
6. & I & \models & F & \text{by 4 and semantics of } \rightarrow
\end{array}
$$

?Why

$$F: \; P \wedge Q \;\rightarrow\; P \vee \neg Q$$

$$I: \; \{P \mapsto \mathsf{true}, \; Q \mapsto \mathsf{false}\}$$

INDUCTIVE DEFINITION - EXAMPLE

$$
\begin{array}{llll}
1. & I \models P & & \text{since } I[P] = \mathsf{true} \\
2. & I \not\models Q & & \text{since } I[Q] = \mathsf{false} \\
3. & I \models \neg Q & & \text{by 2 and semantics of } \neg \\
4. & I \not\models P \wedge Q & & \text{by 2 and semantics of } \wedge \\
5. & I \models P \vee \neg Q & & \text{by 1 and semantics of } \vee \\
6. & I \models F & & \text{by 4 and semantics of } \rightarrow
\end{array}
$$

?Why

$$I \models F_1 \rightarrow F_2 \qquad \text{iff, if } I \models F_1 \text{ then } I \models F_2$$

PROPOSITIONAL LOGIC (PL)

SYNTAX

How formulas are built

SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SATISFIABILITY AND VALIDITY

F **satisfiable** $\leftrightarrow$ there exists an interpretation I such that $\mathrm{I} \vDash F$.

F **valid** $\leftrightarrow$ for all interpretations I, $\mathrm{I} \vDash F$.

$$\boxed{F \text{ is valid} \leftrightarrow \neg F \text{ is unsatisfiable}}$$

Why it works?

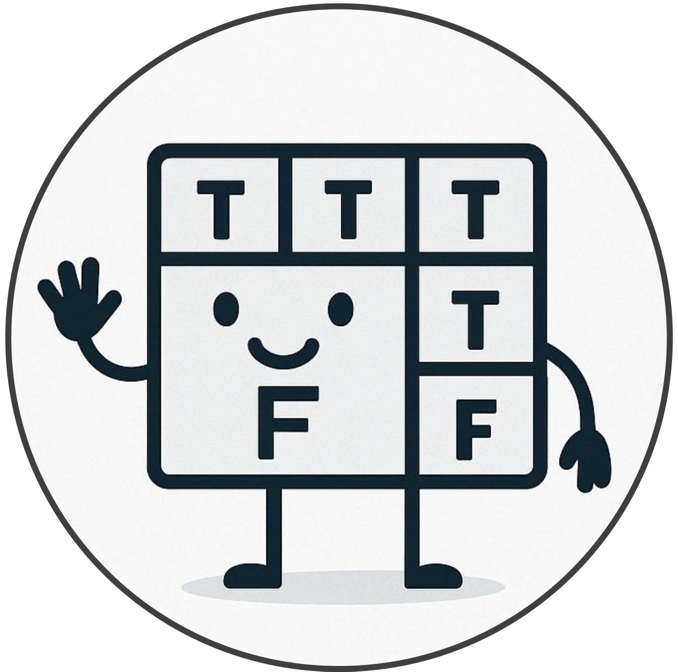
If F is valid,

then for every I, $\mathrm{I} \vDash F \rightarrow$

so $\mathrm{I} \nvDash \neg F \rightarrow$

$\neg F$ is unsatisfiable

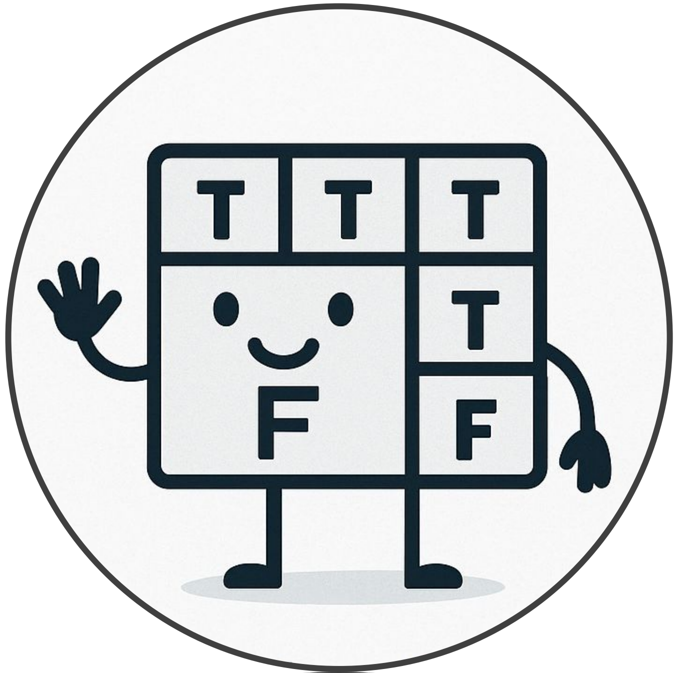
METHODS FOR VALIDITY & SATISFIABILITY



Truth tables

Semantic arguments

METHODS FOR VALIDITY & SATISFIABILITY



Truth tables

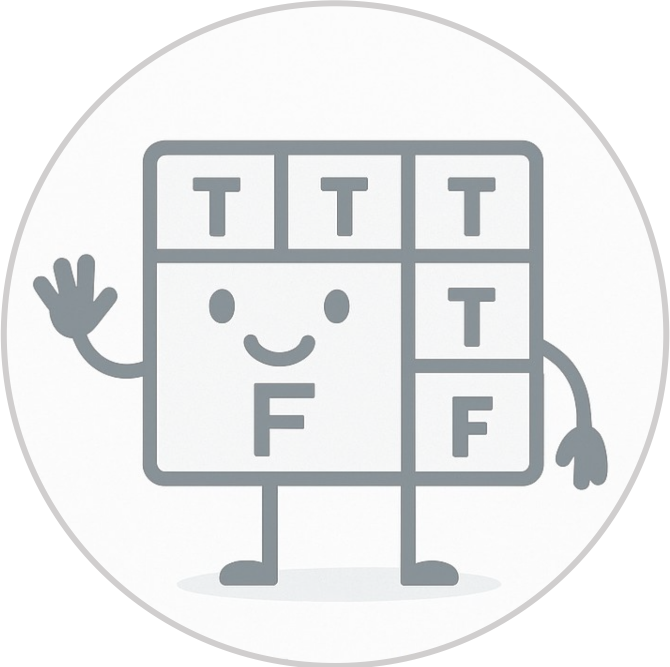
Semantic arguments

METHOD 1: TRUTH TABLES

Example: $F: \; P \wedge Q \;\rightarrow\; P \vee \neg Q \; .$

P	Q	$P \wedge Q$	$\neg Q$	$P \vee \neg Q$	F
0	0	0	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	1	1

METHODS FOR VALIDITY & SATISFIABILITY



Truth tables

Semantic arguments

METHOD 2: SEMANTIC ARGUMENTS

Semantic Arguments provide an alternative to truth tables for proving validity.

Approach:

- Assume the formula F is invalid (i.e., there is a falsifying interpretation I such that $I \not\models F$)
- Apply semantic definitions to deduce consequences
- Derive a contradiction → proves F is valid

METHOD 2: SEMANTIC ARGUMENTS - EXAMPLE

$$\boxed{F : P \land Q \rightarrow P \lor \neg Q}$$

1	$I \not\models F$	(assum.)
2	$I \models P \land Q$	(1 and $\rightarrow$)
3	$I \not\models P \lor \neg Q$	(1 and $\rightarrow$)
4	$I \models P$	(2 and $\land$)
5	$I \not\models P$	(3 and $\lor$)
6	$\perp$	(4 and 5)

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F} \qquad \frac{I \models F \land G}{I \models F \qquad I \models G}$$

$$\frac{I \not\models F \land G}{I \not\models F \quad \mid \quad I \not\models G} \qquad \frac{I \models F \lor G}{I \models F \quad \mid \quad I \models G}$$

$$\frac{I \not\models F \lor G}{I \not\models F \qquad I \not\models G} \qquad \frac{I \models F \rightarrow G}{I \not\models F \quad \mid \quad I \models G}$$

$$\frac{I \not\models F \rightarrow G}{I \models F \qquad I \not\models G}$$

Found a contradiction, so F is valid

SEMANTIC ARGUMENT

PROOF RULES

NEGATION (¬)

$$\frac{I \models \neg F}{I \not\models F} \qquad \frac{I \not\models \neg F}{I \models F}$$

CONJUNCTION (∧)

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}} \leftarrow\text{and} \qquad \frac{I \not\models F \wedge G}{I \not\models F \mid I \not\models G}$$

or

DISJUNCTION (∨)

$$\frac{I \models F \vee G}{I \models F \mid I \models G} \qquad \frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

IMPLICATION (→)

$$\frac{I \models F \rightarrow G}{I \not\models F \mid I \models G} \qquad \frac{I \not\models F \rightarrow G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

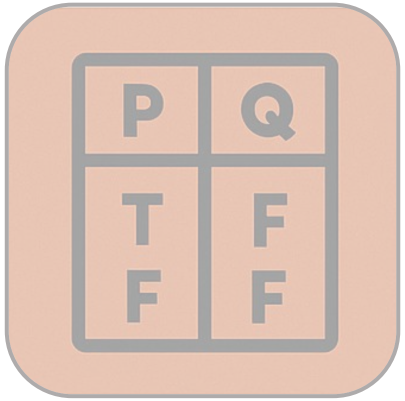
BICONDITIONAL (↔)

$$\frac{I \models F \leftrightarrow G}{I \models F \wedge G \mid I \not\models F \vee G} \qquad \frac{I \not\models F \leftrightarrow G}{I \models F \wedge \neg G \mid I \models \neg F \wedge G}$$

CONTRADICTION RULE

$$\frac{\begin{array}{l} I \models F \\ I \not\models F \end{array}}{I \models \bot}$$

LECTURE OUTLINE



Propositional Logic (PL)

The logic of statements that are either true or false.

First Order Logic (FOL)

Logic with predicates, functions, and quantifiers.

First Order Theories

Logical frameworks for reasoning about numbers, arrays, and data.

FIRST ORDER LOGIC (FOL)

MOTIVATION

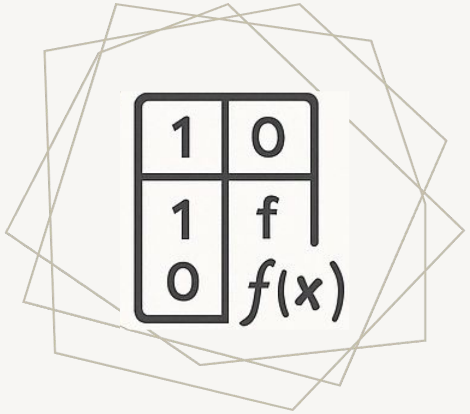
```python
def g(ig):
    og = ig
    for i in range(2):
        og = f(og, ig)
    return og
```

```python
def h(ih):
    oh = f(f(ih,ih),ih)
    return oh
```

FIRST ORDER LOGIC (FOL)

SYNTAX

How formulas are built



SEMANTICS

What formulas mean

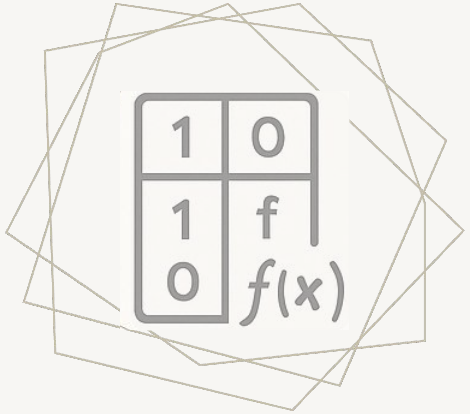
SATISFIABILITY AND VALIDITY

When formulas are true

FIRST ORDER LOGIC (FOL)

SYNTAX

How formulas are built



SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SYNTAX

FOL extends PL with:

- **Terms** – the basic ones are **variables** and **constants**. More complex terms are **functions** which takes n terms as arguments.
- **Predicates** – generalization of propositional variables of PL. An n-ary predicate takes n terms as arguments and return truth value.
- **Quantifiers** - existential quantifier $\exists x. F[x]$, universal quantifier $\forall x. F[x]$.

SYNTAX

The following are all terms:

- x, a variable;
- $f(a)$, a unary function f applied to a constant;
- $g(x,b)$, a binary function g applied to a variable x and a constant b;
- $f(g(x,f(b)))$

SYNTAX

- **atoms** – $\top$, $\bot$, or an n-ary predicate applied to n terms.
- **literals** - An atom or its negation.

$$\overbrace{p(f(x),y)}^{atom/literal}\ ,\qquad \overbrace{\neg p(\underbrace{f(y)}_{term},g(x))}^{literal}$$

SYNTAX

The most simple and basic FOL formula is just one literal

$$p(x, y)$$

But we can create complex FOL formulae using logical connectives,

like $\wedge$, $\vee$, $\longrightarrow$, $\neg$ or $\leftrightarrow$.

$$\overbrace{p(f(x), y)}^{atom/literal} \wedge r(g(y), x) \longrightarrow \overbrace{\neg p(\underbrace{f(y)}_{term}, g(x))}^{literal}$$

SYNTAX

There are two FOL quantifiers:

- the **existential** quantifier $\exists$x. F[x], read “there exists an x such that F[x]”;
- and the **universal** quantifier $\forall$x. F[x], read “for all x, F[x]”.

$$\forall x.\ p(f(x), x) \rightarrow \left(\exists y. p\left(f\left(g(x, y)\right), g(x, y)\right)\right) \land q(x, f(x))$$

SYNTAX

A variable is **free** in formula F[x] if there is an occurrence of x that is not bound by any quantifier. Denote by free(F) the set of free variables of a formula F.

A variable is **bound** in formula F[x] if there is an occurrence of x in the scope of a binding quantifier $\forall x$ or $\exists x$. Denote by bound(F) the set of bound variables of a formula F.

A formula F is **closed** if it does not contain any free variables.

SYNTAX

For example, consider the following FOL formula:

$$F: \ \forall x.\ p(f(x), y) \ \rightarrow \ \forall y.\ p(f(x), y)$$

x only occurs bound, while y appears both free (in the antecedent) and bound (in the consequent). Thus, free(F) = {y} and bound(F) = {x,y}.

So in this case we have: $free(F) \cap \ bound(F) \neq \phi$

SYNTAX

The **subformulae** of a FOL formula are defined according to an extension of the PL definition of subformula:

- the only subformula of $p(t_1, \dots, t_n)$, where the t_1 are terms, is $p(t_1, \dots, t_n)$;
- the subformulae of ¬F are ¬F and the subformulae of F;
- the subformulae of $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$ are the formula itself and the subformulae of F_1 and F_2;
- the subformulae of ∃x. F and ∀x. F are the formula itself and the subformulae of F.

SYNTAX

The **subterms** of a FOL term are defined as follows:

- the only subterm of constant a or variable x is a or x itself.
- and the subterms of $\mathrm{f}(t_1, \dots, t_n)$ are the term itself and the subterms of $t_1, \dots, t_n$.

SYNTAX :SUBFORMULAS - EXAMPLE

$$F: \ \forall x.\ p(f(x), y) \ \rightarrow \ \forall y.\ p(f(x), y)$$

The subformulae of F are:

F

$p(f(x), y) \rightarrow \forall y. p(f(x), y)$

$\forall y. p(f(x), y)$

$p(f(x), y)$

SYNTAX :SUBFORMULAS - EXAMPLE

$$T: \; g\left(f(x), f\left(h(f(x))\right)\right)$$

The subterm of F are:

T

$f(x)$

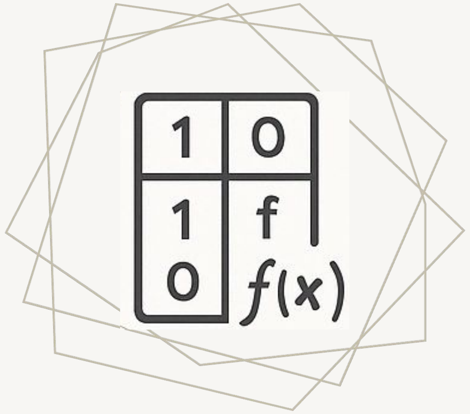
$f\left(h(f(x))\right)$

$h(f(x))$

FIRST ORDER LOGIC (FOL)

SYNTAX

How formulas are built



SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SEMANTICS

We must remember that both PL and FOL formulas are evaluated to the truth values true and false. However, **terms** of FOL formulae evaluate to values from a specified domain.

That lead us to definition of **FOL interpretation**.

An **interpretation I** is a pair (D_I, α_I) which consist of a domain and an assignment.

SEMANTICS

The domain D_I of an interpretation I is a nonempty set of values or objects, such as integers, real numbers, dogs, people, or merely abstract objects.

$|D_I|$ denotes the cardinality, or size, of D_I.

SEMANTICS

The assignment α_I of interpretation I maps constant, function, and predicate symbols to elements, functions, and predicates over D_I.

It also maps variables to elements of D_I:

- each variable symbol x is assigned a value x_i from D_I;
- each $n-ary$ function symbol f is assigned an $n-ary$ function $f_i : D_I^n \rightarrow D_I$ that maps n elements of D_I to an element of D_I;
- each $n-ary$ predicate symbol p is assigned an $n-ary$ predicate p_i: $D_I^n \rightarrow$ {true, false} that maps n elements of D_I to a truth value.

SEMANTICS

let's clarify these ideas with the following example:

$$F:\ x+y>z \ \longrightarrow\ y>z-x$$

$$F':\ p(f(x,y),z) \ \longrightarrow\ p\big(y,g(z,x)\big).$$

SEMANTICS

$$F: \; x + y > z \;\longrightarrow\; y > z - x$$

now let's construct a "standard" interpretation:

- The domain is the integers, $D_I = \mathbb{Z}$.
- $\alpha_I \coloneqq \{+ \mapsto +_{\mathbb{Z}}, \; - \mapsto -_{\mathbb{Z}}, > \mapsto >_{\mathbb{Z}}, \; x \mapsto 13, \; y \mapsto 42, \; z \longmapsto 13\}$

SEMANTICS

Given a FOL formula F and interpretation, I : $(D_I\ ,\ \alpha_I\ )$, we want to compute if F evaluates to true under interpretation I, $I \vDash F$, or if F evaluates to false under interpretation I, $I \ \nvDash\ F$.

We define the semantics inductively, we saw the assignment $\boldsymbol{\alpha_I}$ gives meaning $\boldsymbol{\alpha_I}$ [x], $\boldsymbol{\alpha_I}$ [c], and $\boldsymbol{\alpha_I}$ [f] to variables x, constants c, and functions f.

SEMANTICS

Evaluate arbitrary terms recursively: $\alpha_I[f(t_1, \dots, t_n)] = \alpha_I[f](\alpha_I[t_1], \dots, \alpha_I[t_n])$, for terms $t_1, \dots, t_n$.

That is, define the value of $f(t_1, \dots, t_n)$ under α_I by evaluating the function α_I $\alpha_I[f]$ over the terms $\alpha_I[t_1], \dots, \alpha_I[t_n]$.

Similarly, evaluate arbitrary atoms recursively:

$$\alpha_I[p(t_1, \dots, t_n)] = \alpha_I[p](\alpha_I[t_1], \dots, \alpha_I[t_n])$$

Then $I \vDash p(t_1, \dots, t_n) \leftrightarrow \alpha_I[p](\alpha_I[t_1], \dots, \alpha_I[t_n]) = true$

SEMANTICS

after covering the base case of the inductive semantics, we can turn to the inductive step.

$$
\begin{array}{lcll}
I & \models & \neg F & \text{iff } I \not\models F \\
I & \models & F_1 \wedge F_2 & \text{iff } I \models F_1 \text{ and } I \models F_2 \\
I & \models & F_1 \vee F_2 & \text{iff } I \models F_1 \text{ or } I \models F_2 \\
I & \models & F_1 \rightarrow F_2 & \text{iff, if } I \models F_1 \text{ then } I \models F_2 \\
I & \models & F_1 \leftrightarrow F_2 & \text{iff } I \models F_1 \text{ and } I \models F_2\text{, or } I \not\models F_1 \text{ and } I \not\models F_2
\end{array}
$$

SEMANTICS

$$F:\ x+y>z\ \rightarrow\ y>z-x$$

of Example 2.7 and the interpretation $I:(\mathbb{Z},\alpha_I)$, where

$$\alpha_I:\ \{+\mapsto +_{\mathbb{Z}},\ -\mapsto -_{\mathbb{Z}},\ >\mapsto >_{\mathbb{Z}},\ x\mapsto 13_{\mathbb{Z}},\ y\mapsto 42_{\mathbb{Z}},\ z\mapsto 1_{\mathbb{Z}}\}\ .$$

Compute the truth value of F under I as follows:

1.	$I \models$	$x+y>z$	since $\alpha_I[x+y>z] = 13_{\mathbb{Z}} +_{\mathbb{Z}} 42 >_{\mathbb{Z}} 1_{\mathbb{Z}}$
2.	$I \models$	$y>z-x$	since $\alpha_I[y>z-x] = 42_{\mathbb{Z}} >_{\mathbb{Z}} 1_{\mathbb{Z}} -_{\mathbb{Z}} 13_{\mathbb{Z}}$
3.	$I \models$	F	by 1, 2, and the semantics of $\rightarrow$

SEMANTICS

For the quantifiers, let x be a variable. Define an x-variant of an interpretation $I = (D_I, \alpha_I)$ as an interpretation $J = (D_J, \alpha_J)$ such that:

- $D_I = D_J$
- And $\alpha_I[y] = \alpha_J$[y] for all constants, free variable, function and predicate symbols y, except possibly x.

That is, I and J agree on everything except possibly the value of variable x.

SEMANTICS

Denote by $J: I \leftarrow \{x \mapsto v\}$ the x-variant of I in which α_J[x] = v for some vϵ D_I.

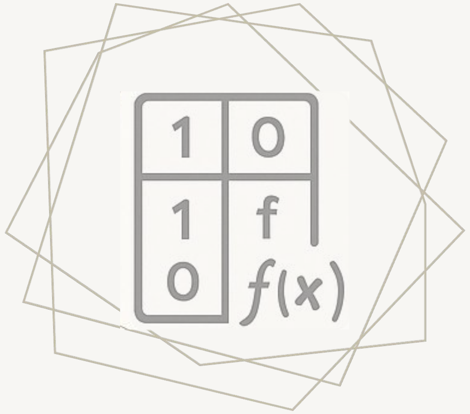
Then:

- $I \vDash \forall x. F$ iff for all $v \in D_I$, $I \leftarrow \{x \mapsto v\} \mid = F$
- $I \vDash \exists x. F$ iff there exist $v \in D_I$, such that $I \leftarrow \{x \mapsto v\} \mid = F$

FIRST ORDER LOGIC (FOL)

SYNTAX

How formulas are built



SEMANTICS

What formulas mean

SATISFIABILITY AND VALIDITY

When formulas are true

SATISFIABILITY AND VALIDITY

- A formula F is said to be **satisfiable** ↔ there exists an interpretation I such that $I \vDash F$.
- A formula F is said to be **valid** ↔ for all interpretations I, $I \vDash F$.
- Determining satisfiability and validity of formulae are important tasks in FOL.
- Recall that **satisfiability and validity are dual**:
  F is valid ↔ ¬F is unsatisfiable.

SATISFIABILITY AND VALIDITY

According to the semantics of universal quantification, from $I \vDash \forall x. F$, deduce $I \leftarrow \{x \mapsto v\} \vDash F$ for any vϵ D_I.

- $\dfrac{I \vDash \forall x.F}{I \leftarrow \{x \mapsto v\} \mid= F}$ for any vϵ D_I

- $\dfrac{I \nvDash \exists x.F}{I \leftarrow \{x \mapsto v\} \nvDash F}$ for any vϵ D_I

- $\dfrac{I \vDash \exists x.F}{I \leftarrow \{x \mapsto v\} \mid= F}$ for some vϵ D_I

- $\dfrac{I \nvDash \forall x.F}{I \leftarrow \{x \mapsto v\} \nvDash F}$ for a some vϵ D_I

SATISFIABILITY AND VALIDITY

We add another rule which helps us determine a contradiction.

- A contradiction exists if two variants of the original interpretation I disagree on the truth value of an n-ary predicate p for a given tuple of domain values.

$$\frac{\begin{array}{lcl} J : I \triangleleft \cdots & \models & p(s_1, \ldots, s_n) \\ K : I \triangleleft \cdots & \not\models & p(t_1, \ldots, t_n) \end{array}}{I \models \bot} \quad \text{for } i \in \{1, \ldots, n\},\ \alpha_J[s_i] = \alpha_K[t_i]$$

SATISFIABILITY AND VALIDITY

We want to prove that $(\forall x.\, p(x)) \longrightarrow (\forall y.\, p(y))$ is valid.

Suppose not, then there is an interpretation I such that $I \not\models F$:

1. $I \models \forall x.\, p(x)$ assumption and semantics of $\longrightarrow$
2. $I \not\models \forall y.\, p(y)$ assumption and semantics of $\longrightarrow$
3. $I \leftarrow \{y \mapsto v\} \not\models p(y)$ 2 and semantics of $\forall$, for some $v \epsilon D_I$
4. $I \leftarrow \{x \mapsto v\} \models p(x)$ 1 and semantics of $\forall$
5. Under $I, p(v)$ is false by 3 and true by 4.

SATISFIABILITY AND VALIDITY

```python
def g(ig):
    og = ig
    for i in range(2):
        og = f(og, ig)
    return og
```

```python
def h(ih):
    oh = f(f(ih,ih),ih)
    return oh
```

SATISFIABILITY AND VALIDITY

```python
def g_unroll(ig):
    og = ig
    og = f(og, ig)
    og = f(og, ig)
    return og
```

```python
def g(ig):
    og = ig
    for i in range(2):
        og = f(og, ig)
    return og
```

```python
def h(ih):
    oh = f(f(ih,ih),ih)
    return oh
```

$$\psi_g:\; og1 = ig \land og2 = f(og1, ig) \land og3 = f(og2, ig)$$

$$\psi_h:\; oh1 = f(f(ih, ih), ih)$$

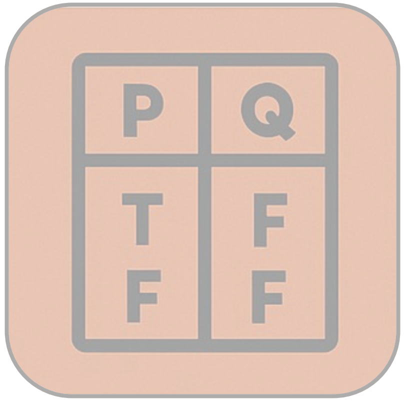
SATISFIABILITY AND VALIDITY

Two functions are equivalent if and only if for every two equal inputs the outputs are the same, that lead us to the formula:

$$\psi \;\coloneqq\; \left(ih = ig \wedge \psi_h \wedge \psi_g\right) \longrightarrow og3 = oh1$$

Finally: $\quad g\ and\ h\ are\ equivalent\ \leftrightarrow \psi\ is\ valid$

LECTURE OUTLINE



Propositional Logic (PL)

The logic of statements that are either true or false.

First Order Logic (FOL)

Logic with predicates, functions, and quantifiers.

First Order Theories

Logical frameworks for reasoning about numbers, arrays, and data.

FIRST ORDER THEORIES

FIRST-ORDER THEORIES

How logic meets real data:

- Numbers, arrays, lists, trees
- Equality, structure, behavior
- Useful for software verification

FIRST ORDER THEORIES

First-Order Theories

Symbols and axioms

Equality

Rules of equality

Natural Numbers and Integers

Logic over whole numbers

Rationals and Reals

Logic over numeric values

Recursive Data Structures

Logic for lists and trees

Arrays

Logic for read/write memory

FIRST ORDER THEORIES

First-Order Theories
Symbols and axioms

Equality
Rules of equality

Natural Numbers and Integers
Logic over whole numbers

Rationals and Reals
Logic over numeric values

Recursive Data Structures
Logic for lists and trees

Arrays
Logic for read/write memory

FIRST-ORDER THEORIES

Definition: A first-order theory **T** is defined by:

- **Signature (Σ):** A set of constant, function, and predicate symbols.
- **Axioms (A):** A set of **closed** FOL formulas using only symbols from Σ.

Σ-formulas are built from these symbols, variables, connectives, and quantifiers.

- The symbols themselves have no built-in meaning.
- Their meaning comes from the axioms A.

$$\Sigma = \{a, b, f, P\}$$
$$A = \{P(a), \forall x. P(x) \rightarrow P(f(x))\}$$

SATISFIABILITY AND VALIDITY

T-validity ($T \vDash F$): A formula F is **valid** in theory T if every interpretation I that satisfies all axioms A also satisfies F.

T-satisfiability: A formula F is **satisfiable** in T if there is some interpretation I (a **T-interpretation**) such that $I \vDash F$.

Example: Let T contain axiom: $\forall x.\ P(x) \rightarrow Q(x)$
Then the formula: $P(a) \rightarrow Q(a)$ is T-valid if $P(a)$ holds under T.

SATISFIABILITY AND VALIDITY

- **Complete theory:** For every formula F: either $T \vDash F$ or $T \vDash \neg F$.
- **Consistent theory:** There exists at least one interpretation satisfying all axioms.
  - This means we cannot have both $T \vDash F$ and $T \vDash \neg F$.
  - Otherwise, $T \vDash F \land \neg F \rightarrow T \vDash \bot$ (contradiction).

Equivalence: Formulas F_1 and F_2 are **T-equivalent** if $T \vDash F_1 \leftrightarrow F_2$.

FRAGMENTS AND DECIDABILITY

Fragment: A syntactically restricted subset of formulas.

- e.g., The **quantifier-free fragment** contains formulas with no quantifiers.

Decidability:

- A theory T is **decidable** if there exists an algorithm that can decide for every Σ-formula F whether $T \models F$.
- A fragment is decidable if this holds within the fragment's restrictions.

COMBINING THEORIES

Union of Theories:

- $T_1 \cup T_2$ has signature $\Sigma_1 \cup \Sigma_2$ and axioms $A_1 \cup A_2$.
- An interpretation of $T_1 \cup T_2$ satisfies both theories.

Implications:

- If a formula is valid in T_1 or $T_2 \rightarrow$ valid in $T_1 \cup T_2$.
- If a formula is satisfiable in $T_1 \cup T_2 \rightarrow$ satisfiable in both T_1 and T_2.

Why we care:

- First-Order Logic (FOL) itself is undecidable.
- Many **theories and fragments** are **decidable**.
- These are central to automated reasoning and verification.

FIRST ORDER THEORIES

First-Order Theories Symbols and axioms	**Equality** Rules of equality	**Natural Numbers and Integers** Logic over whole numbers
Rationals and Reals Logic over numeric values	**Recursive Data Structures** Logic for lists and trees	**Arrays** Logic for read/write memory

EQUALITY

$T_{\mathbf{E}}$ is the simplest first-order theory.

- **Signature $\Sigma_{\mathbf{E}}$:**
  - = (equality) — binary predicate, interpreted via axioms
  - All other symbols (constants, functions, predicates) are uninterpreted — except as they relate to equality

EQUALITY

Axioms of $T_{\mathbf{E}}$:

1. **Reflexivity:** $\forall x.\ x = x$

2. **Symmetry:** $\forall x, y.\ x = y \rightarrow y = x$

3. **Transitivity:** $\forall x, y, z.\ x = y \land y = z \rightarrow x = z$

4. **Function congruence:** $\forall \bar{x}, \bar{y}.\ (\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(\bar{x}) = f(\bar{y})$

5. **Predicate congruence:** $\forall \bar{x}, \bar{y}.\ (\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow \left(p(\bar{x}) \leftrightarrow p(\bar{y})\right)$

$\mathcal{T}_E$ IS UNDECIDABLE

- $\mathcal{T}_E$ includes **all** constant, function, and predicate symbols.
- So, **any FOL formula** can be rephrased as a $\mathcal{T}_E$ formula.
- That makes $\mathcal{T}_E$ **undecidable**, just like FOL.

But the **quantifier-free fragment** *is* decidable — and very useful.

VALIDITY AND SATISFIABILITY IN T_E

Although T_E as a whole is undecidable, Its **quantifier-free fragment** is both interesting and **efficiently decidable**.

Example:

F: $a = b \land b = b \rightarrow g(f(a), b) = g(f(c), a)$ // constants

F′: $x = y \land y = z \rightarrow g(f(x), y) = g(f(z), x)$ // free variables

F is T_E-valid $\Leftrightarrow$ F′ is T_E-valid

F is T_E-satisfiable $\Leftrightarrow$ F′ is T_E-satisfiable

SEMANTIC ARGUMENT — PROVING VALIDITY IN T_E

We can use a semantic argument to prove that a formula is valid in T_E.

Example: Prove that: $F: \; a = b \;\wedge\; b = c \;\rightarrow\; g(f(a), b) = g(f(c), a)$

is T_E -valid Assume it's false.

1.	I	$\not\models$	F	assumption
2.	I	$\models$	$a = b \;\wedge\; b = c$	$1, \rightarrow$
3.	I	$\not\models$	$g(f(a), b) = g(f(c), a)$	$1, \rightarrow$
4.	I	$\models$	$a = b$	$2, \wedge$
5.	I	$\models$	$b = c$	$2, \wedge$
6.	I	$\models$	$a = c$	4, 5, (transitivity)
7.	I	$\models$	$f(a) = f(c)$	6, (function congruence)
8.	I	$\models$	$b = a$	4, (symmetry)
9.	I	$\models$	$g(f(a), b) = g(f(c), a)$	7, 8 (function congruence)
10.	I	$\models$	$\bot$	3, 9

FIRST ORDER THEORIES

First-Order Theories
Symbols and axioms

Equality
Rules of equality

Natural Numbers and Integers
Logic over whole numbers

Rationals and Reals
Logic over numeric values

Recursive Data Structures
Logic for lists and trees

Arrays
Logic for read/write memory

NATURAL NUMBERS AND INTEGERS

We explore three foundational arithmetic theories:

- **Peano Arithmetic (T_{PA}):** supports addition and multiplication over $\mathbb{N}$.
- **Presburger Arithmetic (TN)**: only supports addition.
- **Theory of Integers (TZ)**: simplifies reasoning over $\mathbb{Z}$.

PEANO ARITHMETIC (T_{PA})

Signature $\Sigma_{PA} : \{0,\ 1,\ +,\ \cdot,\ =\}$

- 0, 1: constants
- +, $\cdot$: binary functions (addition, multiplication)
- =: equality predicate

Axioms:

1. Zero: $\forall x.\ \neg(x + 1 = 0)$
2. Successor: $\forall x, y.\ x + 1 = y + 1 \rightarrow x = y$
3. Induction: $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$
4. plus zero: $\forall x.\ x + 0 = x$
5. plus successor: $\forall x, y.\ x + (y + 1) = (x + y) + 1$
6. times zero: $\forall x.\ x \cdot 0 = 0$
7. times successor: $\forall x, y.\ x \cdot (y + 1) = x \cdot y + x$

INTENDED INTERPRETATION OF T_{PA}

The standard model of Peano Arithmetic assumes:

- **Domain**: $\mathbb{N}$ (natural numbers)
- $I[0], I[1]$: $0_{\mathbb{N}}, 1_{\mathbb{N}} \in \mathbb{N}$
- $I[+]$: $+_{\mathbb{N}}$,addition over $\mathbb{N}$
- $I[\times]$: $\times_{\mathbb{N}}$,multiplication over $\mathbb{N}$
- $I[=]$: $=_{\mathbb{N}}$,numeric equality over $\mathbb{N}$

LIMITS OF PEANO ARITHMETIC

- Satisfiability and validity in $T_{\mathbf{PA}}$ are undecidable
- Even quantifier-free $T_{\mathbf{PA}}$ is undecidable!
- Moreover, $T_{\mathbf{PA}}$ is incomplete (Gödel, 1930)
- Upshot: there are valid arithmetic propositions that are $T_{\mathbf{PA}}$ -invalid

Lesson: Multiplication is hard! Let’s try something easier.

PRESBURGER ARITHMETIC ($T_{\mathbb{N}}$)

Signature: $\Sigma_{\mathbb{N}}: \; \{0, \; 1, \; +, \; =\}$

Axioms: Subset of T_{PA}

1. All of the equality axioms: reflexivity, symmetry, transitivity, and congruence
2. **Zero:** $\forall x. \neg(x + 1 = 0)$
3. **Additive identity:** $\forall x. x + 0 = x$
4. **Successor:** $\forall x, y. (x + 1 = y + 1) \rightarrow x = y$
5. **Plus successor:** $\forall x, y. x + (y + 1) = (x + y) + 1$
6. **Induction:** $F[0] \wedge (\forall x.\ F[x] \rightarrow F[x + 1]) \rightarrow \forall x.\ F[x]$

PRESBURGER ARITHMETIC ($T_{\mathbb{N}}$)

The intended interpretation is over $\mathbb{N}$, like with T_{PA}

But, unlike T_{PA}, it has nice properties:

- Validity is **decidable**!
- But complexity is high: $O(2^{2^n})$
- **Complete**: For every $T_{\mathbb{N}}$-formula F, either $\vDash F$ or $\vDash \neg F$
- Admits **quantifier elimination**: For any $T_{\mathbb{N}}$-formula F, there is an equivalent quantifier-free F'
- Validity in quantifier-free fragment is **coNP-complete**

THEORY OF INTEGERS ($T_{\mathbb{Z}}$)

Signature: $\Sigma_{\mathbb{Z}}: \ \{\ldots, -2, -1, 0,\ 1,\ 2,\ \ldots, -3\cdot, -2\cdot,\ 2\cdot,\ 3\cdot,\ \ldots,\ +,\ -,\ =,\ >\}$

- Constants: ..., -2, -1, 0, 1, 2, ...
- Coefficient functions: ..., -2·, 2·, ...
- Binary functions: +, -
- Predicates: =, >

Claim: $T_{\mathbb{Z}}$ is reducible to $T_{\mathbb{N}}$

- Their expressive power is the same, so we won't bother axiomitizing
- $T_{\mathbb{Z}}$ is more convenient, and thus more commonly-used, than $T_{\mathbb{N}}$

EXAMPLE – REDUCING INTEGER LOGIC TO NATURALS

Goal: Rewrite a formula over $\mathbb{Z}$ using only $\mathbb{N}$.

Original:

$$F_0 : \ \forall w, x.\ \exists y, z.\ x + 2y - z - 13 > -3w + 5$$

Idea: Represent each integer v using two naturals: v_p, v_n, such that $v = v_p - v_n$.

Step 1 – Encode subtraction:

$$F_1 : \ \begin{array}{l} \forall w_p, w_n, x_p, x_n.\ \exists y_p, y_n, z_p, z_n. \\ \quad (x_p - x_n) + 2(y_p - y_n) - (z_p - z_n) - 13 > -3(w_p - w_n) + 5 \end{array}$$

EXAMPLE – REDUCING INTEGER LOGIC TO NATURALS

$$F_1 : \begin{array}{l} \forall w_p, w_n, x_p, x_n.\ \exists y_p, y_n, z_p, z_n. \\ \quad (x_p - x_n) + 2(y_p - y_n) - (z_p - z_n) - 13 > -3(w_p - w_n) + 5 \end{array}$$

Step 2 – Eliminate – by rearranging:

$$F_2 : \begin{array}{l} \forall w_p, w_n, x_p, x_n.\ \exists y_p, y_n, z_p, z_n. \\ \quad x_p + 2y_p + z_n + 3w_p > x_n + 2y_n + z_p + 13 + 3w_n + 5\ . \end{array}$$

Step 3 – Remove constants and > using extra variable $u \neq 0$:

$$F_3 : \begin{array}{l} \forall w_p, w_n, x_p, x_n.\ \exists y_p, y_n, z_p, z_n.\ \exists u. \\ \quad \neg(u = 0)\ \land \\ \quad x_p + y_p + y_p + z_n + w_p + w_p + w_p \\ \quad\quad = x_n + y_n + y_n + z_p + w_n + w_n + w_n + u \\ \quad\quad\quad + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ \quad\quad\quad + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1\ . \end{array}$$

THEORY OF INTEGERS ($T_{\mathbb{Z}}$)

Examples in $T_{\mathbb{Z}}$:

- **Example 1:** $\forall x.\ \exists y.\ x = y + 1$ ➔ $\forall x \geq 0 \rightarrow \exists y \geq 0.\ x = y + 1$
- **Example 2:** $\forall x,y,z.\ x > z \land y \geq 0 \rightarrow x + y > z$
- **Example 3:** $\forall x,y.\ x > 0 \land (x = 2y \lor x = 2y + 1) \rightarrow x - y > 0$

All are $T_{\mathbb{Z}}$-valid and show how to reason semantically over integers.

ARITHMETIC THEORY SUMMARY

Theory	Domain	Ops	Decidable?
	$\mathbb{N}$	$+, \cdot$	No
	$\mathbb{N}$	$+$	Yes
	$\mathbb{Z}$	$+, -$	Yes

$\boldsymbol{T_{\mathbf{PA}}}$: powerful but undecidable

$\boldsymbol{T_{\mathbb{N}}}$: limited but decidable and complete

$\boldsymbol{T_{\mathbb{Z}}}$: useful, intuitive, and reducible to $T_{\mathbb{N}}$

FIRST ORDER THEORIES

First-Order Theories Symbols and axioms	Equality Rules of equality	Natural Numbers and Integers Logic over whole numbers
Rationals and Reals **Logic over numeric values**	Recursive Data Structures Logic for lists and trees	Arrays Logic for read/write memory

REAL & RATIONAL ARITHMETIC THEORIES

We study two important arithmetic theories:

Theory	**Domain**	**Ops Supported**	**Notes**
	$\mathbb{R}$	$+$, -, $\cdot$, $=$, $\geq$	Field with roots, ordered, supports multiplication
	$\mathbb{Q}$	$+$, -, $=$, $\geq$	

THEORY OF REALS ($T_{\mathbb{R}}$)

Signature $\Sigma_{\mathbb{R}}: \{0,\ 1,\ +,\ -,\ \cdot,\ =,\ \geq\}$

Includes addition, multiplication, and ordering over $\mathbb{R}$

Abelian Group Axioms (Addition):

1. $\forall x, y, z.\ (x + y) + z = x + (y + z)$ (+ associativity)
2. $\forall x.\ x + 0 = x$ (+ identity)
3. $\forall x.\ x + (-x) = 0$ (+ inverse)
4. $\forall x, y.\ x + y = y + x$ (+ commutativity)

THEORY OF REALS ($T_{\mathbb{R}}$)

Ring Axioms (Multiplication & Distributivity):

1. $\forall x, y, z.\ (xy)z = x(yz)$ (· associativity)
2. $\forall x.\ x1 = x$ (· left identity)
3. $\forall x.\ 1x = x$ (· right identity)
4. $\forall x, y, z.\ x(y + z) = xy + xz$ (left distributivity)
5. $\forall x, y, z.\ (x + y)z = xz + yz$ (right distributivity)

Field Axioms:

1. $\forall x, y.\ xy = yx$ (· commutativity)
2. $0 \neq 1$ (separate identities)
3. $\forall x.\ x \neq 0\ \rightarrow\ \exists y.\ xy = 1$ (· inverse)

THEORY OF REALS ($T_{\mathbb{R}}$)

Order Axioms (≥):

1. $\forall x, y.\ x \geq y \ \wedge \ y \geq x \ \rightarrow \ x = y$ (antisymmetry)
2. $\forall x, y, z.\ x \geq y \ \wedge \ y \geq z \ \rightarrow \ x \geq z$ (transitivity)
3. $\forall x, y.\ x \geq y \ \vee \ y \geq x$ (totality)

Real Closure Axioms:

1. $\forall x, y, z.\ x \geq y \ \rightarrow \ x + z \geq y + z$ (+ ordered)
2. $\forall x, y.\ x \geq 0 \ \wedge \ y \geq 0 \ \rightarrow \ xy \geq 0$ (· ordered)
3. $\forall x.\ \exists y.\ x = y^2 \ \vee \ x = -y^2$ (square-root)
4. for each odd integer n,

$$\forall \overline{x}.\ \exists y.\ y^n + x_1 y^{n-1} + \cdots + x_{n-1} y + x_n = 0 \qquad \text{(at least one root)}$$

THEORY OF REALS: COMPLETE AXIOMATIZATION

1. $\forall x, y.\ x \geq y \ \wedge\ y \geq x \ \rightarrow\ x = y$ (antisymmetry)
2. $\forall x, y, z.\ x \geq y \ \wedge\ y \geq z \ \rightarrow\ x \geq z$ (transitivity)
3. $\forall x, y.\ x \geq y \ \vee\ y \geq x$ (totality)

4. $\forall x, y, z.\ (x + y) + z = x + (y + z)$ (+ associativity)
5. $\forall x.\ x + 0 = x$ (+ identity)
6. $\forall x.\ x + (-x) = 0$ (+ inverse)
7. $\forall x, y.\ x + y = y + x$ (+ commutativity)
8. $\forall x, y, z.\ x \geq y \ \rightarrow\ x + z \geq y + z$ (+ ordered)

9. $\forall x, y, z.\ (xy)z = x(yz)$ (· associativity)
10. $\forall x.\ 1x = x$ (· identity)
11. $\forall x.\ x \neq 0 \ \rightarrow\ \exists y.\ xy = 1$ (· inverse)
12. $\forall x, y.\ xy = yx$ (· commutativity)
13. $\forall x, y.\ x \geq 0 \ \wedge\ y \geq 0 \ \rightarrow\ xy \geq 0$ (· ordered)

14. $\forall x, y, z.\ x(y + z) = xy + xz$ (distributivity)
15. $0 \neq 1$ (separate identities)

16. $\forall x.\ \exists y.\ x = y^2 \ \vee\ -x = y^2$ (square-root)
17. for each odd integer n,

$$\forall \overline{x}.\ \exists y.\ y^n + x_1 y^{n-1} + \cdots + x_{n-1} y + x_n = 0$$ (at least one root)

EXAMPLE – $T_{\mathbb{R}}$ QUANTIFIER ELIMINATION

Given: $F: \exists x.\ ax^2 + bx + c = 0$

Transformed to: $F': b^2 - 4ac \geq 0$

$\rightarrow T_{\mathbb{R}}$ supports quantifier elimination for algebraic reasoning

THEORY OF RATIONALS ($T_{\mathbb{Q}}$)

Signature $\Sigma_{\mathbb{Q}} : \{0,\ 1,\ +,\ -,\ =,\ \geq\}$

Axioms:

1. $\forall x, y.\ x \geq y \ \wedge\ y \geq x \ \rightarrow\ x = y$	(antisymmetry)
2. $\forall x, y, z.\ x \geq y \ \wedge\ y \geq z \ \rightarrow\ x \geq z$	(transitivity)
3. $\forall x, y.\ x \geq y \ \vee\ y \geq x$	(totality)
4. $\forall x, y, z.\ (x + y) + z = x + (y + z)$	(+ associativity)
5. $\forall x.\ x + 0 = x$	(+ identity)
6. $\forall x.\ x + (-x) = 0$	(+ inverse)
7. $\forall x, y.\ x + y = y + x$	(+ commutativity)
8. $\forall x, y, z.\ x \geq y \ \rightarrow\ x + z \geq y + z$	(+ ordered)
9. for each positive integer n, $\forall x.\ nx = 0 \ \rightarrow\ x = 0$	(torsion-free)
10. for each positive integer n, $\forall x.\ \exists y.\ x = ny$	(divisible)

RATIONAL ≈ REAL (FOR LINEAR FORMULAS)

- Every $\Sigma_{\mathbb{Q}}$-formula behaves the same in $\mathbb{Q}$ and $\mathbb{R}$
- No formula can distinguish rational from real domain

$\rightarrow$ For linear logic, $\mathbb{R} \equiv \mathbb{Q}$ under $T_{\mathbb{Q}}$

Example: $\forall x, y.\ \exists z.\ x + y > z$

$\rightarrow$ $\forall x, y.\ \exists z.\ \neg(x + y = z)\ \wedge\ x + y \geq z\ .$

$T_{\mathbb{R}}$ VS $T_{\mathbb{Q}}$ – COMPARISON

Feature		
Supports multiplication	✔️	❌
Handles square roots	✔️	❌
Odd-degree polynomial roots	✔️	❌
Quantifier elimination	✔️	✔️
Decidable?	Yes (complex)	Yes (simpler)

- Use $\boldsymbol{T_{\mathbb{Q}}}$ for linear rational problems
- Use $\boldsymbol{T_{\mathbb{R}}}$ when full algebraic expressiveness is needed

FIRST ORDER THEORIES

First-Order Theories
Symbols and axioms

Equality
Rules of equality

Natural Numbers and Integers
Logic over whole numbers

Rationals and Reals
Logic over numeric values

Recursive Data Structures
Logic for lists and trees

Arrays
Logic for read/write memory

RECURSIVE DATA STRUCTURES (RDS)

- Describe data structures common in programming
- Examples: lists, stacks, binary trees

Non-Recursive vs Recursive

- **Non-recursive**: Like C's struct — a variable with multiple fields
- **Recursive**: A structure that refers to itself (e.g., a list containing another list)

The Theory TRDS

- **TRDS** = Theory of Recursive Data Structures
- Builds on T_{E} (Theory of Equality)
- Helps reason formally about recursive structures

THEORY OF LISTS (T_{CONS})

Focus: LISP-style lists

- Tcons is the theory of lists
- Signature: $\Sigma_{cons} = \{cons, car, cdr, atom, =\}$,Where:

  cons(a, b) – list constructed by concatenating a and b

  car(x) – left projector of x: $car(cons(a, b)) = a$

  cdr(x) – right projector of x: $car(cons(a, b)) = b$

  atom(x) – true $\leftrightarrow$ x is a single-element list

  = - equality predicate

THEORY OF LISTS (T_{CONS})

Example:

- $cons(a, cons(b, c)) =$ list of three elements
- $car\big(cons(a, cons(b, c))\big) = a$
- $cdr\big(cons(a, cons(b, c))\big) = cons(b, c)$

AXIOMS OF T_{CONS}

Based on T_E (Theory of Equality)

- Reflexivity, Symmetry, Transitivity (from T_E)
- Function congruence:

$$\forall x_1, x_2, y_1, y_2.\ x_1 = x_2 \ \wedge\ y_1 = y_2 \ \rightarrow\ \mathsf{cons}(x_1, y_1) = \mathsf{cons}(x_2, y_2)$$

$$\forall x, y.\ x = y \ \rightarrow\ \mathsf{car}(x) = \mathsf{car}(y)$$

$$\forall x, y.\ x = y \ \rightarrow\ \mathsf{cdr}(x) = \mathsf{cdr}(y)$$

- predicate congruence: $\forall x, y.\ x = y \ \rightarrow\ (\mathsf{atom}(x) \leftrightarrow \mathsf{atom}(y))$
- left projection $\forall x, y.\ \mathsf{car}(\mathsf{cons}(x, y)) = x$
- right projection $\forall x, y.\ \mathsf{cdr}(\mathsf{cons}(x, y)) = y$
- Construction $\forall x.\ \neg\mathsf{atom}(x) \ \rightarrow\ \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)) = x$
- atom $\forall x, y.\ \neg\mathsf{atom}(\mathsf{cons}(x, y))$

PROPERTIES OF T_{CONS}

- Equality of lists depends on parts (extensionality)
- Forward: equal lists → equal parts
- Backward: equal parts → equal lists
- Seen also in arrays

GENERAL THEORY OF RDS

- T_{cons} is one example of T_{RDS}.
- Each recursive structure (RDS) has:
  - An $n - ary$ constructor C
  - Projections: $\pi_1^C, \ldots, \pi_n^C$.
  - Predicate: atom_{C}.

GENERAL THEORY OF RDS

Axiom schema for each RDS:

1. T_{E} axioms
2. Function congruence for C, π_i^C
3. Predicate congruence for $\mathsf{atom}_{\mathsf{C}}$.
4. Projection: for each $i \in \{1, \ldots, n\}$, $\forall x_1, \ldots, x_n.\ \pi_i^{\mathsf{C}}(\mathsf{C}(x_1, \ldots, x_n)) = x_i$
5. Reconstruct: $\forall x.\ \neg\mathsf{atom}_{\mathsf{C}}(x) \rightarrow \mathsf{C}(\pi_1^{\mathsf{C}}(x), \ldots, \pi_n^{\mathsf{C}}(x)) = x$
6. Constructor not an atom: $\forall x_1, \ldots, x_n.\ \neg\mathsf{atom}_{\mathsf{C}}(\mathsf{C}(x_1, \ldots, x_n))$

ACYCLIC LISTS AND ATOMS

- T^{+}_{cons} = acyclic version of T_{cons}.
  - Adds axioms like: car(x) ≠ x, cdr(cdr(x)) ≠ x, etc.
- Decidable, unlike full T_{cons}.
- Specified atoms: $T^{\text{atom}}_{\text{cons}}$.
  - Adds: $\forall x.\ \text{atom}(x) \rightarrow \text{atom}(\text{car}(x)) \wedge \text{atom}(\text{cdr}(x))$
  - Makes satisfiability NP-complete

LISTS WITH EQUALITY ($T_{\text{CONS}}^{=}$)

- $\boldsymbol{T_{\text{cons}} + T_{\text{E}} = T_{\text{cons}}^{=}}$ - Combines T_{cons} with T_{E}.
- More expressive: allows uninterpreted symbols
- Example proof:
  - Assume: car(a) = car(b), cdr(a) = cdr(b)
  - Then: a = b → f(a) = f(b) by congruence
  - So: F is valid under $T_{\text{cons}}^{=}$

FIRST ORDER THEORIES

First-Order Theories Symbols and axioms	**Equality** Rules of equality	**Natural Numbers and Integers** Logic over whole numbers
Rationals and Reals Logic over numeric values	**Recursive Data Structures** Logic for lists and trees	**Arrays** Logic for read/write memory

THEORY OF ARRAYS (T_{A})

What are arrays?

- Common data structure in programming
- Similar to functions, but **can be updated**
- T_{A} = Theory of Arrays

Signature:

- $a[i]$: read value at position i
- $a\langle i \lhd v\rangle$: write value v to position i
- $=$: equality predicate

FUNCTIONAL ARRAYS

- Arrays are treated like functions
- Example:
  - $a\langle i \triangleleft v\rangle$ is the new array
  - $a\langle i \triangleleft v\rangle[j] = \begin{cases} v, & j = i \\ a[j], & j \neq i \end{cases}$
- Multiple writes:
  - $a\langle i \triangleleft v\rangle\langle j \triangleleft w\rangle[k] = \begin{cases} w, & k = j \\ v, & k = i \land k \neq j \\ a[k], & otherwise \end{cases}$

AXIOMS OF $T_{\mathbf{A}}$

- **From** $T_{\mathbf{E}}$ **:** reflexivity, symmetry, transitivity
- **Array congruence:** $\forall a, i, j.\ i = j \ \rightarrow\ a[i] = a[j]$
- **Read-over-write 1:** $\forall a, v, i, j.\ i = j \ \rightarrow\ a\langle i \triangleleft v\rangle[j] = v$
- **Read-over-write 2:** $\forall a, v, i, j.\ i \neq j \ \rightarrow\ a\langle i \triangleleft v\rangle[j] = a[j]$

ARRAY EQUALITY ISSUE

Note: $=$ is only defined for array elements

$$F: \; a[i] = e \;\rightarrow\; a\langle i \triangleleft e\rangle = a$$

not $T_{\mathsf{A}} - valid$, but

$$F': \; a[i] = e \;\rightarrow\; \forall j.\; a\langle i \triangleleft e\rangle[j] = a[j] \;,$$

is $T_{\mathsf{A}} - valid$.

T_{A} is undecidable,
Quantifier-free fragment of T_{A} is decidable

EXTENDED THEORY $T_A^=$

Adds extensionality axiom:

$$\forall a, b.\ (\forall i.\ a[i] = b[i]) \ \leftrightarrow\ a = b$$

Now array equality is well-defined

Example: $F:\ a[i] = e\ \rightarrow\ a\langle i \triangleleft e\rangle = a$

is $T_{\mathrm{A}} - valid.$

DECIDABILITY TABLE

Theory	Description	Full	QFF
T_{E}	equality	no	yes
T_{PA}	Peano arithmetic	no	no
$T_{\mathbb{N}}$	Presburger arithmetic	yes	yes
$T_{\mathbb{Z}}$	linear integers	yes	yes
$T_{\mathbb{R}}$	reals (with $\cdot$)	yes	yes
$T_{\mathbb{Q}}$	rationals (without $\cdot$)	yes	yes
T_{RDS}	recursive data structures	no	yes
T^{+}_{RDS}	acyclic recursive data structures	yes	yes
T_{A}	arrays	no	yes
$T^{=}_{\mathsf{A}}$	arrays with extensionality	no	yes

* QFF = Quantifier-Free Fragment

COMPLEXITY TABLE

Theory	**Complexity**
PL	NP-complete
$T_{\mathbb{N}}$, $T_{\mathbb{Z}}$	$\Omega\left(2^{2^{n}}\right)$, $O\left(2^{2^{2^{kn}}}\right)$
$T_{\mathbb{R}}$	$O\left(2^{2^{kn}}\right)$
$T_{\mathbb{Q}}$	$\Omega\left(2^{n}\right)$, $O\left(2^{2^{kn}}\right)$
T_{RDS}^{+}	not elementary recursive

FINAL RECAP EXAMPLE

Back to Our Example

SATISFIABILITY AND VALIDITY

Recall our example:

```python
def g(ig):
    og = ig
    for i in range(2):
        og = f(og, ig)
    return og
```

```python
def h(ih):
    oh = f(f(ih,ih),ih)
    return oh
```

$$\psi_g: \ \ og1 = ig \wedge og2 = f(og1, ig) \wedge og3 = f(og2, ig)$$

$$\psi_h: \ \ oh1 = f(f(ih, ih), ih)$$

$$\psi \ \coloneqq \ \left(ih = ig \wedge \ \psi_h \wedge \psi_g\right) \longrightarrow og3 = oh1$$

SATISFIABILITY AND VALIDITY

$$\psi \ := \ (ih = ig \wedge \ \psi_h \wedge \psi_g) \longrightarrow og3 = oh1$$

For the equivalence of g and h, it's enough to show that ψ is T_E -valid

Suppose not, then there is an interpretation I such that $I \nvDash F$:

1. $I \vDash (ih = ig \wedge \ \psi_h \wedge \psi_g)$ assumption and semantics of $\longrightarrow$
2. $I \nvDash og3 = oh1$ assumption and semantics of $\longrightarrow$
3. $I \vDash (ih = ig)$ 1, and semantics of $\wedge$
4. $I \vDash (\ \psi_h)$ 1, and semantics of $\wedge$
5. $I \vDash (\psi_g)$ 1, and semantics of $\wedge$

SATISFIABILITY AND VALIDITY

$$\psi_g\colon\ og1 = ig \wedge og2 = f(og1, ig) \wedge og3 = f(og2, ig)$$

$$\psi_h\colon\ oh1 = f(f(ih, ih), ih)$$

6. $I \vDash (og1 = ig)$ — *5, and semantics of* $\wedge$
7. $I \vDash (og1 = ih)$ — *3, 6 and transitivity*
8. $I \vDash f(og1, ig) = f(ih, ih)$ — 7, 3 and Function congruence
9. $I \vDash og2 = f(og1, ig)$ — 5, and semantics of $\wedge$
10. $I \vDash og2 = f(ih, ih)$ — *8, 9 and transitivity*
11. $I \vDash f(og2, ig) = f(f(ih, ih), ih)$ — 10, 3 and Function congruence
12. $I \vDash og3 = f(og2, ig)$ — *5, and semantics of* $\wedge$

SATISFIABILITY AND VALIDITY

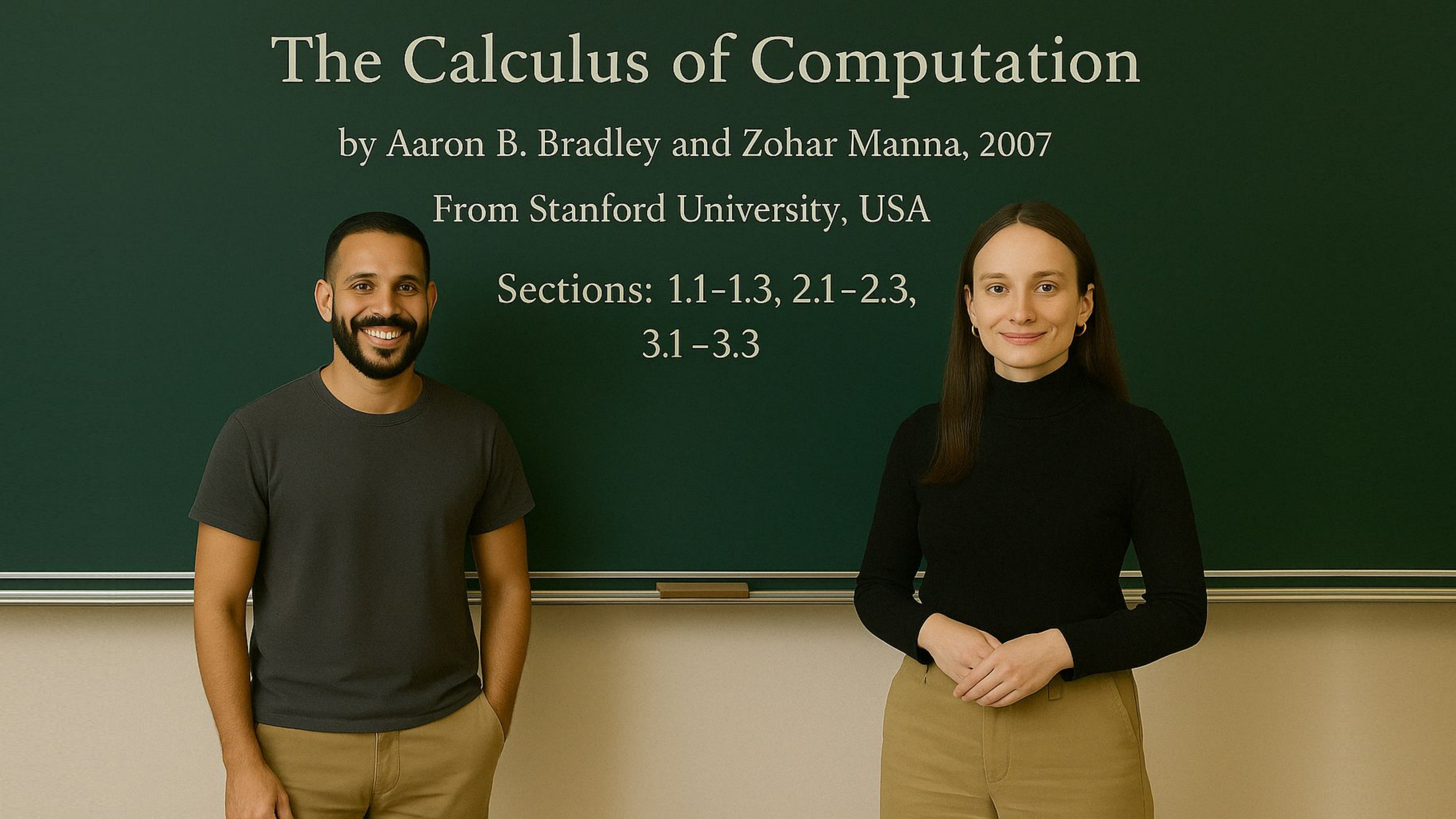
$$\psi_g: \ og1 = ig \wedge og2 = f(og1, ig) \wedge og3 = f(og2, ig)$$

$$\psi_h: \ oh1 = f(f(ih, ih), ih)$$

13. $I \vDash og3 = f(f(ih, ih), ih)$ *11, 12 and transitivity*
14. $I \vDash og3 = oh1$ *13, 4 and transitivity*

We found a contradiction between our result and our initial assumption, so we got that ψ is T_E -valid.

THANK YOU


The Calculus of Computation
by Aaron B. Bradley and Zohar Manna, 2007
From Stanford University, USA
Sections: 1.1-1.3, 2.1-2.3,
3.1-3.3