Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500
Fall 2004
Software Foundations

8 September

CIS 500, 8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September

Course Overview

Why Study Software Foundations?

What is "Software Foundations"?

CIS 500, 8 September

Software Foundations

Fall 2004

CIS 500

8 September
Whystudysoftwarefoundations?

To be able to prove specific facts about particular programs (i.e., program verification) difficult and expensive

Important in some domains (safety-critical systems, hardware design, verification)

To prove general facts about all the programs in a given programming language

difficult and expensive

Important in some domains (safety-critical systems, hardware design, verification)

To develop intuitions for informal reasoning about programs...
Why study software foundations?

To be able to prove specific facts about particular programs (e.g., program verification)

Important in some domains (e.g., security-critical systems, hardware design)

To prove general facts about all the programs in a given programming language

Founds the “material science” of computer science...

What this course is and is not

What you can expect to get out of the course

N.B.: Most good software designers are language designers

Powerful tools for language design, description, and analysis

Deep intuition about how programs work, and how to reason about them

How to prove programs and prove properties about them

A more sophisticated perspective on programs, programming languages, and the activity of programming

Typesystems...
Program meaning can be approached in many different ways.

Denotational semantics and domain theory view programs as simple mathematical objects, abstracting away their flow of control and concentrating on their input-output behavior.

Program logics such as Hoare logic and dependent type theories focus on systems of logical rules for reasoning about programs.

Operational semantics describes program behaviors by means of abstract machines. This approach is somewhat lower-level than the others, but is extremely flexible.

Process calculi focus on the communication and synchronization behaviors of complex concurrent systems.

Types systems describe approximations of program behaviors, concentrating on the shapes of the values passed between different parts of the program.
Approaches

"Program meaning" can be approached in many different ways. Let's take a look at a few of them:

- **Denotational semantics** and **domain theory** abstract away their flow of control and concentrating on their input-output behavior.

- **Operational semantics** describe program behaviors by means of abstract machines. This approach is somewhat low-level than the others, but is extremely flexible.

- **Program logics** such as **Hoare logic** and dependent type theories focus on systems of logical rules for reasoning about programs.

- **Process calculi** focus on the communication and synchronization behaviors of complex concurrent systems.

- **Types systems** describe approximations of program behaviors, concentrating on the shapes of the values passed between different parts of the program.

Overview

In this course, we will concentrate on operational techniques and type systems.

- **Part I: Background**
  - A taste of OCaml
  - Functional programming style
  - Denotational semantics and domain theory

- **Part II: Modelling programming languages**
  - Syntax and operational semantics
  - Inductive proof techniques
  - Operational logics such as Hoare logic and dependent type theories

- **Part III: Object-oriented features (case study)**
  - Simple types
  - Type safety
  - References
  - Subtyping
  - A simple imperative object model
  - An analysis of core Java

Administrative Stuff
Personnel

Send email all staff: cis500@cis.upenn.edu

Instructor: Stephanie Weirich
Levine 510
sweirich@cis.upenn.edu

Office hours today: Wed, 3:00-4:00

Office hours next week: Oct 13 and 15.

Guest lectures for next three lectures:

Temporary Personnel

I will be away from September 10 to September 22.

Administrative Assistant

Cheryl Hickey, Levine 502
If you are unable to reach me, please contact Cheryl Hickey, 215-898-3538 or cherylh@central.cis.upenn.edu.

You may find your class folder in the filing cabinet outside of Room 502 Levine for all graded homework and extra credit.

If you are unable to reach me, please contact Cheryl Hickey, 215-898-3538 or cherylh@central.cis.upenn.edu.

Teaching Assistants:

Nate Foster
Office hours: Thurs 4:30-5:30 in GRW 565

Dimitrios Vytiniotis
Office hours: Tues 2:00-3:00 in GRW 565

Information

Textbook: Types and Programming Languages, Benjamin C. Pierce, MIT Press, 2002

Webpage:
http://www.seas.upenn.edu/~cis500

Newsgroup:
penn.cis.cis500

Guest lectures for next three lectures:

Dr. Benjamin Pierce, September 10.

Dr. Val Tannen, September 15.

Dr. Ben Benjamin, September 22.

Teaching Assistants:

Nate Foster
Office hours: Thurs 4:30-5:30 in GRW 565

Dimitrios Vytiniotis
Office hours: Tues 2:00-3:00 in GRW 565

Office hours:
Wed, 2:00-6:00 and Thu, 10:00-2:00

Office hours location:
Levine 110

Administrative Assistant:
Cheryl Hickey, Levine 502

Send email all staff: cis500@cis.upenn.edu
**Exams**

1. First mid-term: Wed. October 13
2. Second mid-term: Wed. November 15
3. Final: TBA

Additional administrative information will be posted as necessary during the semester. Keep an eye on the course webpage and (especially) the newsgroup.

**Final exam**

The final for this course has been scheduled by the registrar for Thurs. 12/16/04, 8:30-10:30. The final for CIS 501 has been scheduled by the registrar for Thurs. 12/16/04, 11-1.

Additional administrative information will be posted as necessary during the semester. Keep an eye on the course webpage and (especially) the newsgroup.

**Grading**

Final course grades will be computed as follows:
- Homework: 20%
- 2 midterms: 20% each
- Final: 40%

Final exam can be rescheduled by a petition signed by every registered student. I'll pass the petition around after the add period is over (Friday, Sept. 24).

If you have constraints, other than CIS courses, send them to cis500@cis.upenn.edu.

CIS 500, 8 September
Extra Credit

Course grades can be improved after the semester ends in two ways:

1. A 1/3 letter grade improvement can be obtained by doing a substantial extra credit project during the Spring semester.

2. Larger grade improvements can only be obtained by doing a substantial extra credit project (~30 hours of work) during the Spring semester.

Collaboration

Collaboration on homework is strongly encouraged.

— Anon.

Homework

Readings from TAPL should be completed before lecture (see course webpage).

Written homework is due in class or recitation.

All assignments must be typeset and submitted electronically. The use of LaTeX is strongly encouraged.

All assignments are posted on the course web page.

Written homework is due a week from Monday, September 20.

1. Text is strongly encouraged.

2. Larger grade improvements can only be obtained by solving a substantial extra credit project (~30 hours of work) during the Spring semester.

Homework

Homework will not be accepted after the announced deadline.

Late (non)-policy: Homework will not be accepted after the announced deadline.

Deadline:

Deadline: before looking

Deadline: reading in the back of the book. Write your answer down.

Deadline: reading in the back of the book. We may not grade every problem.

Some solutions are in the back of the book. If you find this assignment easy, you

will make a group for extra assignment.

The first assignment is due. Even if you find that assignment easy, you

should be completing before recitation.

Small part of your grade, get a large part of your understanding.

First homework assignment is due a week from Monday, September 20.

First homework assignment is due a week from Monday, September 20.

Assignment is posted on the course web page.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.

Assignment is due a week from Monday, September 20.
Recitations

Everyone in the class should attend one of the recitation sections:

There are two kinds of recitation sections:

1. Review sections will focus on material close to what is presented in class and on homeworks.
2. Advanced sections will introduce additional related material.

Recitations will meet as follows:

- Wed 3:30-4:00 PM, DRBL 213
- Thurs 1:30-2:30 PM, DRBL 4C2
- Fri 9:30-10:30 AM, Towne 307
- Thurs 10:30-11:30 AM, Towne 307
- Review
- Thurs 1:30-2:30 PM, Towne 307
- Thurs 10:30-11:30 AM, Towne 307
- Review

You do not need to be enrolled in the course to take the exam for WPE credit.

You may take the exam for WPE credit even if you are not currently enrolled in the PhD program.

Pass/Fail for the WPE exam.

You will receive two grades: a letter grade for the course and a Pass/Fail for the WPE exam.

The WPE-I syllabus

Review sections will focus on material close to what is presented in class and on homeworks.

Reading knowledge of core OCaml

WPE-I reading list:

- Chapters 1-11 and 13-19 of TAPL
- Reading knowledge of core OCaml

The WPE-I syllabus (continued)

You do not need to be enrolled in the course to take the exam for WPE credit.

If you are enrolled in the course and also take the exam for WPE credit,

You may take the exam for WPE credit even if you are not currently enrolled in the PhD program.

Pass/Fail for the WPE exam.

You will receive two grades: a letter grade for the course and a Pass/Fail for the WPE exam.

The WPE-I syllabus

Review sections will focus on material close to what is presented in class and on homeworks.

Reading knowledge of core OCaml

WPE-I reading list:

- Chapters 1-11 and 13-19 of TAPL
- Reading knowledge of core OCaml

The WPE-I syllabus (continued)
Announcement

The department is hosting a Faculty Research Seminar most weeks during the Fall semester.

First-year CIS PhD students are required to attend. Others are welcome.

Speakers and topics are announced on the CIS newsletter.

Friday afternoons, 3:30 – 4:30, in Levine Auditorium

What is a programming language?

Deﬁning a programming language

Syntax

Here is a BNF grammar for a very simple language of boolean expressions:

```
t ::= terms
true
constanttrue
false
constantfalse
not t
cond t
if t then t else t
```

Terminology:

condition
negation
constant false
constant true
cond t
if t then t else t
not t
false
true
Another form of the definition

The set $B$ of boolean terms is the smallest set such that

1. $(true, false) \in B$;
2. If $t_1, t_2, t_3, t_4, t_5 \in B$, then $(not(t)) \in B$;
3. If $t_1, t_2, t_3, t_4, t_5, t_6 \in B$, then if $t_1$ then $t_2$ else $t_3$ \in B.

Q1: Does this grammar define a set of character strings, a set of token lists, or a set of abstract syntax trees?

The set $B$ of boolean terms is the smallest set such that

1. $(true, false) \in B$;
2. If $t_1, t_2, t_3, t_4, t_5 \in B$, then $(not(t)) \in B$;
3. If $t_1, t_2, t_3, t_4, t_5, t_6 \in B$, then if $t_1$ then $t_2$ else $t_3$ \in B.

Q1: Does this grammar define a set of character strings, a set of token lists, or a set of abstract syntax trees?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?

Abstract vs. concrete syntax

Q: So, are $not$ $false$ $not (false)$ $(((not (((((false)))))))))$ the same term?
Abstract Syntax, not Semantics

We've only defined the abstract syntax of our language. That means our language is just a set of terms. We haven't assigned any meanings to those terms yet. So there is no reason to doubt the truth of not false, they are just uninterpreted terms.

Soon we will start talking about how we can decide what those terms mean.

In this course we will concentrate on operational semantics.

1. Operational Semantics describes the meaning of a program through laws that describe the behavior of programs, much like an interpreter.
2. Denotational Semantics translates programs into a domain that we already know the meaning of: mathematics. The meaning of a term is a mathematical object like a function.
3. Axiomatic Semantics describes the meaning of a program through laws that describe its behavior.

In this course we will concentrate on operational semantics.

Operational Semantics

Operational Semantics describe the evaluation of programs on an abstract machine. Defined by a relation between each program and its result of evaluation.

Methods of Semantics

1. Operational Semantics describe the behavior of programs, much like an interpreter.
2. Denotational Semantics translates programs into a domain that we already know the meaning of: mathematics.
3. Axiomatic Semantics describe the meaning of a program through laws that describe its behavior.

As well as defining the syntax of a programming language, we also need to define what a language "means". We've only defined the abstract syntax of our language. That means our language is just a set of terms. Soon we will start talking about how we can decide what those terms mean.
It is the smallest set such that:

\[ \text{read is a relation between terms in } b. \]

\[ \text{eval} \]

Operational Semantics

If \( \text{false, false} \in \text{read} \) then:

\[ \text{eval} ; \]
Operational Semantics

Eval is a relation between terms in $B$. It is the smallest set such that:

- $(\text{true}, \text{true}) \in \text{Eval}$
- $(\text{false}, \text{false}) \in \text{Eval}$
- $(\neg \text{true}, \text{true}) \in \text{Eval}$
- $(\neg \text{false}, \text{false}) \in \text{Eval}$
- $(\text{if} \ t_1 \ \text{then} \ t_2 \ \text{else} \ t_3, t) \in \text{Eval}$ when either:
  - $(t_1; \text{true}) \in \text{Eval}$ and $(t_2; t) \in \text{Eval}$
  - $(t_1; \text{false}) \in \text{Eval}$ and $(t_3; t) \in \text{Eval}$

Now, let's define the syntax and semantics of the boolean language.

Properties of boolean language

How do we show that these properties are true?

- There is only one meaning for each term (Eval is deterministic).
- All boolean terms have meanings (Eval is total).
- Not false and true have the same meaning.
- $(\text{true}, \text{true}) \notin \text{Eval}$

Operational Semantics

Eval is a relation between terms in $B$. It is the smallest set such that:

- $(\text{true}, \text{true}) \in \text{Eval}$
- $(\text{false}, \text{false}) \in \text{Eval}$
- $(\neg \text{true}, \text{true}) \in \text{Eval}$
- $(\neg \text{false}, \text{false}) \in \text{Eval}$
- $(\text{if} \ t_1 \ \text{then} \ t_2 \ \text{else} \ t_3, t) \in \text{Eval}$ when either:
  - $(t_1; \text{true}) \in \text{Eval}$ and $(t_2; t) \in \text{Eval}$
  - $(t_1; \text{false}) \in \text{Eval}$ and $(t_3; t) \in \text{Eval}$

If $(t_1, t_2) \in \text{Eval}$, we say that $t_2$ is the meaning of $t_1$.
We want to show that a property \( P \) is true for all \( n \in \mathbb{N} \).

**Example:** Natural number induction

**Theorem:**

Suppose that \( P \) is a predicate on the natural numbers. Then:

1. \( P(0) \) holds for all \( n \in \mathbb{N} \).
2. For all \( n \in \mathbb{N} \), \( P(n) \) implies \( P(n+1) \).

Let \( d(n) \) be \( 2^n \).

**Proof:**

Theorem: \( 2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1 \) for every \( n \).

**Example:**

For shorthand, we sometimes abbreviate \( 0 + 1 + 1 + \ldots + 1 \) as \( 3 \), etc.

The set \( \mathbb{N} \) is the smallest set such that:

1. \( 0 \in \mathbb{N} \).
2. If \( n \in \mathbb{N} \), then \( n + 1 \in \mathbb{N} \).

The reason that we have an induction principle for natural numbers is because they are defined in a certain way:

Natural numbers
Example

Theorem: \(2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1\), for every \(n\).

Proof:
- Let \(P(i)\) be "\(2^0 + 2^1 + \ldots + 2^i = 2^{i+1} - 1\)".
- Show that \(P(i)\) implies \(P(i+1)\):
  - \(2^0 + 2^1 + \ldots + 2^i + 2^{i+1} = (2^{i+1} + 1) + 2^{i+1} - 1 = 2^{i+2} - 1\)
  - by IH

Example

Theorem: \(2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1\), for every \(n\).

Proof:
- Let \(P(i)\) be "\(2^0 + 2^1 + \ldots + 2^i = 2^{i+1} - 1\)".
- Show that \(P(i)\) implies \(P(i+1)\):
  - \(2^0 + 2^1 + \ldots + 2^i + 2^{i+1} = (2^{i+1} + 1) + 2^{i+1} - 1 = 2^{i+2} - 1\)
  - by IH

Example

Theorem: \(2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1\), for every \(n\).

Proof:
- Let \(P(i)\) be "\(2^0 + 2^1 + \ldots + 2^i = 2^{i+1} - 1\)".
- Show that \(P(i)\) implies \(P(i+1)\):
  - \(2^0 + 2^1 + \ldots + 2^i + 2^{i+1} = (2^{i+1} + 1) + 2^{i+1} - 1 = 2^{i+2} - 1\)
  - by IH

Example

Theorem: \(2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1\), for every \(n\).

Proof:
- Let \(P(i)\) be "\(2^0 + 2^1 + \ldots + 2^i = 2^{i+1} - 1\)".
- Show that \(P(i)\) implies \(P(i+1)\):
  - \(2^0 + 2^1 + \ldots + 2^i + 2^{i+1} = (2^{i+1} + 1) + 2^{i+1} - 1 = 2^{i+2} - 1\)
  - by IH

Shorthand form

Theorem: \(2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1\), for every \(n\).

Proof:
- Base case (\(n = 0\)): \(2^0 - 1 = 1 - 1\)
- Inductive case (\(n - 1 + 1\)):
  - \(2^0 + 2^1 + \ldots + 2^n = (2^n + 1) + 2^{n+1} - 1 = 2^{n+2} - 1\)

The result \(P(n)\) for all \(n\) follows by the principle of induction.
Inductive Definitions

This is the same way we defined what boolean terms were. The set $B$ is the smallest set such that:

1. If $t \in \{true, false\}$, then $t \in B$.
2. If $t \in B$, then $\neg t \in B$.
3. If $t_1, t_2 \in B$, and $t_1 \neg t_2 \in B$.

We can also use induction for boolean terms. The way we defined lets us use induction principle.

Structural Induction

Wecanalsouseinductionforbooleanterms.Thewaywehavedefinedtermsgivesusaninductionprinciple:

For all $t \in B$, $P(t)$ is true if and only if:

1. $\neg t \in B$, $true \in B$, and $false \in B$.
2. $t_1 \neg t_2 \in B$, $true \in B$, and $false \in B$.
3. $t_1, t_2 \in B$, and $t_1 \neg t_2 \in B$.

This is the same way we defined what boolean terms were.

Proofs by Induction

We'll proventhatevaluationisdeterminist.Inotherwords:For all $t$ there exists at most one $t_0$ such that $(t; t_0) \in Eval$.

This givesustheproperty:

$P(t) = \exists t_0 \text{ s.t. } (t; t_0) \in Eval$.

So we want to show:

$P(true) = \exists t_0 \text{ s.t. } (true; t_0) \in Eval$.

We prove this evaluation is deterministic. In other words: For all $t$ there exists at most one $t_0$ such that $(t; t_0) \in Eval$. Inductive Definitions