

Administrivia • Recitations begin (in fact, have already begun!) this week. • Section E (Tue 3:00 - 4:30) will meet in the IRCS "back conference room," room 413 in the 3401 Walnut building (above Starbucks). Check the course web page for detailed directions. • If you are not sure which recitation you are in, email bracy@gradient.

CIS 500, 16 September

Plan for this week

We've seen enough of OCaml for now. (If time permits, we'll come back later to some more advanced programming idioms later in the course.) Time to get started with the book.

General plan:

- discuss material in class [This week: induction, basic operational semantics]
- read in book [Chapters 3 and 4]
- homework assignment [covering chapters 3 and 4; handed out Thu, due next Thu]
- discuss in recitation
- finish off homework assignment

Reading Assignment

ASAP: Read Preface and Chapter 1 and skim Chapter 2 of TAPL on your own.

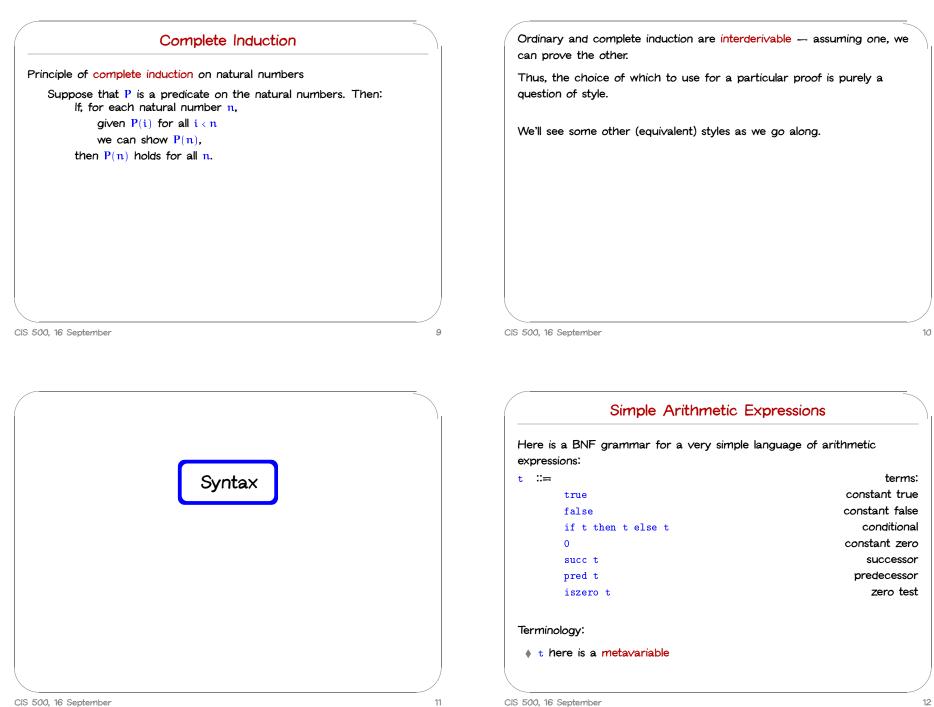
By Thursday or Friday: Read Chapters 3 and 4.

CIS 500, 16 September

3

1

Review	Principle of ordinary induction on natural numbers Suppose that P is a predicate on the natural numbers. Then: If $P(0)$ and, for all i, $P(i)$ implies $P(i + 1)$, then $P(n)$ holds for all n.		
500, 16 September 5	CIS 500, 16 September		
Example	Shorthand form		
·	$\label{eq:Shorthand form} \hline$ Theorem: $2^{\circ}+2^{1}++2^{n}=2^{n+1}-1,$ for every $n.$		
heorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. roof: • Let P(i) be " $2^{0} + 2^{1} + + 2^{i} = 2^{i+1} - 1$."			
Theorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. Proof: • Let P(i) be " $2^{0} + 2^{1} + + 2^{i} = 2^{i+1} - 1$." • Show P(0): $2^{0} = 1 = 2^{1} - 1$	Theorem: $2^0 + 2^1 + + 2^n = 2^{n+1} - 1$, for every n. Proof: By induction on n. \blacklozenge Base case (n = 0):		
Theorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. Proof: • Let P(i) be " $2^{0} + 2^{1} + + 2^{i} = 2^{i+1} - 1$." • Show P(0):	Theorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. Proof: By induction on n. \bullet Base case (n = 0): $2^{0} = 1 = 2^{1} - 1$		
Theorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. Proof: • Let P(i) be " $2^{0} + 2^{1} + + 2^{i} = 2^{i+1} - 1$." • Show P(0): $2^{0} = 1 = 2^{1} - 1$ • Show that P(i) implies P(i + 1): $2^{0} + 2^{1} + + 2^{i+1} = (2^{0} + 2^{1} + + 2^{i}) + 2^{i+1}$ $= (2^{i+1} - 1) + 2^{i+1}$ IH $= 2 \cdot (2^{i+1}) - 1$	Theorem: $2^{0} + 2^{1} + + 2^{n} = 2^{n+1} - 1$, for every n. Proof: By induction on n. • Base case (n = 0): $2^{0} = 1 = 2^{1} - 1$ • Inductive case (n = i + 1): $2^{0} + 2^{1} + + 2^{i+1} = (2^{0} + 2^{1} + + 2^{i}) + 2^{i+1}$ $= (2^{i+1} - 1) + 2^{i+1}$ IH $= 2 \cdot (2^{i+1}) - 1$		



CIS 500, 16 September

Abstract vs. concrete syntax

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

CIS 500, 16 September

13

Abstract vs. concrete syntax

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

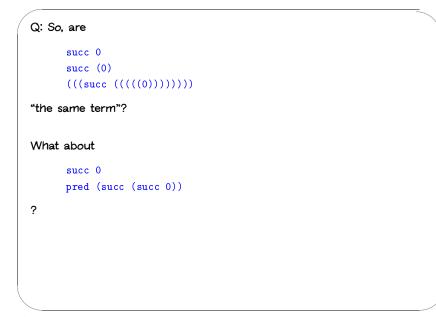
A: In a sense, all three. But we are primarily interested, here, in abstract syntax trees.

For this reason, grammars like the one on the previous slide are sometimes called abstract grammars. An abstract grammar defines a set of abstract syntax trees and suggests a mapping from character strings to trees.

We then write terms as character strings rather than trees simply for convenience. If there is any potential confusion about what tree is intended, we use parens to disambiguate.

CIS 500, 16 September

13-a



A more explicit form of the definition

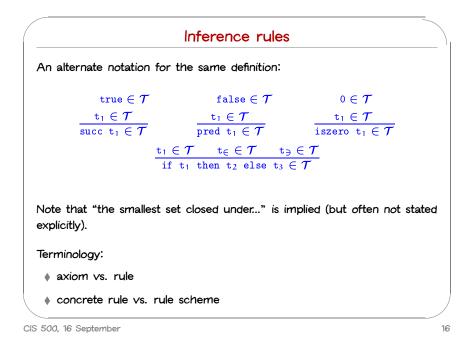
The set of terms is the smallest set T such that

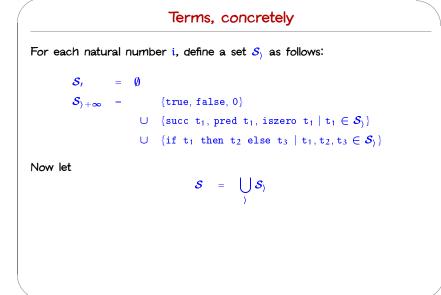
1. {true, false, 0} $\subseteq \mathcal{T}$;

2. if $t_1 \in \mathcal{T}$, then {succ t_1 , pred t_1 , iszero t_1 } $\subseteq \mathcal{T}$;

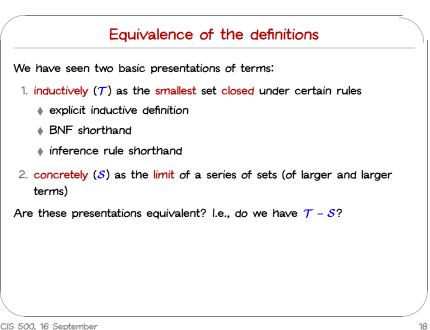
3. if $t_1 \in \mathcal{T}$, $t_2 \in \mathcal{T}$, and $t_3 \in \mathcal{T}$, then if t_1 then t_2 else $t_3 \in \mathcal{T}$.

CIS 500, 16 September





CIS 500, 16 September



Inductive Function Definitions				
The set of constants appearing as follows:	in a	term t, written $Consts(t)$, is defined		
Consts (true)	=	{true}		
Consts (false)	=	{false}		
Consts(0)	=	{0}		
Consts(succ t1)	=	Consts(t1)		
Consts (pred t ₁)	=	$Consts(t_1)$		
Consts (iszero t ₁)	=	Consts(t1)		
Consts (if t_1 then t_2 else t_3)	=	$Consts(t_1) \cup Consts(t_2) \cup Consts(t_3)$		

Simple, right?

First question: In what sense is this a "definition"?

(Normally, a "definition" just assigns a convenient name to a previously-known thing. But here, the "thing" on the right-hand side involves the very name that we are "defining"!)

CIS 500, 16 September

20

Second question: Suppose we had written this instead...

The set of constants appearing in a term ${\tt t}$, written ${\tt BadConsts}({\tt t}),$ is defined as follows:

BadConsts (true)	=	{true}
$\pmb{BadConsts}(\mathtt{false})$	=	{false}
BadConsts(0)	=	{0}
BadConsts(0)	=	{}
$BadConsts(succ t_1)$	=	$BadConsts(t_1)$
${\color{black}{\textbf{BadConsts}}}({\color{black}{pred t_1}})$	=	$BadConsts(t_1)$
${\color{black}{\textbf{BadConsts}}}({\color{black}{\texttt{iszero t}_1}})$	=	$BadConsts(iszero (iszero t_1))$

CIS 500, 16 September

21

What is the essential difference between these two definitions? How do we tell the difference between well-formed inductive definitions and ill-formed ones?

What, exactly, does a well-formed inductive definition mean?

First, recall that a function is just a two-place relation with certain properties:

- It is total: every element of its domain occurs at least once in its "graph"
- It is deterministic: every element of its domain occurs at mostd once in its graph.

CIS 500, 16 September

We have seen how to define relations inductively. E.g	This definition certainly defines a relation (i.e., the smallest one with
Let Consts be the smallest two-place relation closed under the following rules:	certain property). How can we tell that this relation is a function?
$(\texttt{true}, \{\texttt{true}\}) \in \textbf{Consts}$	
$(false, \{false\}) \in Consts$	
(0, {0}) ∈ Consts	
$\frac{(\texttt{t}_1, \texttt{C}) \in \texttt{Consts}}{(\texttt{succ t}_1, \texttt{C}) \in \texttt{Consts}}$	
$\frac{(\texttt{t}_1, \texttt{C}) \in \texttt{Consts}}{(\texttt{pred t}_1, \texttt{C}) \in \texttt{Consts}}$	
$\frac{(t_1, C) \in Consts}{(iszero t_1, C) \in Consts}$	
$(t_1, C_1) \in Consts$ $(t_2, C_2) \in Consts$ $(t_3, C_3) \in Consts$	

CIS 500, 16 September

Prove it!

This definition certainly defines a relation (i.e., the smallest one with a certain property). How can we tell that this relation is a function?

24

Theorem: The relation Consts defined by the inference rules a couple of slides ago is total and deterministic.

Proof: (Exercise.)

CIS 500, 16 September

CIS 500, 16 September



[values]		[concrete definitions for booleans]	
5 500, 16 September	31	CIS 500, 16 September	
[Proof trees		[Normal forms.	
Inference rules as an inductive de	finition of valid proof tion)]	Theorem: normal forms = values.]	

[Multi-step evaluation.]

CIS 500, 16 September