# Formal Methods

## Lecture 4

### (B. Pierce's slides for the book "Types and Programming Languages")

# Going Meta…

The functional programming style used in OCaml is based on treating *programs as data* — i.e., on writing functions that manipulate other functions as their inputs and outputs.

Everything in this course is based on treating *programs as mathematical objects* — i.e., we will be building mathematical theories whose basic objects of study are programs (and whole programming languages).

Jargon: We will be studying the *metatheory* of programming languages.

# Basics of Induction
# (Review)

# Induction

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* $n$.

# 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 $P(0)$:

$$2^0 = 1 = 2^1 - 1$$

- Show that $P(i)$ implies $P(i+1)$:

$$\begin{aligned} 2^0 + 2^1 + \ldots + 2^{i+1} &= (2^0 + 2^1 + \ldots + 2^i) + 2^{i+1} \\ &= (2^{i+1} - 1) + 2^{i+1} \\ &= 2 \cdot (2^{i+1}) - 1 \\ &= 2^{i+2} - 1 \end{aligned}$$

- The result ($P(n)$ for all $n$) follows by the principle of (ordinary) induction.

# Shorthand form

Theorem: $2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$, for every $n$. Proof: By induction on $n$.

- Base case ($n = 0$):

$$2^0 = 1 = 2^1 - 1$$

- Inductive case ($n = i + 1$):

$$\begin{aligned} 2^0 + 2^1 + \ldots + 2^{i+1} &= (2^0 + 2^1 + \ldots + 2^i) + 2^{i+1} \\ &= (2^{i+1} - 1) + 2^{i+1} \\ &= 2 \cdot (2^{i+1}) - 1 \\ &= 2^{i+2} - 1 \end{aligned}$$

# Complete Induction

Principle of *complete induction* on natural numbers:

*Suppose that $P$ is a predicate on the natural numbers.*
*Then:*

*If, for each natural number $n$,*

*given $P(i)$ for all $i < n$*
*we can show $P(n)$,*

*then $P(n)$ holds for all $n$.*

# Complete versus ordinary induction

Ordinary and complete induction are *interderivable* — assuming one, we can prove the other.

Thus, the choice of which to use for a particular proof is purely a question of style.

# Syntax

# Simple Arithmetic Expressions

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

| | | |
|---|---|---|
| `t ::=` | | *terms* |
| | `true` | *constant true* |
| | `false` | *constant false* |
| | `if t then t else t` | *conditional* |
| | `0` | *constant zero* |
| | `succ t` | *successor* |
| | `pred t` | *predecessor* |
| | `iszero t` | *zero test* |

Terminology:

- `t` here is a *metavariable*

# Abstract vs. concrete syntax

Q: 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: 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 linear character strings rather than trees simply for convenience. If there is any potential confusion about what tree is intended, we use parentheses to disambiguate.

Q: So, are

```
succ 0
succ (0)
(((succ (((((0))))))))
```

“the same term”?

What about

```
succ 0
pred (succ (succ 0))
```

?

# A more explicit form of the definition

The set $\mathcal{T}$ of *terms* is the smallest set such that

1. $\{\texttt{true}, \texttt{false}, \texttt{0}\} \subseteq \mathcal{T}$;
2. if $\texttt{t}_1 \in \mathcal{T}$, then $\{\texttt{succ t}_1, \texttt{pred t}_1, \texttt{iszero t}_1\} \subseteq \mathcal{T}$;
3. if $\texttt{t}_1 \in \mathcal{T}$, $\texttt{t}_2 \in \mathcal{T}$, and $\texttt{t}_3 \in \mathcal{T}$, then $\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in \mathcal{T}$.

# Inference rules

An alternate notation for the same definition:

$$\mathtt{true} \in \mathcal{T} \qquad \mathtt{false} \in \mathcal{T} \qquad \mathtt{0} \in \mathcal{T}$$

$$\frac{t_1 \in \mathcal{T}}{\mathtt{succ}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\mathtt{pred}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\mathtt{iszero}\ t_1 \in \mathcal{T}}$$

$$\frac{t_1 \in \mathcal{T} \qquad t_2 \in \mathcal{T} \qquad t_3 \in \mathcal{T}}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \in \mathcal{T}}$$

Note that "the smallest set closed under..." is implied (but often not stated explicitly).

Terminology:

- axiom vs. rule
- concrete rule vs. rule scheme

# Terms, concretely

Define an infinite sequence of sets, $S_0$, $S_1$, $S_2$, . . . , as follows:

$$
\begin{aligned}
S_0 &= \varnothing \\
S_{i+1} &= \{\text{true}, \text{false}, 0\} \\
&\cup \{\text{succ } t_1, \text{pred } t_1, \text{iszero } t_1 \mid t_1 \in S_i\} \\
&\cup \{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \mid t_1, t_2, t_3 \in S_i\}
\end{aligned}
$$

Now let

$$S = \bigcup S_i$$

# Comparing the definitions

We have seen two different presentations of terms:

1. as the *smallest* set that is *closed* under certain rules ($\mathcal{T}$)
   - explicit inductive definition
   - BNF shorthand
   - inference rule shorthand
2. as the *limit* ($\mathcal{S}$) of a series of sets (of larger and larger terms)

What does it mean to assert that "these presentations are equivalent"?

# Why two definitions?

The two ways of defining the set of terms are both useful:

1. the definition of terms as the smallest set with a certain closure property is compact and easy to read
2. the definition of the set of terms as the limit of a sequence gives us an *induction principle* for proving things about terms...

# Induction on Syntax

# Induction on Terms

*Definition:* The *depth* of a term $t$ is the smallest $i$ such that $t \in S_i$.

From the definition of $S$, it is clear that, if a term $t$ is in $S_i$, then all of its immediate subterms must be in $S_{i-1}$, i.e., they must have strictly smaller depths.

This observation justifies the *principle of induction on terms*. Let $P$ be a predicate on terms.

> *If, for each term* $s$,
> *given* $P(r)$ *for all immediate subterms* $r$ *of* $s$
> *we can show* $P(s)$,
> *then* $P(t)$ *holds for all* $t$.

# Inductive Function Definitions

The set of constants appearing in a term $t$, written *Consts*($t$), is defined as follows:

$$
\begin{array}{lcl}
\mathit{Consts}(\mathtt{true}) & = & \{\mathtt{true}\} \\
\mathit{Consts}(\mathtt{false}) & = & \{\mathtt{false}\} \\
\mathit{Consts}(\mathtt{0}) & = & \{\mathtt{0}\} \\
\mathit{Consts}(\mathtt{succ}\ t_1) & = & \mathit{Consts}(t_1) \\
\mathit{Consts}(\mathtt{pred}\ t_1) & = & \mathit{Consts}(t_1) \\
\mathit{Consts}(\mathtt{iszero}\ t_1) & = & \mathit{Consts}(t_1) \\
\mathit{Consts}(\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3) & = & \mathit{Consts}(t_1) \cup \mathit{Consts}(t_2) \\
 & & \cup \mathit{Consts}(t_3)
\end{array}
$$

Simple, right?

*First question:*

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"!

So in what sense is this a definition??

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

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

$$
\begin{array}{lcl}
\mathit{BadConsts}(\mathtt{true}) & = & \{\mathtt{true}\} \\
\mathit{BadConsts}(\mathtt{false}) & = & \{\mathtt{false}\} \\
\mathit{BadConsts}(0) & = & \{0\} \\
\mathit{BadConsts}(0) & = & \{\} \\
\mathit{BadConsts}(\mathtt{succ}\ t_1) & = & \mathit{BadConsts}(t_1) \\
\mathit{BadConsts}(\mathtt{pred}\ t_1) & = & \mathit{BadConsts}(t_1) \\
\mathit{BadConsts}(\mathtt{iszero}\ t_1) & = & \mathit{BadConsts}(\mathtt{iszero}\ (\mathtt{iszero}\ t_1))
\end{array}
$$

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?

# What is a function?

Recall that a *function* $f$ from $A$ (its domain) to $B$ (its co-domain) can be viewed as a two-place *relation* (called the "graph" of the function) with certain properties:

- It is *total*: Every element of its domain occurs at least once in its graph. More precisely:
  *For every* $a \in A$*, there exists some* $b \in B$ *such that* $(a, b) \in f$ .

- It is *deterministic*: every element of its domain occurs at most once in its graph. More precisely:
  *If* $(a, b_1) \in f$ *and* $(a, b_2) \in f$ *, then* $b_1 = b_2$.

We have seen how to define relations inductively. E.g....
Let *Consts* be the smallest two-place relation closed under the following rules:

$$(\mathrm{true}, \{\mathrm{true}\}) \in \mathit{Consts}$$

$$(\mathrm{false}, \{\mathrm{false}\}) \in \mathit{Consts}$$

$$(0, \{0\}) \in \mathit{Consts}$$

$$\frac{(\mathrm{t}_1, C) \in \mathit{Consts}}{(\mathrm{succ}\ \ \mathrm{t}_1, C) \in \mathit{Consts}}$$

$$\frac{(\mathrm{t}_1, C) \in \mathit{Consts}}{(\mathrm{pred}\ \ \mathrm{t}_1, C) \in \mathit{Consts}}$$

$$\frac{(\mathrm{t}_1, C) \in \mathit{Consts}}{(\mathrm{iszero}\ \ \mathrm{t}_1, C) \in \mathit{Consts}}$$

$$\frac{(\mathrm{t}_1, C_1) \in \mathit{Consts} \qquad (\mathrm{t}_2, C_2) \in \mathit{Consts} \qquad (\mathrm{t}_3, C_3) \in \mathit{Consts}}{(\mathrm{if}\ \ \mathrm{t}_1\ \mathrm{then}\ \mathrm{t}_2\ \mathrm{else}\ \mathrm{t}_3, (\mathit{Consts}(\mathrm{t}_1) \cup \mathit{Consts}(\mathrm{t}_2) \cup \mathit{Consts}(\mathrm{t}_3))) \in \mathit{Consts}}$$

This definition certainly defines a *relation* (i.e., the smallest one with a certain closure property).

Q: How can we be sure that this relation is a *function*?

A: *Prove it!*

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

I.e., for each term $t$ there is exactly one set of terms $C$ such that $(t, C) \in Consts$.

Proof: By induction on $t$.
To apply the induction principle for terms, we must show, for an arbitrary term $t$, that if

> for each immediate subterm $s$ of $t$, there is exactly one set of terms $C_s$ such that $(s, C_s) \in Consts$

then

> there is exactly one set of terms $C$ such that $(t, C) \in Consts$.

Proceed by cases on the form of $\mathtt{t}$.

- If $\mathtt{t}$ is $0$, $\mathtt{true}$, or $\mathtt{false}$, then we can immediately see from the definition of *Consts* that there is exactly one set of terms $C$ (namely $\{\mathtt{t}\}$) such that $(\mathtt{t}, C) \in Consts$.

- If $\mathtt{t}$ is $\mathtt{succ}\ \mathtt{t}_1$, then the induction hypothesis tells us that there is exactly one set of terms $C_1$ such that $(\mathtt{t}_1, C_1) \in Consts$. But then it is clear from the definition of *Consts* that there is exactly one set $C$ (namely $C_1$) such that $(\mathtt{t}, C) \in Consts$.

  Similarly when $\mathtt{t}$ is $\mathtt{pred}\ \mathtt{t}_1$ or $\mathtt{iszero}\ \mathtt{t}_1$.

- If $t$ is $\texttt{if } s_1 \texttt{ then } s_2 \texttt{ else } s_3$, then the induction hypothesis tells us

  - there is exactly one set of terms $C_1$ such that $(t_1, C_1) \in Consts$
  - there is exactly one set of terms $C_2$ such that $(t_2, C_2) \in Consts$
  - there is exactly one set of terms $C_3$ such that $(t_3, C_3) \in Consts$

But then it is clear from the definition of $Consts$ that there is exactly one set $C$ (namely $C_1 \cup C_2 \cup C_3$) such that $(t, C) \in Consts$.

How about the bad definition?

$$(\mathtt{true}, \{\mathtt{true}\}) \in \mathit{BadConsts}$$

$$(\mathtt{false}, \{\mathtt{false}\}) \in \mathit{BadConsts}$$

$$(0, \{0\}) \in \mathit{BadConsts}$$

$$(0, \{\}) \in \mathit{BadConsts}$$

$$\frac{(\mathtt{t}_1, C) \in \mathit{BadConsts}}{(\mathtt{succ}\ \mathtt{t}_1, C) \in \mathit{BadConsts}}$$

$$\frac{(\mathtt{t}_1, C) \in \mathit{BadConsts}}{(\mathtt{pred}\ \mathtt{t}_1, C) \in \mathit{BadConsts}}$$

$$\frac{(\mathtt{iszero}\ (\mathtt{iszero}\ \mathtt{t}_1), C) \in \mathit{BadConsts}}{(\mathtt{iszero}\ \mathtt{t}_1, C) \in \mathit{BadConsts}}$$

This set of rules defines a perfectly good *relation* — it's just that this relation does not happen to be a function!

Just for fun, let's calculate some cases of this relation...

- For what values of $C$ do we have $(\mathtt{false}, C) \in \mathit{BadConsts}$?
- For what values of $C$ do we have $(\mathtt{succ\ 0}, C) \in \mathit{BadConsts}$?
- For what values of $C$ do we have $(\mathtt{if\ false\ then\ 0\ else\ 0}, C) \in \mathit{BadConsts}$?
- For what values of $C$ do we have $(\mathtt{iszero\ 0}, C) \in \mathit{BadConsts}$?

# Another Inductive Definition

$$
\begin{aligned}
\mathit{size}(\mathtt{true}) &= 1\\
\mathit{size}(\mathtt{false}) &= 1\\
\mathit{size}(\mathtt{0}) &= 1\\
\mathit{size}(\mathtt{succ}\ \mathtt{t}_1) &= \mathit{size}(\mathtt{t}_1) + 1\\
\mathit{size}(\mathtt{pred}\ \mathtt{t}_1) &= \mathit{size}(\mathtt{t}_1) + 1\\
\mathit{size}(\mathtt{iszero}\ \mathtt{t}_1) &= \mathit{size}(\mathtt{t}_1) + 1\\
\mathit{size}(\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3) &= \mathit{size}(\mathtt{t}_1) + \mathit{size}(\mathtt{t}_2) + \mathit{size}(\mathtt{t}_3) + 1
\end{aligned}
$$

# Another proof by induction

Theorem: The number of distinct constants in a term is at most the size of the term. I.e., $|Consts(t)| \leq size(t)$.

Proof:

# Another proof by induction

Theorem: The number of distinct constants in a term is at most the size of the term. I.e., $|Consts(t)| \leq size(t)$.

Proof: By induction on $t$.

# Another proof by induction

Theorem: The number of distinct constants in a term is at most the size of the term. I.e., $|Consts(\mathtt{t})| \leq size(\mathtt{t})$.

Proof: By induction on $\mathtt{t}$.
Assuming the desired property for immediate subterms of $\mathtt{t}$, we must prove it for $\mathtt{t}$ itself.

There are "three" cases to consider:

Case: $\mathtt{t}$ is a constant

Immediate: $|Consts(\mathtt{t})| = |\{\mathtt{t}\}| = 1 = size(\mathtt{t})$.

Case: $\mathtt{t} = \mathtt{succ}\ \mathtt{t}_1$, $\mathtt{pred}\ \mathtt{t}_1$, or $\mathtt{iszero}\ \mathtt{t}_1$

By the induction hypothesis, $|Consts(\mathtt{t}_1)| \leq size(\mathtt{t}_1)$. We now calculate as follows:

$|Consts(\mathtt{t})| = |Consts(\mathtt{t}_1)| \leq size(\mathtt{t}_1) < size(\mathtt{t})$.

Case: $t = \texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3$

By the induction hypothesis, $|Consts(t_1)| \leq size(t_1)$, $|Consts(t_2)| \leq size(t_2)$, and $|Consts(t_3)| \leq size(t_3)$. We now calculate as follows:

$$
\begin{aligned}
|Consts(t)| &= |Consts(t_1) \cup Consts(t_2) \cup Consts(t_3)| \\
&\leq |Consts(t_1)| + |Consts(t_2)| + |Consts(t_3)| \\
&\leq size(t_1) + size(t_2) + size(t_3) \\
&< size(t).
\end{aligned}
$$

# Operational Semantics

# Abstract Machines

An *abstract machine* consists of:

- a set of *states*
- a *transition relation* on states, written $\longrightarrow$

We read "$t \longrightarrow t'$" as "$t$ evaluates to $t'$ in one step".

A state records *all* the information in the machine at a given moment. For example, an abstract-machine-style description of a conventional microprocessor would include the program counter, the contents of the registers, the contents of main memory, and the machine code program being executed.

# Abstract Machines

For the very simple languages we are considering at the moment, however, the term being evaluated is the whole state of the abstract machine.

Nb. Often, the transition relation is actually a partial function: i.e., from a given state, there is at most one possible next state. But in general there may be many.

# Operational semantics for Booleans

*Syntax of terms and values*

| | | |
|---|---|---|
| t ::= | | *terms* |
| | true | *constant true* |
| | false | *constant false* |
| | if t then t else t | *conditional* |
| v ::= | | *values* |
| | true | *true value* |
| | false | *false value* |

# Evaluation relation for Booleans

The evaluation relation $t \longrightarrow t'$ is the smallest relation closed under the following rules:

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

# Terminology

*Computation* rules:

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

*Congruence* rule:

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

Computation rules perform “real” computation steps.
Congruence rules determine *where* computation rules can be applied next.

# Evaluation, more explicitly

$\longrightarrow$ is the smallest two-place relation closed under the following rules:

$$((\text{if true then } t_2 \text{ else } t_3), t_2) \in \longrightarrow$$

$$((\text{if false then } t_2 \text{ else } t_3), t_3) \in \longrightarrow$$

$$\frac{(t_1, t_1') \in \longrightarrow}{((\text{if } t_1 \text{ then } t_2 \text{ else } t_3), (\text{if } t_1' \text{ then } t_2 \text{ else } t_3)) \in \longrightarrow}$$

The notation $t \longrightarrow t'$ is short-hand for $(t, t') \in \longrightarrow$.

# Simple Arithmetic Expressions

The set T of terms is defined by the following abstract grammar:

| | | |
|---|---|---|
| t | ::= | *terms* |
| | true | *constant true* |
| | false | *constant false* |
| | if t then t else t | *conditional* |
| | 0 | *constant zero* |
| | succ t | *successor* |
| | pred t | *predecessor* |
| | iszero t | *zero test* |

# Inference Rule Notation

More explicitly: The set T is the *smallest* set *closed* under the following rules.

$$\texttt{true} \in T \qquad \texttt{false} \in T \qquad \texttt{0} \in T$$

$$\frac{t_1 \in T}{\texttt{succ}\ t_1 \in T} \qquad \frac{t_1 \in T}{\texttt{pred}\ t_1 \in T} \qquad \frac{t_1 \in T}{\texttt{iszero}\ t_1 \in T}$$

$$\frac{t_1 \in T \qquad t_2 \in T \qquad t_3 \in T}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \in T}$$

# Generating Functions

Each of these rules can be thought of as a *generating function* that, given some elements from $\mathsf{T}$, generates some other element of $\mathsf{T}$. Saying that $\mathsf{T}$ is closed under these rules means that $\mathsf{T}$ cannot be made any bigger using these generating functions — it already contains everything "justified by its members."

$$\mathtt{true} \in \mathsf{T} \qquad \mathtt{false} \in \mathsf{T} \qquad 0 \in \mathsf{T}$$

$$\frac{t_1 \in \mathsf{T}}{\mathtt{succ}\ t_1 \in \mathsf{T}} \qquad \frac{t_1 \in \mathsf{T}}{\mathtt{pred}\ t_1 \in \mathsf{T}} \qquad \frac{t_1 \in \mathsf{T}}{\mathtt{iszero}\ t_1 \in \mathsf{T}}$$

$$\frac{t_1 \in \mathsf{T} \qquad t_2 \in \mathsf{T} \qquad t_3 \in \mathsf{T}}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \in \mathsf{T}}$$

Let's write these generating functions explicitly.

$$
\begin{aligned}
F_1(U) &= \{\mathtt{true}\} \\
F_2(U) &= \{\mathtt{false}\} \\
F_3(U) &= \{\mathtt{0}\} \\
F_4(U) &= \{\mathtt{succ}\ \mathtt{t}_1 \mid \mathtt{t}_1 \in U\} \\
F_5(U) &= \{\mathtt{pred}\ \mathtt{t}_1 \mid \mathtt{t}_1 \in U\} \\
F_6(U) &= \{\mathtt{iszero}\ \mathtt{t}_1 \mid \mathtt{t}_1 \in U\} \\
F_7(U) &= \{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \mid \mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_3 \in U\}
\end{aligned}
$$

Each one takes a set of terms $U$ as input and produces a set of "terms justified by $U$" as output.

If we now define a generating function for the whole set of inference rules (by combining the generating functions for the individual rules),

$$F(U) \quad = \quad F_1(U) \cup F_2(U) \cup F_3(U) \cup F_4(U) \cup F_5(U) \cup F_6(U) \cup F_7(U)$$

then we can restate the previous definition of the set of terms $\mathrm{T}$ like this:

Definition:

- A set $U$ is said to be "closed under $F$" (or "F-closed") if $F(U) \subseteq U$.
- The set of terms $\mathrm{T}$ is the smallest $F$-closed set.
  (I.e., if $\mathrm{O}$ is another set such that $F(\mathrm{O}) \subseteq \mathrm{O}$, then $\mathrm{T} \subseteq \mathrm{O}$.)

Our alternate definition of the set of terms can also be stated using the generating function $F$ :

$$
\begin{aligned}
S_0 &= \emptyset \\
S_{i+1} &= F(S_i)
\end{aligned}
$$

$$S = \bigcup_i S_i$$

Compare this definition of $S$ with the one we saw last time:

$$
\begin{aligned}
S_0 &= \emptyset \\
S_{i+1} &= \{\text{true}, \text{false}, 0\} \\
&\cup \{\text{succ } t_1, \text{pred } t_1, \text{iszero } t_1 \mid t_1 \in S_i\} \\
&\cup \{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \mid t_1, t_2, t_3 \in S_i\}
\end{aligned}
$$

$$S = \bigcup_i S_i$$

We have “pulled out” $F$ and given it a name.

Note that our two definitions of terms characterize the same set from different directions:

- "from above," as the intersection of all $F$ -closed sets;
- "from below," as the limit (union) of a series of sets that start from $\varnothing$ and get "closer and closer to being $F$ -closed."

The book shows that these two definitions actually define the same set.

# Structural Induction

The principle of structural induction on terms can also be re-stated using generating functions:

> *Suppose* $T$ *is the smallest* $F$*-closed set.*
>
> *If, for each set* $U$,
> *from the assumption "*$P(u)$ *holds for every* $u \in U$*"*
> *we can show "*$P(v)$ *holds for any* $v \in F(U)$*,"*
> *then* $P(t)$ *holds for all* $t \in T$.

# Structural Induction

Why? Because:

- We assumed that $T$ was the *smallest* $F$-closed set, i.e., that $T \subseteq O$ for any other $F$-closed set $O$.
- But showing

  *for each set* $U$,
  *given* $P(u)$ *for all* $u \in U$
  *we can show* $P(v)$ *for all* $v \in F(U)$

  amounts to showing that "the set of all terms satisfying $P$" (call it $O$) is itself an $F$-closed set.
- Since $T \subseteq O$, every element of $T$ satisfies $P$.

# Structural Induction

Compare this with the structural induction principle for terms from last lecture:

> *If, for each term* s,
>
> *given* $P$(r) *for all immediate subterms* r *of* s
>
> *we can show* $P$(s),
>
> *then* $P$(t) *holds for all* t.

Recall, from the definition of $S$, it is clear that, if a term $t$ is in $S_i$, then all of its immediate subterms must be in $S_{i-1}$, i.e., they must have strictly smaller depths. Therefore:

> *If, for each term* $s$,
>     *given* $P(r)$ *for all immediate subterms* $r$ *of* $s$
>     *we can show* $P(s)$, *then*
> $P(t)$ *holds for all* $t$.

Slightly more explicit proof:

- Assume that for each term $s$, given $P(r)$ for all immediate subterms of $s$, we can show $P(s)$.
- Then show, by induction on $i$, that $P(t)$ holds for all terms $t$ with depth $i$.
- Therefore, $P(t)$ holds for all $t$.