510: Programming Languages

Inductive Definitions

David Walker

Fall, 2002

Jobs This Week

- Inductive Definitions (chapter 1, Harper)
- Syntax (chapter 3, 4, 5, Harper)
- Typing (chapter 6, Harper) – if we get there
- Read over the assignment and get started.

Inductive Definitions

Inductive definitions play a **central role** in the study of programming languages.

These aspects of a language are specified by inductive definitions:

- Its concrete syntax, using **context-free grammars**;

- Its abstract syntax, using **abstract syntax trees**;

- Its static semantics, using **typing rules**;

- Its dynamic semantics, using **evaluation rules**.

Inductive Definitions

An inductive definition consists of

- a set of **rules** for deriving . . .
- one or more **judgements**, or **assertions**.

For example, rules for deriving the judgement n nat:

1. `zero` nat;
2. if n nat, then $\texttt{succ}(n)$ nat.

Inference Rule Notation

Inference rules are normally written in the form

$$\frac{J_1 \quad \cdots \quad J_n}{J}$$

where J and $J_1, \ldots, J_n$ are judgements. (For axioms, $n = 0$.)

For example, the rules for deriving n nat are usually written:

$$\frac{}{\mathtt{zero}\ \mathsf{nat}} \qquad \frac{n\ \mathsf{nat}}{\mathtt{succ}(n)\ \mathsf{nat}}$$

Derivation of Judgements

A judgement J is **derivable** iff either

1. there is an axiom

$$\frac{}{J}$$

   or

2. there is a rule

$$\frac{J_1 \quad \cdots \quad J_n}{J}$$

   such that each of the judgements J_1, . . . , J_n are derivable.

(The second condition subsumes the first, by taking $n = 0$.)

Derivation of Judgements

We may determine whether a judgement is derivable by working backwards.

For example, the judgement $\texttt{succ}(\texttt{succ}(\texttt{zero}))$ nat is derivable:

$$\dfrac{\dfrac{\overline{\texttt{zero}\ \mathsf{nat}}}{\texttt{succ}(\texttt{zero})\ \mathsf{nat}}}{\texttt{succ}(\texttt{succ}(\texttt{zero}))\ \mathsf{nat}}$$

Such a composition of rules is called a **derivation**.

Binary Trees

Here is a set of rules defining the judgement t tree stating that t is a binary tree:

$$\frac{}{\texttt{empty}\ \mathsf{tree}} \qquad \frac{t_1\ \mathsf{tree} \quad t_2\ \mathsf{tree}}{\texttt{node}(t_1, t_2)\ \mathsf{tree}}$$

Check that $\texttt{node}(\texttt{empty}, \texttt{node}(\texttt{empty}, \texttt{empty}))\ \mathsf{tree}$ is derivable according to these rules.

Rule Induction

By definition, every derivable judgement

- is the consequence of some rule ...

- whose premises are derivable.

That is, the rules are an **exhaustive** description of the derivable judgements.

Rule Induction

Thus to show that every derivable judgement has a property P, it is enough to show that

if $J_1, \ldots, J_n$ have property P, then J has property P.

for every rule

$$\frac{J_1 \quad \cdots \quad J_n}{J}.$$

This is the principal of **rule induction**.

Example: Natural Numbers

Consider the rules defining the judgement n nat.

We can prove that the property P holds of every n such that n nat by rule induction.

- Show that P holds of `zero`;
- Assuming that P holds of n, show that P holds of $\texttt{succ}(n)$.

This is just ordinary **mathematical induction**!

Example: Binary Trees

Similarly, we can prove that every binary tree t has a property P by showing that

- `empty` has property P;

- if t_1 has property P and t_2 has property P, then $\texttt{node}(t_1, t_2)$ has property P.

This is called **tree induction**.

Example: The Height of a Tree

Consider the following equations:

$$\begin{array}{rcl} hgt(\texttt{empty}) & = & 0 \\ hgt(\texttt{node}(t_1, t_2)) & = & 1 + \max(hgt(t_1), hgt(t_2)) \end{array}$$

Claim: for every binary tree t there is a unique integer n such that $hgt(t) = n$.

That is, the above equations define a **function**.

Example: The Height of a Tree

We will prove the claim by rule induction:

- If t is derivable by the axiom

$$\frac{}{\texttt{empty}\ \mathsf{tree}}$$

  then $n = 0$ is uniquely determined by the first equation.

- ...

Example: The Height of a Tree

Continuing, . . .

- If t is derivable by the rule

$$\frac{t_1 \ \mathsf{tree} \qquad t_2 \ \mathsf{tree}}{\mathtt{node}(t_1, t_2) \ \mathsf{tree}},$$

  then we may assume that

  1. there exists a unique n_1 such that $hgt(t_1) = n_1$;

  2. there exists a unique n_2 such that $hgt(t_2) = n_2$;

  Hence it follows that there exists a unique n, namely $1+\max(n_1, n_2)$, such that $hgt(t) = n$.

Example: The Height of a Tree

This is awfully pedantic, but it is useful to see the details at least once.

- It is not obvious **a priori** that a tree has a well-defined height!

- Rule induction justifies the existence of the function hgt.

It is obvious from the equations that there is **at most one** n such that $hgt(t) = n$. The proof shows that there exists **at least one**.

Example: Variadic Trees

We often define several judgements simultaneously by a set of (mutually recursive) rules.

For example, these rules define the **variadic trees**:

$$\frac{f \ \text{forest}}{\mathtt{node}(f) \ \text{tree}} \qquad \frac{}{\mathtt{nil} \ \text{forest}} \qquad \frac{t \ \text{tree} \quad f \ \text{forest}}{\mathtt{cons}(t, f) \ \text{forest}}$$

(Variadic = variable number of children at any node.)

Example: Variadic Trees

We may show that every tree t has property T and every forest f has property F by showing

- $\texttt{node}(f)$ has property T, assuming that f has property F.

- $\texttt{nil}$ has property F.

- $\texttt{cons}(t, f)$ has property F, assuming that t has property T and f has property F.

Exercise: justify this by rule induction.

A More Rigorous Treatment

What are the objects in the rules?

1. What is **`zero`**? What is **`succ`**$(-)$?

2. What is **`empty`**? What is **`node`**$(-,-)$?

What licenses reasoning by induction?

Universes

We will work with objects drawn from a specified universe.

Examples:

- Strings over a specified alphabet.
- Abstract syntax trees (ast's) over a specified signature.
- Various other "finitary" objects.

Strings

Let Σ be a countable set of **characters**.

- ASCII or UniCode.

- Binary digits.

Strings

The set Σ^* consists of all finite strings over Σ.

- Null string ε.

- Single-character string: a, where $a \in \Sigma$.

- Concatenation: $s_1\, s_2$ where s_1 and s_2 are strings.

AST's (Terms)

Let $\mathcal{O}$ be a countable set of **operators**.

Let $\alpha : \mathcal{O} \to \mathbb{N}$ be an assignment of **arities** to operators.

- Arity = number of arguments.

- 0-ary = no arguments.

AST's (Terms)

The set $\mathcal{T} = \mathcal{T}(\mathcal{O}, \alpha)$ is the set of **ast's** built from operators in $\mathcal{O}$:

$$\mathcal{T} = \{\, o(t_1, \ldots, t_n) \mid o \in \mathcal{O},\ \alpha(o) = n,\ t_1, \ldots, t_n \in \mathcal{T}\,\}.$$

The ast's t_1,. . . ,t_n are **sub-trees** of $o(t_1, \ldots, t_n)$.

AST's (Terms)

Examples:

- $\mathtt{zero}$ has arity 0, $\mathtt{succ}(-)$ has arity 1.

- $\mathtt{empty}$ has arity 0, $\mathtt{node}(-,-)$ has arity 2.

So, for example, $\mathtt{node}(\mathtt{empty}, \mathtt{empty})$ is an ast.

Finitary Objects

Strings can all be coded up as natural numbers.

- Use Gödel-numbering to establish a 1-1 correspondence between strings and natural numbers.

- Idea: think of a string as a number in base 128.

Ast's can also be encoded by similar means.

In general, a **finitary** object is one that may be encoded as a natural number.

Closure Under Rules

Fix a universe $\mathcal{U}$ and a set of rules $\mathcal{R}$ over $\mathcal{U}$.

A set X of judgements is **closed** under $\mathcal{R}$ iff

- for each rule

  $$\frac{J_1 \quad \cdots \quad J_n}{J}$$

  in $\mathcal{R}$, . . .

- if J_1,. . . , J_n are all in X, then J is in X.

Closure Under Rules

Example: A set X is closed under the rules $\mathcal{R}_N$ for the natural numbers iff

1. $\texttt{zero} \in X$;

2. if n nat $\in X$, then $\texttt{succ}(n)$ nat $\in X$.

Closure Under Rules

The universe $\mathcal{U}$ is closed under **any** rule set!

The universe contains "everything", so it is closed under any rules for putting elements into it.

Closure Under Rules

Suppose that `junk` an operator of arity 0.

The set

$$\{\, \texttt{junk}\ \mathsf{nat} \,\} \cup \{\, \underbrace{\texttt{succ}(\ldots\texttt{succ}(\texttt{zero}))}_{(i\ \ times)}\ \mathsf{nat} \mid i \geq 0 \,\}$$

is **not** closed under $\mathcal{R}_N$.

For example, $\texttt{succ}(\texttt{junk})$ nat is not in the set, even though `junk` nat is in it.

Closure Under Rules

However, the set

$$\{ \underbrace{\texttt{succ}(\ldots\texttt{succ}(\texttt{junk}))}_{(i \ \ times)} \ \textsf{nat} \mid i \geq 0 \}$$

$$\cup$$

$$\{ \underbrace{\texttt{succ}(\ldots\texttt{succ}(\texttt{zero}))}_{(i \ \ times)} \ \textsf{nat} \mid i \geq 0 \}$$

is closed under $\mathcal{R}_N$.

It is **not** the smallest closed set, however!

Inductively Defined Sets

The **smallest** set I closed under $\mathcal{R}$ means that:

- I is closed under $\mathcal{R}$.
- If X is closed under $\mathcal{R}$, then $I \subseteq X$.

Inductively-Defined Sets

Notice that there is at most one smallest set closed under $\mathcal{R}$.

If I and J are both the smallest $\mathcal{R}$-closed set, then

- $I \subseteq J$ because I is smallest,

- $J \subseteq I$ because J is smallest,

hence $I = J$.

Inductively Defined Sets

For rule set $\mathcal{R}$, define

$$I(\mathcal{R}) := \bigcap\{\, X \subset \mathcal{U} \mid X \;\; is \; \mathcal{R} - closed \,\}.$$

This is the smallest subset of $\mathcal{U}$ closed under $\mathcal{R}$.

Rule Induction

Theorem 1

For any rule set $\mathcal{R}$, the set $S = I(\mathcal{R})$ is the smallest set closed under $\mathcal{R}$.

Proof:

- If X is $\mathcal{R}$-closed, then X is among the sets in the intersection defining S, so $S \subseteq X$.
- . . .

$\blacksquare$

Rule Induction

Continuing, . . .

- Suppose that

$$\frac{J_1 \quad \cdots \quad J_n}{J}$$

  is a rule in $\mathcal{R}$ and that each J_i is in S. Then J is in every $\mathcal{R}$-closed set X, so $J \in S$, their intersection.

Rule Induction

This theorem licenses proof by **rule induction**.

> *To prove that every* $J \in I(\mathcal{R})$ *has a property* P*, it suffices to prove that* P *is* $\mathcal{R}$*-closed.*

Example: Palindromes

Fix an alphabet Σ.

Consider the following rules, $\mathcal{R}_P$, over the universe Σ^*:

$$\frac{}{\varepsilon\ \mathsf{pal}} \qquad \frac{}{a\ \mathsf{pal}} \qquad \frac{s\ \mathsf{pal}}{a\,s\,a\ \mathsf{pal}}$$

The judgement $s\,\mathsf{pal}$ states that the string s is a **palindrome**.

Example: Palindromes

Theorem: If $s\,\mathsf{pal}$, then $s = s^R$.

It suffices to show that this property is $\mathcal{R}_P$-closed:

1. $\varepsilon = \varepsilon^R$.

2. $a = a^R$.

3. Suppose that $s = s^R$. Then $(a\,s\,a)^R = a^R\,s^R\,a^R = a\,s^R\,a = a\,s\,a$, as required.

Defining Functions By Induction

Consider the equations

$$\begin{array}{rcl} parity(\varepsilon) & = & 0 \\ parity(a) & = & 1 \\ parity(a\,s\,a) & = & parity(s) \end{array}$$

Claim: if $s\,\mathsf{pal}$, then there exists a unique $b \in \{\,0, 1\,\}$ such that $parity(s) = b$.

Exercise: prove this by rule induction.

Summary

An inductive definition consists of a set of rules defining the derivability of one or more judgements.

A judgement is derivable iff it may be obtained by composing rules.

Rule induction allows us to show that every derivable judgement has a specified property by considering each rule in turn.