Programming Languages and Computation

Week 5: Inductive Reasoning, Arithmetic Expressions

We use this sheet as a refresher on mathematical and structural induction, to help us build better intuition for the definitions we've seen and will be seeing more of. Later questions allow you to practice your understanding of arithmetic expressions, and prove useful facts about their semantics—by induction on their syntax!

Inductive Reasoning

Arithmetic expressions (and other expressions that were defined previously in the unit) were defined inductively: they are defined as the smallest set that contains some base elements (the base cases), and is closed under certain operations (the inductive cases).

A very simple and familiar example of such an inductively defined object is the set $\mathbb{N}$ of natural integers. $\mathbb{N}$ is indeed defined as the smallest set such that:

- $0 \in \mathbb{N}$, and
- if $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$.

* 1. Let's see why it's important to say "the smallest set such that" in inductive definitions, and try to give a definition for *even* integers.
Let $\mathbb{E} \subseteq \mathbb{N}$ be a set such that

- $0 \in \mathbb{E}$, and
- if $n \in \mathbb{E}$, then $n + 2 \in \mathbb{E}$.

(a) Find a set that meets the two conditions we placed on $\mathbb{E}$ but contains odd integers.

Solution

(a) The set $\mathbb{N}$ contains 0, and for any integer $n \in \mathbb{N}$, we also have $n + 2 \in \mathbb{N}$. (This is because $n + 1 \in \mathbb{N}$ by definition of $\mathbb{N}$, and therefore also $n + 2 = n + 1 + 1 \in \mathbb{N}$.) However, it also contains odd integers.
The key here is that $\mathbb{N}$ contains 1, which cannot be constructed by adding a multiple of 2 to 0.

By stating that a set is “the smallest such that”, we say that all its elements *must* be constructed using the closure operations applied to the base elements, and that no other elements belong in that set![1]

You could right now take some time to think about inductive definitions such as those discussed above, and Haskell algebraic datatype definitions: can Haskell datatypes ever contain values that cannot be constructed by applying constructors to ground values?

Mathematical Induction.

Inductively-defined sets are very nicely behaved. Because we know that all the objects they contain are constructed through the given operations from the given base cases, we can prove properties for all objects by *structural induction*.

In the case of $\mathbb{N}$, you already know this as *mathematical induction*. To prove that some property $P : \mathbb{N} \to \{\top, \bot\}$ holds for all $n \in \mathbb{N}$ (we'll write $P(n)$ for “P holds on n"), we prove, that:

- $P(0)$; and
- if $P(n)$, then $P(n+1)$.

These two combined give us a proof for any $n \in \mathbb{N}$: because all elements in $\mathbb{N}$ are either 0 or the result of applying the successor operation to 0 a finite number of times, we know there exists a finite proof for P that can be constructed from the two statements. (And importantly, we don't actually need to ever construct it.)

Structural Induction.

Structural induction does the same for any inductively-defined set (and also functions and relations, since they are sets, too). For example, let's consider a simple set $\mathbb{T}$ of trees that are only a structure—they won't contain any data. The set $\mathbb{T}$ is the smallest set such that:

- $\bullet \in \mathbb{T}$, and
- if $\ell \in \mathbb{T}$ and $r \in \mathbb{T}$, then $N(\ell, r) \in \mathbb{T}$.

As examples, $\bullet$ is a tree in $\mathbb{T}$, but also $N(\bullet, \bullet)$, and $N(N(\bullet, \bullet), \bullet)$, ...

On this set, we can define some neat functions inductively. For example, the $\mathsf{size} : \mathbb{T} \to \mathbb{N}$ function counts the number of $\bullet$ in the tree, and the $\mathsf{depth} : \mathbb{T} \to \mathbb{N}$ function counts the number of “levels” in the tree.

$$\begin{aligned} \mathsf{size}(\bullet) &= 1 & \qquad \mathsf{depth}(\bullet) &= 1 \\ \mathsf{size}(N(\ell, r)) &= \mathsf{size}(\ell) + \mathsf{size}(r) & \qquad \mathsf{depth}(N(\ell, r)) &= 1 + \mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r)) \end{aligned}$$

** 2. Adapting the reasoning principles of mathematical induction to the inductive structure of $\mathbb{T}$, prove the following two facts:

(a) Given any $t \in \mathbb{T}$, we have $1 \leq \mathsf{depth}(t)$ and $1 \leq \mathsf{size}(t)$.

(b) Given any $t \in \mathbb{T}$, we have $\mathsf{depth}(t) \leq \mathsf{size}(t)$.

[1] We kind of gloss over the fact that it is not obvious there exists a smallest set such that the conditions hold, or that it is unique if it exists; that's a topic for another unit. In this unit, we only ever give you inductive definitions for sets that actually exist.

Solution

The structural induction principle for $\mathbb{T}$ says we have to prove the property for $\bullet$, and then, assuming it holds for two trees ℓ and r also prove that it must hold for $N(\ell, r)$.

We write these proofs in slightly more detail than we'd expect in an exam setting.

(a) By structural induction on t, we prove that for any $t \in \mathbb{T}$, we have $1 \leq \mathsf{depth}(t)$ and $1 \leq \mathsf{size}(t)$.

Case $t = \bullet$: In this case, we have $\mathsf{size}(\bullet) = \mathsf{depth}(\bullet) = 1$.

Case $t = N(\ell, r)$: We have two induction hypotheses (one for each of the subtrees):

1. $1 \leq \mathsf{depth}(\ell)$ and $1 \leq \mathsf{size}(\ell)$, and
2. $1 \leq \mathsf{depth}(r)$ and $1 \leq \mathsf{size}(r)$.

Therefore, we have $1 \leq 2 \leq 1 + \mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r)) = \mathsf{depth}(N(\ell, r))$, and $1 \leq 2 \leq \mathsf{size}(\ell) + \mathsf{size}(r) = \mathsf{size}(N(\ell, r))$.

(b) By structural induction on t, we prove that for any $t \in \mathbb{T}$ we have $\mathsf{depth}(t) \leq \mathsf{size}(t)$.

Case $t = \bullet$: In this case, we have $\mathsf{size}(\bullet) = 1 = \mathsf{depth}(\bullet)$, and the property holds.

Case $t = N(\ell, r)$: We have two induction hypotheses (one for each of the subtree):

1. $\mathsf{depth}(\ell) \leq \mathsf{size}(\ell)$, and
2. $\mathsf{depth}(r) \leq \mathsf{size}(r)$.

By definition of size and depth, we have $\mathsf{depth}(N(\ell, r)) = 1 + \mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r))$ and $\mathsf{size}(N(\ell, r)) = \mathsf{size}(\ell) + \mathsf{size}(r)$.

By the induction hypotheses, we know that $1+\mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r)) \leq 1+\mathsf{max}(\mathsf{size}(\ell), \mathsf{size}(r))$. Further, since $1 \leq \mathsf{size}(\ell)$ and $1 \leq \mathsf{size}(r)$, we have $1 + \mathsf{max}(\mathsf{size}(\ell), \mathsf{size}(r)) \leq \mathsf{size}(\ell) + \mathsf{size}(r)$, and the property holds.

It's worth noting that it's in fact easy to prove both properties in a single go, using $1 \leq \mathsf{depth}(t) \leq \mathsf{size}(t)$ as the inductive property.

Let us define the set $\mathbb{Z} \subseteq \mathbb{T}$ of *zig-zags* as the smallest set such that:

- $\bullet \in \mathbb{Z}$;
- if $t \in \mathbb{Z}$, then $N(t, \bullet) \in \mathbb{Z}$; and
- if $t \in \mathbb{Z}$, then $N(\bullet, t) \in \mathbb{Z}$.

*** 3.

> (a) Prove that, given any $t \in \mathbb{T}$, if $\mathsf{depth}(t) = \mathsf{size}(t)$, then $t \in \mathbb{Z}$.

Solution

(a) By structural induction on t, we show that any t such that $\mathsf{depth}(t) = \mathsf{size}(t)$ is a zig-zag.

Case $t = \bullet$: t is a zig-zag by definition.

Case $t = N(\ell, r)$: By induction hypothesis, we have:

1. if $\mathsf{depth}(\ell) = \mathsf{size}(\ell)$, then $\ell \in \mathbb{Z}$;
2. if $\mathsf{depth}(r) = \mathsf{size}(r)$, then $r \in \mathbb{Z}$.

We have $\mathsf{size}(N(\ell,r)) = \mathsf{size}(\ell)+\mathsf{size}(r) = 1+\mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r)) = \mathsf{depth}(N(\ell,r))$. Let us consider the case where $\mathsf{max}(\mathsf{depth}(\ell), \mathsf{depth}(r)) = \mathsf{depth}(\ell)$.

In this case, we have $\mathsf{size}(\ell)+\mathsf{size}(r) = \mathsf{depth}(\ell)+1$. Since $1 \leq \mathsf{depth}(\ell) \leq \mathsf{size}(\ell)$ and $1 \leq \mathsf{size}(r)$, it must be that $1 = \mathsf{size}(r)$ and $\mathsf{size}(\ell) = \mathsf{depth}(\ell)$.

We therefore have $r = \bullet$ and $\ell \in \mathbb{Z}$.[2] and t is therefore a zig-zag.

The case $\mathsf{depth}(N(\ell,r)) = \mathsf{depth}(r)$ gives rise to a symmetrical argument which constructs a zig-zag that starts on the right.

While Expressions

Recall the definitions of syntax and semantics for arithmetic expressions in While.

* 4. For each of the following, indicate whether it is a (syntactically) valid arithmetic expression in the While language.

(a) `x + 1`

(b) `x - 1`

(c) `-x + 1`

(d) `x - +1`

(e) `x + -1`

Solution

(a) Valid

(b) Valid

(c) Invalid (unary negation on a variable)

(d) Invalid (it's just broken)

(e) Valid (integer literals have value in $\mathbb{Z}$)

** 5. Give the semantics of the following expressions (as integer-valued functions of a state):

(a) `112 + 46`

(b) `1 + n * 56`

(c) `10 - (x + 10)`

(d) `42 * (z - 2 * z)`

[2] You may want to prove $\mathsf{size}(t) = 1 \Rightarrow t = \bullet$ also, by induction. But it's obvious enough I wouldn't hold it against you if you didn't.

Solution

(a) $[\![\texttt{112 + 46}]\!]^{\mathscr{A}} = \_ \mapsto 158$

(b) $[\![\texttt{1 + n * 56}]\!]^{\mathscr{A}} = \sigma \mapsto 1 + \sigma(\texttt{n}) \cdot 56$

(c) $[\![\texttt{10 - (x + 10)}]\!]^{\mathscr{A}} = \sigma \mapsto -\sigma(\texttt{x})$

(d) $[\![\texttt{42 * (z - 2 * z)}]\!]^{\mathscr{A}} = \sigma \mapsto -42 \cdot \sigma(\texttt{z})$

** 6. Evaluate the semantics above in the following states:

> (a) $\begin{bmatrix} \texttt{n} & \mapsto & 3 \\ \texttt{x} & \mapsto & 10 \\ \texttt{z} & \mapsto & 2 \end{bmatrix}$
>
> (b) $\begin{bmatrix} \texttt{z} & \mapsto & 3 \\ \texttt{x} & \mapsto & 10 \end{bmatrix}$

Solution

Unless I can't multiply, the results are as follows:

(a) 158, 169, −10, −84

(b) 158, 1, −10, −126

*** 7. We define the *free variables* $\mathsf{FV}(a)$ of an arithmetic expression inductively as follows:

> $$\begin{aligned} \mathsf{FV}(\texttt{n}) &= \emptyset \\ \mathsf{FV}(x) &= \{x\} \\ \mathsf{FV}(a_1 \texttt{ + } a_2) &= \mathsf{FV}(a_1) \cup \mathsf{FV}(a_2) \\ \mathsf{FV}(a_1 \texttt{ - } a_2) &= \mathsf{FV}(a_1) \cup \mathsf{FV}(a_2) \\ \mathsf{FV}(a_1 \texttt{ * } a_2) &= \mathsf{FV}(a_1) \cup \mathsf{FV}(a_2) \end{aligned}$$
>
> Let a be an arithmetic expression, and σ_1, σ_2 be two states such that, for every variable $\texttt{v} \in \mathsf{FV}(a)$, we have $\sigma_1(\texttt{v}) = \sigma_2(\texttt{v})$ (that is, σ_1 and σ_2 agree on the free variables of a). Then, prove—by structural induction—that $[\![a]\!]^{\mathscr{A}}(\sigma_1) = [\![a]\!]^{\mathscr{A}}(\sigma_2)$. (In other words, the semantics of an arithmetic expression depends only on the values given by the state to its free variables—this is known as a *frame rule*.)

Solution

Let σ_1 and σ_2 be two states. We prove by induction over a that, if for every $\texttt{v} \in \mathsf{FV}(a)$ we have $\sigma_1(\texttt{v}) = \sigma_2(\texttt{v})$, then $[\![a]\!]^{\mathscr{A}}(\sigma_1) = [\![a]\!]^{\mathscr{A}}(\sigma_2)$.

Case $\texttt{n}$ We have $[\![\texttt{n}]\!]^{\mathscr{A}}(\sigma_1) = [\![\texttt{n}]\!]^{\mathbb{Z}} = [\![\texttt{n}]\!]^{\mathscr{A}}(\sigma_2)$

Case x We have (by hypothesis) that $\mathsf{FV}(x) = \{x\}$. Therefore $[\![x]\!]^{\mathscr{A}}(\sigma_1) = \sigma_1(x) = \sigma_2(x) = [\![x]\!]^{\mathscr{A}}(\sigma_2)$ (For this case, it is crucial that we keep the implication in the inductive property.)

Case $a_1 + a_2$ By induction hypothesis, we have:

- if, for every $v \in \mathsf{FV}(a_1)$ we have $\sigma_1(v) = \sigma_2(v)$, then $[\![a_1]\!]^{\mathscr{A}}(\sigma_1) = [\![a_1]\!]^{\mathscr{A}}(\sigma_2)$
- if, for every $v \in \mathsf{FV}(a_2)$ we have $\sigma_1(v) = \sigma_2(v)$, then $[\![a_2]\!]^{\mathscr{A}}(\sigma_1) = [\![a_2]\!]^{\mathscr{A}}(\sigma_2)$

We have (by hypothesis) that for every $x \in \mathsf{FV}(a_1 + a_2)$, $\sigma_1(x) = \sigma_2(x)$. Since $\mathsf{FV}(a_1 + a_2) = \mathsf{FV}(a_1) \cup \mathsf{FV}(a_2)$, we also have $\sigma_1(x) = \sigma_2(x)$ for every $x \in \mathsf{FV}(a_1)$ and for every $x \in \mathsf{FV}(a_2)$.

We therefore have $[\![a_1]\!]^{\mathscr{A}}(\sigma_1) = [\![a_1]\!]^{\mathscr{A}}(\sigma_2)$ and $[\![a_2]\!]^{\mathscr{A}}(\sigma_1) = [\![a_2]\!]^{\mathscr{A}}(\sigma_2)$, and we can conclude as follows.

$[\![a_1 + a_2]\!]^{\mathscr{A}}(\sigma_1) = [\![a_1]\!]^{\mathscr{A}}(\sigma_1) + [\![a_2]\!]^{\mathscr{A}}(\sigma_1) = [\![a_1]\!]^{\mathscr{A}}(\sigma_2) + [\![a_2]\!]^{\mathscr{A}}(\sigma_2) = [\![a_1 + a_2]\!]^{\mathscr{A}}(\sigma_2)$.

(For this case, it is therefore crucial that we discharge the condition on equality of states over the free variables of subexpressions.)

Cases $a_1 - a_2$ **and** $a_1 * a_2$ Analogous.