Perhaps the single most important proof technique in the theory of programming languages is *mathematical induction*. Today we'll look at how to use it to prove things about the semantics of ARITH we talked about last time, and why we need induction to do it.

Recall the small-step operational semantics for ARITH we defined last time.

$$\frac{n_1 + n_2 = m}{\overline{n_1} + \overline{n_2} \longrightarrow \overline{m}} \qquad \frac{n_1 * n_2 = m}{\overline{n_1} * \overline{n_2} \longrightarrow \overline{m}}$$

$$\frac{e_1 \longrightarrow e_1'}{e_1 + e_2 \longrightarrow e_1' + e_2} \qquad \frac{e_1 \longrightarrow e_1'}{e_1 * e_2 \longrightarrow e_1' * e_2} \qquad \frac{e \longrightarrow e'}{\overline{n} + e \longrightarrow \overline{n} + e'} \qquad \frac{e \longrightarrow e'}{\overline{n} * e \longrightarrow \overline{n} * e'}$$

# 1 Using Induction to Prove Program Properties

We can use the same structure to prove properties about programs in ARITH. We may wish, for instance, to prove that the operational semantics for ARITH always terminates. To do this we need to define how to chain multiple operational steps together into a longer execution, which can be done very simply using the following (also inductive) multi-step relation.

$$\frac{}{e \longrightarrow^* e} \qquad \frac{e \longrightarrow e' \qquad e' \longrightarrow^* e''}{e \longrightarrow^* e''}$$

Now we can ask a few questions:

1. Formally, what does it mean for an execution to terminate?
2. Do all executions terminate?

To answer the first question, we note that "termination" usually means that an execution cannot take any more steps *because it has reached a value*. In this case, "values" are just integer symbols, so we might say that $e$ terminates if it eventually evaluates to $\overline{n}$ for some $\overline{n}$. Using the multi-step relation above, $e$ terminates if there is some $\overline{n}$ such that $e \longrightarrow^* \overline{n}$.

We can now phrase the answer to our second question as the following soundness theorem.

**Theorem 1** (Soundness)**.** *Evaluation of expressions yields an integer.*

$$\forall e \in \textbf{AExp}.\, \exists \overline{n} \in \textbf{Int}.\, e \longrightarrow^* \overline{n}$$

We will prove this by *structural induction on* $e$.

*Proof.* Let $e \in \textbf{AExp}$ be be an expression. This proof will follow by structural induction on $e$.

**Case** $e = \overline{n}$**:** By the axiom rule of $\longrightarrow^*$, $e = \overline{n} \longrightarrow^* \overline{n}$.

**Case** $e = e_1 + e_2$**:** The principle of induction allows us to assume that the result holds for both $e_1$ and $e_2$, giving us two inductive hypotheses:

1. $\exists \overline{n_1}.\, e_1 \longrightarrow^* \overline{n_1}$
2. $\exists \overline{n_2}.\, e_2 \longrightarrow^* \overline{n_2}$

We can prove (also by induction, though we will not do so here) that the single-step operational semantic rules extend to multi-step rules. Therefore, by the first inductive hypothesis, $e_1 + e_2 \longrightarrow^* \overline{n_1} + e_2$, and by the second inductive hypothesis $\overline{n_1} + e_2 \longrightarrow^* \overline{n_1} + \overline{n_2}$. Putting these together, we get

$$e = e_1 + e_2 \longrightarrow^* \overline{n_1} + e_2 \longrightarrow^* \overline{n_1} + \overline{n_2} \longrightarrow \overline{m}$$

where $n_1 + n_2 = m$.

Using two more lemmas that are straightforward and we will apply without proof here—that $\longrightarrow^*$ is transitive, and that if $e \longrightarrow^* e'$ and $e' \longrightarrow e''$, then $e \longrightarrow^* e''$, we complete the proof that $e \longrightarrow^* \overline{m}$.

**Case** $e = e_1 * e_2$**:** This case is identical to the previous case, but using rules for $*$ instead of $+$. □

In the inductive cases of this proof, we *assumed the theorem was true for smaller subterms*. Why is this allowed? This is the principle of structural induction, and we will now talk about why it works..

## 2 Eval Revisited

Last time we gave an eval function for ARITH:

$$\mathsf{eval}(e) = \begin{cases} n & \text{when } e = \overline{n} \\ \mathsf{eval}(e_1) + \mathsf{eval}(e_2) & = \text{when } e = e_1 + e_2 \\ \mathsf{eval}(e_1) * \mathsf{eval}(e_2) & = \text{when } e = e_1 * e_2. \end{cases}$$

Why does this definition uniquely determine the function eval? If we think of it as an inductively defined relation $\mathsf{eval} \subseteq \mathbf{AExp} \times \mathbb{Z}$, there are two issues.

- *Uniqueness*: Is there (at most) one $n \in \mathbb{Z}$ such that $(e, n) \in \mathsf{eval}$ for all $e \in \mathbf{AExp}$? That is, is eval a function?
- *Existence*: Is eval defined on all ARITH terms? In other words, is it total?

We noted before that that this is a structurally-recursive definition. There are exactly three clauses of eval, one for each clause in the BNF definition **AExp**, and that expressions can be formed only in these three ways. Critically, although eval appears on the right side of three of the five clauses, in each case it is applied to a proper (*proper* = strictly) subterm. Intuitively, this last point means that the term it is applied to is always shrinking, and eval will eventually "bottom out" at the first clause. Now let's make that intuition precise.

## 3 Well-Founded Relations

**Definition 1.** A binary relation $\prec$ is said to be *well-founded* if it has no infinite descending chains.

An *infinite descending chain* is an infinite sequence $a_0, a_1, a_2, \ldots$ such that $a_{i+1} \prec a_i$ for all $i \geq 0$.

Note that a well-founded relation cannot be reflexive, as an infinite stream of the same value would then be an infinite descending chain.

Here are some examples of well-founded relations.

- The successor relation $\{(m, m+1) \mid m \in \mathbb{N}\}$ on the natural numbers $\mathbb{N}$.
- The (strict) less-than relation $<$ on $\mathbb{N}$.
- The (strict) sub-expression relation on ARITH expressions.
- The element-of relation $\in$ on sets. The axiom of foundation (or axiom of regularity) in Zermelo–Fraenkel (ZF) set theory implies that $\in$ is well-founded. Among other things, it prevents a set from containing itself.
- The proper subset relation $\subset$ on *finite* subsets of $\mathbb{N}$.

The following are *not* well-founded relations.

- The predecessor relation $\{(m+1, m) \mid m \in \mathbb{N}\}$ on the natural numbers $\mathbb{N}$. The sequence $0, 1, 2, \ldots$ is an infinite *descending* chain!
- The greater-than relation $>$ on $\mathbb{N}$.
- The less-than relation $<$ on $\mathbb{Z}$. The sequence $0, -1, -2, \ldots$ is an infinite descending chain.
- The less-than relation $<$ on the real interval $[0, 1]$. $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$ is an infinite descending chain.
- The proper subset relation $\subset$ on arbitrary subsets of $\mathbb{N}$. The sequence $\mathbb{N}, \mathbb{N} - \{0\}, \mathbb{N} - \{0, 1\}, \ldots$ is an infinite descending chain.

# 4 Well-Founded Induction

For any well-founded binary relation $\prec$ on a set $A$, the principle of well-founded induction allows us to prove some predicate $P$ holds on all elements of $A$ by proving that, for any $a \in A$ such that $P(b)$ holds whenever $b \prec a$, then $P(a)$ must also hold. In mathematical notation,

$$\forall a \in A.\, (\forall b \in A.\, b \prec a \implies P(b)) \implies P(a) \quad \implies \quad \forall a \in A.\, P(a). \tag{1}$$

The basis for this induction is when $a$ has no $\prec$-predecessors, in which case $\forall b \in A.\, b \prec a \implies P(b)$ holds vacuously.

We can see how this generalizes all forms of induction we have seen so far.

**Example 1.** For the well-founded predecessor relation on $\mathbb{N}$ (pred $= \{(m+1, m) \mid m \in \mathbb{N}\}$), (1) reduces to the standard principle of mathematical induction. Because 0 has no predecessors, $\forall n \in \mathbb{N}.\, (0, n) \in \text{pred} \implies P(n)$ hold vacuously, and we must directly prove $P(0)$. For all other natural numbers $n > 0$, $n = m + 1$, so we must prove $P(m) \implies P(m+1)$.

**Example 2.** For the well-founded relation $<$ on $\mathbb{N}$, (1) reduces to *strong* induction on $\mathbb{N}$: to prove $\forall n \in \mathbb{N}.\, P(n)$, it suffices to prove that $P(n)$ holds whenever *all* of $P(0), P(1), \ldots, P(n-1)$ hold. When $n = 0$, the set of inductive hypotheses is empty, requiring a direct proof of $P(0)$.

**Example 3.** Structural induction on an inductive set is a form of well-founded induction. For instance, the strict sub-term relation on ARITH expression is a well-founded relation where $n$ and $x$ are the base cases, as they have no strict sub-expressions, and each other expression form is an inductive case, as it contains strict sub-expression.

Similarly, structural induction on the step relation $\longrightarrow$ is well founded for the same reason.

## 4.1 Equivalence of Well-Foundedness and the Validity of Induction

In fact, one can show that the induction principle (1) is valid *if and only if* the relation $\prec$ is well-founded.

**Theorem 2.** *The induction principle (1) holds for $\prec$ on set $A$ if and only if $\prec$ is well-founded on $A$.*

*Proof.* This will be a proof by contradiction.

We first prove that, if principle (1) is not valid for $\prec$ on $A$, then there is an infinite descending chain (that is, $\prec$ is not well-founded on $A$). By assumption, (1) does not hold, which means there is some predicate $P$ and some element $a_0 \in A$ such that $\neg P(a_0)$, and yet

$$\forall a \in A.\, (\forall b \in A.\, b \prec a \implies P(b)) \implies P(a).$$

This implication is equivalent to:

$$\forall a \in A.\, \neg P(a) \implies \exists b \in A.\, b \prec a \wedge \neg P(b).$$

Since $\neg P(a_0)$, this means there must be some $a_1 \prec a_0$ such that $\neg P(a_1)$. But because $P(a_1)$ is false, the same logic applies again, giving an $a_2 \prec a_1$ where $\neg P(a_2)$. Using the axiom of dependent choice (a weakened form of the axiom of choice), one can continue this process to construct an infinite descending chain $a_0, a_1, a_2, \ldots$ such that $a_{i+1} \prec a_i$ and $\neg P(a_i)$ for all $i \geq 0$. Therefore $\prec$ is not well-founded on $A$.

To prove the other direction, we assume there is an infinite descending chain $a_0, a_1, a_2, \ldots$ and show that (1) cannot hold. Specifically, consider the predicate $P(a) = a \notin \{a_0, a_1, a_2, \ldots\}$.

The $a_i$s form an infinite descending chain, so if $\forall b \in A.\, b \prec a \implies P(b)$, that means whenever $b \prec a$ that $b \neq a_i$ for all $i \geq 0$, which must mean $a \notin \{a_0, a_1, a_2, \ldots\}$ as well because otherwise *something* smaller than $a$ would also be in the set. Thus the premise of (1) is satisfied for $P$, yet it is clearly not the case that $\forall a \in A.\, P(a)$, as $\neg P(a_i)$ for all $i \geq 0$. □

This equivalence is deep and important. It immediately proves the equivalence of the principles of strong mathematical induction on $\mathbb{N}$ and the well-ordering of $\mathbb{N}$—that every non-empty subset of $\mathbb{N}$ has a unique least element, which is straightforwardly the same as $<$ being well-founded on $\mathbb{N}$. It is not too hard to show that $<$ is well-founded on $\mathbb{N}$ if and only if predecessor is well-founded (assuming a standard definition of $<$ in terms of predecessor), which then includes the principle of weak induction on $\mathbb{N}$ as being equivalent to both strong induction and well-ordering.

# 5 Well-Founded Induction, Structural Induction, and Recursion

Well-founded induction tells us that structural induction works properly and the recursive functions over inductively-defined data exist and are unique.

Take, for instance, the eval definition above. Define $e \sqsubseteq e'$ to be when $e$ is a *proper* subterm of $e'$. That is, $e$ is a subterm of $e'$ and $e \neq e'$. If we think of ARITH expressions as finite labeled trees where the label is which type of term, then $e$ is a subtree of $e'$. Since the trees are finite, $\sqsubseteq$ is well-founded. Fundamentally, this is why structural induction is mathematically sound.

We can now show that $\mathsf{eval}(e)$ is a total function on ARITH expression. In particular, we can prove by induction on $e$ that for every $e \in \mathbf{AExp}$, there is a unique $n \in \mathbb{Z}$ such that $(e, n) \in \mathsf{eval}$. The base case ($e = \overline{n}$) is clear from the definition of eval. The two inductive cases follow by applying the inductive hypotheses and noting that addition and multiplication in $\mathbb{Z}$ have unique results.

Now that we have a strong understanding of what induction is and why it works, we will be using these ideas and principles liberally throughout the rest of the course.