1. Circle

Proof. We intend to prove that, $\forall e.\, \exists v.\, e \Rightarrow v$. We proceed by induction on the height of e and then by case analysis on the form of e. The expression has two forms: e is either `Duck` e' or it is `Goose`.

In the case that e is `Goose`, the Goose rule gives us that $e \Rightarrow$ `Goose`, so $v =$ `Goose` and this case is finished.

In the case that e is `Duck` e', we note that e' is of lesser height than e. By the inductive hypothesis, we have that $e' \Rightarrow v'$ for some v'. The Duck rule gives us that, as a result, `Duck` $e' \Rightarrow v'$. Since $e =$ `Duck` e', we have $e \Rightarrow v'$ and so this case is finished. □

Observe that, in the above proof, we state what we intend to prove. We also state how we intend to prove it. This allows the reader an understanding of the strategy going forward.

When proving a universal ($\forall$) of a metavariable (e), it is common to operate on the metavariable as if it contains a particular but unknown value. If we are able to prove our point about e without knowing anything about it in advance (other than what we know from definitions, such as the grammar of e here), then our reasoning applies to every e and we have satisfied the $\forall$ part of the proof.

When we encounter a case in which multiple options exist (such as the "form" of e, which refers to the particular case of the grammar that was used to construct it), one strategy is to use case analysis. When we state that the expression has two forms, we are basically writing an `match` expression into our proof: it's enough to prove what we need to prove in each case, even if we have to use different reasoning for each case.

In the `Goose` case, the property is trivial: the `Goose` rule axiomatically says that `Goose` evaluates to a value, so we're finished. We have nothing else to show.

In the `Duck` case, the only rule we can use is the `Duck` rule. That requires that the inner expression of `Duck` also evaluate to something. We stated, though, that we would proceed by induction on the height of e; this means that, when dealing with an e of height n, we are permitted to assume that all e' of height $n-1$ or less[†] already have the property we aim to prove. Since we're trying to show that every e has some v to which it evaluates, this inductive hypothesis lets us assume that e' evaluates to some v' (since e' is of height $n-1$). We can use that as the premise for the `Duck` rule to solve this case.

Once all cases are solved, there is nothing else that needs to be said. If you like, you can summarize what has been proven or assert that all cases are complete. The little □ in the bottom right of the `proof` environment, often pronounced as

the letters "Q.E.D."[‡], indicates that we think the proof is finished.

[†]Strictly speaking, induction on a parametric proposition P allows you to assume that $P(n-1)$ is true while proving $P(n)$; that is to say, we're only really allowed to assume that all e' of height $n-1$ evaluate. This form of induction, sometimes called "weak induction", is often enough when dealing with simple data structures like trees; it's certainly enough here. In some cases, though, you might want this property over *all* smaller trees (such as when the expression is a binary operation). In that case, so-called "strong induction" permits us to assume $P(0)$, $P(1)$, and so on up to $P(n-1)$ when proving $P(n)$. This distinction is rarely important; many proofs simply use "induction" to mean strong induction. However, this tidbit is useful in relating the "$P(n-1)$ implies $P(n)$" perspective of induction that is often given in class to the one used in more intricate proofs.

[‡]Q.E.D. is an initialism for the Latin phrase "quod erat demonstrandum": roughly speaking, "that which was to be demonstrated".

2. Robot

Proof. We intend to prove that, $\forall e.\, \exists v.\, e \Rightarrow v$. We proceed by induction on the height of e and then by case analysis on the form of e. The expression has two forms: e is either `Beep` e_1 e_2 or it is `Boop`.

In the case that e is `Boop`, the Boop rule gives us that $e \Rightarrow$ `Boop`. Thus, $v =$ `Boop` and this case is finished.

In the case that e is `Beep` e_1 e_2, we note that e_1 and e_2 are of lesser height than e. By induction, we have that $e_1 \Rightarrow v_1$ and $e_2 \Rightarrow v_2$. These are the premises of the Beep rule, so we have that `Beep` e_1 $e_2 \Rightarrow v_2$. Thus, $e \Rightarrow v_2$ and this case is finished. □

This proof is essentially the same as the previous proof. The key difference is that the language contains a binary operation, `Beep`, as opposed to the unary operation in the previous language, `Duck`. As a consequence, we require *strong* induction: we show that, given $P(0)$, $P(1)$, and so on up to $P(n-1)$ that $P(n)$ is true. This is necessary because it is not always the case that the subexpressions of `Beep` have the same height; for instance, in the expression `Beep Boop (Beep Boop Boop)`, the left subtree has height 0 while the right subtree has height 1.

In this proof, we simply use the term "induction" to refer to the strong induction technique. This is common in proofs over ASTs and similar trees; the author is essentially assuming that the reader is able to infer the induction technique (much like the phrasing "proceed by induction on the height of e" assumes that the reader can figure out what P is).

3. CoinFlip

Proof. We intend to prove that, if $e \overset{\text{H}}{\Rightarrow} v$, then $\texttt{Flip}\ e \overset{\text{T}}{\Rightarrow} v$. We proceed by induction on the height of e and then case analysis on the form of e. The expression has two forms: e is either `Stop` or $\texttt{Flip}\ e'$.

If e is `Stop` then $e \overset{\text{H}}{\Rightarrow} \texttt{Heads}$ by the Stop rule of $\overset{\text{H}}{\Rightarrow}$. In this case, $\texttt{Flip}\ e$ is $\texttt{Flip Stop}$. $\texttt{Flip Stop} \overset{\text{T}}{\Rightarrow} \texttt{Heads}$ by the following proof:

$$\textsc{Flip}\ \dfrac{\textsc{Stop}\ \dfrac{}{\texttt{Stop} \overset{\text{T}}{\Rightarrow} \texttt{Tails}}}{\texttt{Flip Stop} \overset{\text{T}}{\Rightarrow} \texttt{Heads}}$$

We have that e is `Stop`, $e \overset{\text{H}}{\Rightarrow} \texttt{Heads}$, and $\texttt{Flip}\ e \overset{\text{T}}{\Rightarrow} \texttt{Heads}$; thus, this case is finished.

Otherwise, e is $\texttt{Flip}\ e'$. By the inductive hypothesis, we have that $e' \overset{\text{H}}{\Rightarrow} v'$ such that $\texttt{Flip}\ e' \overset{\text{T}}{\Rightarrow} v'$. We have $e = \texttt{Flip}\ e'$, so we know $e \overset{\text{T}}{\Rightarrow} v'$. By $e' \overset{\text{H}}{\Rightarrow} v'$ and the Flip rule of $\overset{\text{H}}{\Rightarrow}$, we have $e \overset{\text{H}}{\Rightarrow} v$ where v is the opposite of v'. By $e \overset{\text{T}}{\Rightarrow} v'$ and the Flip rule of $\overset{\text{T}}{\Rightarrow}$, we have $\texttt{Flip}\ e \overset{\text{T}}{\Rightarrow} v$. In summary, $e \overset{\text{H}}{\Rightarrow} v$ and $\texttt{Flip}\ e \overset{\text{T}}{\Rightarrow} v$, so this case is finished. □

> This case is a little different because *two* proof systems are involved. One proof system, $\overset{\text{H}}{\Rightarrow}$, evaluates `Stop` to `Heads` while the other, $\overset{\text{T}}{\Rightarrow}$, evaluates `Stop` to `Tails`. As a consequence of using two proof systems, we must be mindful about which system's rule we are discussing at any given point in time.
>
> The other thing that's a little different here is the form of the proposition to be proven. The previous propositions have simply started with "for all e"; in those cases, we know nothing about the e. In this case, we are trying to make an assertion about an implication: *if* $e \overset{\text{H}}{\Rightarrow} v$ *then* $\texttt{Flip}\ e \overset{\text{T}}{\Rightarrow} v$. As a consequence, we have an additional fact — $e \overset{\text{H}}{\Rightarrow} v$ — which we can assume in order to reach our goal.

4. Addsolute

Proof. We intend to prove that $e \Rightarrow v$ and $e \Rightarrow v'$ then $v = v'$. We proceed by induction on the height of the proof trees and then by case analysis on the proof rule used in the first proof.

If the Value rule is used in the first proof, then $e = v$. In that case, the only rule which applies to v is the Value rule, so the Value rule must have been used in the second proof. As a result, $e = v = v'$ and this case is finished.

If the Negate rule is used in the first proof, then $e = \texttt{-}e_1$. By the Negate rule, we have that $e_1 \Rightarrow v_1$. Since the Negate rule is the only rule which applies to e, we know that the second proof must use the negate rule and has $e_1 \Rightarrow v_1'$. Because the proofs of $e_1 \Rightarrow v_1$ and $e_1 \Rightarrow v_1'$ have lesser height than the proofs of $e \Rightarrow v$ and $e \Rightarrow v'$, the inductive hypothesis gives us that $v_1 = v_1'$. In this case, v and v' are both the arithmetic negation of v_1 and so this case is finished.

If the Plus rule is used in the first proof, then $e = e_1 \texttt{ + } e_2$. This case proceeds much like the Negate case above. We know from the first proof that $e_1 \Rightarrow v_1$ and $e_2 \Rightarrow v_2$; we also know that the only rule which applies to e is the Plus rule, so it must have been used in the second proof as well. By the Plus rule from the second proof, we have $e_1 \Rightarrow v_1'$ and $e_2 \Rightarrow v_2'$. By the inductive hypothesis, we have $v_1 = v_1'$ and $v_2 = v_2'$, so both v and v' are the arithmetic sum of v_1 and v_2. As a result, this case is finished.

If the Abs Pos rule is used in the first proof, then $e = \texttt{Abs } e_1$. We know from this rule that $e_1 \Rightarrow v$ and that $v \geq 0$. In this case, there are exactly two rules which match the syntax of e and so be used in the second proof: the Abs Pos rule or the Abs Neg rule. We will show that the Abs Neg rule cannot be used.

Suppose for contradiction that the second proof uses the Abs Neg rule. Then we know from its premises that $e_1 \Rightarrow v_1'$, $v_1' < 0$, and $\texttt{-}e_1' \Rightarrow v'$. Because e_1 is a smaller expression, we have by the inductive hypothesis that $v_1 = v_1'$. But then $v_1 < 0$ and $v_1 \geq 0$, which is a contradiction; as a result, the Abs Neg rule cannot be used in this case.

So if the Abs Pos rule is used in the first proof, the Abs Pos rule must be used in the second proof. As a consequence, we have $e_1 \Rightarrow v'$ and $v' \geq 0$. By the inductive hypothesis, we have $v = v'$ and so this case is finished.

Otherwise, the Abs Neg rule is used in the first proof. This case proceeds in the same fashion as the Abs Pos case, using induction for the final premise of the Abs Neg rule. □

This problem is similar to the previous problem in that there are *two proofs* (although here, those two proofs are from the same operational semantics). A new challenge, though, is that we can't prove this by induction on the height of e[†]. If we tried to do so, then the Abs Neg rule would present a problem: for an expression $e =$ `Abs` e' of height n, the premise is that the expression `-`e evaluates but that expression also has height n and so we cannot use the inductive hypothesis.

There are a number of ways to address this; one of them is to proceed by induction on the height of the *proof* rather than the height of the *expression*. In this strategy, we look at each *rule* in the proof (rather than each *node* in the AST) and perform case analysis on the rule which was used. By using this approach, it doesn't matter if the expression gets smaller, since the proof always will. In the Abs Pos case, for instance, we observe that the *proof* of `-`$e_1 \Rightarrow v$ is smaller than the *proof* of `Abs` $e \Rightarrow v$. This is especially important because the *expression* `-`e is *not* smaller than the expression `Abs` e and so induction on the height of the expression would not be sound here.

The other novelty about this proof system is that the rules are not syntactically unique: there are two rules to handle the `Abs` e expression. This is much like the `If` expression in F♭. We can't know which of those two rules was used in the second proof, so we have to consider both cases. Here, we start by pretending that the Abs Neg rule was used. We then follow sound logical steps to prove something impossible: that v is both negative and non-negative at the same time! Because everything we did was valid, the original assumption – that the Abs Neg rule was used – must be false. This is why we can determine that the second proof *also* used the Abs Pos rule.

[†]At least, we can't prove this by the height of e quite so directly. Strictly speaking, it's still possible to do so, but it doesn't proceed in quite the same fashion and it's a little bit messy. Instead, we use the height of the proof tree, both because it solves this problem and it makes for a good demonstration of technique. (The somewhat peculiar form of the Abs Neg rule is chosen to motivate this decision as well; most cases in which induction on a proof tree are natural involve more complex rule sets.)