Announcements

Structural Induction

Boolean terms: Syntax

Recall the definition of the language \( B \):

This was a shorthand notation for the definition of the set \( B \).

If \( t \) is a Boolean term, then:

- \( t \) is \( \text{true} \)
- \( t \) is \( \text{false} \)
- \( t \) is \( \text{true} \) \( \text{false} \)

The set of Boolean terms is the smallest set such that:

1. \( \text{true} \), \( \text{false} \) \( \text{true} \) \( \text{false} \)

First homework assignment is due one week from today.

Fri 9:30-11 AM Levine 712
Fri 12:30-2:30 PM Levine 712
Fri 3:30-5:30 PM Levine 712

Wed 9:30-11 AM Bohannon 101
Wed 12:30-2:30 PM Bohannon 101
Wed 3:30-5:30 PM Bohannon 101

If you need help finding a study group, we will match people up in recitation sections this week.

Recruit as many people as you can.

Induction; Operational Semantics

Fall 2005
Software Foundations
CIS 500
Boolean terms: Semantics

We defined the semantics of Boolean terms using the relation $\text{Eval}$. If $(t_1, t_2) \in \text{Eval}$, then $t_2$ is the meaning of $t_1$. Recall that $\text{Eval}$ is the smallest set closed under the following rules:

1. $(\text{true}, \text{true}) \in \text{Eval}$
2. $(\text{false}, \text{false}) \in \text{Eval}$
3. $(\text{not} t, \text{true}) \in \text{Eval}$ when $(t, \text{false}) \in \text{Eval}$
4. $(\text{not} t, \text{false}) \in \text{Eval}$ when $(t, \text{true}) \in \text{Eval}$
5. $(\text{if} t_1 \text{then} t_2 \text{else} t_3) \in \text{Eval}$ when $(t_1, \text{true}) \in \text{Eval}$ and $(t_2, \text{false}) \in \text{Eval}$ and $(t_3, \text{false}) \in \text{Eval}$

These rules define $\text{Eval}$ as the set of semantic values for Boolean terms.

Proving properties of programming languages

Structural induction

We can use induction for boolean terms. The way we have defined terms gives us an induction principle:

For all $t \in B$, $P(t)$ holds.

This gives us the property:

exists at most one $t \in B$ such that $(t, t_0) \in \text{Eval}$

We prove this using induction. In other words, for all $t$ there exists at most one $t_0$ such that $(t, t_0) \in \text{Eval}$.

Proofs by induction

Suppose we want to prove that evaluation is deterministic. In other words, for all $t$, there exists at most one $t_0$ such that $(t, t_0) \in \text{Eval}$.

We'll prove this by induction on $t$. The way we have defined terms gives us an induction principle:

For all $t \in B$, $P(t)$ holds.

This gives us the property:

exists at most one $t \in B$ such that $(t, t_0) \in \text{Eval}$

We prove this using induction on $t$. In other words, for all $t$ there exists at most one $t_0$ such that $(t, t_0) \in \text{Eval}$.

Proofs of properties of programming languages

We proved the semantic rules of Boolean terms: Semantics
Boolean terms: Semantics

We defined the semantics of \( B \) using the relation \( \text{Eval} \). Recall that \( \text{Eval} \) is the smallest set closed under the following rules:

1. \((\text{true}, \text{true}) \in \text{Eval}\)
2. \((\text{false}, \text{false}) \in \text{Eval}\)
3. \((\text{not}, \text{true}) \in \text{Eval} \) when \((\text{true}, \text{true}) \in \text{Eval}\)
4. \((\text{not}, \text{false}) \in \text{Eval} \) when \((\text{false}, \text{false}) \in \text{Eval}\)
5. \((\text{if}, \text{then}, \text{else}, \text{true}) \in \text{Eval} \) when either:
   - \((\text{true}, \text{true}) \in \text{Eval}\)
   - \((\text{true}, \text{false}) \in \text{Eval}\)
   - \((\text{false}, \text{true}) \in \text{Eval}\)
   - \((\text{false}, \text{false}) \in \text{Eval}\)
   - \((\text{true}, \text{false}) \in \text{Eval}\)
   - \((\text{true}, \text{true}) \in \text{Eval}\)

Alternation: Inference rules

We can also define \( \text{Eval} \) using a shorthand notation. An alternate notation for the same definition:

1. \((\text{true}, \text{true}) \in \text{Eval}\)
2. \((\text{false}, \text{false}) \in \text{Eval}\)
3. \((\text{not}, \text{true}) \in \text{Eval} \) when \((\text{true}, \text{true}) \in \text{Eval}\)
4. \((\text{not}, \text{false}) \in \text{Eval} \) when \((\text{false}, \text{false}) \in \text{Eval}\)
5. \((\text{if}, \text{then}, \text{else}, \text{true}) \in \text{Eval} \) when either:
   - \((\text{true}, \text{true}) \in \text{Eval}\)
   - \((\text{true}, \text{false}) \in \text{Eval}\)
   - \((\text{false}, \text{true}) \in \text{Eval}\)
   - \((\text{false}, \text{false}) \in \text{Eval}\)
   - \((\text{true}, \text{false}) \in \text{Eval}\)
   - \((\text{true}, \text{true}) \in \text{Eval}\)

Alternation: relation symbols

If we abbreviate \((t_1, t) \in \text{Eval} \) as \( t \downarrow t \), we can write these rules even more succinctly:

1. \(\text{true} \downarrow \text{true}\)
2. \(\text{false} \downarrow \text{false}\)
3. \(\text{if} t_1 \text{then} t_2 \text{else} t_3 \downarrow t\)

Note that, just in the BNF notation, “the smallest set closed under...” is implied (but often not stated explicitly).
Naming the rules

It is also useful to give names to each rule, so that we can refer to them later.

Derivations

Use structural induction

To show that evaluation is total, we need \( t \in B \) and \( t \in B \) such that \( t = t + t_0 \).

For all \( t, t_1, t_2 \in B \), if \( P(t_1) \) holds, then \( P(t_2) \) holds.

For all \( t \in B \), \( t \in B \).

How to prove this property?

Last time we showed that the evaluation relation was a function.

Proving properties about evaluation

Use structural induction

For all \( t \in B \), \( t \in B \).

For all \( t \), \( t \).

Use structural induction

Derivations

If \( t_1 \), then \( t_2 \), else \( t_3 \).
Strengthening the induction principle

We cannot show that $P(n)$ is true for $n = 1$, given $P(t_1)$. $P(t_1)$ tells us that $t_1$ evaluates to some $t$, but $t$ not $t_1$ only evaluates if $t_1$ is true of $t_1$. We cannot show that $P(t_1)$ is true.

What to do now? Are we stuck?

To show the second property we need $P(t)$ to be "either $t$ true or $t$ false". To prove the second property implies that the first one is also true.

For all $t$ either $t$ true or $t$ false

Instead of showing $t$ true or false, and we don't know that.

"there exists a $t$ such that $t$ true or false"

The solution is to prove a property that implies the property that we want.

Growing a language

A larger language

The boolean language is an extremely simple language. There is not a lot you can say with it. You can say with it $t + t_0$.

The boolean language is an extremely simple language. There is not a lot you can do with it.

CIS 500, Induction; Operational Semantics

CIS 500, Induction; Operational Semantics
Consider a larger language, called Arith, that includes both booleans and natural numbers:

\[
\begin{align*}
\text{t} &::= \text{true} \\
\text{false} &::= \text{false} \\
\text{if} &::= \text{if } t \text{ then } t \text{ else } t \\
\text{succ} &::= \text{succ} \text{ if } t \text{ else } t \\
\text{pred} &::= \text{pred} \text{ if } t \text{ else } t \\
\text{iszero} &::= \text{iszero} \text{ if } t \text{ else } t
\end{align*}
\]

What is the structural induction principle for this language?

Semanticsof Arith

We use the metavariable \( \alpha \) to indicate terms that are also values.

\[
\begin{align*}
\text{true} + \alpha &::= \text{true} \\
\text{false} + \alpha &::= \text{false} \\
\text{if } t \text{ then } t \text{ else } t + \alpha &::= \text{if } t \text{ then } t + \alpha \text{ else } \text{if } t \text{ then } t + \alpha \text{ else } t
\end{align*}
\]

These are called the values.

Arith then will do the result of evaluation.

To define the semantics of Arith, we will first define a subset of the terms of Arith that will do the result of evaluation.

Languagedefinability (informally)

Note: we are overloading the symbol \( \uparrow \) to refer to two different relations.

Semanticsof Arith

We are shorter:

Learning not means that our induction principle (and therefore our proofs) are shorter with not \( t \) as true for Arith without not \( t \).

However, all is not lost. Whenever we want to say not \( t \), we can write:

This language does not include the term \( \text{false} \).

Consider a larger language, called Arith, that includes both booleans and

natural numbers:

The Language Arith
Some terms, like succ false, are "meaningless" in our semantics.

If we try to use induction to show this theorem, where does the proof break?

There is a counterexample to this theorem. What does succ false evaluate to?

Evaluation is local for all $t, t'$.

Evaluation is not total.

Properties of Arith

We showed that two properties were true of $B$, are these same properties true of $\text{Arith}$?

Evaluation is deterministic: for all $t$, there is at most one $t'$ such that $t' \downarrow$.

Evaluation is total: for all $t$, $t \downarrow$. What if we phrase it as:

Evaluation is total: for all $t$, $t \downarrow v$.

Evaluation is not total.

Metavariables are useful:

New rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Truth Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \downarrow v$</td>
<td>$v$</td>
</tr>
</tbody>
</table>
It's a little unsettling that evaluation isn't total. We want to give meanings to all terms. We want to (abstractly) describe the execution of a computer. Later: some languages contain infinite loops. Those terms won't have meanings with this style of semantics either. Later: some languages contain infinite loops.

Want to distinguish loops from errors like succfalse. We need to describe the "intermediate" steps of evaluation of an abstract machine. A transition relation on states, written \( t \rightarrow t' \). A set of states. An abstract machine consists of:

- Core ideas: describe the "intermediate" steps of evaluation of an abstract machine. Small-step operational semantics. Small-step evaluation is the one step execution of the abstract machine.
- Multi-step evaluation is the reflexive, transitive closure of small-step evaluation.

\( t \rightarrow t' \) is total (because of reflexivity). Small-step evaluation is the reflexive, transitive closure of small-step evaluation.

\( t \rightarrow t' \rightarrow t'' \) is not total (because of "stuck"). It may not be total (when the machine gets "stuck").
A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term that is in normal form is a value, i.e., a term.

A term is a normal form if it is not reducible to any other terms.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

A term is a normal form if it is not reducible to any other terms.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.

The meaning of a term $t$ with small-step semantics is a term $t_0$, such that

\[ t \rightarrow t_0 \]

is the normal form of $t$. We say that $t_0$, is the normal form of $t$.

A term is a normal form if it is not reducible to any other terms.

A normal form is a term that cannot be evaluated any further. It is a value, i.e., a term.

A normal form is a term that is in normal form. It is a value, i.e., a term.
Suppose we wanted to change our evaluation strategy so that the `then` and `else` branches of an `if` expression (in that order) before the guard. How would we need to change the rule?

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics

Properties of this semantics
We've seen three definitions of sets and their associated induction principles. Given a set defined with BNF, it is not too hard to describe the structural induction principle for that set. For example:

```
\text{if } t \text{ then } t' \text{ else } t'' \quad \text{iff}\quad t = 0\text{ or } t = 1\text{ or } t = \text{false}
```

For these sets also have induction principles.

We defined the semantics of these languages using relations, and relations are just sets. Theses sets also have induction principles.

However, these are not the only sets that we've defined so far.

We can define an induction principle for small-step evaluation. Recall the definition (just for booleans, for now):

```
t::= \text{brillig} 
t::= \text{tovesnickert} 
gyretgimblet
```

What is the structural induction principle for this language?

We've seen three definitions of sets and their associated induction principles.
Using this induction principle

For all $t$, $t_0$,

- $P(t)$ if $true$ then $t_2$ else $t_3$.
- $P(t)$ if $false$ then $t_2$ else $t_3$.
- $P(t)$ if $t_1$ then $t_2$ else $t_3$.

Given that $P(t_1)$ then $t_0$.

What does it mean to say $P(t)$ if $t_1$ then $t_2$ else $t_3$?

CIS 500, Induction; Operational Semantics
Forexample,wecanshowthatsmall-stepevaluationisdeterministic.

Theorem:If \( t \rightarrow t_0 \) then if \( t \rightarrow t_{00} \) then \( t_0 = t_{00} \).

Proof: By induction on a derivation \( D \) of \( t \rightarrow t_0 \).

1. Suppose the last rule used in \( D \) is E-IfTrue, with \( t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \) and \( t_1 = \text{true} \) and \( t_0 = t_2 \). This case is similar to the previous.

2. Suppose the last rule used in \( D \) is E-IfFalse, with \( t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \) and \( t_1 = \text{false} \) and \( t_0 = t_3 \). This case is similar to the previous.

3. Suppose the last rule used in \( D \) is E-If, with \( t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \) and \( t_1 \rightarrow t_0 \). By induction \( t_1 = t' \) can only be E-If, so it must be that \( t_1 \rightarrow t' \) and \( t' \rightarrow t_0 \). The last rule in the derivation of \( t \rightarrow t' \) can only be E-If, so it must be that \( t \rightarrow t' \) and \( t' \rightarrow t_0 \).

Well-founded induction

We've proved the same theorem using two different induction principles.

What principle to use?
A question: Why are any of these induction principles true? Why should I believe a proof that employs one?

**Well-founded induction**

Well-founded induction is a generalized form of all of these induction principles. Let $\prec$ be a well-founded relation on a set $A$. Let $d$ be a property. Then $d$ is a well-founded induction if $\forall a \in A. \ d(a) \iff \forall a \prec b \ a \ d(b)$. This is a more general form of all of these induction principles.

**Strong induction**

If $\prec$ is the strictly less than relation, then the principle we get is strong induction.

**Induction**

If $\prec$ is the strictly less than relation then the principle we get is strong induction.

## Simplify:

\[
\begin{align*}
(1 + 1) d & \iff (1) d \lor \forall a \ N \ E \ A \lor (0) d \\
& \iff (a) d \lor (a) d' \ N \ E \ A \lor (a) d' \ N' \ E \ A
\end{align*}
\]

Now, by definition $a$ is either 0 or $1$. For some $i$.

\[
\begin{align*}
(1 + 1) d & \iff (\exists a \ N \ E \ A \lor (a) d \lor (a) d') \lor (0) d \\
& \iff (\exists a \ N \ E \ A \lor (a) d \lor (a) d') \lor (a) d' \ N \ E \ A
\end{align*}
\]

Simplify to:

\[
\begin{align*}
(1 + 1) d & \iff (\exists a \ N \ E \ A \lor (a) d \lor (a) d') \lor (0) d \\
& \iff (\exists a \ N \ E \ A \lor (a) d \lor (a) d') \lor (a) d' \ N \ E \ A
\end{align*}
\]

Well-founded induction

**Strong induction**

If $\prec$ is the strictly less than relation, then the principle we get is strong induction.

**Induction**

If $\prec$ is the strictly less than relation then the principle we get is strong induction.

**Induction**

If $\prec$ is the strictly less than relation then the principle we get is strong induction.
Well-founded relation

The induction principle holds only when the relation is well-founded.

Definition:
A well-founded relation is a binary relation on a set such that there are no infinite descending chains.

Well-founded induction

The induction principle holds only when the relation is well-founded.

Proof of well-founded induction

We'd like to show that:

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

The $(\Rightarrow)$ direction is trivial. We'll show the $(\Leftarrow)$ direction.

Theorem: Let $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

We'd like to show that

Well-founded induction

Structural induction

Well-founded induction also generalizes structural induction.

If $\rightarrow$ is the "immediate subterm" relation for an inductively defined set, then the principle we get is structural induction.

For example, in Arith, the relation $\rightarrow_1$ is an immediate subterm relation.

If $\rightarrow$ is the "immediate subterm" relation for an inductively defined set, then

The principle we get is structural induction.

Well-founded induction also generalizes structural induction.

For example, in Arith, the relation $\rightarrow_1$ is an immediate subterm relation.

If $\rightarrow$ is the "immediate subterm" relation for an inductively defined set, then

Well-founded induction also generalizes structural induction.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.

Theorem: Let $\prec$ be a well-founded relation on a set $A$. Then $A \subseteq \{x \in A : \forall y \prec x \exists z \prec y \}$.
Termination of evaluation

Theorem: For every \( t \) there is some normal form \( t_0 \) such that \( t \rightarrow^* t_0 \).

Proof: For every \( t \) there is some normal form \( t_0 \) such that \( t \rightarrow^* t_0 \).

Induction on derivations

Another example

Theorem: If \( t \rightarrow^* t_0 \) then by the definition of \( \mathrm{size} \), we have \( \mathrm{size}(t) < \mathrm{size}(t_0) \).

Definition. By the induction hypothesis, \( \mathrm{size}(t_1) < \mathrm{size}(t_2) \) and \( t \rightarrow^* t_1 \rightarrow^* t_2 \). Suppose the final rule used in \( t \) is \( \mathrm{E-IfTrue} \). Then the result is immediate from the definition of \( \mathrm{size} \).

Inference rules?

Note: This is yet another shorthand. How would we write the definition with inference rules?

We can define the \( \mathrm{size} \) of a term with the following definition:

An Inductive Definition

Termination of evaluation
Termination of evaluation

Theorem: For every \( t \) there is some normal form \( t_0 \) such that \( t \rightarrow^* t_0 \).

Proof: First, recall that single-step evaluation strictly reduces the size of the term.

\[ \text{Theorem: For every } t \text{ there is some normal form } t_0 \text{ such that } t \rightarrow^* t_0. \]

The proof involves constructing an infinite descending chain in \( W \), which is a well-founded set.

1. Choose a well-founded set \( W \), i.e., a set with a partial order \( \prec \).

2. Define a function \( f \) from \( X \) to \( W \) such that \( f(x) \prec f(y) \) for all \( x, y \in X \).

3. Show \( f(x) \prec f(y) \) implies \( x \rightarrow y \).

4. Conclude that \( f(x) \prec f(y) \) for all \( x, y \in X \).

5. Conclude that there are no infinite descending chains in \( W \).

6. Assume (for a contradiction) that there is an infinite sequence such that \( \ldots \rightarrow f(x_0) \rightarrow f(x_1) \rightarrow f(x_2) \rightarrow \ldots \).

7. Construct an infinite descending chain in \( W \).

But such a sequence cannot exist — contradiction.

\[ \text{Proof: For every } t \text{ there is some normal form } t_0 \text{ such that } t \rightarrow^* t_0. \]