INDUCTIVELY GENERATED DATA

2024 1

Generative processes

- Virtually every infinite set considered in programming is ***generated by a process***.

  The fundamental example is the set $\mathbb{N}$ of natural numbers:

  - **Initial object** ("base"): The number 0 is in $\mathbb{N}$.

Generative processes

- Virtually every infinite set considered in programming is ***generated by a process***.

  The fundamental example is the set $\mathbb{N}$ of natural numbers:

  - **Initial object** ("base"): The number 0 is in $\mathbb{N}$.

    Generative step: If $n \in N$ then "next" of n, $\mathrm{s}\, n$, is $\in \mathbb{N}$

Generative processes

- Virtually every infinite set considered in programming
  is ***generated by a process***.

  The fundamental example is the set $\mathbb{N}$ of natural numbers:

  - **Initial object** ("base"): The number 0 is in $\mathbb{N}$.

    Generative step: If $n \in N$ then "next" of n, $\mathbf{s}\, n$, is $\in \mathbb{N}$

- Implicit assumptions:

  The meanings of 0 and "next" are known and given.

Generating $\{0, 1\}^*$

- **Base.** The ***empty string*** is in $\{0, 1\}^*$.
- **Generative step.**
  If $w \in \{0, 1\}^*$ then $0\,w$ and $1\,w$ are $\in \{0, 1\}^*$

Generating $\{0,1\}^*$

- **Base.** The ***empty string*** is in $\{0,1\}^*$.
- **Generative step.**
  If $w \in \{0,1\}^*$ then $0\,w$ and $1\,w$ are $\in \{0,1\}^*$

- Implicit assumptions:

  The meanings of the empty string
  and of juxtaposition are known.

Generating $\{0,1\}^*$

- **Base.** The ***empty string*** is in $\{0,1\}^*$.
- **Generative step.**
  If $w \in \{0,1\}^*$ then $0\,w$ and $1\,w$ are $\in \{0,1\}^*$

• Implicit assumptions:

The meanings of the empty string
and of juxtaposition are known.

• Note: We generate strings here "from the head";
This conforms with the general use of constructors,
and reflected in the functions *head* and *tail*.

Format of generative definitions

- Two parts in a generative dfn of set S:
  - **Base:**
    Particular known objects are in S.

Format of generative definitions

- Two parts in a generative dfn of set S:
  - **Base:**
    Particular known objects are in S.
  - **Generative steps:**
    If certain objects are in S
    then so are certain objects obtained from those.

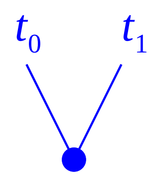
Another example: Binary trees

- $\boxed{\textit{Binary tree}}$ means here a
  finite, ordered, unlabeled binary tree

Base: The singleton tree • is in $\mathbf{BT}$.

Generative step:

If $t_0,\ t_1$ are binary trees then so is



Implicit assumptions:

We know what a singleton tree and juncture of trees mean.

Try this...

- Generate the set E of even natural numbers.

Try this...

- Generate the set E of even natural numbers.
  - Base: 0
  - Generative step: If $n \in E$ then $n{-}2 \in E$.

Boolean terms

- Two ways to define closed boolean terms:

Boolean terms

- Two ways to define closed boolean terms:
- $\mathbf{IBT}$: Infix boolean terms:
  - `0` and `1` are in $\mathbf{IBT}$
  - If $t, t' \in \mathbf{IBT}$ then $(t) \wedge (t') \in \mathbf{IBT}$ and $(t) \vee (t') \in \mathbf{IBT}$

Boolean terms

- Two ways to define closed boolean terms:
- $\mathbf{IBT}$: Infix boolean terms:
  - `0` and `1` are in $\mathbf{IBT}$
  - If $t, t' \in \mathbf{IBT}$ then $(t) \wedge (t') \in \mathbf{IBT}$ and $(t) \vee (t') \in \mathbf{IBT}$
- $\mathbf{PBT}$: Prefix boolean terms:
  - `0` and `1` are in $\mathbf{PBT}$
  - If $t, t' \in \mathbf{PBT}$ then $\wedge\, t\, t' \in \mathbf{PBT}$ and $\vee\, t\, t' \in \mathbf{PBT}$

Boolean terms

- Two ways to define closed boolean terms:
- $\mathbf{IBT}$: Infix boolean terms:
  - `0` and `1` are in $\mathbf{IBT}$
  - If $t, t' \in \mathbf{IBT}$ then $(t) \wedge (t') \in \mathbf{IBT}$ and $(t) \vee (t') \in \mathbf{IBT}$
- $\mathbf{PBT}$: Prefix boolean terms:
  - `0` and `1` are in $\mathbf{PBT}$
  - If $t, t' \in \mathbf{PBT}$ then $\wedge\, t\, t' \in \mathbf{PBT}$ and $\vee\, t\, t' \in \mathbf{PBT}$
- Main difference between IBT and PBT:

  No parentheses in PBT !

Lists of natural numbers

- Generate $\mathbb{L}(\mathbb{N})$ the $\boxed{\textit{lists of natural numbers}}$.
- Fix a textual coding of $\mathbb{N}$, say binary numerals.

Lists of natural numbers

- Generate $\mathbb{L}(\mathbb{N})$ the $\boxed{\textit{lists of natural numbers}}$.
- Fix a textual coding of $\mathbb{N}$, say binary numerals.
  - $\square$ is a list of naturals.
  - If ℓ is a list and k a numeral then $k : \ell$ is a list.

Lists of natural numbers

- Generate $\mathbb{L}(\mathbb{N})$ the $\boxed{\textit{lists of natural numbers}}$.
- Fix a textual coding of $\mathbb{N}$, say binary numerals.
  - $\square$ is a list of naturals.
  - If ℓ is a list and k a numeral then $k : \ell$ is a list.
- Examples: $1 : \square$, $0 : 101 : 10011 : 10 : \square$.

REASONING ABOUT INDUCTIVE DATA

2024 9

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0, 1\}^*$ is infinite.

  But our minds and our proofs are finite.

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0,1\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about* $\mathbb{N}$ *?*

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0,1\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about $\mathbb{N}$?*

- Trying many cases is never sufficient.

  Example: *"For all n at least one of 2^n+1 and 2^n-1 is prime."*

  $2, 3, 5, 7, 17, 31$ Hooray!

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0, 1\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about* $\mathbb{N}$ *?*

- Trying many cases is never sufficient.

  Example: *"For all n at least one of $2^n + 1$ and $2^n - 1$ is prime."*

  $2, 3, 5, 7, 17, 31$ Hooray!

  Oops: Both 63 and 65 are composite.

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{\mathtt{0},\mathtt{1}\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about $\mathbb{N}$?*

- Trying many cases is never sufficient.

  Example: *"For all n at least one of 2^n+1 and 2^n-1 is prime."*

  $2,3,5,7,17,31$ Hooray!

  Oops: Both 63 and 65 are composite.

- Another try (Fermat): *All numbers $2^{2^n}+1$ are prime*

  $3,5,17,257,65537$. Yahoo!

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0, 1\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about $\mathbb{N}$?*

- Trying many cases is never sufficient.

  Example: *"For all n at least one of $2^n + 1$ and $2^n - 1$ is prime."*

  $2, 3, 5, 7, 17, 31$ Hooray!

  Oops: Both 63 and 65 are composite.

- Another try (Fermat): *All numbers $2^{2^n} + 1$ are prime*

  $3, 5, 17, 257, 65537$. Yahoo!

  Oops (Euler): Next one is 4,294,967,297, which is divisible by 641.

Infinite sets, finite minds

- $\mathbb{N}$ is infinite. $\{0,1\}^*$ is infinite.

  But our minds and our proofs are finite.

- *So how can we prove anything about $\mathbb{N}$?*

- Trying many cases is never sufficient.

  Example: *"For all n at least one of 2^n+1 and 2^n-1 is prime."*

  $2, 3, 5, 7, 17, 31$ Hooray!

  Oops: Both 63 and 65 are composite.

- Another try (Fermat): *All numbers $2^{2^n}+1$ are prime*

  $3, 5, 17, 257, 65537$. Yahoo!

  Oops (Euler): Next one is 4,294,967,297, which is divisible by 641.

- *So how can we hope to prove*

  that all natural numbers are such-and-such ?

Finitely generated infinities!

- The secret is that inductive data is generated by ***finite rules.***
- Therefore we have a finite tool for proving that all generated objects satisfy certain properties.

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 ***1***

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 1 2

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 1 2 3

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 1 2 3 4

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 1 2 3 4

- They all come out green:
  As we generate $\mathbb{N}$ we make sure that we start with green,
  and that each step maintains green-ness.

Following the process

- Suppose we generate $\mathbb{N}$ using a green pen.
  - 0 is a green natural.
  - If x is a green natural, then so is its successor.

0 1 2 3 4

- They all come out green:
  As we generate $\mathbb{N}$ we make sure that we start with green,
  and that each step maintains green-ness.
- Green-ness is here the process' ***invariant:***
  True at the outset, and preserved by the steps.

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.

  $P(x)$ abbreviates here *"x has the property P"*

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.
  $P(x)$ abbreviates here *"x has the property P"*
- Assume:
  - ***Base.*** $P(0)$ and
  - ***Step.*** For all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$.

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.
  $P(x)$ abbreviates here *"x has the property P"*
- Assume:
  - ***Base.*** $P(0)$ and
  - ***Step.*** For all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$.
- Conclude: $P(x)$ for all $x \in \mathbb{N}$.

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.
  $P(x)$ abbreviates here "*x has the property P*"
- Assume:
  - ***Base.*** $P(0)$ and
  - ***Step.*** For all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$.
- Conclude: $P(x)$ for all $x \in \mathbb{N}$.
- As natural numbers are being generated,
  they all come out satisfying P.

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.
  $P(x)$ abbreviates here *"x has the property P"*

- Assume:

  - ***Base.*** $P(0)$ and

  - ***Step.*** For all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$.

- Conclude: $P(x)$ for all $x \in \mathbb{N}$.

- As natural numbers are being generated,
  they all come out satisfying P.

- *A property of natural numbers that holds for zero*
  and is invariant under successor
  is true of every natural number.

The principle of induction for $\mathbb{N}$

- Suppose $P(x)$ is a property of natural numbers x.
  $P(x)$ abbreviates here *"x has the property P"*
- Assume:
  - ***Base.*** $P(0)$ and
  - ***Step.*** For all $n \in \mathbb{N}$, $P(n)$ implies $P(n+1)$.
- Conclude: $P(x)$ for all $x \in \mathbb{N}$.
- As natural numbers are being generated,
  they all come out satisfying P.
- *A property of natural numbers that holds for zero*
  and is invariant under successor
  is true of every natural number.
- The premise of the STEP is often called the "induction assumption"
  or the Induction Hypothesis (IH).

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$. What is the property?

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- $P(x)$ is $2^x < 2^{x+1}$

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- If we know that
  - $2^x < 2^{x+1}$ is true for $x = 0$; and
  - $2^x < 2^{x+1}$ for $x = n$
    implies that $2^x < 2^{x+1}$ for $x = n + 1$
- then $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- If we know that
  - $2^x < 2^{x+1}$ is true for $x = 0$; and
  - $2^x < 2^{x+1}$ for $x = n$
    implies that $2^x < 2^{x+1}$ for $x = n + 1$
- then $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- But we do have
  - Base: $2^0 = 1 < 2 = 2^{0+1}$

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- If we know that
  - $2^x < 2^{x+1}$ is true for $x = 0$; and
  - $2^x < 2^{x+1}$ for $x = n$
    implies that $2^x < 2^{x+1}$ for $x = n + 1$
- then $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- But we do have
  - Base: $2^0 = 1 < 2 = 2^{0+1}$
  - Step: If $2^n < 2^{n+1}$ ($P(x)$ for $x = n$) then
    $2^{n+1} = 2^n + 2^n < 2^{n+1} + 2^{n+1} = 2^{n+2}$
    ($P(x)$ for $x = n{+}1$)

Example

- Show that $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- If we know that
  - $2^x < 2^{x+1}$ is true for $x = 0$; and
  - $2^x < 2^{x+1}$ for $x = n$
    implies that $2^x < 2^{x+1}$ for $x = n + 1$
- then $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.
- But we do have
  - Base: $2^0 = 1 < 2 = 2^{0+1}$
  - Step: If $2^n < 2^{n+1}$ ($P(x)$ for $x = n$) then
    $2^{n+1} = 2^n + 2^n < 2^{n+1} + 2^{n+1} = 2^{n+2}$
    ($P(x)$ for $x = n{+}1$)
- By Induction, $2^x < 2^{x+1}$ for all $x \in \mathbb{N}$.

Try this...

- Prove by induction on $\mathbb{N}$ that $x \leqslant 2^x$ for all $x \in \mathbb{N}$.
  We are given that exponentiation is an increasing function.
- By Induction $x \leqslant 2^x$ for all $x \in \mathbb{N}$.

Try this...

- Prove by induction on $\mathbb{N}$ that $x \leqslant 2^x$ for all $x \in \mathbb{N}$.
  We are given that exponentiation is an increasing function.
  - Base: For $x = 0$ we have $x^2 = 0 < 1 = 2^x$.
  - Step: Assume $n \leqslant 2^n$. Then

$$\begin{aligned} n+1 &\leqslant 2^n + 1 && \text{(IH)} \\ &= 2^n + 2^0 \\ &\leqslant 2^n + 2^n && \text{(exponentiation is increasing)} \\ &= 2^{n+1} \end{aligned}$$

- By Induction $x \leqslant 2^x$ for all $x \in \mathbb{N}$.

Example: Divisibility

- $P(x): \quad x^3 + 2x$ is divisible by 3.
- By Induction:

Example: Divisibility

- $P(x): \quad x^3 + 2x$ is divisible by 3.
- By Induction:
  - **Base.** For $x = 0$
    $x^3 + 2x = 0^3 + 2 \cdot 0 = 0$ which is divisible by 3.

Example: Divisibility

- $P(x): \quad x^3 + 2x$ is divisible by 3.
- By Induction:
  - **Base.** For $x = 0$
    $x^3 + 2x = 0^3 + 2 \cdot 0 = 0$ which is divisible by 3.
  - **Step.** Assume $P(n)$ (IH). Then for $x = n+1$

$$\begin{aligned} x^3 + 2x &= (n+1)^3 + (2n+2) \\ &= (n^3 + 3n^2 + 3n + 1) + (2n+2) \\ &= (n^3 + 2n) + 3(n^2 + n + 1) \end{aligned}$$

$x^3 + 2x$ is the sum of numbers divisible by 3,
and is therefore divisible by 3.

Example: Divisibility

- $P(x): \quad x^3 + 2x$ is divisible by 3.
- By Induction:
  - **Base.** For $x = 0$
    $x^3 + 2x = 0^3 + 2 \cdot 0 = 0$ which is divisible by 3.
  - **Step.** Assume $P(n)$ (IH). Then for $x = n{+}1$

$$
\begin{aligned}
x^3 + 2x &= (n+1)^3 + (2n+2) \\
&= (n^3 + 3n^2 + 3n + 1) + (2n+2) \\
&= (n^3 + 2n) + 3(n^2 + n + 1)
\end{aligned}
$$

$x^3 + 2x$ is the sum of numbers divisible by 3,
and is therefore divisible by 3.

- By Induction $x^3 + 2x$ is divisible by 3, for all $x \in \mathbb{N}$.

Iterated summation

$$(\star) \qquad 1 + 3 + 5 + \cdots + (2x - 1) \;=\; x^2$$

Iterated summation

$$(\star) \qquad 1+3+5+\cdots+(2x-1) \;=\; x^2$$

By Induction:

Iterated summation

$$(\star) \qquad 1+3+5+\cdots+(2x-1) \;=\; x^2$$

By Induction:

- **Base.** $(\star)$ is true for $x=0$:
  the empty sum $=0=0^2$.

Iterated summation

$$(\star) \quad 1+3+5+\cdots+(2x-1) \;=\; x^2$$

By Induction:

- **Base.** $(\star)$ is true for $x=0$:
  the empty sum $=0=0^2$.
- **Step.** Assume $(\star)$ for $x=n$.

Then, for $x=n+1$,

$$\begin{aligned} 1+3+\cdots+(2x-1) &= 1+3+\cdots+(2n-1)+(2n+1) \\ &= n^2+(2n+1) && \text{(IH)} \\ &= (n+1)^2 \end{aligned}$$

That is, $(\star)$ for $x=n+1$.

Iterated summation

$$(\star) \quad 1+3+5+\cdots+(2x-1) \;=\; x^2$$

By Induction:

- **Base.** $(\star)$ is true for $x=0$:
  the empty sum $=0=0^2$.
- **Step.** Assume $(\star)$ for $x=n$.

Then, for $x=n+1$,

$$\begin{aligned} 1+3+\cdots+(2x-1) &= 1+3+\cdots+(2n-1)+(2n+1) \\ &= n^2+(2n+1) && \text{(IH)} \\ &= (n+1)^2 \end{aligned}$$

That is, $(\star)$ for $x=n+1$.

Conclude: $(\star)$ holds for every $x \in \mathbb{N}$.

Another iterated summation

$$(\star) \qquad 1 + 2 + \cdots + x \;=\; x(x+1)/2$$

Another iterated summation

$$(\star) \qquad 1 + 2 + \cdots + x \; = \; x(x+1)/2$$

By Induction.

Another iterated summation

$$(\star) \qquad 1+2+\cdots+x \;=\; x(x+1)/2$$

By Induction.

- **Base.** $(\star)$ is true for $x=0$:
  The empty sum $=0=0\cdot 1/2$.

Another iterated summation

$$(\star) \quad 1+2+\cdots+x \;=\; x(x+1)/2$$

By Induction.

- **Base.** $(\star)$ is true for $x=0$:

  The empty sum $=0=0\cdot 1/2$.

- **Step.** Assume $(\star)$ for $x=n$.

Then, for $x=n{+}1$,

$$\begin{aligned}
0+1+\cdots+x &= 0+1+\cdots+n+(n{+}1) \\
&= \tfrac{n(n+1)}{2}+(n{+}1) && \text{(IH)} \\
&= (n+1)(\tfrac{1}{2}n+1) \\
&= \tfrac{1}{2}(n{+}1)\cdot(n{+}2) \\
&= \tfrac{1}{2}\,x\,(x{+}1)
\end{aligned}$$

That is, $(\star)$ for $x=n{+}1$.

Another iterated summation

$$(\star) \qquad 1+2+\cdots+x \;=\; x(x+1)/2$$

By Induction.

- **Base.** $(\star)$ is true for $x=0$:

  The empty sum $=0=0\cdot 1/2$.

- **Step.** Assume $(\star)$ for $x=n$.

Then, for $x=n{+}1$,

$$\begin{aligned}
0+1+\cdots+x \;&=\; 0+1+\cdots+n+(n{+}1) \\
&=\; \tfrac{n(n+1)}{2}+(n{+}1) && \text{(IH)} \\
&=\; (n+1)(\tfrac{1}{2}n+1) \\
&=\; \tfrac{1}{2}(n{+}1)\cdot(n{+}2) \\
&=\; \tfrac{1}{2}x\,(x{+}1)
\end{aligned}$$

That is, $(\star)$ for $x=n{+}1$.

Conclude: $(\star)$ holds for every $x\in\mathbb{N}$.

Involving other data

- A property of natural numbers may refer to non-numeric data!

  ($\star$) *Every set with* x *elements has* 2^x *subsets*

- By Induction.

Involving other data

- A property of natural numbers may refer to non-numeric data!

  ($\star$) *Every set with x elements has 2^x subsets*

- By Induction.

  - **Base.** $x = 0$. The only set with 0 elements is $\emptyset$, which has just $2^0 = 1$ subset, namely $\emptyset$ itself.

Involving other data

- A property of natural numbers may refer to non-numeric data!

  ($\star$) *Every set with x elements has 2^x subsets*

- By Induction.

  - **Base.** $x = 0$. The only set with 0 elements is $\emptyset$, which has just $2^0 = 1$ subset, namely $\emptyset$ itself.

  - **Step.** Assume $P(n)$ (IH).

  For $x = n+1$ let S be a set with $n+1$ elements.
  Choose $a \in S$ (S can't be empty!) and let $S^- =_{\mathrm{df}} S - \{a\}$.

Involving other data

- A property of natural numbers may refer to non-numeric data!

  ($\star$) *Every set with x elements has 2^x subsets*

- By Induction.

  - **Base.** $x = 0$. The only set with 0 elements is $\emptyset$, which has just $2^0 = 1$ subset, namely $\emptyset$ itself.
  - **Step.** Assume $P(n)$ (IH).

    For $x = n+1$ let S be a set with $n+1$ elements.

    Choose $a \in S$ (S can't be empty!) and let $S^- =_{\text{df}} S - \{a\}$.

    By IH S^- has 2^n subsets $A_1, \ldots, A_{2^n}$.

    Subsets of S: $A_1, \ldots, A_{2^n},\ A_1 \cup \{a\},\ \ldots,\ A_{2^n} \cup \{a\}$

    which are all different. So S has $2^n + 2^n = 2^{n+1}$ subsets.

Involving other data

- A property of natural numbers may refer to non-numeric data!

  ($\star$) *Every set with x elements has 2^x subsets*

- By Induction.

  - **Base.** $x = 0$. The only set with 0 elements is $\emptyset$, which has just $2^0 = 1$ subset, namely $\emptyset$ itself.
  - **Step.** Assume $P(n)$ (IH).

  For $x = n+1$ let S be a set with $n+1$ elements.
  Choose $a \in S$ (S can't be empty!) and let $S^- =_{\text{df}} S - \{a\}$.
  By IH S^- has 2^n subsets $A_1, \ldots, A_{2^n}$.
  Subsets of S: $A_1, \ldots, A_{2^n}$, $A_1 \cup \{a\}, \ldots, A_{2^n} \cup \{a\}$
  which are all different. So S has $2^n + 2^n = 2^{n+1}$ subsets.

- By Induction ($\star$) for all $x \in \mathbb{N}$.

Starting Induction elsewhere

- *Show* $x^2 > x$ *for all* $x > 1$.
- We wish to start induction from 2.

Starting Induction elsewhere

- *Show $x^2 > x$ for all $x > 1$.*
- We wish to start induction from 2.

  But that's the same as Induction

  for the property $(x+2)^2 > (x+2)$!

Starting Induction elsewhere

- *Show* $x^2 > x$ *for all* $x > 1$.
- We wish to start induction from 2.

  But that's the same as Induction

  for the property $(x+2)^2 > (x+2)$!
- We refer to this as Shifted Induction:
  - **Base.** $2^2 = 4 > 2$
  - **Step.** $n^2 > n$ implies

$$\begin{aligned} (n+1)^2 &= n^2 + 2n + 1 \\ &> n + 2n + 1 && \text{(IH)} \\ &> n + 1 && \text{since } n > 0) \end{aligned}$$

Starting Induction elsewhere

- *Show* $x^2 > x$ *for all* $x > 1$.
- We wish to start induction from 2.

  But that's the same as Induction

  for the property $(x+2)^2 > (x+2)$!
- We refer to this as Shifted Induction:
  - **Base.** $2^2 = 4 > 2$
  - **Step.** $n^2 > n$ implies

$$\begin{aligned} (n+1)^2 &= n^2 + 2n + 1 \\ &> n + 2n + 1 \quad \text{(IH)} \\ &> n + 1 \quad \text{since } n > 0) \end{aligned}$$

- Conclusion: $x^2 > x$ for all integers $x > 1$.

Shifted Induction

- The template for such reasoning is $\boxed{\textit{Shifted Induction}}$
- Given a property $P(x)$ of natural numbers, and $b \in \mathbb{N}$,
- Assume:
  - **Shifted Base.** P true of b; and
  - **Shifted Step.** For all $n \geqslant b$, $P(n)$ implies $P(n+1)$
- Conclude: $P(x)$ for all $x \geqslant b$.
- Induction is a special case, with $b = 0$.

Another example

- $3^n > 5 \cdot 2^n$ for all $n \geqslant 4$.
- By Shifted Induction with initial value 4.

Another example

- $3^n > 5 \cdot 2^n$ for all $n \geqslant 4$.
- By Shifted Induction with initial value 4.
  - Basis. $3^4 = 81 > 80 = 5 \cdot 2^4$

Another example

- $3^n > 5 \cdot 2^n$ for all $n \geqslant 4$.
- By Shifted Induction with initial value 4.
  - Basis. $3^4 = 81 > 80 = 5 \cdot 2^4$
  - Step. If $3^n > 5 \cdot 2^n$ then

$$\begin{array}{rcll} 3^{n+1} & = & 3 \cdot 3^n & \\ & > & 3 \cdot (5 \cdot 2^n) & \text{(IH)} \\ & > & 2 \cdot 5 \cdot 2^n & \\ & = & 5 \cdot 2^{n+1} & \end{array}$$

Is this new?

- *But is Shifted Induction a new method?*
  Or is it just syntactic sugar for particular form of Induction?

Shifting, in general

- Given
  - Shifted Base. $P(b)$ and
  - Shifted Step. $P(n)$ implies $P(n+1)$ for all $n \geqslant b$

  we prove by ***Induction*** that $P(x)$ for all $x \geqslant b$:
- Let $P'(x)$ be $P(x-b)$
  - **Base.** $P'(0)$, because $P(b)$ by the Shifted Base.
  - **Step.** $P'(n)$ implies $P'(n+1)$ for $n \geqslant 0$
    because $P(n)$ implies $P(n+1)$ for all $n \geqslant b$,
    by the Shifted Base.
- By Induction, $P'(x)$ for all $x \geqslant 0$,
  so $P(x)$ for all $x \geqslant b$.

Another example

- To prove $3^x > 5 \cdot 2^x$ for $x \geqslant 2$

  use Induction to prove $3^{x+2} > 5 \cdot 2^{x+2}$ for $x \geqslant 0$:

  - **Shifted Basis.** $3^x > 5 \cdot 2^x$ for $x = 2$,

    i.e. $3^{x+2} > 5 \cdot 2^{x+2}$ for $x = 0$

  - **Step.** $3^n > 5 \cdot 2^n$ implies $3^{n+1} > 5 \cdot 2^{n+1}$ for $n \geqslant 2$,

    i,e, $3^{n+2} > 5 \cdot 2^{n+2}$ implies $3^{(n+2)+1} > 5 \cdot 2^{(n+2)+1}$ for $n \geqslant 0$

- Conclusion by Induction:

  $3^{x+2} > 5 \cdot 2^{x+2}$ for $x \geqslant 0$

  i.e. $3^x > 5 \cdot 2^x$ for $x \geqslant 2$.

Another shortcoming of Induction?

- **Theorem.** *Every positive integer is the product of primes*
- *Induction?*

  No useful relation between factoring n and factoring $n+1$!

Another shortcoming of Induction?

- **Theorem.** *Every positive integer is the product of primes*
- E.g. 1 is the empty product;
  3 a singleton product;
  9 the product of 3 used twice.
- *Induction?*
  No useful relation between factoring n and factoring $n+1$!

Cumulative Induction

- **Theorem.** *Every positive integer is the product of primes*.

Cumulative Induction

- **Theorem.** *Every positive integer is the product of primes*.
- Proof by Induction for the property

  $(\star)$ Every positive integer $\leqslant x$ is product of primes.

Cumulative Induction

- **Theorem.** *Every positive integer is the product of primes*.
- Proof by Induction for the property

  ($\star$) Every positive integer $\leqslant x$ is product of primes.

  - **Basis.** $x = 0$. No positive integers $x \leqslant 0$ so ($\star$) vacuously.

Cumulative Induction

- **Theorem.** *Every positive integer is the product of primes*.
- Proof by Induction for the property

  ($\star$) Every positive integer $\leqslant x$ is product of primes.

  - **Basis.** $x = 0$. No positive integers $x \leqslant 0$ so ($\star$) vacuously.
  - **Step.** Assume ($\star$) for $x = n$, show ($\star$) for $x = n+1$

Cumulative Induction

- **Theorem.** *Every positive integer is the product of primes*.
- Proof by Induction for the property

  ($\star$) Every positive integer $\leqslant x$ is product of primes.

  - **Basis.** $x = 0$. No positive integers $x \leqslant 0$ so ($\star$) vacuously.
  - **Step.** Assume ($\star$) for $x = n$, show ($\star$) for $x = n+1$
    - Case 1: $n+1$ is 1, which is the empty product.
    - Case 2. $n+1$ is a prime, ok.
    - Case 3. $n+1 = y \cdot z$ for some $y, z \in [2..n]$.

      By IH y, z are products of primes, so $n+1$ is too.

The template of Cumulative Induction

- $\boxed{\textit{Cumulative Induction}}$ template:
- ***Assume:***
  - **Base.** $P(x)$ is true for $x = 0$.
  - **Step.** For every x,
    if $P(y)$ is true for every $y \leqslant x$ then P for $x+1$.
- ***Conclude:*** $P(x)$ for all $x \in \mathbb{N}$.
- Induction is a special case, where the IH is just $P(n)$.
- Cumulative Induction is also dubbed "Strong Induction",
  even though it is not stronger than Induction, as we show next.

Cumulative Induction is not more general

- Let "P progressive" abbreviate

  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."

Cumulative Induction is not more general

- Let "P progressive" abbreviate

  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."

- Cumulative induction reads:

  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$

Cumulative Induction is not more general

- Let "P progressive" abbreviate

  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."

- Cumulative induction reads:

  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$

- Proof by Induction.

Cumulative Induction is not more general

- Let "P progressive" abbreviate

  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."

- Cumulative induction reads:

  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$

- Proof by Induction.

  - Assume P is progressive.

Cumulative Induction is not more general

- Let "P progressive" abbreviate

  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."

- Cumulative induction reads:

  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$

- Proof by Induction.

  - Assume P is progressive.
  - Let $Q(x)$ abbreviate "$P(y)$ *for every* $y \in [0..x)$."

Cumulative Induction is not more general

- Let "P progressive" abbreviate
  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."
- Cumulative induction reads:
  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$
- Proof by Induction.
  - Assume P is progressive.
  - Let $Q(x)$ abbreviate "$P(y)$ *for every* $y \in [0..x)$."
  - Base: $Q(0)$, because $[0..0) = \emptyset$

Cumulative Induction is not more general

- Let “P progressive” abbreviate

  “if $P(y)$ for every $y \in [0..x)$ then $P(x)$.”

- Cumulative induction reads:

  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$

- Proof by Induction.
  - Assume P is progressive.
  - Let $Q(x)$ abbreviate “$P(y)$ *for every* $y \in [0..x)$.”
  - Base: $Q(0)$, because $[0..0) = \emptyset$
  - Step: P is progressive, so $Q(x)$ implies $Q(x+1)$.

Cumulative Induction is not more general

- Let "P progressive" abbreviate
  "if $P(y)$ for every $y \in [0..x)$ then $P(x)$."
- Cumulative induction reads:
  If P is progressive then $P(x)$ for all $x \in \mathbb{N}$
- Proof by Induction.
  - Assume P is progressive.
  - Let $Q(x)$ abbreviate "$P(y)$ *for every* $y \in [0..x)$."
  - Base: $Q(0)$, because $[0..0) = \emptyset$
  - Step: P is progressive, so $Q(x)$ implies $Q(x{+}1)$.
- By Induction, $Q(x)$ for every $x \in \mathbb{N}$.
  But then $P(x)$ for every $x \in \mathbb{N}$.

INDUCTIVE REASONING IN GENERAL

Induction over generated sets

- The principle of inductive reasoning applies
  to any inductively generated set S, not just $\mathbb{N}$.
- If $P(x)$ makes sense for $x \in S$,
  is true for every base element of S
  and remains true under the generative steps for S,
  then $P(x)$ is true for all $x \in S$.

Induction over generated sets

- The principle of inductive reasoning applies
  to any inductively generated set S, not just $\mathbb{N}$.
- If $P(x)$ makes sense for $x \in S$,
  is true for every base element of S
  and remains true under the generative steps for S,
  then $P(x)$ is true for all $x \in S$.
- The underlying reason is the same as for $\mathbb{N}$:
  as the elements of S are generated,
  the property P invariantly holds.

Induction on strings

- Let $P(x)$ be a property of Σ-strings.

Induction on strings

- Let $P(x)$ be a property of Σ-strings.
- ***Assume:***
  - **Base.** $P(\varepsilon)$
  - **Steps.** For each $\sigma \in \Sigma$ and $w \in \Sigma^*$
    $P(w)$ implies $P(\sigma w)$

Induction on strings

- Let $P(x)$ be a property of Σ-strings.
- ***Assume:***
  - **Base.** $P(\varepsilon)$
  - **Steps.** For each $\sigma \in \Sigma$ and $w \in \Sigma^*$
    $P(w)$ implies $P(\sigma w)$
- ***Conclude:*** $P(w)$ for all $w \in \Sigma^*$.

Example: Swapping

- For $w \in \{0,1\}^*$ let $\wr(w)$ ("swap w") be w with $\mathtt{0}$ and $\mathtt{1}$ interchanged: $\wr\mathtt{001} = \mathtt{110}$.

  We show $\quad (\star) \quad \wr(\wr(w)) = w$

Example: Swapping

- For $w \in \{0,1\}^*$ let $\wr(w)$ ("swap w") be w with 0 and 1 interchanged: $\wr 001 = 110$.

  We show $\quad (\star) \quad \wr(\wr(w)) = w$

- The proof is by induction on $\{0,1\}^*$.

Example: Swapping

- For $w \in \{0,1\}^*$ let $\wr(w)$ ("swap w") be w with $\mathtt{0}$ and $\mathtt{1}$ interchanged: $\wr\mathtt{001} = \mathtt{110}$.

  We show $\quad (\star) \quad \wr(\wr(w)) = w$

- The proof is by induction on $\{0,1\}^*$.
  - **Basis.** $\wr(\wr(\varepsilon)) = \wr(\varepsilon) = \varepsilon$

Example: Swapping

- For $w \in \{0,1\}^*$ let $\wr(w)$ ("swap w") be w with 0 and 1 interchanged: $\wr 001 = 110$.

  We show $(\star)$ $\wr(\wr(w)) = w$

- The proof is by induction on $\{0,1\}^*$.
  - **Basis.** $\wr(\wr(\varepsilon)) = \wr(\varepsilon) = \varepsilon$
  - **Step for 0.** If $\wr(\wr(x)) = x$

  then $\wr(\wr(0x)) = \wr(1 \wr (x))$
  $= 0 \wr (\wr(x))$
  $= 0x$ (IH)

  Step for 1 is similar.

Example: Swapping

- For $w \in \{0, 1\}^*$ let $\wr(w)$ ("swap w") be w with 0 and 1 interchanged: $\wr 001 = 110$.

  We show $\quad (\star) \quad \wr(\wr(w)) = w$

- The proof is by induction on $\{0, 1\}^*$.
  - **Basis.** $\wr(\wr(\varepsilon)) = \wr(\varepsilon) = \varepsilon$
  - **Step for 0.** If $\wr(\wr(x)) = x$

  then
  $$\begin{aligned} \wr(\wr(0x)) &= \wr(1 \wr (x)) \\ &= 0 \wr (\wr(x)) \\ &= 0x \qquad \text{(IH)} \end{aligned}$$

  Step for 1 is similar.

- By induction on $\{0, 1\}^*$ $(\star)$ for all $w \in \{0, 1\}^*$.

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).
- *Problem: This is a property of a pair of strings!*

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).
- Solution: Read it as a property of one x:

$$|x \cdot u| = |x| + |u| \quad \text{for all } u \in \Sigma^* \tag{$\star$}$$

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).
- Solution: Read it as a property of one x:

$$|x \cdot u| = |x| + |u| \quad \text{for all } u \in \Sigma^* \tag{$\star$}$$

  - Basis: $x = \varepsilon$.

$$\begin{aligned} |\varepsilon \cdot u| &= |u| && \text{since } \varepsilon \cdot u = u \\ |\varepsilon| + |u| &= 0 + |u| = |u| \end{aligned}$$

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).
- Solution: Read it as a property of one x:

$$|x \cdot u| = |x| + |u| \quad \text{for all } u \in \Sigma^* \tag{$\star$}$$

  - Basis: $x = \varepsilon$.

$$\begin{aligned} |\varepsilon \cdot u| &= |u| && \text{since } \varepsilon \cdot u = u \\ |\varepsilon| + |u| &= 0 + |u| = |u| \end{aligned}$$

  - Step: Assume $(\star)$ for $x = w$.

    For $x = \sigma w$ we have for all $u \in \Sigma^*$

$$\begin{aligned} |\sigma w \cdot u| &= |\sigma(w \cdot u)| \\ &= 1 + |w \cdot u| \\ &= 1 + |w| + |u| \quad \text{(IH)} \\ &= (|\sigma w|) + |u| \end{aligned}$$

Dealing with several inputs

- Prove $|x \cdot u| = |x| + |u|$ ($x, u \in \Sigma^*$).
- Solution: Read it as a property of one x:

$$|x \cdot u| = |x| + |u| \quad \text{for all } u \in \Sigma^* \qquad (\star)$$

  - Basis: $x = \varepsilon$.

$$\begin{aligned} |\varepsilon \cdot u| &= |u| && \text{since } \varepsilon \cdot u = u \\ |\varepsilon| + |u| &= 0 + |u| = |u| \end{aligned}$$

  - Step: Assume $(\star)$ for $x = w$.

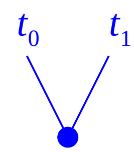
    For $x = \sigma w$ we have for all $u \in \Sigma^*$

$$\begin{aligned} |\sigma w \cdot u| &= |\sigma(w \cdot u)| \\ &= 1 + |w \cdot u| \\ &= 1 + |w| + |u| \quad \text{(IH)} \\ &= (|\sigma w|) + |u| \end{aligned}$$

- By induction on Σ^* conclude $(\star)$ for all $x \in \Sigma^*$.

Induction over binary trees

- Recall that the set of binary trees is generated from a base tree $\bullet$ by juncture:

  if t_0, t_1 are binary trees then so is 

- Let $P(x)$ be a property that makes sense for any binary tree t.
- If we can show that
  - **Base:** $P(\bullet)$; and
  - **Step:** If both $P(t_0)$ and $P(t_1)$ then $P(t)$ for the juncture t above of t_0 and t_1

  then $P(t)$ is true for all binary trees t.

Example: Odd size of binary trees

- *Can a binary tree have an even number of nodes?*

Example: Odd size of binary trees

- Every binary tree has an odd number of nodes.

Example: Odd size of binary trees

- Every binary tree has an odd number of nodes.
- Let $P(t)$ be the property

  "t has an odd number of nodes"

Example: Odd size of binary trees

- Every binary tree has an odd number of nodes.
- Let $P(t)$ be the property

  "t has an odd number of nodes"

  Induction on trees:

  - ***Basis:*** $P(\bullet)$ (since 1 is odd)

Example: Odd size of binary trees

- Every binary tree has an odd number of nodes.
- Let $P(t)$ be the property

  "t has an odd number of nodes"

  Induction on trees:

  - ***Basis:*** $P(\bullet)$ (since 1 is odd)
  - ***Step:*** Suppose t_0, t_1 are trees of odd sizes n_0 and n_1.

    Let t be obtained from t_0 and t_1.

    The size of t is $n_0 + n_1 + 1$, which is again odd.

Example: Odd size of binary trees

- Every binary tree has an odd number of nodes.
- Let $P(t)$ be the property

  "t has an odd number of nodes"

  Induction on trees:

  - ***Basis:*** $P(\bullet)$ (since 1 is odd)
  - ***Step:*** Suppose t_0, t_1 are trees of odd sizes n_0 and n_1.

    Let t be obtained from t_0 and t_1.

    The size of t is $n_0 + n_1 + 1$, which is again odd.
- By induction on binary tree we conclude that $P(t)$ for all binary trees t.

Invariants: A dynamic view of induction

- Euclid GCD algorithm
- Eating lots of chocolate
- A game of coins

GENERATING SETS

Generating the star of a language

- Example: We generated Σ^* starting with an alphabet Σ:
  - Base: Each $\sigma \in \Sigma$ is in Σ^*.
  - Generative step: If $x \in \Sigma^*$ and $\sigma \in \Sigma$ then $\sigma x \in \Sigma^*$.
- We started from a finite number of initial objects. More broadly we can start with any set.
- For any language L we generate L^* :
  - Base: Each $w \in L$ is in L^*.
  - Generative step: If $x \in L^*$ and $w \in L$ then $w \cdot x \in L^*$.
- This defines the mapping $*$ between languages: each language L is mapped to L^* .

Generating the star of a mapping

- Combining languages by concatenation yields a new language.
  The iteration of concatenating with L,
  starting with the unit language $\{\varepsilon\}$,
  yields L^*.

Generating the star of a mapping

- Combining languages by concatenation yields a new language.

  The iteration of concatenating with L,

  starting with the unit language $\{\varepsilon\}$,

  yields L^*.

- Similarly, combining ***mappings*** $F : A \Rightarrow A$ by composition

  yields a new mapping.

  The iteration of composing with F, starting with $\mathbf{Id}_A : A \to A$

  yields the $\boxed{\textit{star of } F}$, denoted F^*.

Generating the star of a mapping

- Similarly, combining ***mappings*** $F : A \Rightarrow A$ by composition

  yields a new mapping.

  The iteration of composing with F, starting with $\mathbf{Id}_A : A \to$

  yields the $\boxed{\textit{star of } F}$, denoted F^*.

- That is, $x\,(F)^*\,y$ iff

$$= z_0\,(F)\,z_1\,(F)\ \cdots\ (F)\,z_k = y$$

  for some $k \geqslant 0$ and $z_0, .., z_k\ A$.

COMPUTING BY RECURRENCE

Explicit function definitions (reminder)

- $f(x) = 3 \times (x + 2)$ defines a function
  in terms of given constants and function-identifiers.
- The definition
  1. introduces a new function identifier (f); and
  2. uses a defining expression $3 \times (x + 2))$
     built from known functions $+$ and $\times$ and the
     argument (i.e. input) of f .

Computing by recurrence

- Suppose $s(x) = x+1$ is the only function available. Define:

$$\begin{array}{rcl} d(0) & = & 0 \\ d(s(x)) & = & s(s(d(x))) \end{array}$$

- *What is* $d(1)$*?* $d(2)$*?* $d(x)$*?*

Computing by recurrence

$$\begin{aligned} d(0) &= 0 \\ d(s(x)) &= s(s(d(x))) \end{aligned}$$

- As the input is being generated,
  successive outputs grow by double-increments:
  - $d(0) = 0$ is given
  - $d(1) = s(s(d(0))) = s(s(0)) = 2$
  - $d(2) = s(s(d(1))) = s(s(2)) = 4$ etc.
- f is defined by providing a value for input 0
  and an explicit definition of $f(x+1)$ in terms of $f(x)$.

The general template

- A function $f : \mathbb{N} \to \mathbb{N}$ is defined here by providing
  1. a value for $f(0)$ and
  2. an explicit definition of $f(x+1)$ in terms of $f(x)$

The general template

- A function $f : \mathbb{N} \to \mathbb{N}$ is defined here by providing
  1. a value for $f(0)$ and
  2. an explicit definition of $f(x+1)$ in terms of $f(x)$
- Our next examples illustrate the potential of recurrence to yield rapidly-increasing functions.

Exponentiation

- Using the doubling-function d
  we define a new function:

$$\begin{aligned} e(0) &= 1 \\ e(s(x)) &= d(e(x)) \end{aligned}$$

Exponentiation

- Using the doubling-function d
  we define a new function:

$$\begin{aligned} e(0) &= 1 \\ e(s(x)) &= d(e(x)) \end{aligned}$$

- *What are* $e(1)$*?* $e(2)$*?* $e(n)$*?*

Exponentiation

- Using the doubling-function d

  we define a new function:

$$\begin{aligned} e(0) &= 1 \\ e(s(x)) &= d(e(x)) \end{aligned}$$

- *What are* $e(1)$*?* $e(2)$*?* $e(n)$*?*
- How do we prove it?

Exponentiation

- Using the doubling-function d
  we define a new function:

$$\begin{aligned} e(0) &= 1 \\ e(s(x)) &= d(e(x)) \end{aligned}$$

- *What are* $e(1)$*?* $e(2)$*?* $e(n)$*?*

  By Induction!

Shooting for the stars

- We defined the function e of exponentiation base 2. Using Iteration again we define a new function

$$
\begin{aligned}
h(0) &= 1 \\
h(s(x)) &= e(h(x))
\end{aligned}
$$

- $$
\begin{aligned}
h(1) &= 2^1 &= 2 \\
h(2) &= 2^2 &= 4 \\
h(3) &= 2^4 &= 16 \\
h(4) &= 2^{16} &= 15384 \\
h(5) &= 2^{15384} &> 10^{5000}
\end{aligned}
$$ no physical meaning!

Recursion vs. recurrence

- *Recurrence* and *recursion*
  are practically identical as English words,
  but they have different mathematical meanings.
  - ***Recursion:*** $f(x) = \cdots$
    f used as you wish
    eg $f(x) = f(f(x+1))$
  - ***Recurrence:*** $f(x) = \cdots$
    f used for input $x-1$

  Much weaker than recursion.
  - ***Cumulative recurrence:*** $f(x) = \cdots$ uses
    f for inputs $< x$

  Has same power as recurrence.

Output need not be numeric

- Strings as output:

$$\begin{aligned} f(0) &= \varepsilon \\ f(s(x)) &= \mathtt{a} \cdot f(x) \end{aligned}$$

Output need not be numeric

- Strings as output:

$$\begin{aligned} f(0) &= \varepsilon \\ f(s(x)) &= \mathtt{a} \cdot f(x) \end{aligned}$$

- $f(n) =$

Output need not be numeric

- Strings as output:

$$\begin{aligned} f(0) &= \varepsilon \\ f(s(x))) &= \mathtt{a} \cdot f(x) \end{aligned}$$

- $f(n) = \mathtt{a}^n$

Outputs need not be numeric

- Sets as output:

$$
\begin{aligned}
f(0) &= \emptyset \\
f(s(x))) &= f(x) \cup \{f(x)\}
\end{aligned}
$$

Outputs need not be numeric

- Sets as output:
$$\begin{aligned} f(0) &= \emptyset \\ f(s(x))) &= f(x) \cup \{f(x)\} \end{aligned}$$
-
  - $f(1) = \{\emptyset\}$
  - $f(2) = \{\emptyset, \{\emptyset\}\}$
  - $f(3) = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$

Outputs need not be numeric

- Sets as output:
$$\begin{aligned} f(0) &= \emptyset \\ f(s(x))) &= f(x) \cup \{f(x)\} \end{aligned}$$
-
  - $f(1) = \{\emptyset\}$
  - $f(2) = \{\emptyset,\ \{\emptyset\}\}$
  - $f(3) = \{\emptyset,\ \{\emptyset\}, \{\emptyset,\ \{\emptyset\}\}\}$
- In general, $f(n) = \{f(0), \ldots, f(n-1)\}$
  Every "number" is the set of previous numbers.

Outputs need not be numeric

- Sets as output:
$$\begin{aligned} f(0) &= \emptyset \\ f(s(x))) &= f(x) \cup \{f(x)\} \end{aligned}$$
-
  - $f(1) = \{\emptyset\}$
  - $f(2) = \{\emptyset,\ \{\emptyset\}\}$
  - $f(3) = \{\emptyset,\ \{\emptyset\}, \{\emptyset,\ \{\emptyset\}\}\}$
- In general, $f(n) = \{f(0), \ldots, f(n-1)\}$
  Every "number" is the set of previous numbers.
- So natural numbers can be simulated by abstract sets!

Outputs need not be numeric

- Languages as output:

$$\begin{aligned} f(0) &= \{\varepsilon\} \\ f(s(x)) &= f(x) \cdot \{\mathtt{a}, \mathtt{b}\} \end{aligned}$$

Outputs need not be numeric

- Languages as output:

$$\begin{aligned} f(0) &= \{\varepsilon\} \\ f(s(x)) &= f(x) \cdot \{\mathtt{a}, \mathtt{b}\} \end{aligned}$$

- $f(n) =$

Outputs need not be numeric

- Languages as output:

$$\begin{aligned} f(0) &= \{\varepsilon\} \\ f(s(x)) &= f(x) \cdot \{\mathtt{a}, \mathtt{b}\} \end{aligned}$$

- $f(n) = \{\mathtt{a}, \mathtt{b}\}^n$

RECURRENCE ON STRINGS

Example: The swap function

$\wr : \{0,1\}^* \to \{0,1\}^*$

For instance: $\wr(01011) = 10100$

Example: The swap function

$\wr : \{0,1\}^* \to \{0,1\}^*$

For instance: $\wr(01011) = 10100$

$$
\begin{aligned}
\wr(\varepsilon) &= \varepsilon \\
\wr(0w) &= 1 \wr (w) \\
\wr(1w) &= 0 \wr (w)
\end{aligned}
$$

Example: The swap function

$\wr : \{0,1\}^* \to \{0,1\}^*$

For instance: $\wr(01011) = 10100$

$$
\begin{aligned}
\wr(\varepsilon) &= \varepsilon \\
\wr(0w) &= 1 \wr (w) \\
\wr(1w) &= 0 \wr (w)
\end{aligned}
$$

As the values in $\{0,1\}^*$ are generated,
the corresponding output values are obtained.

Template of recurrence on strings

- For $\Sigma = \{0, 1\}$:
- Given functions g_0 and g_1 over Σ^* ,

$$\begin{aligned} f(\varepsilon) &= c \\ f(0w) &= g_0(f(w)) \\ f(1w) &= g_1(f(w)) \end{aligned}$$

An entry for each $\sigma \in \Sigma$
and where each g_σ is a previously defined function

Template of recurrence on strings

- For $\Sigma = \{0, 1\}$:
- Given functions g_0 and g_1 over Σ^* ,

$$\begin{aligned} f(\varepsilon) &= c \\ f(0w) &= g_0(f(w)) \\ f(1w) &= g_1(f(w)) \end{aligned}$$

- A generic template for alphabets Σ :

$$\begin{aligned} f(\varepsilon) &= c \\ f(\sigma w) &= g_\sigma(f(w)) \end{aligned}$$

  An entry for each $\sigma \in \Sigma$
  and where each g_σ is a previously defined function

Template of recurrence on strings

- For $\Sigma = \{0, 1\}$:
- Given functions g_0 and g_1 over Σ^* ,

$$\begin{aligned} f(\varepsilon) &= c \\ f(0w) &= g_0(f(w)) \\ f(1w) &= g_1(f(w)) \end{aligned}$$

- A generic template for alphabets Σ :

$$\begin{aligned} f(\varepsilon) &= c \\ f(\sigma w) &= g_\sigma(f(w)) \end{aligned}$$

An entry for each $\sigma \in \Sigma$

and where each g_σ is a previously defined function

Another example

- $\Sigma = \{\mathtt{0}, \mathtt{1}\}$,
  $f : \Sigma^* \to \Sigma^*$ replaces in input w the first $\mathtt{0}$ by $\mathtt{1}$.
- Example: $f(\mathtt{11001}) = \mathtt{11101}$.

Another example

- $\Sigma = \{0, 1\}$,
  $f : \Sigma^* \to \Sigma^*$ replaces in input w the first $\mathtt{0}$ by $\mathtt{1}$.
- Example: $f(\mathtt{11001}) = \mathtt{11101}$.
  - $f(\varepsilon) =$
  - $f(\mathtt{1}w) =$
  - $f(\mathtt{0}w) =$

Another example

- $\Sigma = \{0, 1\}$,
  $f : \Sigma^* \to \Sigma^*$ replaces in input w the first 0 by 1.
- Example: $f(11001) = 11101$.
  - $f(\varepsilon) = \varepsilon$
  - $f(1w) =$
  - $f(0w) =$

Another example

- $\Sigma = \{0, 1\}$,
  $f : \Sigma^* \to \Sigma^*$ replaces in input w the first $\mathtt{0}$ by $\mathtt{1}$.
- Example: $f(\mathtt{11001}) = \mathtt{11101}$.
  - $f(\varepsilon) = \varepsilon$
  - $f(\mathtt{1}w) = \mathtt{1}f(w)$
  - $f(\mathtt{0}w) =$

Another example

- $\Sigma = \{0, 1\}$,
  $f : \Sigma^* \to \Sigma^*$ replaces in input w the first 0 by 1.
- Example: $f(11001) = 11101$.
  - $f(\varepsilon) = \varepsilon$
  - $f(1w) = 1f(w)$
  - $f(0w) = 1w$

The output need not be strings

- The length function $\ell : \{0,1\}^* \to \mathbb{N}$, i.e. $\ell(w) = |w|$:

$$\begin{aligned} \ell(\varepsilon) &= 0 \\ \ell(0w) &= 1 + \ell(w) \\ \ell(1w) &= 1 + \ell(w) \end{aligned}$$

Another example

- $v\ell : \{\mathtt{0}, \mathtt{1}\}^* \to \mathbb{N}$

$$\begin{aligned} v\ell(\varepsilon) &= 0 \\ v\ell(\mathtt{0}w) &= 2 \cdot v\ell(w) \\ v\ell(\mathtt{1}w) &= 1 + 2 \cdot v\ell(w) \end{aligned}$$

What is $v\ell$ *?*

A valuation function

$$
\begin{aligned}
v\ell(\varepsilon) &= 0 \\
v\ell(\mathtt{0}w) &= 2 \cdot v\ell(w) \\
v\ell(\mathtt{1}w) &= 1 + 2 \cdot v\ell(w)
\end{aligned}
$$

- $v\ell(w) =$ the numeric value of w^R in binary.
- For example,

$$
\begin{aligned}
v\ell(\mathtt{011}) &= 2 \cdot v\ell(\mathtt{11}) \\
&= 2 \cdot (1 + 2 \cdot v\ell(\mathtt{1})) \\
&= 2 \cdot (1 + 2 \cdot (1 + v\ell(\varepsilon))) \\
&= 2 \cdot (1 + 2 \cdot (1 + 0)) \\
&= 6 \\
&= \text{the numeric value of } \mathtt{110} \text{ in binary.}
\end{aligned}
$$