

specifying sets

enumeration of its elements

$\{red, green, blue\}$

set builder notation / set comprehensions

flavor 1

$\{x \in \mathbb{N}^+ : x \text{ is prime}\}$

$\{x \in \mathbb{N}^+ \mid x \text{ is prime}\}$

flavor 2

$\{x : x \in \mathbb{N}^+, x \text{ is prime}\}$

$\{2x : x \in \mathbb{N}^+\}$

bad flavor

$\{(x : x \notin x)\}$

$\{x : x = x\}$

recursive definition

(we will discuss this later)

enumeration w/ suspension points/ellipsis

$\{1, 2, 3, 4, 5, \dots\}$

(informal stand-in for a recursive definition)



$12 \in \{2, 3, 5, 8, \dots\}$?



set algebra

some properties of intersection, union, and set difference:

idempotence $A \cup A = A \cap A = A$

commutativity $A \cup B = B \cup A$

commutativity $A \cap B = B \cap A$

associativity $(A \cup B) \cup C = A \cup (B \cup C)$

associativity $(A \cap B) \cap C = A \cap (B \cap C)$

$$A \cup \emptyset = A$$

$$A \cap \emptyset = \emptyset$$

$$A \supseteq A \cap B \subseteq B$$

$$A \subseteq A \cup B \supseteq B$$

distributivity $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

distributivity $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

$$A \cap B \subseteq A \cup B$$

$$A \setminus A = \emptyset$$

$$A \setminus \emptyset = A$$

$$-(-B) = B$$

(more in the exercises of
1.4.1, 1.4.2, and 1.4.3 in SLAM)

terminology: source, target, domain, range

For binary relations $R \subseteq A \times B$:

A is a *source*.

B is a *target*.

Note that for any R , source and target are *not uniquely determined*:

$$R \subseteq A \times B$$

For any $A' \supseteq A$ and $B' \supseteq B$, we have $A \times B \subseteq A' \times B'$.

$$R \subseteq A \times B \subseteq A' \times B'$$

By contrast, these *are* uniquely determined:

the *domain* of R : $\text{dom}(R) = \{a : (a, b) \in R \text{ for some } b\}$

the *range* of R : $\text{range}(R) = \{b : (a, b) \in R \text{ for some } a\}$

For any relation $R \subseteq A \times B$ it is *always* the case that

$$\text{dom}(R) \subseteq A \quad \text{and} \quad \text{range}(R) \subseteq B$$

converse vs complement

$$R = \{(a, a), (a, b), (b, b), (b, c), (c, a), (c, c)\}$$

R	a	b	c
a	1	1	0
b	0	1	1
c	1	0	1

$$R^{-1} = \{(a, a), (a, c), (b, a), (b, b), (c, b), (c, c)\}$$

R	a	b	c
a	1	0	1
b	1	1	0
c	0	1	1

converse: mirror at the diagonal

$$\overline{R} = \{(a, c), (b, a), (c, b)\}$$

R	a	b	c
a	0	0	1
b	1	0	0
c	0	1	0

complement: flip zeros and ones

properties: a(anti)symmetry

Consider \leq and $<$ on the natural numbers. Neither is symmetric, but in slightly different ways.

For $<$, it is **never** the case that $a < b$ and $b < a$.

This is called **asymmetry**.

For \leq , it sometimes is, but only when $a = b$.

This is called **antisymmetry**.

Both relations are antisymmetric. Only $<$ is asymmetric.

A binary relation $R \subseteq A \times A$ is **asymmetric** iff for all $a, b \in A$
if aRb then not bRa

A binary relation $R \subseteq A \times A$ is **antisymmetric** iff for all $a, b \in A$
if aRb and bRa then $a = b$

equivalence relations

A binary relation $\approx \subseteq A \times A$ is an *equivalence relation* iff it is

1. reflexive
2. symmetric
3. transitive

What about these:

- equality
- divides: $m | n$ iff there is $k \geq 1 : km = n$
- relatively prime: $m \perp n$ iff there is no $k \geq 2 : k | m$ and $k | n$



equivalence relations

A binary relation $\approx \subseteq A \times A$

1. reflexive
2. symmetric
3. transitive

equality: yes, as it is reflexive, symmetric, and transitive.

divides: no, because it is not symmetric;

2 divides 4, but 4 does not divide 2.

relatively prime: no, it is neither reflexive (3 is not relatively prime to 3), nor transitive (2 is relatively prime to 5, 5 to 6, but not 2 to 6).

What about these:

- equality

- divides:

- relatively prime:

$m | n$ iff there is $k \geq 1 : km = n$

$m \perp n$ iff there is no $k \geq 2 : k | m$ and $k | n$



partitions

Given a set A , a *partition* of A is a set of pairwise disjoint sets $\{B_i : i \in I\}$, such that

$$A = \bigcup_{i \in I} B_i$$



A: EU citizens, I: EU member states, B_i : citizens of country i

A: atoms, I: elements, B_i : atoms of element i

A: natural numbers, I: primes, B_i : multiples of i (excluding i)



partitions

Given a set A , a partition
 $\{B_i : i \in I\}$, such that

EU citizens: no, since a EU citizen may be a citizen of more than one country

atoms: yes, all atoms belong to exactly one element

natural numbers: no, 1. the multiple of different primes are not disjoint, and 2. without the primes themselves their union does not include all natural numbers.



A: EU citizens, I: EU member states, B_i : citizens of country i

A: atoms, I: elements, B_i : atoms of element i

A: natural numbers, I: primes, B_i : multiples of i (excluding i)

equivalence class, quotient set

Equivalence relations can be used to generate partitions.

Given a set A and an equivalence relation \approx on A , for any $a \in A$ we define the equivalence class of a $[a]_{\approx}$ as $[a]_{\approx} = \{b \in A : a \approx b\}$

Alternative syntax:

SLAM $\left\{ \begin{array}{l} [a] \\ |a| \\ |a|_{\approx} \end{array} \right\}$ when the relation is understood

Given a set A and an equivalence relation \approx on A , the quotient (set) A/\approx is defined as $A/\approx = \{|a|_{\approx} : a \in A\}$

SLAM 2.5.4:

1. Every partition is the quotient of an equivalence relation.
2. Every quotient set is a partition.



Review the proof in the book. Connect it to these definitions.

order relation, poset

A binary relation $\preceq \subseteq A \times A$ is an (*inclusive or non-strict*) (*partial*) order iff it is

1. reflexive
2. antisymmetric
3. transitive



What about these:

- divides: $m \mid n$ iff there is $k \geq 1 : km = n$
- set inclusion: \subseteq
- on numbers: \leq and $<$
- proper set inclusion: \subset

divides: yes
set inclusion: yes
less than or equal to: yes
less than: no
proper set inclusion: no

order relation, poset

A binary relation $\preceq \subseteq A \times A$ is an (*inclusive or non-strict*) (*partial*) *order* iff it is

1. reflexive
2. antisymmetric
3. transitive



What about these:

- divides: $m \mid n$ iff there is $k \geq 1 : km = n$
- set inclusion: \subseteq
- on numbers: \leq and $<$
- proper set inclusion: \subset

A pair (A, \preceq) where A is a set and $\preceq \subseteq A \times A$ a partial order on A is called a *partially ordered set* or *poset*.

Examples: $(\mathbb{N}^+, |)$

$(\mathcal{P}(A), \subseteq)$

strict (partial) order

A binary relation $\prec \subseteq A \times A$ is a *strict (partial) order* iff it is

1. irreflexive
2. transitive

Note: Irreflexivity and transitivity imply asymmetry.



How?

irreflexivity:

$$a \not\prec a$$

transitivity:

if $a \prec b$ and $b \prec c$ then $a \prec c$

asymmetry:

if $a \prec b$ then $b \not\prec a$

total (or linear) order

A binary relation $\preceq \subseteq A \times A$ is a (*non-strict*) total (or linear) order iff it is

1. reflexive

2. antisymmetric

3. transitive

4. total (complete): $a \preceq b$ or $b \preceq a$

divides: no, not total
set inclusion: no, not total
less then or equal: yes
less than: no, not reflexive



What about these:

- divides: $m \mid n$ iff there is $k \geq 1 : km = n$

- set inclusion: \subseteq

- on numbers: \leq and $<$

transitive closure

The *transitive closure* R^+ of a binary relation $R \subseteq A \times A$ is defined as follows:

$$R^+ = \bigcup_{i \in \mathbb{N}} R_i \text{ with}$$

$$R_0 = R$$

$$R_{n+1} = R_n \cup \{(a, c) : \text{if } aR_n b \text{ and } bR_n c \text{ for some } b \in A\}$$

R^*
alternative syntax
(SLAM)

$$\bowtie = \{(F, E), (E, F), (F, B), (B, F), (F, D), (D, F), (F, CH), (CH, F), (F, I), (I, F), (B, NL), (NL, B), (B, D), (D, B), (D, NL), (NL, D), (D, CH), (CH, D), (CH, I), (I, CH), (GB, IRL), (IRL, GB)\}$$

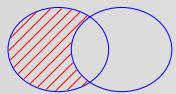


What is the meaning of \bowtie^+ ?
What are its properties?



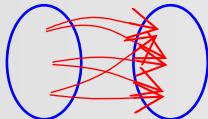
$$R = \{x : x \notin x\}$$

sets



$$\heartsuit \subseteq P \times Q$$

relations

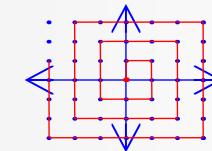


$$f : A \longrightarrow B$$

functions

$$A \hookrightarrow B$$

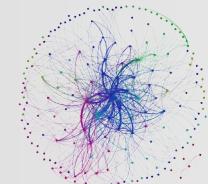
investigate



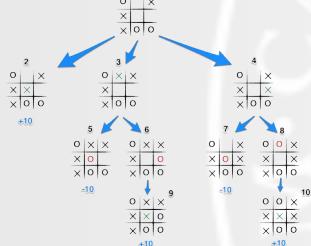
infinity

working with infinite
(or arbitrarily large) stuff

graphs



trees



definition, construction,
recursion, induction
(also: proofs, logic)

domain, range, codomain

When talking about functions, some of the terminology is the same as for relations in general, some is not:

	$R \subseteq A \times B$	$f : A \rightarrow B$
actual values, left	domain	domain
actual values, right	range	range
A	source	domain
B	target	codomain

dom(f) = A

about B^A ...

Why B^A ?

f	1	2	3	4
a	1	0	0	0
b	0	0	1	0
c	1	0	0	0



we make a choice
for each of these, #(A) times

← each time, we choose from
these #(B) options

$$\underbrace{\#(B) \cdot \dots \cdot \#(B)}_{\#(A) \text{ times}} = \#(B)^{\#(A)}$$

So for the number of functions from A to B, we have $\#(B^A) = \#(B)^{\#(A)}$.



How many *relations* $R \subseteq A \times B$?



about B^A ...

Why B^A ?

f	1	2	3	4
a	1	0	0	0
b	0	0	1	0
c	1	0	0	0

← each time, we chose from these #(B) options

we make a choice
for each of these #(A) times

So for the number of

This question asks for the number of subsets of $A \times B$,
i.e.

$$\#\mathcal{P}(A \times B) = 2^{\#(A \times B)} = 2^{\#A \#B}$$



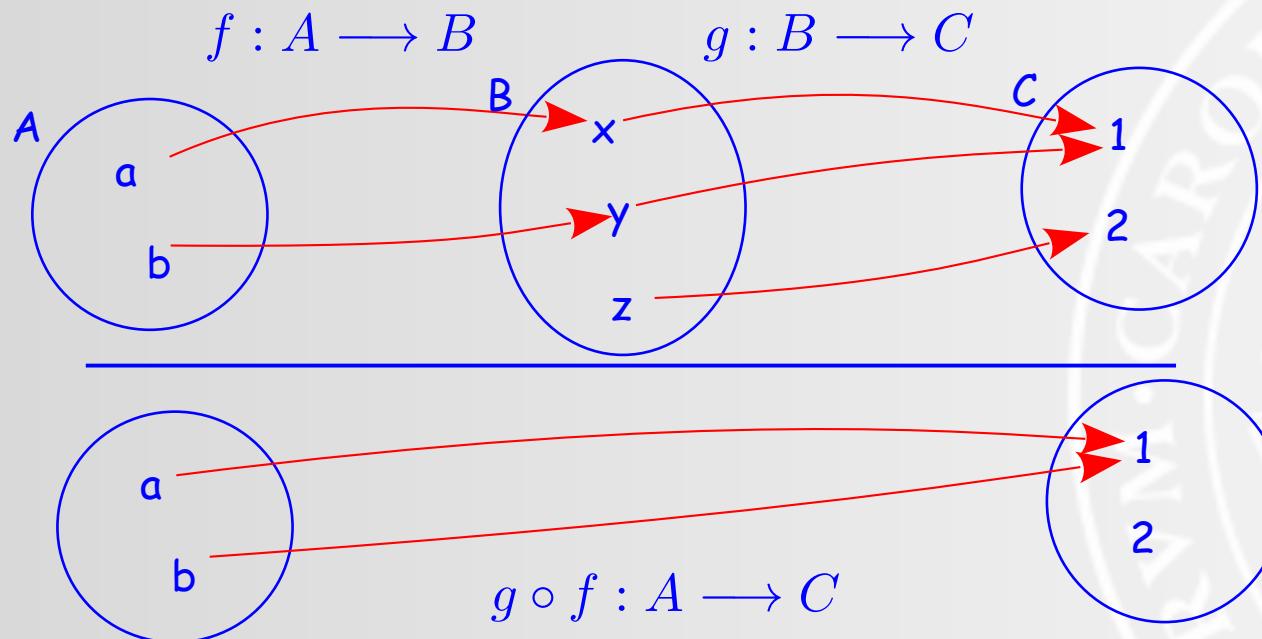
How many *relations* $R \subseteq A \times B$?

composition

Given functions $f : A \rightarrow B$ and $g : B \rightarrow C$ their composition
 $g \circ f : A \rightarrow C$ defined as:

$$g \circ f(a) = g(f(a))$$

Function composition is just a special case of composition of relations:



It is associative:
 $(h \circ g) \circ f = h \circ (g \circ f)$

So we can omit the parentheses and write
 $h \circ g \circ f$

\mathbb{Q} : the rational numbers

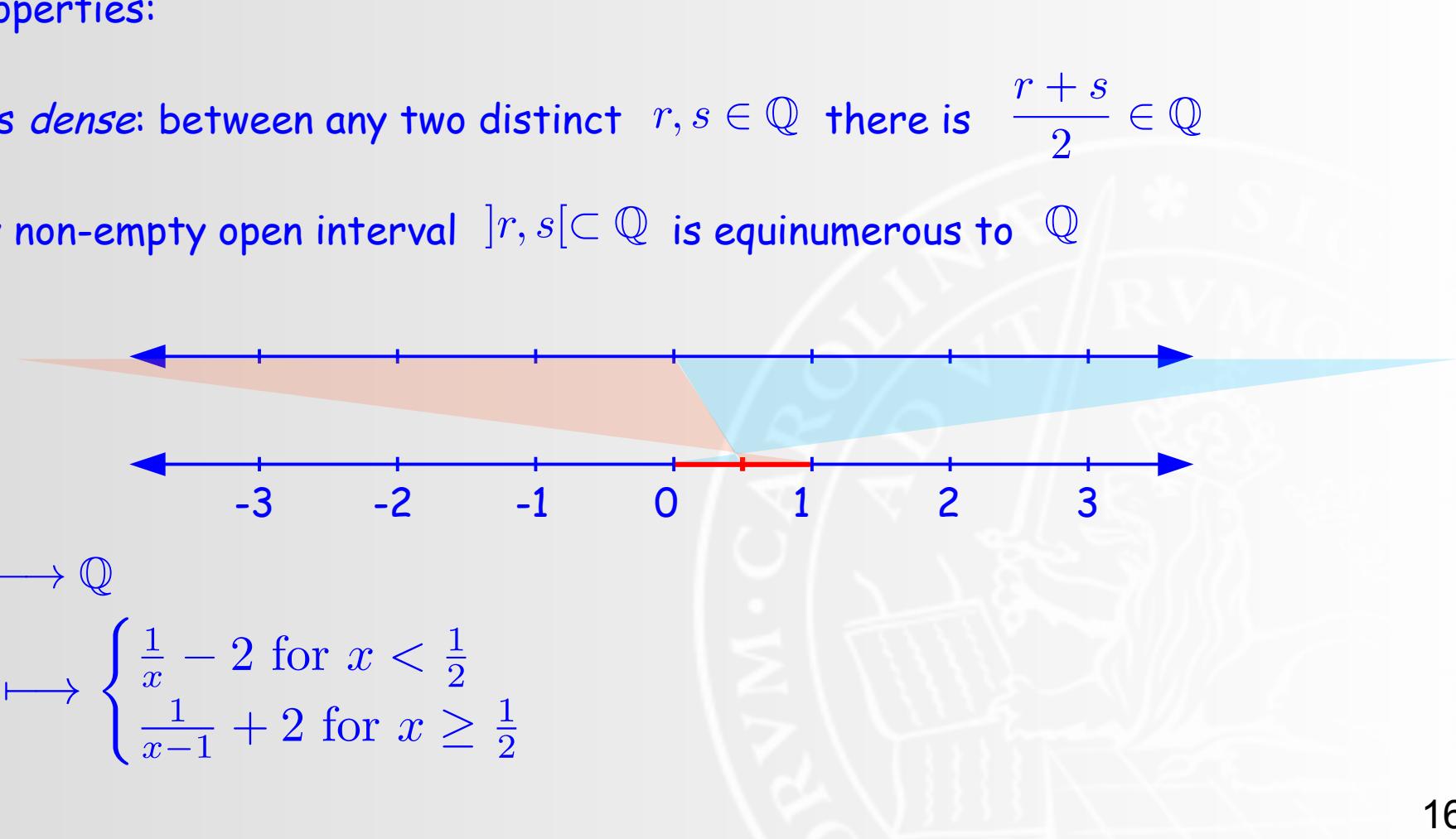
Some properties:

1. \mathbb{Q} is dense: between any two distinct $r, s \in \mathbb{Q}$ there is $\frac{r+s}{2} \in \mathbb{Q}$
2. Any non-empty open interval $]r, s[\subset \mathbb{Q}$ is equinumerous to \mathbb{Q}

For example:

$$p_{\mathbb{Q}} :]0, 1[\mathbb{Q} \longleftrightarrow \mathbb{Q}$$

$$x \mapsto \begin{cases} \frac{1}{x} - 2 & \text{for } x < \frac{1}{2} \\ \frac{1}{x-1} + 2 & \text{for } x \geq \frac{1}{2} \end{cases}$$



function definitions

What, exactly, is the relationship between the $f: x \mapsto \dots$ style of defining a function, and concept of a function as a set of pairs?

$$\begin{aligned} f : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\mapsto 2x \end{aligned}$$

$$f = \{(x, 2x) : x \in \mathbb{R}\}$$

$$\begin{array}{ccc} \{(x, x) : x \in \mathbb{R}, x \geq 0\} & & g = \{(x, x) : x \in \mathbb{R}, x \geq 0\} \cup \{(x, -x) : x \in \mathbb{R}, x < 0\} \\ g : \mathbb{R} \longrightarrow \mathbb{R} & \diagup & \\ x \mapsto \begin{cases} x & \text{for } x \geq 0 \\ -x & \text{for } x < 0 \end{cases} & & \\ & \diagdown & \\ & \{(x, -x) : x \in \mathbb{R}, x < 0\} & \end{array}$$

recursive function definitions (v 1.0)

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

$$n \mapsto \begin{cases} 1 & \text{for } n = 0 \\ n \cdot f(n - 1) & \text{for } n > 0 \end{cases}$$

$$f = \bigcup_{i \in \mathbb{N}} f_i$$

$$f_0 = \emptyset$$

$$f_{i+1} = f_i \cup \{(0, 1)\} \cup \{(n, n \cdot v) : n \in \mathbb{N}, v \in f_i(n - 1), n > 0\}$$

$$S_0 = X$$

$$S_{k+1} = S_k \cup F(S_k)$$

$$S = \bigcup_{k \in \mathbb{N}} S_k$$



Note that f_i is a relation, so $f_i(n - 1)$ computes the image of $n - 1$ under f_i , which is a set of values. If it is empty, nothing bad happens, there just is no value for v .

recursive function definitions

$$f = \bigcup_{i \in \mathbb{N}} f_i \quad \begin{aligned} f_0 &= \emptyset \\ f_{i+1} &= f_i \cup \{(0, 1)\} \cup \{(n, n \cdot v) : n \in \mathbb{N}, v \in f_i(n-1), n > 0\} \end{aligned}$$

i	f_i	$\{(n, n \cdot v) : n \in \mathbb{N}, v \in f_i(n-1), n > 0\}$
0	\emptyset	\emptyset
1	$\{(0, 1)\}$	$\{(1, 1)\}$
2	$\{(0, 1), (1, 1)\}$	$\{(1, 1), (2, 2)\}$
3	$\{(0, 1), (1, 1), (2, 2)\}$	$\{(1, 1), (2, 2), (3, 6)\}$
4	$\{(0, 1), (1, 1), (2, 2), (3, 6)\}$	$\{(1, 1), (2, 2), (3, 6), (4, 24)\}$

closure under relation

Given a relation $R \subseteq A \times A$ and a set $X \subseteq A$, the closure of X under R $R[X]$ is defined as the smallest $Y \subseteq A$ such that

$$X \subseteq Y \text{ and } R(Y) \subseteq Y$$

Construction:

$$Y_0 = X$$

$$Y_{n+1} = Y_n \cup R(Y_n)$$

$$R[X] = \bigcup_{i \in \mathbb{N}} Y_i$$



$R[C]$?

$R[\{\varepsilon\}]$?

$R[C]$: the set of all odd-length palindromes (in C)

$R[\{\varepsilon\}]$: the set of all even-length palindromes

Example:

$$C = \{"a", \dots, "z"\}$$

$$R \subseteq C^* \times C^*$$

$$R = \{(s, asa) : s \in C^*, a \in C\}$$



How to express the set of all palindromes?

closure under relations, rules, generators

Given a set A , a family of relations on A $R = \{R_i \subseteq A^{n_i} : i \in I\}$ and a set $X \subseteq A$, the closure of X under R $R[X]$ is defined as the smallest $Y \subseteq A$ such that

$$X \subseteq Y \text{ and } R_i(Y^{n_i-1}) \subseteq Y \text{ for all } i \in I$$

The elements of R are also called *rules, constructors, generators.*

Example:

$$R = \{R_1, R_2, R_3\}, C = \text{UTF-16}$$

$$R_1 = \{(s, " - " s) : s \in C^*\}$$

$$R_2 = \{(s_1, s_2, "(" s_1 "+" s_2 ")"') : s_1, s_2 \in C^*\}$$

$$R_3 = \{(s_1, s_2, "(" s_1 "*" s_2 ")"') : s_1, s_2 \in C^*\}$$

Construction:

$$Y_0 = X$$

$$Y_{n+1} = Y_n \cup \bigcup_{i \in I} R_i(Y_n^{n_i-1})$$

$$R[X] = \bigcup_{i \in \mathbb{N}} Y_i$$

$R[V]$ is the set of all expressions with unary - and binary + and *, and the variable symbols in V

$$V = \{"a", ..., "z"\}^* \setminus \{\varepsilon\}$$



$$R[V] ?$$

structural recursion

$$R = \{R_1, R_2, R_3\}, C = \text{UTF-16}$$

$$V = \{"a", \dots, "z"\}^* \setminus \{\varepsilon\}$$

$$R_1 = \{(s, "-" s) : s \in C^*\}$$

$$R_2 = \{(s_1, s_2, "(" s_1 "+" s_2 ")") : s_1, s_2 \in C^*\}$$

$$R_3 = \{(s_1, s_2, "(" s_1 "*" s_2 ")") : s_1, s_2 \in C^*\}$$

Let's write a function that evaluates an expression in $R[V]$.

Assume a function $E : V \rightarrow \mathbb{R}$ that assigns every variable a value.

$$\text{eval}_E : R[V] \rightarrow \mathbb{R}$$

$$\text{eval}_E : s \mapsto \begin{cases} E(s) & \text{for } s \in V \\ -\text{eval}_E(s') & \text{for } s' \in R[V], s = "-" s' \\ \text{eval}_E(s_1) + \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = "(" s_1 "+" s_2 ")" \\ \text{eval}_E(s_1) \cdot \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = "(" s_1 "*" s_2 ")" \end{cases}$$

Note how the structure is decomposed (or deconstructed) in these clauses!



Write a function that returns for every expression s the set of variables that occur in it.

unique decomposability

$$R = \{R_1, R_2\}, C = \text{UTF-16}$$

$$R_1 = \{(s_1, s_2, s_1 \text{ "+" } s_2) : s_1, s_2 \in C^*\}$$

$$V = \{"a", ..., "z"\}^* \setminus \{\varepsilon\}$$

$$R_2 = \{(s_1, s_2, s_1 \text{ "-" } s_2) : s_1, s_2 \in C^*\}$$

Let's look at a variant of the previous example:

$$\text{eval}_E : s \mapsto \begin{cases} E(s) & \text{for } s \in V \\ \text{eval}_E(s_1) + \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = s_1 \text{ "+" } s_2 \\ \text{eval}_E(s_1) - \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = s_1 \text{ "-" } s_2 \end{cases}$$



What is the problem with this function definition?

It does **not** uniquely decompose a string such as “a – b + c – d”.

unique decomposability

Suppose we have generators $R = \{R_1, \dots, R_k\}$ with $R_k \subseteq A^{n_k+1}$, and basis $X \subseteq A$.

The general form of a recursive function $f : R[X] \rightarrow V$ is

$$f : x \mapsto \begin{cases} h_X(x) & \text{for } x \in X \\ h_1(f(x_1), \dots, f(x_{n_1})) & \text{for } x_i, x \in R[X], (x_1, \dots, x_{n_1}, x) \in R_1 \\ \dots \\ h_k(f(x_1), \dots, f(x_{n_k})) & \text{for } x_i, x \in R[X], (x_1, \dots, x_{n_k}, x) \in R_k \end{cases}$$

It is only well-defined if all $x \in R[X]$ are *uniquely decomposable*.



cf. section 4.6.3 in SLAM.

the idea

$$A(m, n) = \begin{cases} n + 1 & \text{for } m = 0 \\ A(m - 1, 1) & \text{for } m > 0, n = 0 \\ A(m - 1, A(m, n - 1)) & \text{otherwise} \end{cases}$$

We need to "order" the elements in domains such as $R[V]$ and $\mathbb{N} \times \mathbb{N}$ in such a way that

- (a) recursive calls are always made with "smaller" argument values than the call received that they are made from, and
- (b) from any possible argument value, you cannot "descend" forever --- eventually, the function will call itself with a value for which there is no "smaller" value to recursively call itself with.

$$\text{eval}_E : R[V] \longrightarrow \mathbb{R}$$

$$\text{eval}_E : s \mapsto \begin{cases} E(s) & \text{for } s \in V \\ -\text{eval}_E(s') & \text{for } s' \in R[V], s = "-" s' \\ \text{eval}_E(s_1) + \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = "(" s_1 "+" s_2 ")" \\ \text{eval}_E(s_1) \cdot \text{eval}_E(s_2) & \text{for } s_1, s_2 \in R[V], s = "(" s_1 "*" s_2 ")" \end{cases}$$

well-founded sets

A poset $(A, <)$ is well-founded iff all non-empty subsets $X \subseteq A$ have a minimal element, i.e.

for some $m \in X$ and all $a \in X, a \not< m$

Intuitively, this means there are no infinite descending chains:

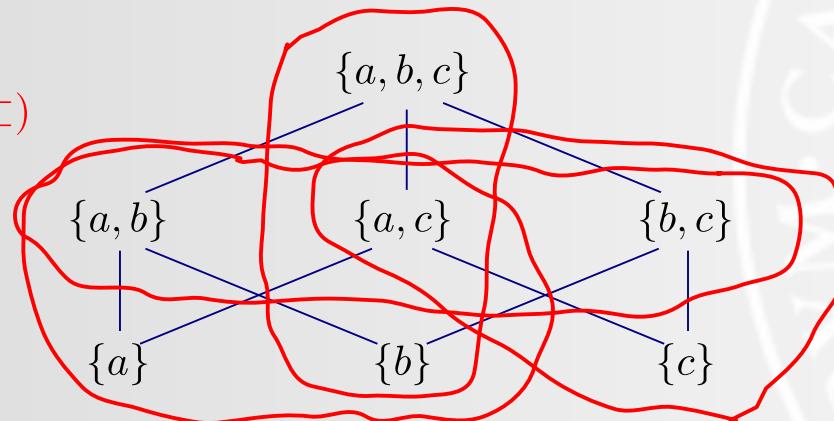
$$\dots < a_n < a_{n-1} < \dots < a_2 < a_1 < a_0$$



minimal element != minimum element:

- minimal means that there is no smaller element
- minimum means all other elements are greater

$(\mathcal{P}(\{a, b, c\}) \setminus \{\emptyset\}, \subset)$



well-founded sets

A poset $(A, <)$ is **well-founded** iff all non-empty subsets $X \subseteq A$ have a minimal element, i.e.
for some $m \in X$ and all $a \in X, a \not< m$



$(\mathbb{N}, <)$ ✓

$(\mathbb{Z}, <)$ ✗

$(\mathbb{Q}_0^+, <)$ ✗

$(\mathcal{P}(\mathbb{N}), \subset)$ ✗

$(\mathbb{N}^2, <_{\text{lex}})$ ✓

$(\mathbb{N}^2, <_{\text{prod}})$ ✓

$(a_1, a_2) <_{\text{prod}} (b_1, b_2)$ iff $a_1 < b_1$ and $a_2 < b_2$

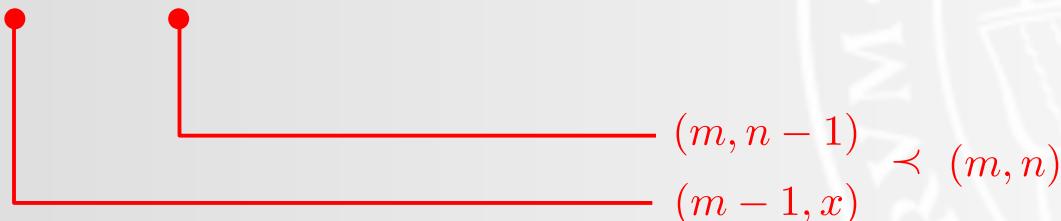
$(a_1, a_2) <_{\text{lex}} (b_1, b_2)$ iff $a_1 < b_1$ or $(a_1 = b_1 \text{ and } a_2 < b_2)$

well-founded recursion

Given a well-founded set (W, \prec) and a recursive function definition for a function $f : W \rightarrow X$, f is *well-defined* if it computes the value for every $w \in W$ only depending on values of f for $v \prec w$.

So how do we use this in practice? How can we tell that the Ackermann function is well-defined?

$$A : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$
$$A(m, n) = \begin{cases} n + 1 & \text{for } m = 0 \\ A(m - 1, 1) & \text{for } m > 0, n = 0 \\ A(m - 1, A(m, n - 1)) & \text{otherwise} \end{cases}$$



$$(m - 1, 1) \prec (m, 0)$$
$$(m - 1, x) \prec (m, n)$$
$$(m, n - 1) \prec (m, n)$$

well-founded recursion

Given a well-founded set $(W, <)$ and a recursive function definition for a function $f : W \rightarrow X$,
f is *well-defined* if it computes the value for every $w \in W$ only depending on values of f for $v < w$.

$$(a_1, a_2) <_{\text{lex}} (b_1, b_2) \text{ iff } a_1 < b_1 \text{ or } (a_1 = b_1 \text{ and } a_2 < b_2)$$



Which order would make $\mathbb{N} \times \mathbb{N}$ well-founded, such that the inequalities on the right would be true?

$$\begin{aligned}(m - 1, 1) &\prec (m, 0) \\ (m - 1, x) &\prec (m, n) \\ (m, n - 1) &\prec (m, n)\end{aligned}$$



Now take a look at the “buggy” version of the Ackermann function, and track down, how it would fail to meet the requirements of well-founded recursion.

a simple inductive proof

$$\text{sum}(n) = \sum_{i=1 \dots n} i = 1 + \dots + n$$

Hypothesis: $\text{sum}(n) = \frac{n(n+1)}{2}$



Basis: $\text{sum}(1) = \frac{1(1+1)}{2} = 1$



Induction step:

induction hypothesis

Assuming that

$$\text{sum}(k) = \frac{k(k+1)}{2}$$

show that

$$\text{sum}(k+1) = \frac{(k+1)(k+2)}{2}$$

induction goal

$$\text{sum}(k+1) = \text{sum}(k) + (k+1)$$

def sum()

$$= \frac{k(k+1)}{2} + (k+1)$$

IH

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+1)(k+2)}{2}$$

a simple induction principle

Hypothesis: $P(n)$ for all n

Basis: Show that $P(1)$ (or $P(0)$ or some other, depending on circumstances)

Induction step: Assuming that $\underbrace{P(k)}$ show that $\underbrace{P(k + 1)}$

induction hypothesis induction goal

Things that often go wrong:

- mixing basis and induction step: do not try to do everything at once
- confusing induction hypothesis and induction goal
- not using the induction hypothesis: it ain't cheating!
- getting lost: proving the goal can be messy, keep eyes on prize



cumulative (complete) induction

Cumulative, also *complete* or *strong*, induction uses an induction hypothesis that assumed the truth of the hypothesis for all smaller values, instead of just the previous one.

Hypothesis: $g(n) \leq 2^n$



Basis: $g(0) \leq 2^0 = 1$



$$g(1) \leq 2^1 = 2$$

Induction step: induction hypothesis

Assuming that $g(m) \leq 2^m$ for all $m < k$

show that $g(k) \leq 2^k$

induction goal

$$g(n) = \begin{cases} 1 & \text{for } n \leq 1 \\ g(n-1) + g(n-2) & \text{otherwise} \end{cases}$$

$$g(k) = g(k-1) + g(k-2) \quad \text{def } g()$$

$$\leq 2^{k-1} + 2^{k-2}$$

IH

$$= \frac{1}{2}2^k + \frac{1}{4}2^k$$

$$\leq 2^k$$

cumulative induction principle

Hypothesis: $P(n)$

Basis: $P(0), \dots$

Induction step:

Assuming that $\underbrace{P(m) \text{ for all } m < k}_{\text{induction hypothesis}}$ show that $\underbrace{P(k)}_{\text{induction goal}}$

Note that the basis is the vacuous form of the induction step, for $k=0$.
As a result, the basis is subsumed by the induction step.
In practice, it is often treated separately.

structural induction

$$R = \{R_1, R_2, R_3\}, C = \text{UTF-16}$$

$$V = \{"a", ..., "z"\}^* \setminus \{\varepsilon\}$$

$$R_1 = \{(s, "-" s) : s \in C^*\}$$

$$R_2 = \{(s_1, s_2, "(" s_1 "+" s_2 ")") : s_1, s_2 \in C^*\}$$

$$R_3 = \{(s_1, s_2, "(" s_1 "*" s_2 ")") : s_1, s_2 \in C^*\}$$

Hypothesis: Every $s \in R[V]$ contains equal numbers of opening and closing parentheses.



Basis: Every $s \in V$ contains equal numbers of opening and closing parentheses.



Induction step:

induction hypothesis

All rules $R_i \in R$ preserve the property: if their "input" objects have it, then so does their "output" object.

induction goal

$$R_1 = \{(s, "-" s) : s \in C^*\}$$

$$R_2 = \{(s_1, s_2, "(" s_1 "+" s_2 ")") : s_1, s_2 \in C^*\}$$

$$R_3 = \{(s_1, s_2, "(" s_1 "*" s_2 ")") : s_1, s_2 \in C^*\}$$



structural induction principle

Structural induction is a variant of cumulative induction, but instead of natural numbers we induce over the structure of rule-generated objects in some structurally-recursive set $R[X]$.

Hypothesis: $P(x)$ for all $x \in R[X]$

Basis: $P(x)$ for all $x \in X$

Induction step:

All rules $R_i \in R$ preserve the property: if their “input” objects have it, then so does their “output” object.

induction hypothesis

induction goal

well-founded induction



Cumulative (complete, strong) induction assumed that a property needed to be shown over the natural numbers, structural induction that the set was generated inductively through generators.

Well-founded induction generalizes the idea to all well-founded sets.

Given a well-founded set $(A, <)$ and a property $P(a), a \in A$, if

for all $a \in A : P(w)$ for all $w < a$ implies $P(a)$

then $P(a)$ for all $a \in A$

induction hypothesis

induction goal

Emmy Noether
1882-1935

As with cumulative induction, the base case is subsumed by the induction step.
In practice, it is still sometimes handled separately.

well-founded induction

Hypothesis: Every $n \geq 2$ can be factored into primes.

well-founded order: $(\{n \in \mathbb{N} : n \geq 2\}, |_{\neq})$ (divides-relation, strict version)



Descending chains?
Minimal elements?

Descending chains look like this: $7|_{\neq} 21|_{\neq} 42|_{\neq} 126|_{\neq} 252|_{\neq} \dots$

The minimal elements are the primes.

well-founded induction

Hypothesis: Every $n \geq 2$ can be factored into primes.

well-founded order: $(\{n \in \mathbb{N} : n \geq 2\}, |_{\neq})$ (divides-relation, strict version)

(Base:) Trivially true for every minimal element in $(\{n \in \mathbb{N} : n \geq 2\}, |_{\neq})$

Induction step: If n is not minimal, then there is a k such that $k|_{\neq} n$, and thus there is some m such that $n = km$. k and m can be prime-factored, therefore so can n .

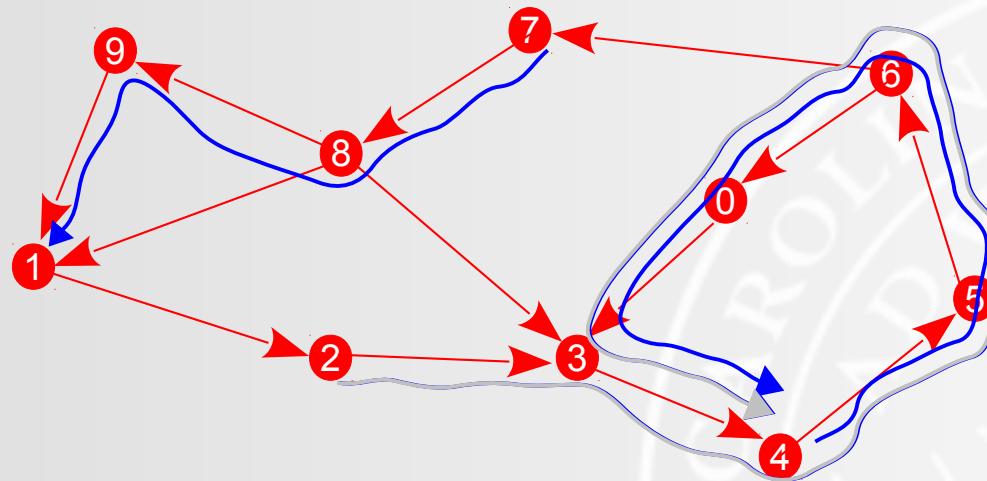


How do we know that m can be factored into primes?

Because $m|_{\neq} n$, too.

paths

Given a graph (V, E) , a *path* is a finite sequence a_0, \dots, a_n in V with $n \geq 1$ such that $(a_{k-1}, a_k) \in E$ for $1 \leq k \leq n$. The *length* of the path is n .

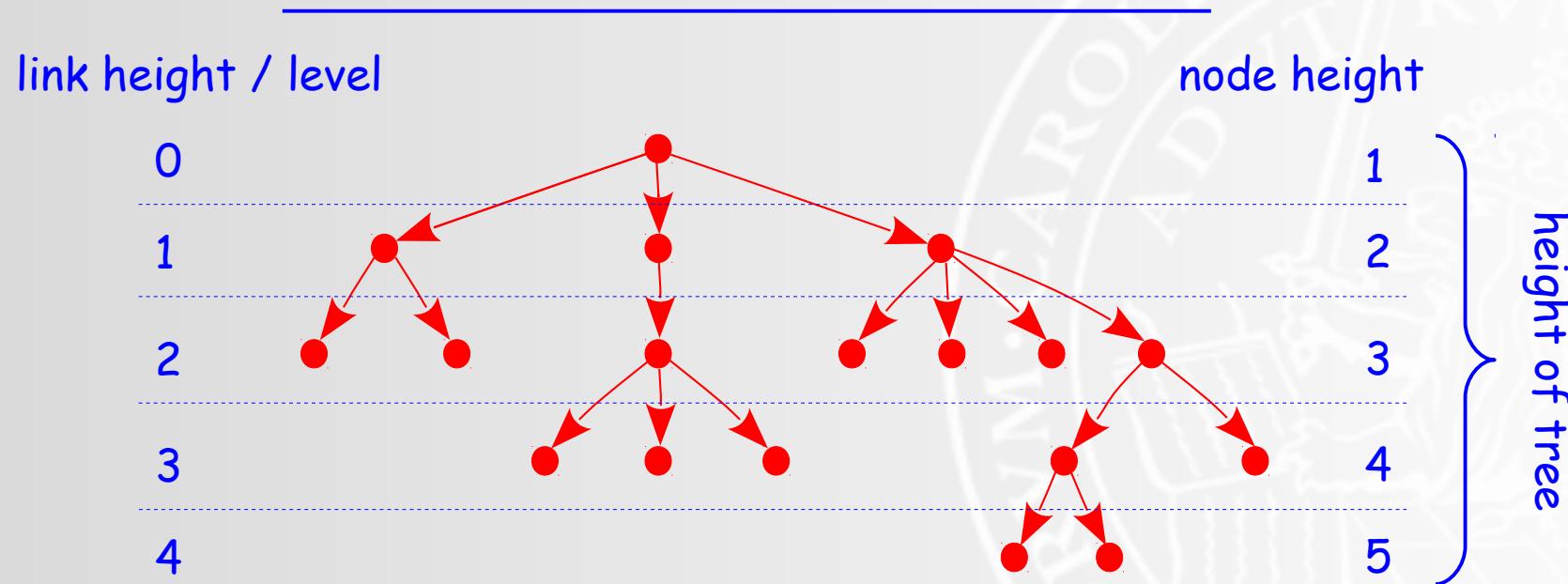


A *cycle* is a path a_0, \dots, a_n where $a_0 = a_n$.

A graph that does not contain cycles is called *acyclic*.

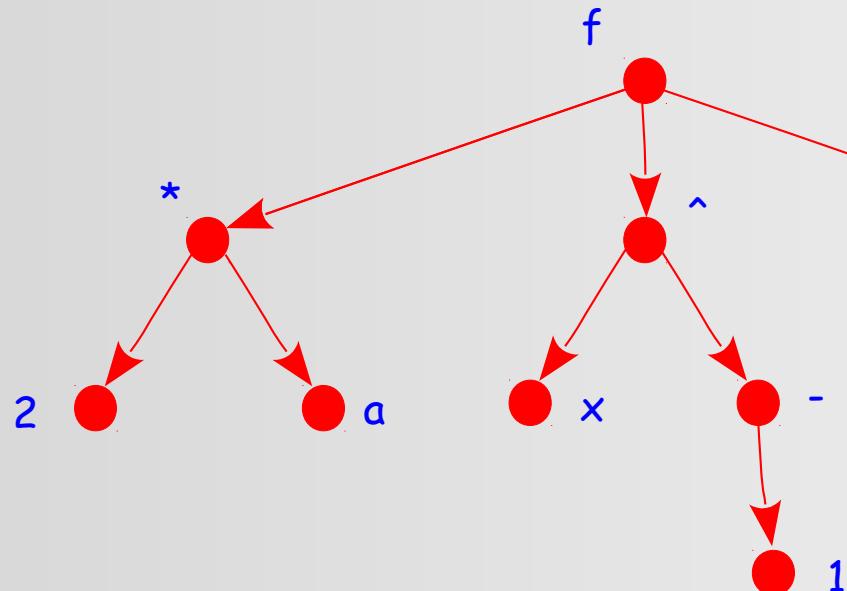
properties

- (1) Any non-empty tree has a unique root.
- (2) A root has no parent.
- (3) Every non-root has exactly one parent.
- (4) A tree with n nodes has $n-1$ links.
- (5) A tree contains no cycles.



labeled trees

Given a tree (T, R) , and a set of labels L , a labeling is a function $\lambda : T \rightarrow L$
A tree with a labeling is called a labeled tree.



In practice, the labeling function is often realized by adding data to the nodes of a tree.

1 Not uniquely – the tree definition contains no ordering information.
It represents just as well
 $f(1, 2^*a, (-1)^x)$



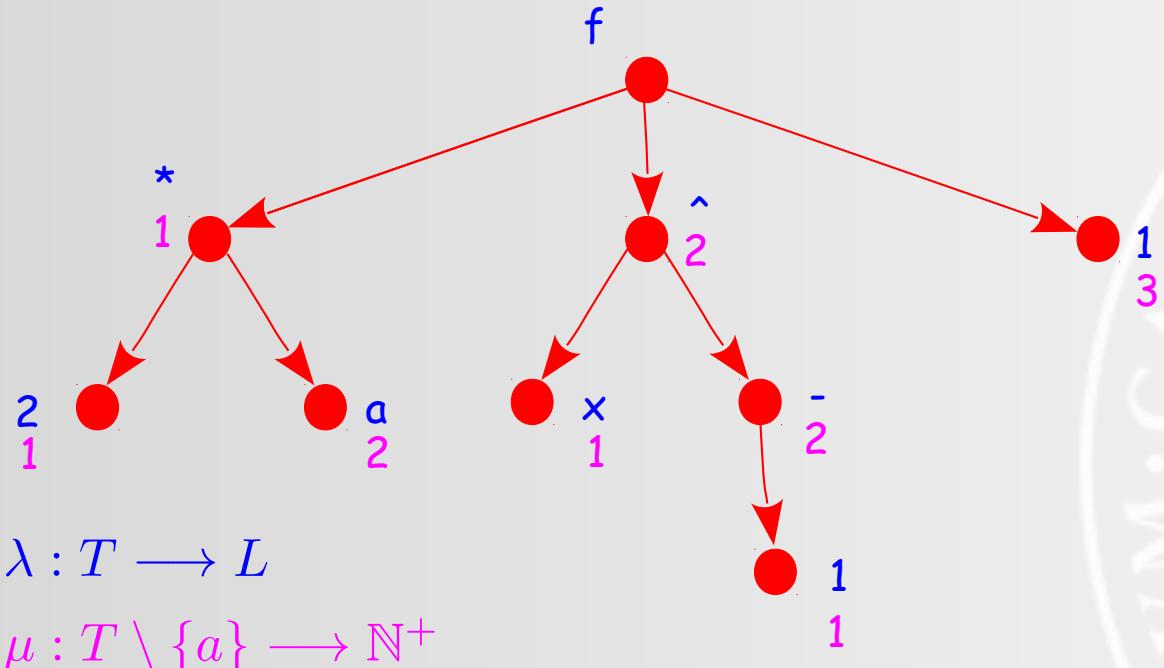
What expression might the tree represent?
Does it?

ordered trees

Given a tree (T, R) with root a , we say it is *ordered* if there is a function

$$\mu : T \setminus \{a\} \longrightarrow \mathbb{N}^+$$

such that for every node its n children are labeled $1..n$.



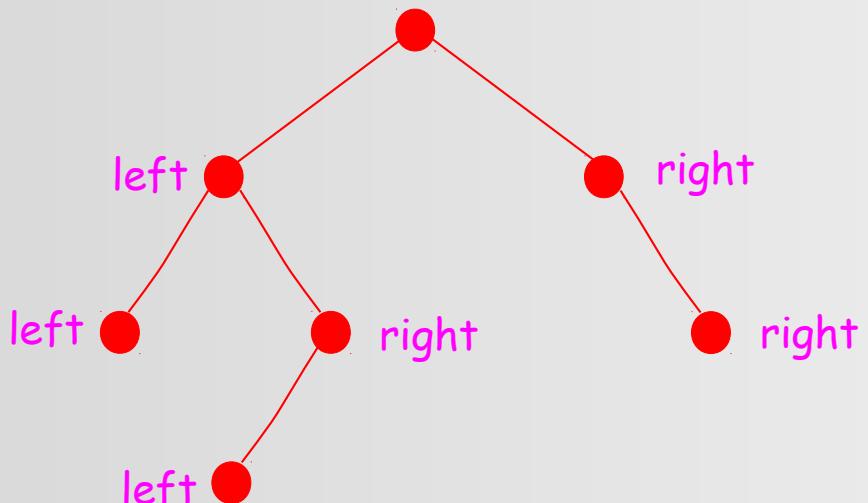
Ordering is usually represented implicitly by the left-to-right order of child nodes in the tree data structure.

binary trees

Given a tree (T, R) with root a , we say it is *binary* if every node has at most two children and there is a labeling function

$$\beta : T \setminus \{a\} \longrightarrow \{\text{left}, \text{right}\}$$

such that no two children of the same node are labeled identically.



$$\beta : T \setminus \{a\} \longrightarrow \{\text{left}, \text{right}\}$$



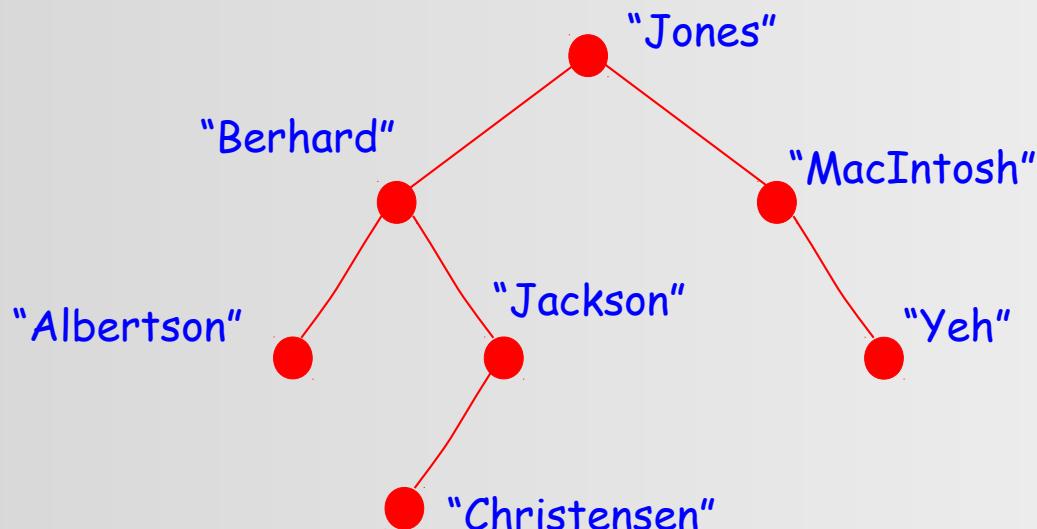
Labels are usually represented by the left-to-right order of child nodes and angled links.

binary search trees

Given a binary tree (T, R) , with root a , binary labels $\beta : T \rightarrow \{left, right\}$, and labeling function $\lambda : T \rightarrow L$ and a total order \leq on L .
It is a *binary search tree* iff for all nodes a in T :
any label in their left subtree, and less than a .

It could't.
The labels of an ordered tree establish a total order *among the children* of each node.

The labels in a binary tree also relate the children with respect to the parent.



Could the same be achieved with an ordered tree, instead of a binary tree?

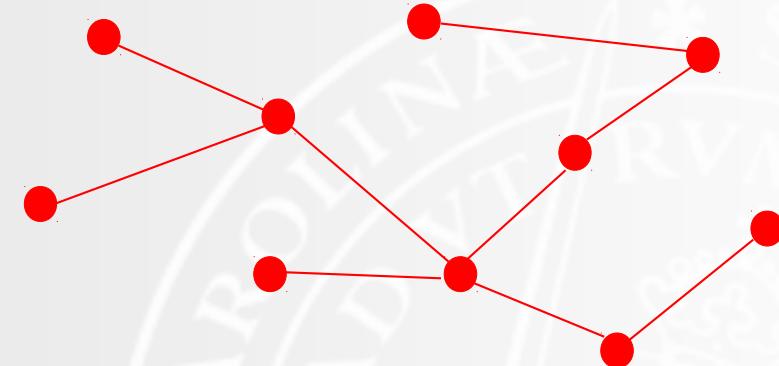
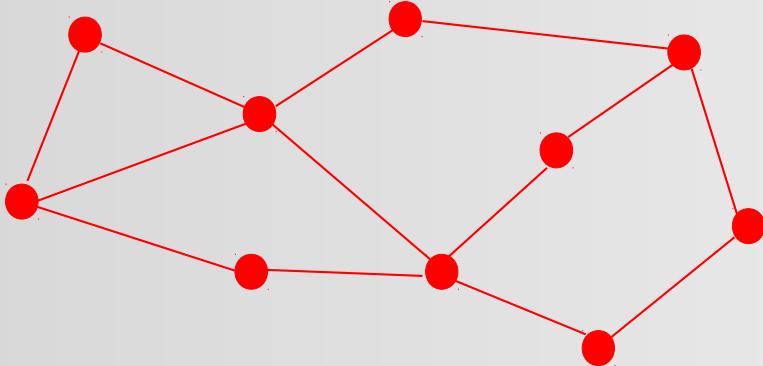
IOW, is a binary tree a special case of an ordered tree, or something else?

EDAA40 - Discrete Structures in Computer Science

6.3 - unrooted trees

spanning trees

Given an undirected graph (T, S) , an unrooted tree (T, R) is a *spanning tree* for it iff
 $R \subseteq S$



There may be many spanning trees for any given graph.

from truth table to formula

p	q	r	f(p, q, r)
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	0

← $p \wedge q \wedge r$

← $p \wedge \neg q \wedge r$



Are there truth tables for which this method does not produce a formula?

What does this mean for the connectives not, and, or?

Only the one where always $f(\dots) = 0$.
This can be expressed as $f(p, q, r) = p \wedge \neg p$.

It means these connectives are universal.
All propositional formulae can be expressed using not, and, or.

$$f : (p, q, r) \mapsto (p \wedge q \wedge r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge q \wedge \neg r)$$

tautological implication

Given a set of formulae A and a formula β we say that A tautologically implies β if there is no valuation v such that

$$v(\alpha) = 1 \text{ for all } \alpha \in A \text{ and } v(\beta) = 0$$

$$A \vdash \beta$$

$$A \vDash \beta$$

Example:

(modus ponens)

$$p, p \rightarrow q \vdash q$$

p	q	$p \rightarrow q$
1	1	1
1	0	0
0	1	1
0	0	1

Name	LHS	RHS
Simplification, \wedge^-	$\alpha \wedge \beta$	α
	$\alpha \wedge \beta$	β
Conjunction, \wedge^+	α, β	$\alpha \wedge \beta$
Disjunction, \vee^+	α	$\alpha \vee \beta$
	β	$\alpha \vee \beta$
Modus ponens, MP, \rightarrow^-	$\alpha, \alpha \rightarrow \beta$	β
Modus tollens, MT	$\neg \beta, \alpha \rightarrow \beta$	$\neg \alpha$
Disjunctive syllogism, DS	$\alpha \vee \beta, \neg \alpha$	β
Transitivity	$\alpha \rightarrow \beta, \beta \rightarrow \gamma$	$\alpha \rightarrow \gamma$
Material implication	β	$\alpha \rightarrow \beta$
	$\neg \alpha$	$\alpha \rightarrow \beta$
Limiting cases	γ	$\beta \vee \neg \beta$
	$\alpha \wedge \neg \alpha$	γ

tautological equivalence

Given two formulae α and β , we say that they are *tautomorphically equivalent* if they tautologically imply each other.

$$\alpha \dashv\vdash \beta$$

Name	LHS	RHS
Double negation	α	$\neg\neg\alpha$
Commutation for \wedge	$\alpha \wedge \beta$	$\beta \wedge \alpha$
Association for \wedge	$\alpha \wedge (\beta \wedge \gamma)$	$(\alpha \wedge \beta) \wedge \gamma$
Commutation for \vee	$\alpha \vee \beta$	$\beta \vee \alpha$
Association for \vee	$\alpha \vee (\beta \vee \gamma)$	$(\alpha \vee \beta) \vee \gamma$
Distribution of \wedge over \vee	$\alpha \wedge (\beta \vee \gamma)$	$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
Distribution of \vee over \wedge	$\alpha \vee (\beta \wedge \gamma)$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
Absorption	α	$\alpha \wedge (\alpha \vee \beta)$
	α	$\alpha \vee (\alpha \wedge \beta)$
Expansion	α	$(\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)$
	α	$(\alpha \vee \beta) \wedge (\alpha \vee \neg\beta)$
de Morgan	$\neg(\alpha \wedge \beta)$	$\neg\alpha \vee \neg\beta$
	$\neg(\alpha \vee \beta)$	$\neg\alpha \wedge \neg\beta$
	$\alpha \wedge \beta$	$\neg(\neg\alpha \vee \neg\beta)$
	$\alpha \vee \beta$	$\neg(\neg\alpha \wedge \neg\beta)$
Limiting cases	$\alpha \wedge \neg\alpha$	$\beta \wedge \neg\beta$
	$\alpha \vee \neg\alpha$	$\beta \vee \neg\beta$

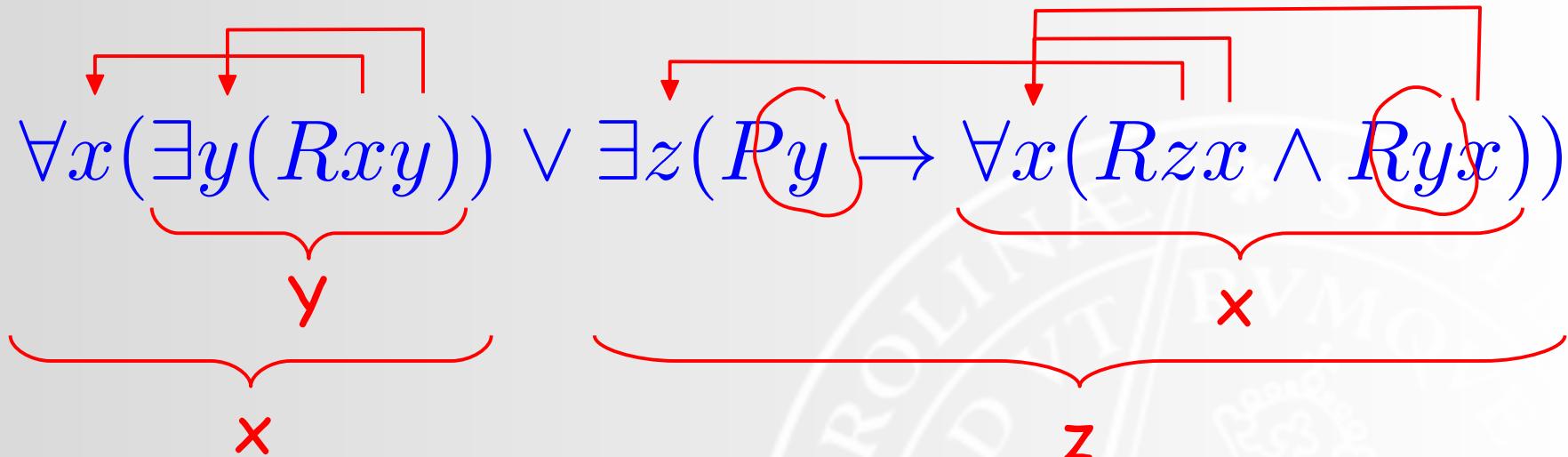


tautological equivalence

Name	LHS	RHS
→ Contraposition	$\alpha \rightarrow \beta$ $\alpha \rightarrow \neg \beta$ $\neg \alpha \rightarrow \beta$	$\neg \beta \rightarrow \neg \alpha$ $\beta \rightarrow \neg \alpha$ $\neg \beta \rightarrow \alpha$
→ Import/export	$\alpha \rightarrow (\beta \rightarrow \gamma)$ $\alpha \rightarrow (\beta \rightarrow \gamma)$	$(\alpha \wedge \beta) \rightarrow \gamma$ $\beta \rightarrow (\alpha \rightarrow \gamma)$
→ Consequential mirabilis (miraculous consequence)	$\alpha \rightarrow \neg \alpha$ $\neg \alpha \rightarrow \alpha$	$\neg \alpha$ α
Commutation for \leftrightarrow	$\alpha \leftrightarrow \beta$	$\beta \leftrightarrow \alpha$
Association for \leftrightarrow	$\alpha \leftrightarrow (\beta \leftrightarrow \gamma)$	$(\alpha \leftrightarrow \beta) \leftrightarrow \gamma$
¬ through \leftrightarrow	$\neg(\alpha \leftrightarrow \beta)$	$\alpha \leftrightarrow \neg \beta$
→ Translations between connectives	$\alpha \leftrightarrow \beta$ $\alpha \leftrightarrow \beta$ $\alpha \rightarrow \beta$ $\alpha \rightarrow \beta$ $\alpha \vee \beta$	$(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$ $(\alpha \wedge \beta) \vee (\neg \alpha \wedge \neg \beta)$ $\neg(\alpha \wedge \neg \beta)$ $\neg \alpha \vee \beta$ $\neg \alpha \rightarrow \beta$
Translations of negations of connectives	$\neg(\alpha \rightarrow \beta)$ $\neg(\alpha \wedge \beta)$ $\neg(\alpha \leftrightarrow \beta)$	$\alpha \wedge \neg \beta$ $\alpha \rightarrow \neg \beta$ $(\alpha \wedge \neg \beta) \vee (\beta \wedge \neg \alpha)$

quantifier scopes

variable uses, and the quantifier they are bound by



quantifier scopes, and the variables bound in/by them

free and bound variable occurrences

A variable occurrence is *bound* iff it occurs inside the scope of a quantifier that binds that variable.

It is *free* otherwise.

A formula with no free variable occurrences is called *closed*.
A closed formula is a *sentence*.

free occurrences

•

$$\forall z[Rxz \rightarrow \exists y(Rzy)]$$

bound occurrences

•

• •

definitions, theorems, proofs

A **definition** is a statement that gives a precise meaning to a term or a symbol.

$$A \subseteq B \text{ iff } \forall x (x \in A \rightarrow x \in B)$$

$$n \in \mathbb{Z} \text{ is even iff } \exists k \in \mathbb{Z} (n = 2k)$$

$$n \in \mathbb{Z} \text{ is odd iff } \exists k \in \mathbb{Z} (n = 2k + 1)$$

A **theorem** is a statement that needs to be proven based on definitions (and axioms).

$$A \times (B \cap C) = A \times B \cap A \times C$$

$$\#(\mathbb{N}) < \#(2^{\mathbb{N}})$$

There are infinitely many prime numbers.

Other words for theorem:
proposition, lemma, corollary.

A **proof** is a chain of logical reasoning showing the truth of a theorem.

kinds of proofs

Proofs come in different flavors, which depend on the **form of the theorem**, and the chain of reasoning best suited to prove it.

Many theorems are conditional statements, i.e. they have the form "premise implies conclusion, or

$$P \rightarrow C$$

$$\forall x \in \mathbb{Z} \ (x \text{ is odd} \rightarrow x^2 \text{ is odd})$$

$$\forall a, b, c \in \mathbb{Z} \ ((a|b \wedge b|c) \rightarrow a|c)$$

P	C	$P \rightarrow C$
T	T	T
T	F	F
F	T	T
F	F	T

direct proof

Theorem: If P, then C.

Proof: Suppose P.

...

Therefore C.

Theorem:

x is odd $\rightarrow x^2$ is odd

Proof:

Suppose x is odd.

Therefore, there is an integer k such that $x = 2k + 1$.

Thus $x^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$.

Note that $2k^2 + 2k$ is an integer.

Thus there is an integer n such that $x^2 = 2n + 1$.

Therefore x^2 is odd.

direct proof with cases

Sometimes, the premise consists of several *cases*, and it becomes easier to study each case by itself.

n	$1 + (-1)^n(2n - 1)$
1	0
2	4
3	-4
4	8
5	-8
6	12

Theorem: If $n \in \mathbb{N}$ then $1 + (-1)^n(2n - 1)$ is a multiple of 4.

Proof: Suppose $n \in \mathbb{N}$. Then n is either even or odd.

Case 1: Suppose n is even. Then $n = 2k$ for some $k \in \mathbb{Z}$.

$$\text{Thus } 1 + (-1)^{2k}(2(2k) - 1) = 1 + 1^k(4k - 1) = 4k.$$

That is a multiple of 4.

Case 2: Suppose n is odd. Then $n = 2k + 1$ for some $k \in \mathbb{Z}$.

$$\text{Thus } 1 + (-1)^{2k+1}(2(2k + 1) - 1) = 1 - (4k + 2 - 1) = -4k.$$

That is also a multiple of 4.

The result in both cases is a multiple of 4.

contrapositive proof

In some cases, it is easier to reason about a theorem in *contrapositive form*.

Theorem:

If $x^2 - 6x + 5$ is even, then x is odd.

Proof:

Suppose $x^2 - 6x + 5$ is even, i.e. there exists an integer a such that $x^2 - 6x + 5 = 2a$.

...

Thus there is an integer b such that $x = 2b + 1$.

Therefore b is odd.

direct proof:

Theorem: If P, then C.

Proof: Suppose P.

...

Therefore C.

contrapositive proof

Contrapositive form: $\neg C \rightarrow \neg P$

P	C	$P \rightarrow C$	$\neg C$	$\neg P$	$\neg C \rightarrow \neg P$
T	T	T	F	F	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Theorem: If P, then C.

Proof: Suppose not C.

...

Therefore not P.

Theorem:

If $x^2 - 6x + 5$ is even, then x is odd.

Proof:

Suppose x is even.

There is an integer a such that $x = 2a$.

$$x^2 - 6x + 5 = 4a^2 - 12a + 4 + 1 = 2(2a^2 - 6a + 2) + 1$$

So there is an integer b s.t. $x^2 - 6x + 5 = 2b + 1$.

Therefore $x^2 - 6x + 5$ is not even.

proof by contradiction

Suppose we want to prove a proposition P , not necessarily in conditional form.

Proof by contradiction uses the fact that if we can show that not P results in a logical contradiction, e.g. it implies some conclusion C as well as its opposite, not C , then not P cannot be true, and so P must be true.

Theorem:

If $a, b \in \mathbb{Z}$ then $a^2 - 4b \neq 2$.

Proof:

Suppose there are $a, b \in \mathbb{Z}$ s.t. $a^2 - 4b = 2$.

Since this implies $a^2 = 4b + 2 = 2(2b + 1)$, a^2 is even.

Hence a is even, so $a = 2c$ for some integer c .

Thus $4c^2 - 4b = 2$, i.e. $2c^2 - 2b = 1$.

Therefore $2(c^2 - b) = 1$ with $c^2 - b \in \mathbb{Z}$.

So 1 is even.

P	C	$\neg P$	$C \wedge \neg C$	$\neg P \rightarrow C \wedge \neg C$
T	T	F	F	T
T	F	F	F	T
F	T	T	F	F
F	F	T	F	F

Theorem: P .

Proof: Suppose not P .

...
Or any other
false proposition!

Therefore C and not C .