Logic and discrete mathematics (HKGAB4)

http://www.ida.liu.se/∼HKGAB4/

Organization

- **Discrete Mathematics**:
  - lectures: 5×2 hours
  - seminars: 10×2 hours
  - exam: together with the logic course
  - recommended book:
    K. Eriksson and H. Gavel
    "Diskret matematik och diskreta modeller"
    published by Studentlitteratur, Lund
- **Logic**:
  - lectures: 9×2 hours
  - seminars: 6×2 hours
  - supervised labs: 12×2 hours
  - unsupervised labs: 8×2 hours
  - exam: 1×5 hours
  - manual:
    J. Barwise, J. Etchemendy
    "Language, Proof and Logic"
    published by CSLI Publications

Discrete mathematics: contents

1. Sets: equality and inclusion, operations, Venn diagrams.
2. Relations: graphs, properties of relations.
3. Functions. Discrete structures.
4. Definitions, recursion and induction.
5. Formal Languages. Chomsky hierarchy

Logic: contents

Logic curse will be focused on practical reasoning. In particular we will:

- define a general framework for logics
- show how logics are defined and used
- show the connections between natural language phenomena and logics, in particular discuss intensional (modal) notions
- show the connections between commonsense reasoning and logics.

Sets

Intuitively a *set* is any "abstract collection" of objects, called *elements (members) of the set*. The *empty set*, denoted by $\emptyset$, is the set containing no elements.

Examples

1. the set of all persons studying in Linköping
2. the set of meals served in a given restaurant
3. the set of names in a phone book
4. the set of week days
5. the set of natural numbers $0, 1, 2, 3, \ldots$
6. the set of 3 years old kids studying computer science in Linköping (empty set).

Membership is denoted by $\in$. Expression $e \in S$ means that object e is a member of (belongs to) the set S. By writing $e \notin S$ we indicate that e is not a member of the set S.

Examples

1. August Strindberg $\in$ the set of Swedish dramatists
2. Artur Connan Doyle $\notin$ the set of Swedish dramatists
3. Wednesday $\in$ the set of weekdays
4. Wednesday $\notin$ the set of weekend days.

Important:

- we often limit consideration to a particular sets of objects, called a *domain* or a *universe*
- we also distinguish between constants and variables. *Constants* have a fixed value while *variables* are used to represent a range of possible values.

Examples

Consider the universe consisting of dates.

1. constants 18-03-1899, 27-01-2015 represent concrete days
2. if we want to represent any day, e.g., between the above two dates, we use variable, say x, and write 18-03-1899$< x <$27-01-2015.

Notation 1 (list notation): sets are denoted by $\{e_1, e_2, \ldots\}$, i.e., we use brackets "{" and "}" to denote sets and list all elements, separating them by commas ",".

Examples

1. $\{John, Mary, Paul\}$ is the set consisting of $John$, $Mary$ and $Paul$
2. $\{0, 1, 2, 3, 4\}$ is the set consisting of $0, 1, 2, 3$ and 4.

Notation 2 (extended list notation): sets are also denoted by $\{e_1, e_2, \ldots, e_n\}$, i.e., we additionally use dots "..." as an abbreviation, in the case when all elements between e_2 and e_n are known from context.

Examples

1. $\{1, \ldots, 9\}$ is the set consisting of $1, 2, 3, 4, 5, 6, 7, 8$ and 9
2. $\{Tuesday, \ldots, Friday\}$ is the set consisting of $Tuesday, Wednesday, Thursday$ and $Friday$.

Notation 3 (predicate notation): sets are also denoted by $\{x \mid$ variable x satisfies a given condition$\}$, i.e., we additionally use sign "|" and some condition.

Examples

1. $\{x \mid x$ studies psychology in Sweden$\}$ is the set consisting of all persons studying psychology in Sweden
2. $\{x \mid x$ is a weekday$\}$ is the set consisting of all weekdays.

WARNING! Notation 3 sometimes leads to paradoxes!

Example: a paradox

Consider the situation, where in a small village, say Småby,

> the barber shaves all and only those male inhabitants of Småby, who do not shave themselves.

Assume that the barber is a male and an inhabitant of Småby, too.

We are interested in the set:

$$\{x \mid x \text{ is a male inhabitant of Småby who shaves himself}\}$$

and ask a question whether the barber is a member of this set.

We have two cases:

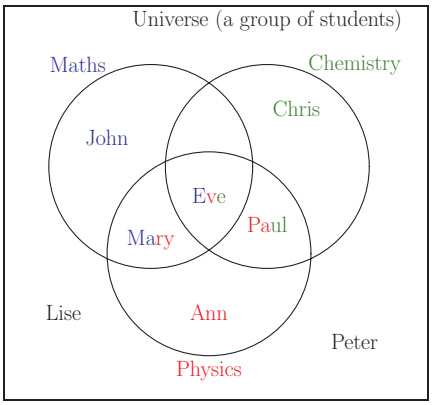
1. Case 1: the answer is "yes".
   Then the barber satisfies the condition that he is a male inhabitant of Småby who shaves himself. But he shaves only those who do not shave themselves.
   So the answer cannot be "yes".
2. Case 2: the answer is "no".
   Then the barber does not satisfy the condition that he is a male inhabitant of Småby who shaves himself. So he does not shave himself. But we said that the barber shaves those who do not shave themselves, so he shaves himself.
   So the answer cannot be "no".

We then have a paradox!

Venn diagrams

In *Venn diagrams* sets are visualized as circles. Members of sets are depicted as points. Sometimes all circles are surrounded by a rectangle representing all considered elements.

Example



Creating Venn diagrams

To create a Venn diagram, proceed as follows:

1. gather information about the considered situation:
   (a) what is known about the considered situation?
   (b) what are the most important elements of the situation?
   (c) what characteristics do the elements have in common?
   (d) what characteristics do not the elements have in common?
2. concentrate on the overlap areas
   (they show relationships among sets):
   (a) what is the intersection of chosen circles?
   (b) how can overlapping areas be named?
   (c) what relationships do they show?
3. try to discover laws illustrated by diagrams.

Set equality and set inclusion

Two sets are *equal* (or *identical*) iff they have the same elements. Set equality is denoted by $=$ and inequality is denoted by $\neq$.

Examples

1. $\{Wednesday, Friday\} = \{Wednesday, Friday\}$
2. $\{x \mid x \text{ is a weekend day}\} = \{Saturday, Sunday\}$
3. $\{x \mid x \text{ is a weekend day}\} \neq \{Wednesday, Friday\}$.

A set, say A is *included* (or, in other words, *is a subset of*) set B, denoted by $A \subseteq B$ iff all members of A are also members of B. To indicate that A is not a subset of B, we write $A \not\subseteq B$.

Examples

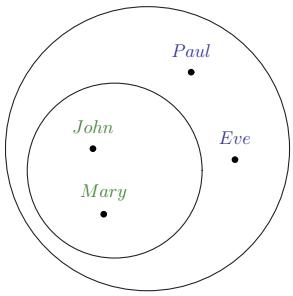
1. $\{Wednesday, Friday\} \subseteq \{Wednesday, Friday\}$
2. $\{Wednesday\} \subseteq \{Wednesday, Friday\}$
3. $\{Friday\} \subseteq \{Wednesday, Friday\}$
4. $\{x \mid x \text{ is a weekend day}\} \subseteq \{Friday, Saturday, Sunday\}$
5. $\{x \mid x \text{ is a weekend day}\} \not\subseteq \{Wednesday, Friday\}$.

A set A is *properly included* (or *is a proper subset of*) set B if both $A \subseteq B$ and $A \neq B$. Proper inclusion is denoted by $A \subsetneq B$.

Examples

1. $\{Wednesday\} \subsetneq \{Wednesday, Friday\}$
2. $\{x \mid x \text{ is a weekend day}\} \subsetneq \{Friday, Saturday, Sunday\}$.

Venn diagrams for inclusion



$\{John, Mary\} \subseteq \{John, Mary, Paul, Eve\}$

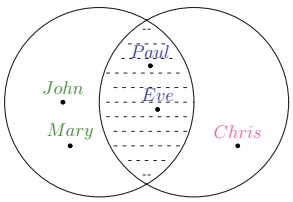
Operations on sets: intersection

Intersection of sets A and B, denoted by $A \cap B$, is the set containing all elements which are members of both A and B.

Examples

1. $\{Monday, Tuesday\} \cap \{Tuesday, Friday\} = \{Tuesday\}$
2. $\{1, 2, 4, 6\} \cap \{2, 3, 4, 5\} = \{2, 4\}$.

Venn diagrams for intersection



$\{John, Mary, Paul, Eve\} \cap \{Paul, Eve, Chris\} = \{Paul, Eve\}$

Observe that, for any two sets A, B,
$A \cap B \subseteq A$ and $A \cap B \subseteq B$.

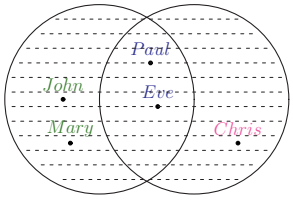
Operations on sets: union

Union of A and B, denoted by $A \cup B$, is the set containing all elements which are members of A or of B (or of both).

Examples

1. $\{Monday, Tuesday\} \cup \{Tuesday, Friday\} =$
   $= \{Monday, Tuesday, Friday\}$
2. $\{1, 2, 4, 6\} \cup \{2, 3, 4, 5\} = \{1, 2, 3, 4, 5, 6\}$.

Venn diagrams for union



$\{John, Mary, Paul, Eve\} \cup \{Paul, Eve, Chris\} =$
$= \{John, Mary, Paul, Eve, Chris\}$

Observe that, for any two sets A, B,
$A \subseteq A \cup B$ and $B \subseteq A \cup B$.

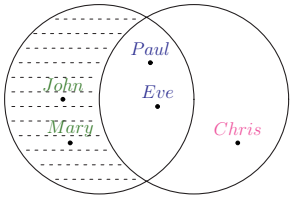
Operations on sets: difference

The *difference* of sets A and B, denoted by $A\backslash B$ (sometimes by $A - B$), is the set of all members of A which are not members of B.

Examples

1. $\{1,2,3,4\}\backslash\{2,4\} = \{1,3\}$
2. $\{dog, cat, rabbit\}\backslash\{cat, mouse\} = \{dog, rabbit\}$.

Venn diagrams for difference



$\{John, Mary, Paul, Eve\}\backslash\{Paul, Eve, Chris\} =$
$= \{John, Mary\}$

Operations on sets: complement

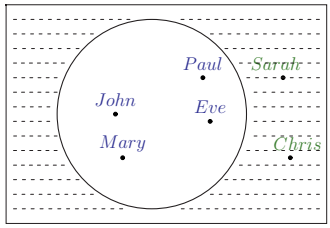
Let U be the universe of all considered objects. The *complement* of a set A, denoted by A' (sometimes by A^c or by $-A$), is the set of all objects in U, which are not in A.

Examples

Let U be the set of weekdays.

1. $\{Saturday, Sunday\}' =$
$= \{Monday, Tuesday, Wednesday, Thursday, Friday\}$
2. $\{Monday, Tuesday, Wednesday, Thursday, Friday\}' =$
$= \{Saturday, Sunday\}$.

Venn diagrams for complement



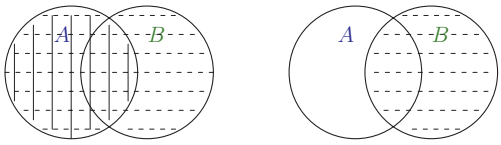
$\{John, Mary, Paul, Eve\}' = \{Sarah, Chris\}$

Applying Venn diagrams

Using Venn diagrams one can discover general properties of sets and operations on sets.

Our first goal is to check whether $A \cap (B \cup A) \stackrel{?}{=} B$.

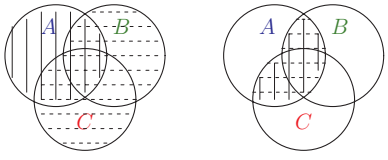
$A \cap (B \cup A)$ $\qquad$ B



Answer: NO! But one can discover that $A \cap (B \cup A) = A$.

Let us now check whether $A \cap (B \cup C) \stackrel{?}{=} (A \cap B) \cup (A \cap C)$.

$A \cap (B \cup C)$ $\qquad$ $(A \cap B) \cup (A \cap C)$



Answer: YES!

Set equalities

- *Idempotent Laws*:
  - $A \cup A = A$
  - $A \cap A = A$
- *Identity Laws*:
  - $A \cup \emptyset = A$
  - $A \cap \emptyset = \emptyset$
- *Commutative Laws*:
  - $A \cup B = B \cup A$
  - $A \cap B = B \cap A$
- *Complement Laws*:
  - $(A')' = A$
  - $A \cup A' = U$,
    where U is the universe of considered objects
  - $A \cap A' = \emptyset$
  - $A\backslash B = A \cap B'$
- *DeMorgan's Laws*:
  - $(A \cup B)' = A' \cap B'$
  - $(A \cap B)' = A' \cup B'$

Set equalities

- *Associative Laws*:
  - $A \cup (B \cup C) = (A \cup B) \cup C$
  - $A \cap (B \cap C) = (A \cap B) \cap C$
- *Distributive Laws*:
  - $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  - $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Consistency Principles

- $A \subseteq B$ iff $A \cup B = B$
- $A \subseteq B$ iff $A \cap B = A$
- $\emptyset \subseteq A$
- $A \subseteq U$, where U is the universe of considered objects.

Example

Simplify the expression $(A \cup B) \cap B'$.

$$\begin{array}{rcll} (A \cup B) \cap B' & = & (A \cap B') \cup (B \cap B') & \text{(by Distributive Law)} \\ & = & (A \cap B') \cup \emptyset & \text{(by Complement Law)} \\ & = & (A \cap B') & \text{(by Identity Law)} \\ & = & A \backslash B & \text{(by Complement Law).} \end{array}$$

Cardinality of sets

Cardinality of a finite set A, denoted by $|A|$, is the number of elements in the set. Warning: cardinality of infinite sets is a much more complicated notion and will not be discussed during these lectures.

Examples

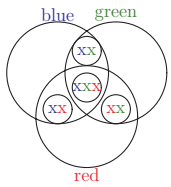
1. $|\{John, Mary, Paul\}| = 3$
2. the cardinality of the set of weekdays is 7.

Example: intersection, union and cardinality of sets

Consider balls colored by two or three colors from the set of colors blue, green, red, where exactly one ball has all three colors, and:

- exactly two balls have colors green and blue
- exactly two balls have colors green and red
- exactly two balls have colors blue and red.

What is the cardinality of the set of balls described above?



Cartesian product of sets

By the *Cartesian product of sets* (introduced by Descartes) A and B, denoted by $A \times B$, we understand the set of all pairs $\langle a, b \rangle$ such that $a \in A$ and $b \in B$.

Examples

1. $\{John, Eve\} \times \{Smith, Wolf\} =$
   $\{\langle John, Smith\rangle, \langle John, Wolf\rangle, \langle Eve, Smith\rangle, \langle Eve, Wolf\rangle\}$
2. $\{1, 2, 3\} \times \{a, b\} = \{\langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle, \langle 2, b\rangle, \langle 3, a\rangle, \langle 3, b\rangle\}$.

By the *Cartesian product of sets* $A_1, \ldots, A_k$, denoted by $A_1 \times \ldots \times A_k$, we understand the set of all *k-tuples* $\langle a_1, \ldots, a_k \rangle$ such that $a_1 \in A_1, \ldots, a_k \in A_k$.

The powerset of a set

By the *powerset of set* A, denoted by 2^A, we understand the set of all subsets of the set A.

Examples

1. $2^{\{John, Eve\}} = \Big\{\emptyset, \{John\}, \{Eve\}, \{John, Eve\}\Big\}$
2. $2^{\{1,2,3\}} = \Big\{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\Big\}$.

What you should have learnt from Lecture I?

- what is a set?
- what is a member of a set?
- what are notations for sets?
- when sets are equal?
- when a set is a subset of another set?
- what is the relationship between notion of subsets and inclusion?
- what is set intersection, union, complement?
- what are and how to use Venn diagrams?
- what is the cardinality of a set?
- what is the Cartesian product?
- what is the powerset of a given set?

Relations

Intuitively: a *relation* is a formal way to express *relationships* between objects.

Examples

1. consider a set of persons; we might isolate many relationships, including:
   - "a person x is a parent of a person y"
   - "a person x is a friend of a person y"
   - "persons x and y look similar to each other"
2. consider a set of cars on a road; one might find useful the following relationships:
   - "a car x is behind a car y"
   - "a car x is close to a car y"
   - "a car x is more comfortable than a car y"
3. consider persons and cars; one often deals with the following relations:
   - "a person x is inside of a car y"
   - "a person x drives a car y"
   - "a person x likes a car y".
4. to illustrate one-argument relations consider a set of animals, and:
   - "an animal x is a dog"
   - "an animal x is a mammal"
   - "an animal x is a fish".

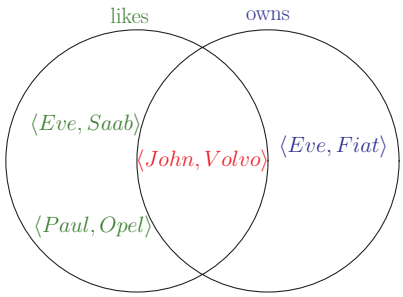
Notation for relations

Tuple notation: relations are denoted by listing tuples which are in the relation, i.e., $name = \{ list\ of\ tuples \}$.

Examples

1. $parent = \{\langle John, Mary\rangle, \langle Eve, Mary\rangle, \langle John, Paul\rangle\}$
2. $likes = \{\langle John, Volvo\rangle, \langle Eve, Saab\rangle, \langle Paul, Opel\rangle\}$
3. $owns = \{\langle John, Volvo\rangle, \langle Eve, Fiata\rangle\}$.

Important: any relation is a set of tuples. Thus *relation* is defined as any subset of a Cartesian product of some sets.



Prefix notation: relations are denoted by $name(object_1, \ldots, object_n)$.
The number of arguments, n, is called the *arity* of relation *name*. One-argument relations are called *unary*, two-argument - *binary* and n-argument - *n-ary* relations.

Examples

Relationship	Notation
"a person x is a parent of a person y"	$parent(x, y)$
"a person x is a friend of a person y"	$friend(x, y)$
"persons x and y look similar to each other"	$similar(x, y)$
"a car x is behind a car y"	$behind(x, y)$
"a car x is close to a car y"	$close(x, y)$
"a car x is more comfortable than a car y"	$moreComf(x, y)$
"a person x is inside of a car y"	$inside(x, y)$
"a person x drives a car y"	$drives(x, y)$
"a person x likes a car y"	$likes(x, y)$

Infix notation: in the case of two-argument relations we often write $x\ name\ y$ rather than $name(x, y)$.

Examples

1. $x \leq 10$ rather than "$\leq (x, 10)$"
2. $x = 100$ rather than "$= (x, 100)$".

Set-theoretical operations on relations

As observed, relations are sets of tuples of a fixed arity. In order to make sure that the union of relations is a relation we assume that relations subject to union are of the same arity.

Example

Consider two relations,

$$cars_1 = \{\langle Volvo, red\rangle, \langle Fiat, green\rangle\}$$
$$cars_2 = \{\langle Opel, blue\rangle, \langle Volvo, red\rangle\}.$$

Then:

$$cars_1 \cup cars_2 = \{\langle Volvo, red\rangle, \langle Fiat, green\rangle, \langle Opel, blue\rangle\}$$

and

$$cars_1 \cap cars_2 = \{\langle Volvo, red\rangle\}.$$

Complement of a relation is, as in the case of sets, relative to a universe of all considered tuples of a given arity.

Example

If the considered universe is

$$\{\langle Volvo, red\rangle, \langle Fiat, green\rangle, \langle Opel, blue\rangle\}$$

then $\{\langle Fiat, green\rangle\}' = \{\langle Volvo, red\rangle, \langle Opel, blue\rangle\}$.

Tabular representation of relations

Relations are often represented in a form of tables, where columns contain values of some *attributes* and rows contain tuples describing particular objects. This representation is basic in *relational databases*.

Example

The following table represents an exemplary relation.

Name	Age	Weight	Town
Eve	26	56	Linköping
John	12	42	Kisa
Sarah	9	24	Linköping
Paul	19	71	Norrköping

More precisely, the relation is over

$$Names \times Numbers \times Numbers \times Towns,$$

and is equal to:

$$\{\ \langle Eve, 26, 56, Linköping\rangle,\ \langle John, 12, 42, Kisa\rangle,\ \langle Sarah, 9, 24, Linköping\rangle,\ \langle Paul, 19, 71, Norrköping\rangle\ \}$$

Selection from relations

In many applications, in particular databases, one is interested in selecting, from given relations, tuples satisfying some condition. Below we consider SQL-like selection operation (SQL is a standard query language in relational databases).

Consider fixed relations $R_1(x_1, \ldots, x_k), \ldots, R_n(y_1, \ldots, y_m)$.
Selection, denoted by
SELECT $z_1, \ldots, z_r$ FROM $R_1, \ldots, R_n$ WHERE C,
where C is a condition and variables $z_1, \ldots, z_r$ are chosen from variables $x_1, \ldots, x_k, \ldots, y_1, \ldots, y_m$,
is defined to be the relation consisting of tuples

$$\{\langle z_1, \ldots, z_r\rangle \mid R_1(x_1, \ldots, x_k), \ldots, R_n(y_1, \ldots, y_m) \text{ and condition } C \text{ holds}\}$$

Observe that selection results in a new relation.

In terms of tabular representation, selection allows us to select pieces of information from one or more tables, satisfying certain conditions.
The obtained result is itself organized as a table with columns labelled by $z_1, \ldots, z_r$ and rows filled by items taken from tables $R_1, \ldots, R_n$, respectively.

Example

Consider relations:

- ▷ $person(name, age, profession)$, consisting of tuples $\{\langle John, 22, driver\rangle, \langle Eve, 20, student\rangle, \langle Paul, 27, teacher\rangle\}$
- ▷ $phone(name, place, number)$, consisting of tuples $\{\langle John, Kisa, 22\,33\,44\rangle, \langle Eve, Ljungby, 33\,44\,55\rangle\}$
- ▷ $likes(name, item)$, consisting of tuples $\{\langle John, books\rangle, \langle Eve, films\rangle, \langle Paul, films\rangle, \langle Sarah, books\rangle\}$.

Now:

1. SELECT $name, profession$ FROM $person$
   WHERE $age \leq 22$
   results in $\{\langle John, driver\rangle, \langle Eve, student\rangle\}$
2. SELECT $name, number$ FROM $phone$
   WHERE $place = Kisa$
   results in $\{\langle John, 22\,33\,44\rangle\}$
3. SELECT $name, place, profession$ FROM $person, phone$
   WHERE $age \leq 22$
   results in $\{\langle John, Kisa, driver\rangle, \langle Eve, Ljungby, student\rangle\}$
4. SELECT $age, profession$ FROM $person, likes$
   WHERE $item = films$
   results in $\{\langle 20, student\rangle, \langle 27, teacher\rangle\}$

Visualizing binary relations

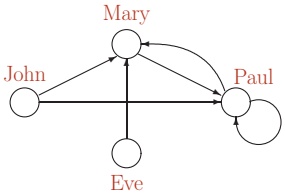
To visualize a binary relation one often uses *graphs*, where *nodes*, depicted as circles, represent objects, and *edges*, depicted as arrows, represent relationships.

Example

Consider relation *likes* such that:

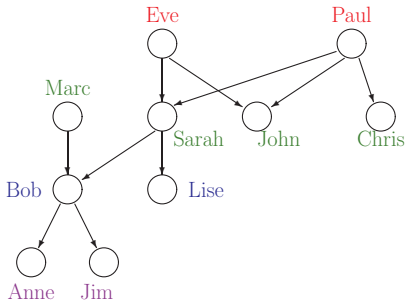
John likes Mary,
John likes Paul,
Mary likes Paul,
Eve likes Mary,
Paul likes Mary,
Paul likes Paul.

The corresponding graph:



Example

Consider parent-child graph (family tree):



It represents relation containing pairs:

Tuple notation	Prefix notation	Infix notation
$\langle Eve, Sarah\rangle$	$parent(Eve, Sarah)$	$Eve\ parent\ Sarah$
$\langle Eve, John\rangle$	$parent(Eve, John)$	$Eve\ parent\ John$
$\langle Paul, Sarah\rangle$	$parent(Paul, Sarah)$	$Paul\ parent\ Sarah$
$\langle Paul, John\rangle$	$parent(Paul, John)$	$Paul\ parent\ John$
$\langle Paul, Chris\rangle$	$parent(Paul, Chris)$	$Paul\ parent\ Chris$
$\langle Marc, Bob\rangle$	$parent(Marc, Bob)$	$Marc\ parent\ Bob$
$\langle Sarah, Bob\rangle$	$parent(Sarah, Bob)$	$Sarah\ parent\ Bob$
$\langle Sarah, Lise\rangle$	$parent(Sarah, Lise)$	$Sarah\ parent\ Lise$
$\langle Bob, Anne\rangle$	$parent(Bob, Anne)$	$Bob\ parent\ Anne$
$\langle Bob, Jim\rangle$	$parent(Bob, Jim)$	$Bob\ parent\ Jim$

Properties of binary relations

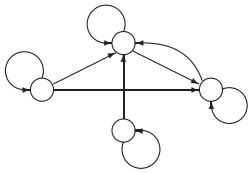
Reflexivity: every object is in relation with itself.

Examples

1. relation "person x is in the same age as person y" is reflexive
2. relation "person x is a brother of person y" is not reflexive, since nobody is his/her brother
3. relation "person x is a parent of person y" is not reflexive, since nobody is his/her own parent
4. is relation "object x is similar to object y" reflexive?

Graphs representing reflexive relations contain one-node cycles (i.e., cycles connecting every element with itself).

Example



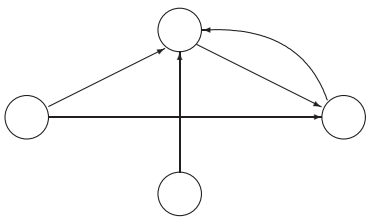
Irreflexivity: no object is in relation with itself.

Examples

1. relation "person x is in the same age as person y" is not irreflexive
2. relation "person x is a brother of person y" is irreflexive
3. relation "person x is a parent of person y" is irreflexive
4. is relation "object x is similar to object y" irreflexive?

Graphs representing irreflexive relations do not contain any one-node cycles.

Example



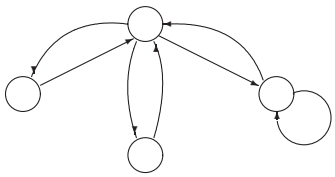
Symmetry: for every two objects, if the first one is in relation with the second then the second is in the relation with the first.

Examples

1. relation "person x is in the same age as person y" is symmetric
2. relation "person x is a brother of person y" is not symmetric (why?)
3. relation "person x is a parent of person y" is not symmetric
4. is relation "object x is similar to object y" symmetric?

Graphs representing symmetric relations contain "back" arrows for every arrow.

Example



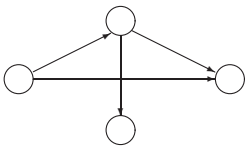
Anti-symmetry: for every two objects, distinct or not, if the first one is in relation with the second then the second is not in the relation with the first.

Examples

1. relation "person x is in the same age as person y" is not anti-symmetric
2. relation "person x is a brother of person y" is not anti-symmetric (why?)
3. relation "person x is a parent of person y" is anti-symmetric
4. is relation "object x is similar to object y" anti-symmetric?

Graphs representing anti-symmetric relations never contain "back" arrows (in particular one-node cycles).

Example



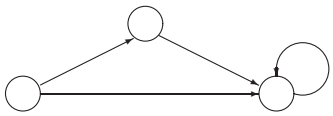
Weak anti-symmetry: for every two distinct objects, if the first one is in relation with the second then the second is not in the relation with the first.

Examples

1. relation "person x is in the same age as person y" is not weakly anti-symmetric
2. relation "person x is a brother of person y" is not weakly anti-symmetric
3. relation "person x is a parent of person y" is weakly anti-symmetric
4. is relation "object x is similar to object y" weakly anti-symmetric?

Graphs representing weakly anti-symmetric relations never contain "back" arrows between distinct elements, but can contain one-node cycles).

Example



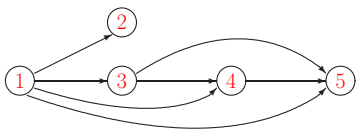
Transitivity: for every three objects, if the first one is in relation with the second and the second is in relation with the third, then the first is in relation with the third.

Examples

1. relation "person x is in the same age as person y" is transitive
2. relation "person x is a brother of person y" is not transitive (why?)
3. relation "person x is a parent of person y" is not transitive
4. is relation "object x is similar to object y" transitive?

Graphs representing transitive relations contain direct arrows between any two nodes among which there is an indirect connection.

Example



This graph represents relation:

$$\Big\{\langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \langle 3,4\rangle, \langle 3,5\rangle, \langle 4,5\rangle\Big\}$$

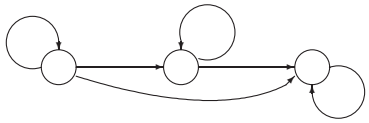
Linearity: reflexivity plus the condition that for any two different objects either the fist one is in the relation with the second or vice versa.

Examples

1. relation "person x is not older than a person y" is linear
2. the usual relation $\leq$ on numbers is linear
3. is relation "object x is similar to object y" linear?

Graphs representing linear relations contain an one-element cycles and, for any two nodes there is an arrow between them.

Example



Strict linearity: the condition that for any two different objects either the fist one is in the relation with the second or vice versa (including irreflexivity).

Important types of relations: orderings

Strict partial order is a binary relation which is irreflexive, anti-symmetric and transitive.

Examples

1. the ancestor relation between persons is a strict partial order on persons
2. exam results strictly partially order the set of students (here we assume that x, y are in the relation if x has a lower grade than y)
3. the usual relation $<$ on natural numbers is a strict partial order
4. inclusion of sets $\subseteq$ is not a strict partial ordering, since it is not irreflexive.

Partial ordering is a binary relation which is reflexive, weakly anti-symmetric and transitive.

Examples

1. the ancestor relation between persons is not a partial ordering on persons
2. the usual relation $\leq$ on natural numbers is a partial order, but is not a strict partial order
3. inclusion of sets $\subseteq$ is a partial ordering.

Linear ordering: a linear partial order.

Examples

1. the usual relation $\leq$ on numbers is a linear order
2. relation between persons "to be not younger than" is a linear order
3. the alphabetical ordering of names "to be not later in a phone book" is a linear order
4. set inclusion is not a linear order.

Strict linear ordering: a strict linear partial order.

Examples

1. the usual relation $<$ on numbers is a strict linear order
2. relation between persons "to be older than" is a strict linear order
3. the alphabetical ordering of names "to be earlier in a phone book" is a strict linear order
4. set inclusion is not a strict linear order.

Important types of relations: similarities

A *similarity* relation is any relation which is reflexive and symmetric.

- Reflexivity is the requirement that any object is similar to itself.
- Symmetry is the requirement that whenever an object, say A is similar to an object B, then B is to be similar to A, too.

Examples

1. closeness between objects is a similarity relation
2. relation $simTemp(t_1, t_2)$, meaning that "temperature t_1 differs from temperature t_2 no more than by 1°C", is a similarity relation (remark: observe that it is not transitive).

Important types of relations: equivalence relations

An *equivalence* relation is any relation which is reflexive, symmetric and transitive.

Examples

1. equality between numbers is an equivalence relation
2. closeness between objects is not an equivalence relation
3. $parL(l_1, l_2)$, meaning that "lines l_1 and l_2 are parallel" is an equivalence relation.

What you should have learnt from Lecture II?

- what is a relation?
- what are notations for relations?
- understanding relations as sets of tuples, understanding set-theoretical operations on relations
- graphical representation of binary relations
- tabular representation of relations
- what is selection?
- properties of relations: reflexivity, symmetry, anti-symmetry, weak anti-symmetry, transitivity
- what is a partial order? what is a linear order?
- what is a similarity relation?
- what is an equivalence relation?

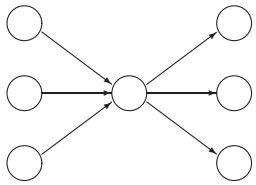
Many-to-many relationships

If many ("many" means "more than one") objects can be in a given relation with many objects then the relation is a *many-to-many* relation.

Examples

1. relation "to be a relative of" is a many-to-many relation.
2. relation "to be a child of" is a many-to-many relation
3. "to be married to" is not a many-to-many relation.

Graphs of many-to-many relations contain nodes with many input and many output arrows.



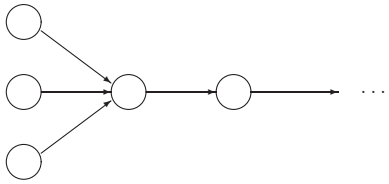
Many-to-one relationships

If many objects can be in a given relation with one object and every object can be in relation with exactly one object then the relation is a *many-to-one* relation.

Examples

1. relation "to be a relative of" is not a many-to-one relation.
2. relation "to be a father of" is a many-to-one relation
3. "to be married to" is not a many-to-one relation.

Graphs of many-to-one relations contain nodes with many input arrows and exactly one output arrow.



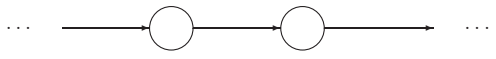
One-to-one relationships

If exactly one object can be in a given relation with exactly one object then the relation is a *one-to-one* relation.

Examples

1. relation "to be a relative of" is not a one-to-one relation.
2. relation "to be a child of" is not a one-to-one relation
3. "to be married to" is a one-to-one relation.

Graphs of one-to-one relations contain nodes with exactly one input and one output arrow.



Important types of relations: functions

Functions are relations which are many-to-one or one-to-one.

Standard notation: $name(expr_1, \ldots, expr_n)$ denotes a *function value*, where ***name*** is a *function name* and $expr_1, \ldots, expr_n$ are *function arguments*.
The number of arguments is called the *arity* of the function.
In order to indicate the universes of arguments and result, we write $name : U_1 \times U_2 \times \ldots \times U_n \longrightarrow U$ meaning that the first argument is to take a value from U_1, the second from U_2, ..., the n-th from U_n and the result if a value from U.
A two-argument function id called *binary* and a one-argument function is called *unary*. For binary functions one often uses infix notation, some functions have traditionally accepted special notations.

Examples

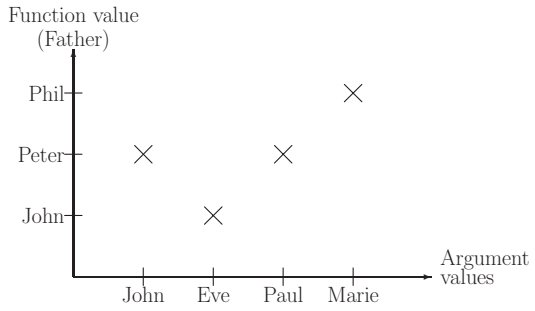
1. Let $mother : Persons \longrightarrow Persons$ have the commonsense meaning. Then $mother(x)$ is a function.
2. $son(x)$ is not a function (why?)
3. $x + y$, $x * y$ (infix notation);
   what notation "$+ : [0,1] \times [2,5] \longrightarrow [2,6]$" means?
4. $\sqrt{x+y}$, $2x^2{+}3x{+}7$, $x^2{+}2xy{+}y^3$ (examples of special notations).

Graphical representation of unary functions: charts

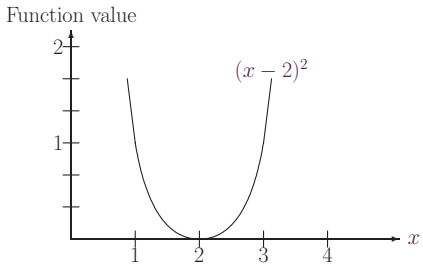
Descartes (as in Cartesian co-ordinates) and Fermat noticed that there is a relationship between a unary function and a curve.
There are two lines in such a chart: horizontal line represents values of arguments and vertical line represents values of functions.

Examples of charts

A discrete case:



A continuous case:



Lambda notation: denotes functions formed from expressions: $\lambda x_1, \ldots, x_k.[expression]$, where $x_1, \ldots, x_k$ are function arguments.

Examples

1. $\lambda x, y.[x^2+2xy+y^3]$ – denotes binary function resulting, for given x, y, in value $x^2 + 2xy + y^3$
2. $\lambda x.[father(father(x))]$ as well as $\lambda x.[father(mother(x))]$ denote "grandfather"
3. $\lambda x.[mother(father(x))]$ as well as $\lambda x.[mother(mother(x))]$ denote "grandmother"
4. $\lambda x.[brother(x, y)]$ returning TRUE when x is a a brother of y and FALSE otherwise, is a correct function
5. $\lambda x.[brother(x)]$ is not correct (why?).

Conditional definitions are of the form:

$$\lambda x_1, \ldots, x_k. \begin{cases} f_1(x_1, \ldots, x_k) & \text{when } x_1, \ldots, x_k \text{ satisfy condition 1} \\ f_2(x_1, \ldots, x_k) & \text{when } x_1, \ldots, x_k \text{ satisfy condition 2} \\ \ldots & \ldots \\ f_r(x_1, \ldots, x_k) & \text{when } x_1, \ldots, x_k \text{ satisfy condition } r \\ 0 & \text{otherwise} \end{cases}$$

Examples

1. Tax calculations are often based on conditional definitions, e.g.,

$$\lambda x. \begin{cases} 0.06 * price(x) & \text{when } x \text{ is a book} \\ 0.12 * price(x) & \text{when } x \text{ is a passenger transport service} \\ \ldots & \ldots \\ 0.25 * price(x) & \text{otherwise,} \end{cases}$$

   defines a (small) part of VAT regulations.

2. The following function defines an income of a person:

$$\lambda x. \begin{cases} salary(x) & \text{when } x \text{ is employed} \\ pension(x) & \text{when } x \text{ retired} \\ scholarship(x) & \text{when } x \text{ is a student} \\ 0 & \text{otherwise.} \end{cases}$$

What is a (discrete) structure?

A *structure* $\langle U, \mathcal{F}, \mathcal{R} \rangle$ consists of a universe U of elements, together with functions $\mathcal{F}$ and relations $\mathcal{R}$ defined on U.

Examples

1. the set of natural numbers with functions of addition and multiplication and the usual relations $=, \leq$
2. the set of real numbers with functions of addition and multiplication and the usual relations $=, \leq$
3. the set of persons with family relationships (like "mother", "father", etc.).

Intuitively: a *discrete structure* is a structure with universe consisting of elements whose "close neighborhoods" do not contain other elements of the considered universe. In a sense, the elements of a discrete structure are separated from each other.

Examples

1. the set of natural numbers is a discrete universe
2. the set of real numbers is not a discrete universe
3. any finite universe is a discrete universe.

Discrete structures: directed graphs

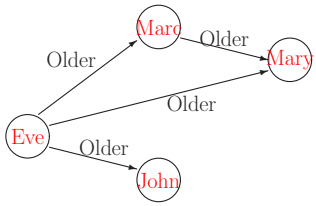
A *directed graph* (*dag*, for short) is a structure $\langle N, E\rangle$, where

- N is a finite set of *nodes*
- $E = \{\langle n, m\rangle \mid n, m \in N\}$ is the set of *directed edges* (sometimes called *arcs*).

Examples

1. $\langle Persons, Likes\rangle$, where $\langle x, y\rangle \in Likes$ if person x likes person y
2. $\langle Persons, Parent\rangle$, where $\langle x, y\rangle \in Parent$ if person x is a parent of person y
3. $\langle Towns, directConnection\rangle$, where $\langle u, v\rangle \in directConnection$ if there is a direct connection from town u to town v

The following figure illustrates graph $\langle Persons, Older\rangle$, where $\langle x, y\rangle \in Older$ if person x is older than person y



Discrete structures: undirected graphs

An *undirected graph* (*graph*, for short) is a structure $\langle N, E\rangle$, where

- N is a finite set of *nodes*
- $E = \{\{n, m\} \mid n, m \in N \text{ and } m \neq n\}$ is the set of *undirected edges* (*edges*, for short).

Undirected graphs illustrate symmetric relationships (i.e, the order of arguments does not matter).

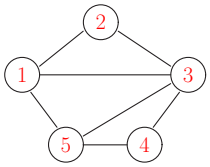
Examples

1. $\langle Persons, Partners\rangle$, where $\{x, y\} \in Partners$ if person x is a partner of person y
2. $\langle Persons, sameProfession\rangle$, where $\{x, y\} \in sameProfession$ if person x has the same profession as person y
3. $\langle Cars, Connected\rangle$, where $\langle u, v\rangle \in Connected$ if there is a connector between cars u and v
4. $\langle Towns, Highway\rangle$, where $\{s, t\} \in Highway$ if there is a highway between towns u and v.

Walks, paths, trails, cycles and circuits

- A *walk* is any sequence of nodes: $n_0 - n_1 - n_2 - \ldots - n_{k-1} - n_k$.
- A *path* is a walk in which a node can appear at most once.
- A *closed path* (a *cycle*) is a walk which starts and ends in the same node and in which all other nodes can appear at most once.
- A *trail* is a walk in which an edge can appear at most once.
- A *closed trail* (a *circuit*) is a walk which starts and ends in the same node and in which all edges can appear at most once.

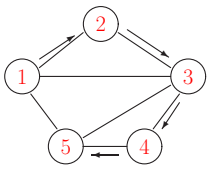
Examples



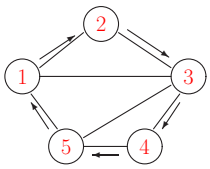
1. 1-2-3-5-3-2 is a walk, but is neither a path nor a trail
2. 1-2-3-5 is both a path and a trail
3. 1-2-3-1-5 is a trail but not a path
4. 1-2-3-5-1 is both a cycle and a circuit.

Euler and Hamiltonian cycles of graphs

A *Hamiltonian path* is a path traversing each graph node exactly once.
A *Hamiltonian cycle* is a closed Hamiltonian path.

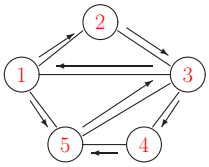


Hamiltonian path: 1-2-3-4-5



Hamiltonian cycle: 1-2-3-4-5-1

A *Eulerian trail* is a trail traversing each graph edge exactly once. A *Eulerian circuit* is a closed Eulerian trail.

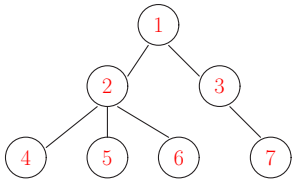


Eulerian circuit 1-2-3-1-5-3-4-5

Discrete structures: trees

A graph is *connected* if there is a walk between any two nodes. A *tree* is an (undirected) connected graph without cycles.

Examples



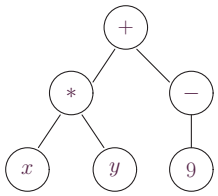
1. a family tree
2. an expression tree
3. a decision tree
4. a company organization chart.

A *binary tree* is a tree in which every node is connected with at most two other nodes. A *rooted tree* is a tree with one distinguished node called the *root* of the tree.

Traversing binary trees

- *In-order*: visit the left sub-tree, then root, then the right sub-tree.
- *Pre-order*: visit root, then the left sub-tree, then the right sub-tree.
- *Post-order*: visit the left sub-tree, then the right sub-tree, then root.

Example



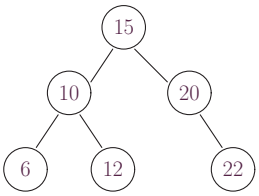
In-order:	$(x * y) + (-9)$	standard notation
Pre-order:	$+ \ * \ x \ y \ - \ 9$	prefix notation
Post-order:	$x \ y \ * \ 9 \ -$	postfix notation

Binary Search Trees (BST)

By a *binary search tree* (BST) we understand a binary tree with nodes containing numbers and satisfying condition that, for any subtree of the tree,

- value in the root is greater than all values in the left subtree
- value in the root is smaller than all values in the right subtree.

Example



Traversing the tree in in-order implements sorting (placing elements in the tree in increasing order). This follows from the observation that, in in-order, we always visit the left subtree, then the root and then the right subtree. Left subtree contains elements smaller than the one in the root and the right subtree contains elements greater than the one in the root.

Discrete structures: words

Let Σ be a finite set of symbols, called an *alphabet*. By a *word* over alphabet Σ we understand any finite sequence of symbols from Σ.

Examples

1. Let $\Sigma_1 = \{a, b, c, \ldots, z, A, B, C, \ldots Z\}$ be an alphabet. Then *cat*, *John* are words over Σ_1. Also $fjKkHt$ is a word over Σ_1.
2. Let $\Sigma_2 = \{0, 12, 3, 4, 5, 6, 7, 8, 9\}$ be an alphabet. Then 123 and 4459 are words over Σ_2, but *John* is not a word over Σ_2 since it contains letters which are not in Σ_2.

Concatenation of words

Let w and v be words over a given alphabet Σ. By the *concatenation* of w and v, denoted by $w \circ v$, we understand word wv.

Examples

1. $John \circ Smith = JohnSmith$.
2. $123 \circ 456 = 123456$.

Do numbers form discrete structures?

- real numbers form a continuous (thus not discrete) line
- natural numbers $0, 1, 2, 3, \ldots$ ordered by $\leq$ form a discrete structure
- all finite sets are discrete. Since a given computer can store only a finite amount of information, all numbers represented in computers form discrete structures (in particular, not all reals or natural numbers can be represented).

Examples

1. Number $\frac{1}{3}$ can be represented accurately on computers as a pair $\langle 1, 3\rangle$, but not using a decimal representation.
2. Some numbers, like $\sqrt{2}$ or π cannot be accurately represented on computers, but can be approximated with arbitrary accuracy.
3. There are real numbers, in particular in interval $[0, 1]$, that cannot even be approximated with arbitrary accuracy using the current model of computers.

Questions

1. is the set of rational numbers discrete?
2. is the set of integers discrete?

Representing numbers: decimal number system

Any natural number can be represented as a word $d_k d_{k-1} \ldots d_1 d_0$ over alphabet $\{0, 1, \ldots, 9\}$. Elements of this alphabet are called *decimal digits*. The value of number $d_k d_{k-1} \ldots d_1 d_0$ is

$$d_k * 10^k + d_{k-1} * 10^{k-1} + \ldots + d_1 * 10^1 + d_0 * \underbrace{10^0}_{=1}.$$

Example

$523 = 5 * 10^2 + 2 * 10^1 + 3 * 10^0$.

Any non-negative real number can be represented as two words $d_k d_{k-1} \ldots d_1 d_0 . f_1 f_2 \ldots$ over alphabet $\{0, 1, \ldots, 0\}$, separated by dot. The first word represents the integer part and the second word - the fractional part of the number. The value of number $d_k d_{k-1} \ldots d_1 d_0 . f_1 f_2 \ldots$ is

$$d_k * 10^k + d_{k-1} * 10^{k-1} + \ldots + d_1 * 10^1 + d_0 * 10^0 + f_1 * 10^{-1} + f_2 * 10^{-2} + \ldots.$$

(Recall that $10^{-p} \stackrel{\text{def}}{=} \frac{1}{10^p}$)

Example

$$523.12 = 5 * 10^2 + 2 * 10^1 + 3 * 10^0 + 1 * 10^{-1} + 2 * 10^{-2} = 5 * 10^2 + 2 * 10^1 + 3 * 10^0 + 1 * \frac{1}{10} + 2 * \frac{1}{100}$$

Binary number system

In *binary representation*, natural numbers are words (called *binary numbers*) over the alphabet $\{0, 1\}$. The *binary digits* $0, 1$ are often called *bits*. The value of number $d_k d_{k-1} \ldots d_1 d_0$ is

$$d_k * 2^k + d_{k-1} * 2^{k-1} + \ldots + d_1 * 2^1 + d_0 * 2^0.$$

Examples

1. binary number 1101 represents $1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$, which in decimal representation is $8 + 4 + 1 = 13$
2. decimal number 22 is represented by binary number 10110 (since $22 = 16 + 4 + 2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0$).

Example in binary arithmetics: addition

Observe that in binary system:

$$0 + 0 = 0,\ 0 + 1 = 1 + 0 = 1,\ 1 + 1 = 10$$

hence, when we add $1 + 1$, we obtain two output digits: a *sum* and a *carry*.

Thus we add binary numbers as follows:

```
    1 1 0 1 0 1 1
+   1 0 0 1 0 0 1
  ----------------
= 1 0 1 1 0 1 0 0
```


What you should have learnt from Lecture III?

- what are one-to-one, many-to-one and many-to-many correspondences?
- what are functions?
- what are notations for functions?
- what are discrete structures?
- what are graphs?
- what are walks, trails, paths, cycles and circuits?
- what are Hamiltonian and Euler circuits
- what are trees and rooted trees?
- traversing trees
- what are binary search trees (BST)?
- decimal and binary numbers
- binary arithmetics.

Definitions

A *definition* is a formal way to introduce new concepts.
A definition has three parts:

1. the *definiendum*: the concept to be defined
2. the *copula*: link between definiendum and definiens
3. the *definiens*: the defining part.

Examples

1. In Aristotle's famous definition "human is a rational animal":
   - "human" is the definiendum
   - "is" is the copula
   - "a rational animal" is the definiens.
2. In the definition of tree ("A tree is an (undirected) connected graph without cycles"):
   - "tree" is the definiendum
   - "is" is the copula
   - "an (undirected) connected graph without cycles" is the definiens
3. In many definitions we shall use the following copulas:
   - symbol "$\stackrel{\text{def}}{=}$" standing for "is by definition equal to", e.g., $f(n) \stackrel{\text{def}}{=} n^2 + 1$
   - symbol "$\stackrel{\text{def}}{\equiv}$" standing for "is by definition equivalent to", e.g., $son(x, y) \stackrel{\text{def}}{\equiv} [child(x, y) \text{ and } male(x)]$.

What are "good definitions"

1. Definitions should not be too broad.
   A definition is *too broad* if the definiens includes more items than it properly should.

   Example: "a dog is an animal having four legs."
   – Wrong since there are also other animals having four legs.
2. Definitions should not be too narrow.
   A definition is *too narrow* if the definiens improperly excludes some items.

   Example: "a thief is a person who steals money."
   – Wrong since thieves steal not only money, but other goods, too.
3. Definitions should not be circular.
   A definition is *circular* if definiens improperly refers to definiendum.

   Example: "a line is a linear path"
4. Definitions should use a precise language, avoiding metaphors and figures.

   For example, "a camel is the ship of the desert" is not a good definition.

All-and-only test

A clear way to check whether the definition is not too broad or too narrow.

The definiens should apply to all possible instances of the definiendum and only to those instances.

Examples

1. To apply this rule to Aristotle's definition of a human, we ask:
   - are all humans rational animals?
   - are only humans rational animals?

   In both cases the answer is "yes', so the test is passed.
2. To apply this rule to the considered definition of dogs we ask:
   - are all dogs animals having four legs? — yes
   - are only dogs animals having four legs?
     — no, thus the definition is too broad
3. To apply this rule to the considered definition of thieves we ask:
   - are all thieves persons stealing money?
     — no, thus the definition is too narrow
   - are only thieves that are persons stealing money? — yes.

Recursion

In *recursive (inductive) definitions* definiendum appears in definiens.

Examples

1. an *ancestor of a person* is

   either a parent of the person
   or is an *ancestor* of a parent of the person
2. a *tree* is

   either a single node
   or is composed of a root node and *trees* attached to this node
3. a *future* is

   either the next observable time moment
   or a *future* of the next observable time moment
4. *factorial*, denoted by $n!$, where n is a natural number, is defined by:

$$n! \stackrel{\text{def}}{=} \begin{cases} 1 & \text{when } n = 0 \text{ or } n = 1 \\ n * (n-1)! & \text{when } n > 1. \end{cases}$$

5. *Fibonacci numbers*, denoted by $F(n)$, where n is a natural number, are defined as follows:

$$F(n) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{when } n = 0 \text{ or } n = 1 \\ F(n-1) + F(n-2) & \text{when } n > 1. \end{cases}$$

How to compute the results?

1. Consider the following definition:

$$c(n) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{when } n = 0 \\ 2 * c(n-1) + 1 & \text{when } n > 0. \end{cases}$$

We can compute values of $c(n)$, for $n = 0, 1, 2, 3, 4 \ldots$ as follows:

$$\begin{aligned} c(0) &\stackrel{\text{def}}{=} 1 \\ c(1) &\stackrel{\text{def}}{=} 2 * c(0) + 1 = 2 * 1 + 1 = 3 \\ c(2) &\stackrel{\text{def}}{=} 2 * c(1) + 1 = 2 * 3 + 1 = 7 \\ c(3) &\stackrel{\text{def}}{=} 2 * c(2) + 1 = 2 * 7 + 1 = 15 \\ c(4) &\stackrel{\text{def}}{=} 2 * c(3) + 1 = 2 * 15 + 1 = 31 \\ &\ldots \end{aligned}$$

2. We can compute values of Fibonacci numbers $F(n)$, for $n = 0, 1, 2, 3, 4 \ldots$ as follows:

$$\begin{aligned} F(0) &\stackrel{\text{def}}{=} 1 \\ F(1) &\stackrel{\text{def}}{=} 1 \\ F(2) &\stackrel{\text{def}}{=} F(1) + F(0) = 1 + 1 = 2 \\ F(3) &\stackrel{\text{def}}{=} F(2) + F(1) = 2 + 1 = 3 \\ F(4) &\stackrel{\text{def}}{=} F(3) + F(2) = 3 + 2 = 5 \\ &\ldots \end{aligned}$$

Observe that calculations start with the "starting clauses" of recursive definitions and then we apply the clauses that allow us to construct new elements on the basis of the previous ones.

WARNING!
There is a risk of circularity when recursive definitions are not properly used!

Examples

Consider the following definitions:

1. $a(n) \stackrel{\text{def}}{=} 1 + a(n)$, where n is a natural number
   — right or wrong? why?
2. $b(n) \stackrel{\text{def}}{=} 2 * b(n+1)$, where n is a natural number
   — right or wrong? why?
3. let $\langle V, E \rangle$ be a graph and define:

$$path(x,y) \stackrel{\text{def}}{\equiv} \langle x, y \rangle \in E \text{ or there is a node } z \in V \text{ such that } \langle x, z \rangle \in E \text{ and } path(z, y)$$

   — right or wrong? why?
4. x is a *human* iff mother and father of x are *humans*, too.
   This definition could be stated more formally as follows:

$$human(x) \stackrel{\text{def}}{\equiv} [human(mother(x)) \text{ and } human(father(x))].$$

   — right or wrong? why?
   — what if the universe of considered persons is finite?

Rules for providing good recursive definitions

A recursive definition of a set (a relation) always consists of three distinct clauses:

1. the *basis* of the definition establishes that certain objects are in the set (satisfy a given relation)
2. the *inductive clause* (or simply *induction*) of the definition establishes the ways in which elements of the set (relation) can be combined to produce new elements of the set (relation)
3. the *closure clause* asserts that unless an object can be shown to be a member of the set (relation) by applying the basis and inductive clauses a finite number of times, the object is not a member of the set (relation). This clause is usually implicit in recursive definitions.

Further examples of recursive definitions

1. Definition of natural numbers:
   - basis: 0 is a natural number
   - inductive clause:
     if n is a natural number
     then $n+1$ is a natural number, too
   - the closure clause: only objects obtained from 0 by adding 1 a finite number of times are natural numbers.

   Thus natural numbers are constructed as follows:

$$0,\ 0+1=1,\ 1+1=2,\ 2+1=3,\ 3+1=4,\ \ldots$$

2. Definition of odd numbers:
   - basis: 1 is an odd number
   - inductive clause:
     if n is an odd number
     then $n+2$ is an odd number, too
   - the closure clause: only objects obtained from 1 by adding 2 a finite number of times are odd numbers.

   Thus odd numbers are constructed as follows:

$$1,\ 1+2=3,\ 3+2=5,\ 5+2=7,\ 7+2=9,\ \ldots$$

3. Definition of crowd:
   - basis: five people are a crowd (but not less than five)
   - inductive clause:
     if there is a crowd and one person joins it,
     then that is also a crowd.
   - what is the meaning of the closure clause here?

   How to construct a crowd?

---

4. Definition of function of cardinality of finite sets:
   - basis: $|\emptyset| = 0$, i.e., 0 is the cardinality of the empty set $\emptyset$
   - inductive clause:
     if $a \notin A$ then $|A \cup \{a\}| = |A| + 1$
   - what is the meaning of the closure clause here?

   Exercise: calculate the cardinality of the set $\{2, 6, 9, 7\}$, using the above definition.

   One of solutions:

$$|\{2\}| = |\emptyset \cup \{2\}| = |\emptyset| + 1 = 0 + 1 = 1$$
$$|\{2, 6\}| = |\{2\} \cup \{6\}| = |\{2\}| + 1 = 1 + 1 = 2$$
$$|\{2, 6, 9\}| = |\{2, 6\} \cup \{9\}| = |\{2, 6\}| + 1 = 2 + 1 = 3$$
$$|\{2, 6, 9, 7\}| = |\{2, 6, 9\} \cup \{7\}| = |\{2, 6, 9\}| + 1 = 3 + 1 = 4.$$

Induction principle

Induction principle (*mathematical induction*):
In order to show that a property holds for a set defined inductively (recursively):

1. *base step*: show that the property holds for all elements defined in the basis of inductive definition
2. *induction step*: show that the property is preserved by all inductive clauses (i.e., if the property holds for simpler elements, then it holds for elements constructed by inductive clauses from the considered simpler elements).

Example

Consider a family whose first known member, say J, was rich. Assume further that in this family the following inheritance holds:
if a person is rich then all children of the person become rich, too.

Let a successor of person x be recursively defined as follows:

- basis: children of x are successors of x
- inductive clause:
  if y is a successor of x then all children of y are successors of x.

Then we can conclude that all successors of J are rich:

1. the base step: J was rich
2. the induction step:
   if x is rich then all children of the person become rich, too.

Induction in natural numbers

Induction in natural numbers reflects the general induction principle stated above and the recursive definition of natural numbers provided in the previous slide.

In order to show that a property is satisfied by all natural numbers, show:

1. base step: the property is satisfied for 0
2. induction step: assuming that the property is satisfied for a given natural number n then it is also satisfied for $n+1$.

Example

Consider sequence:

$$a(n) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{when } n = 0 \\ 2 * a(n-1) & \text{otherwise.} \end{cases}$$

Show that $a(n) = 2^n$.

1. the base step ($n = 0$): $a(0) \stackrel{\text{def}}{=} 1 = 2^0$.
2. the induction step:
   assume $a(n) = 2^n$;
   we want to show that the equality holds for $n + 1$,
   i.e., that $a(n+1) = 2^{n+1}$:

$$a(n+1) \stackrel{\text{def}}{=} 2 * \underbrace{a(n)}_{2^n} = 2 * 2^n = 2^{n+1}.$$

Example

Show that $0 + 1 + 2 + \ldots + n = \frac{n * (n+1)}{2}$.

1. the base step ($n = 0$):
   - lefthand side $= 0$,
   - righthand side $= \frac{0 * (0+1)}{2} = 0$.

   Thus the lefthand side is equal to the righthand side.
2. the induction step:
   assume that $0 + 1 + 2 + \ldots + n = \frac{n * (n+1)}{2}$ holds;
   we have to show that this equality holds for $n + 1$, i.e., that

$$0 + 1 + 2 + \ldots + n + (n+1) = \frac{(n+1) * (n+2)}{2}$$

   holds.
   We start with the lefthand side written for $n + 1$:

$$\underbrace{0 + 1 + 2 + \ldots + n}_{\frac{n*(n+1)}{2}} + (n+1) =$$
$$= \frac{n * (n+1)}{2} + \frac{2 * (n+1)}{2} =$$
$$= \frac{(n+1)}{2} * (n+2) =$$
$$= \frac{(n+1) * (n+2)}{2},$$

   which is indeed what we wanted to show.

Induction in commonsense reasoning and learning

The induction principle considered so far is idealized. It is important, since it allows us to show formally facts hat surely hold in some idealized world.
On the other hand, due to the incompleteness and uncertainty of knowledge, in commonsense reasoning and learning, the term *induction* frequently refers to an inference (called also the *inductive inference*) from repeatedly observed instances of a given property to some of it's unobserved instances.

Examples

1. Kids learn the gravitation law from repeatedly dropping things.
2. From repeated observations we infer that the sun will rise tomorrow or that it raised 10 000 years ago.
3. From the fact that we and people we know have never been poisoned in a restaurant we infer that a dinner served in a restaurant will not poison us.

Let us analyze the second inference from the perspective of the principle of induction. Assume that 0 denotes the first day of Earth and all the next days are numbered using the successive natural numbers.

- the base step: sun raised in the beginning of day 0
- the induction step: if sun raises in a given day, say n then it will raise in day $n+1$.

Are the above steps valid?

The case of finite universes

In the case of finite universes one often computes objects satisfying a given recursive definition applying the following algorithm, where R denotes the set of objects constructed by the considered recursive definition:

1. initially assume that R is empty (i.e., $R = \emptyset$)
2. while R changes, add to R all objects that can be constructed from objects in R, using the recursive definition.

Intuitively:

1. initially we make no assumptions whether objects are or are not in the defined set R
2. in the first step we add to R elements defined in the basis of the considered recursive definition
3. in the second step we add to R elements constructed from elements added in the first step, according to the inductive clause
4. in the third step we add to R elements constructed from elements added in the first and the second step, according to the inductive clause
5. ...
6. in the k-th step we add to R elements constructed from elements added in steps $1, 2, \ldots, k-1$, according to the inductive clause
7. ... — a finite number of steps — why?

Example: "humans" example revisited

Recall the definition of humans considered previously:

$$human(x) \stackrel{\text{def}}{\equiv} [human(mother(x)) \text{ and } human(father(x))].$$

Assume that we additionally know that:

$$human(John),\ human(Eve), human(Marc)$$
$$father(Mary) = father(Jack) = John,$$
$$mother(Mary) = mother(Jack) = Eve,$$
$$mother(Joe) = Mary,\ father(Joe) = Marc$$
$$mother(Jim) = Eve.$$

In order to construct the set of humans we start with the empty set $\emptyset$.

The next iterations of the algorithm result in sets:

$$\{John, Eve, Marc\}$$
$$\{John, Eve, Marc\} \cup \{Mary, Jack\} =$$
$$= \{John, Eve, Marc, Mary, Jack\}$$
$$\{John, Eve, Marc, Mary, Jack\} \cup \{Joe\} =$$
$$= \{John, Eve, Marc, Mary, Jack, Joe\}$$

What you should have learnt from Lecture IV?

- what is a definition? what are parts of definitions?
- how to construct good definitions?
- what is a recursive (inductive) definition?
- how to construct recursive definitions?
- induction principle, induction in natural numbers,
- understanding inductive inference in learning and commonsense reasoning
- how to compute sets defined by recursive definitions?

Formal languages

Let Σ be a finite alphabet. Recall that words have been defined as finite sequences of alphabet symbols.
By a *formal language* (*language*, for short) over Σ we understand any set of words over Σ. Language words are also called *phrases*.

Examples

1. Consider the alphabet $\Sigma = \{a, b, c \ldots, z, A, B, C, \ldots, Z, \_\}$ (symbol _ stands for space). We can define *English* to be the set of all correct sentences of the English language.

   Observe that "sentence" is a finite sequence of symbols of our alphabet, i.e., corresponds to what we called "phrase". Examples of phrases of *English* (representing sentences of English) are the following:

   - Jack_reads_many_books
   - What_time_is_it

   Observe that we cannot use some symbols, like question mark, since these are not in the alphabet

2. Consider the alphabet $\Sigma = \{0, 1, 2, \ldots, 9\}$. We define *NatNb*, to be the language of all words over Σ.

   NatNb consists of words representing natural numbers using decimal digits.

Production schemas

Definitions of languages we gave before are very informal and cannot really be used in practice.

In order to define formal languages one usually uses production schemas. In the style presented here they have been introduced by Thue.

By a *production schema* (or simply, *production*) over a set of symbols *Sym* we understand any expression of the form
$w \longrightarrow u$,
where w and u are words over *Sym*.
The meaning of $w \longrightarrow u$ is "w can be replaced by u".

Examples

Let *Sym* be the set of letters.

1. $abc \longrightarrow aac$ is a production
2. $animal \longrightarrow cat$, $animal \longrightarrow dog$ is a production
3. $a\#b \longrightarrow aac$ is not a production, since $\# \notin Sym$.

Production $abc \longrightarrow aac$ means that abc can be replaced by aac.

Having this production defined we can replace word $dabceee$ by word $daaceee$.

Similarly, having the production from the second example, $animal\ runs$ can be replaced by $cat\ runs$ as well as by $dog\ runs$.

Derivations

Given a set of productions P, we say that a word u can be *directly derived* from a word w, and denote this by $w \Rightarrow u$, if $w = w_1 w_2 w_3$, $u = w_1 w_2' w_3$ and there is a production $w_2 \longrightarrow w_2'$ in P.
The word u can be *derived* from w, which is denoted by $w \stackrel{*}{\Rightarrow} u$, if there is a sequence of words $s_0, s_1, s_2, \ldots, s_k$, satisfying conditions:

- $s_0 = w$
- for all $1 \leq i \leq k$ we have that $s_{i-1} \Rightarrow s_i$
- $s_k = u$.

Example

Consider a production $aa \longrightarrow a$. Then we can have the following derivation:

$$repeaaaat \Rightarrow repeaaat \Rightarrow repeaat \Rightarrow repeat$$

i.e.,

$$repeaaaat \stackrel{*}{\Rightarrow} repeat.$$

Formal grammars

Formal grammars are used to define well formed expressions of formal languages.

A *formal grammar* (or simply, *grammar*) we understand any tuple $\langle \Sigma, \Delta, S, R \rangle$, where:

- Σ is an alphabet;
  (elements of Σ are also called *terminal symbols*)
- Δ is a finite set of *non-terminal symbols*
- $S \in \Delta$ is the *initial symbol*
- R is the set of productions over $\Sigma \cup \Delta$, called the *grammar rules*.

The meaning of the listed items is:

- an alphabet consists of symbols that are allowed to appear as parts of words of a given language
- non-terminal symbols are auxiliary symbols which are used only in constructing phrases; they are usually used to reflect some category of phrases
- the initial symbol is the one from which the derivation of well-formed words starts
- productions allow to produce well-formed words of a given language starting from the initial symbol.

What language is specified by a grammar?

We say that language L is *generated* (or *specified*) by grammar G iff L consists of all and only words that can be derived from the initial symbol of G by applying productions of G.

Example

Consider the grammar:

- alphabet $\{a, b\}$
- non-terminal symbols $\{D, N\}$
- initial symbol N
- rules (productions):

$$D \to a$$
$$D \to b$$
$$D \to a\,D$$
$$D \to b\,D$$
$$N \to a\,D$$

The non-terminal symbol D generates all words consisting of letters a and b. Thus the language generated by this grammar consists of all words over alphabet $\{a, b\}$ beginning with letter a.

Examples: grammars and languages

1. What language is generated by the grammar:
   - alphabet $\Sigma_1 = \{0, 1, 2, \ldots, 9\}$
   - non-terminal symbols $\Delta_1 = \{D, N\}$
   - initial symbol N
   - rules (productions):

$$D \to 0$$
$$D \to 1$$
$$\ldots$$
$$D \to 9$$
$$N \to D$$
$$N \to DN$$

2. What language is generated by the grammar:
   - alphabet $\Sigma_2 = \Sigma_1 \cup \{-, .\}$
   - non-terminal symbols $\Delta_2 = \Delta_1 \cup \{S\}$
   - initial symbol S
   - rules (productions) – rules of the previous example plus:

$$S \to N$$
$$S \to N.N$$
$$S \to -S$$

Backus-Naur notation for formal grammars

In *Backus-Naur* notation (BNF):

- angle brackets $\langle\rangle$ are used to surround non-terminal symbols
- symbol ::= denotes "$\to$" used in Thue-like style
- symbol | denotes "or".

Example

In the following rules:

$$\langle \text{S} \rangle ::= \langle \text{N} \rangle \mid \langle \text{N} \rangle.\langle \text{N} \rangle$$
$$\langle \text{S} \rangle ::= -\langle \text{S} \rangle$$

we know that that:

- S and N are non-terminal symbols
- . and $-$ are terminal symbols.

Thus the alphabet and nonterminal symbols do not have to be specified separately. One specifies productions and the initial symbol only.

Examples of formal grammars specified in BNF

1. Let $\langle \text{E} \rangle$ be the initial symbol, and let $\langle \text{S} \rangle$ be the non-terminal symbol considered previously and specifying real numbers.

   The following productions:

$$\langle \text{E} \rangle ::= \langle \text{S} \rangle$$
$$\langle \text{E} \rangle ::= \langle \text{E} \rangle + \langle \text{E} \rangle \mid \langle \text{E} \rangle - \langle \text{E} \rangle \mid \langle \text{E} \rangle * \langle \text{E} \rangle \mid \langle \text{E} \rangle / \langle \text{E} \rangle$$
$$\langle \text{E} \rangle ::= -\langle \text{E} \rangle \mid (\langle \text{E} \rangle)$$

   specify arithmetical expressions over reals.

2. Consider the following productions:

$$\langle \text{letter} \rangle ::= \langle \text{lowercase letter} \rangle \mid \langle \text{uppercase letter} \rangle$$
$$\langle \text{lowercase letter} \rangle ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z$$
$$\langle \text{uppercase letter} \rangle ::= A|B|C|D|E|F|G|H|I|J|K|L|M| N|O|P|Q|R|S|T|U|V|W|X|Y|Z$$
$$\langle \text{lowercase word} \rangle ::= \langle \text{lowercase letter} \rangle \mid \langle \text{lowercase letter} \rangle \langle \text{lowercase word} \rangle$$
$$\langle \text{name} \rangle ::= \langle \text{uppercase letter} \rangle \langle \text{lowercase word} \rangle$$

   What language is generated by the grammar consisting of the above productions assuming that $\langle \text{name} \rangle$ is the initial symbol?

Example of a formal grammar for a very small fragment of English

Consider the following productions:

⟨Noun⟩ ::= *house* | *cinema* | *restaurant* | *office*
⟨Name⟩ ::= *John* | *Mary* | *Jack*
⟨Verb⟩ ::= *is* | *works*
⟨Preposition⟩ ::= *in*
⟨Article⟩ ::= *the* | *a*
⟨Adjective⟩ ::= *nice* | *large* | *small*
⟨Sentence⟩ ::=
⟨Name⟩⟨Verb⟩⟨Preposition⟩⟨Article⟩⟨Adjective⟩⟨Noun⟩.

Assume that ⟨Sentence⟩ is the initial symbol.

Then we can derive, e.g., the following phrases:

John works in a large cinema.
John is in a small house.
Mary is in the nice restaurant.

One cannot derive the following phrases:

John is Mary
John is in a cinema.
Mary is in a big nice restaurant.
Cats are mammals.

Chomsky hierarchy

Chomsky hierarchy consists of the following classes of languages:

1. *regular languages*, where productions are of one of the forms (but the second and the third form cannot both appear in the same grammar):

   ⟨non-terminal⟩ ::= ⟨terminal⟩
   ⟨non-terminal⟩ ::= ⟨non-terminal⟩′⟨terminal⟩
   ⟨non-terminal⟩ ::= ⟨terminal⟩⟨non-terminal⟩′

2. *context free languages*, where productions are of the form:

   ⟨non-terminal⟩ ::= ⟨Tn-T⟩

   where ⟨Tn-T⟩ denotes any non-empty sequence of terminals and/or non-terminals

3. *context sensitive languages*, where productions are of the form:

   ⟨Tn-T⟩′⟨non-terminal⟩⟨Tn-T⟩″ ::=
   ⟨Tn-T⟩′⟨Tn-T⟩⟨Tn-T⟩″

   where ⟨Tn-T⟩ is non-empty

4. *type 0 languages*, where productions are of the form

   ⟨Tn-T⟩ ::= ⟨Tn-T⟩′

   where ⟨Tn-T⟩ is non-empty.

Examples of regular languages

1. Language consisting of words of the form $1\ldots10\ldots0$ (block of 1s followed by a block of 0s):

   ⟨block⟩ ::= 1 ⟨block⟩ | 1 ⟨block of 0s⟩
   ⟨block of 0s⟩ ::= 0 | 0 ⟨block of 0s⟩

   How to construct block of 1s followed by a block of 0s and then followed by a block of 1s?

2. binary numbers:

   ⟨binary number⟩ ::= 0 | 1 |
   0 ⟨binary number⟩ |
   1 ⟨binary number⟩
   ⟨signed number⟩ ::= +⟨binary number⟩ |
   −⟨binary number⟩

3. How to specify the language consisting of words over $\{0,1\}$ of an odd length?

4. How to specify the language consisting of words over $\{0,1\}$ containing sequence 0110?

Examples of context free languages

1. Every regular language is a context-free language.

2. The language consisting of words of the form

   $\underbrace{1\ldots1}_{k \text{ times}}\ \underbrace{0\ldots0}_{k \text{ times}}$

   (both blocks contain the same number of digits):

   ⟨A⟩ ::= 10 | 1 ⟨A⟩ 0

   One can prove that this language is not regular.

3. The language of palindromes over alphabet $\{a,b,s,w\}$ (palindromes are words which read the same forwards as backwards, e.g., *www*, *sas*, *abba*):

   ⟨P⟩ ::= a | b | s | w
   ⟨P⟩ ::= a ⟨P⟩ a | b ⟨P⟩ b | s ⟨P⟩ s | w ⟨P⟩ w

4. The language of words over alphabet $\{a,b\}$, containing twice as many b's as a's:

   ⟨P⟩ ::= abb | bab | bba
   ⟨P⟩ ::= ⟨P⟩ abb | a ⟨P⟩ bb | ab ⟨P⟩b | abb ⟨P⟩
   ⟨P⟩ ::= ⟨P⟩ bab | b ⟨P⟩ ab | ba ⟨P⟩b | bab ⟨P⟩
   ⟨P⟩ ::= ⟨P⟩ bba | b ⟨P⟩ ba | bb ⟨P⟩a | bba ⟨P⟩

Examples of context sensitive languages

1. Every context free language is a context sensitive language (just taking the empty context).
2. The language $\{a^i b^j c^k | 1 \leq i \leq j \leq k\}$, where a^i denotes a repeated i-times etc., is not context-free,

   There exist context sensitive grammars that generate this language, e.g.:

   $\langle S\rangle \ ::= \ a\langle S'\rangle b\langle X\rangle \mid ab\langle X\rangle$
   $\langle S'\rangle \ ::= \ a\langle S'\rangle b\langle C\rangle \mid \langle S'\rangle b\langle C\rangle \mid b\langle C\rangle \mid \langle C\rangle$
   $\langle C\rangle b \ ::= \ b\langle C\rangle$
   $\langle C\rangle X \ ::= \ \langle X\rangle c$
   $\langle X\rangle \ ::= \ c$

   Convince yourselves that the above grammar works.

   Hints:

   - The productions for $\langle S\rangle$ and $\langle S'\rangle$ are first used to fix the values i, j, and k.
   - Non-terminal $\langle X\rangle$ is introduced as a marker situated immediately to the right of all the bs.
   - Occurrences of non-terminal $\langle C\rangle$ are moved to the right past the bs before being converted to cs.
   - Finally, $\langle X\rangle$ is itself converted to c.

Natural language example

As an easy illustration of the use of context sensitive rules in describing natural languages, we specify the phenomenon of subject-verb agreement with respect to number, i.e., singular or plural, as reflected in sentences:

- The child runs.
- The men run.

We introduce the following rules:

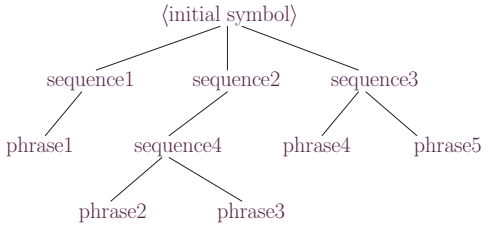
⟨sentence⟩ ::= ⟨noun phrase⟩⟨verb phrase⟩
⟨noun phrase⟩ ::= ⟨determiner⟩⟨noun singular⟩ |
⟨determiner⟩⟨noun plural⟩
⟨noun singular⟩⟨verb phrase⟩ ::=
⟨noun singular⟩⟨verb singular⟩
⟨noun plural⟩⟨verb phrase⟩ ::=
⟨noun plural⟩⟨verb plural⟩
⟨determiner⟩ ::= *the*
⟨noun singular⟩ ::= *man* | *child* | *woman*
⟨noun plural⟩ ::= *men* | *women* | *children*
⟨verb singular⟩ ::= *runs* | *swims* | *laughs*
⟨verb plural⟩ ::= *run* | *swim* | *laugh*

Note that the fourth and fifth rules here are context-sensitive but not context-free. Although it is possible to account for the subject-verb agreement using context free rules only, the two context-sensitive rules capture neatly our intuition that the number of the subject determines that of the verb.

Syntax trees

Let G be a given formal grammar. By a *syntax tree* of a phrase α we understand the rooted tree constructed as follows:

1. the root of the tree consists of the initial symbol of G
2. all final nodes consist of sequences of terminal symbols only; the concatenation of those symbols results in the phrase α
3. all "intermediate" nodes are labelled by sequences of terminal and/or non-terminal symbols of G and are constructed from their immediate predecessors in the tree by applying productions of G.

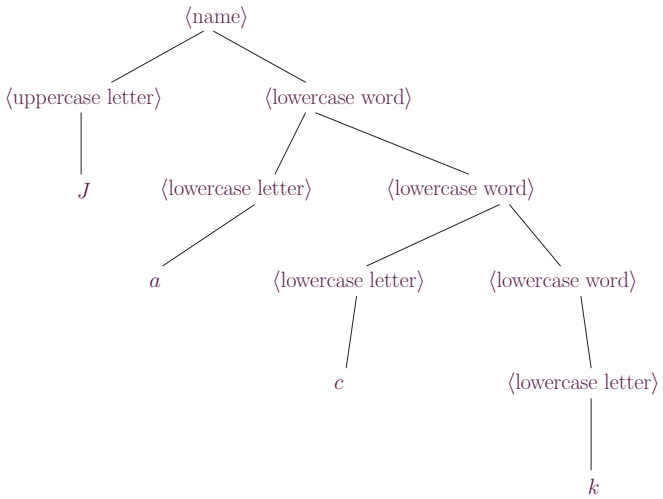


Syntax tree for phrase "phrase1 phrase2 phrase3 phrase4 phrase5".

Example

Consider the second grammar specified in slide 84.

The following tree is the syntax tree for word "Jack".



What you should have learnt from Lecture V?

- what is a formal language?
- what is a production schema?
- what is a formal grammar?
- what are terminal and non-terminal symbols?
- what are grammar rules?
- Backus-Naur notation
- Chomsky hierarchy, in particular:
  - regular languages
  - context-free languages
  - context languages
  - type 0 languages
- syntax trees.

Index