Reference formalisms (T2MP)

Index

- **Language**
- Set theory
- Graphs
- From Sets to Graphs

Chomsky hierarchy

- The Chomsky hierarchy defines a hierarchy of **formal grammars**.
- A formal grammar describes which strings of a given alphabet belong to a given **formal language**.
- We define formal grammar as a set of **production rules** used to rewrite strings by defining a replacement of a particular string (on the left of the rule) with another (on the right).
- The grammar distinguishes between three kinds of symbols in its alphabet: **start, non-terminal, and terminal**.
- The procedure to generate a string in a language starts always by the start symbol (which is also one of the non-terminal).
- The language generated by a grammar is defined by the set of strings with only terminal symbols which are possible to generate using its production rules.

- As notation, we use upper case letters as non-terminals, lower case letters as terminals, and greek letters as strings of terminals and/or non-terminals.
- The letter ϵ is used to represent an empty string.

Type 3 grammars: Regular grammars

- Regular grammars are used to generate regular languages, they are restricted in having production rules only from a singular non-terminal symbol to a single terminal symbol, and from a single non-terminal symbol to a single terminal symbol with another non-terminal symbol on the left or right.
- **Regular expressions** can obtain any language defined by regular grammar.

$$A \to a \quad A \to Ba \quad \text{or} \quad A \to a \quad A \to aB$$

Where A, B are non-terminal, a is terminal.

$A \to \epsilon$ is allowed only if A doesn't appear in the right side of any rule.

Example: Language of all binary numbers

Non-terminal Symbols : $\{A, B\}$
Terminal Symbols : $\{1, 0\}$
Starting Symbol : A
Production rules:

$A \to 1A \mid 1B \mid 1$
$B \to 0B \mid 0A \mid 0$

Type 2 grammars: Context-free grammars

- Context-free grammars allow for more expressivity in the languages that they define than the one defined by regular grammar with the cost of possible ambiguity. Their production rules are restricted to having only one non-terminal symbol on the left, while on the right a string of non-terminal and terminal symbols are allowed.
- Context-free grammars are used to define the syntax of **programming languages**.

$$A \rightarrow \alpha$$

Where A is non-terminal and α is a non-empty string of non-terminal and/or terminal symbols.

Example: Language of all palindrome binary numbers

Non-terminal Symbols : $\{A, B\}$
Terminal Symbols : $\{1, 0\}$
Starting Symbol : A
Production rules:
$A \rightarrow 1A1 \mid 1B1 \mid 11$
$B \rightarrow 0B0 \mid 0A0 \mid 00$

Type 1 grammars: Context-sensitive grammars

- Context-sensitive grammars are used to define context-sensitive languages. The production rules of such grammar allow the specification of the "context" of the non-terminal symbol on the left of the rule, which has to be maintained on the right.

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

Where A is non-terminal and α, β are possibly-empty strings of non-terminal and/or terminal symbols, while γ must be non-empty.

$A \rightarrow \epsilon$ is allowed only if A doesn't appear in the right side of any rule.

Example: Language $L = \{a^n b^n c^n \mid n > 0\}$

Non-terminal Symbols : $\{S, A, B, C, W, Z\}$
Terminal Symbols : $\{a, b\}$
Starting Symbol : S
Production rules:

$S \rightarrow aBC$	$WZ \rightarrow WC$	$bC \rightarrow bc$
$S \rightarrow aSBC$	$WC \rightarrow BC$	$cC \rightarrow cc$
$CB \rightarrow CZ$	$aB \rightarrow ab$	
$CZ \rightarrow WZ$	$bB \rightarrow bb$	

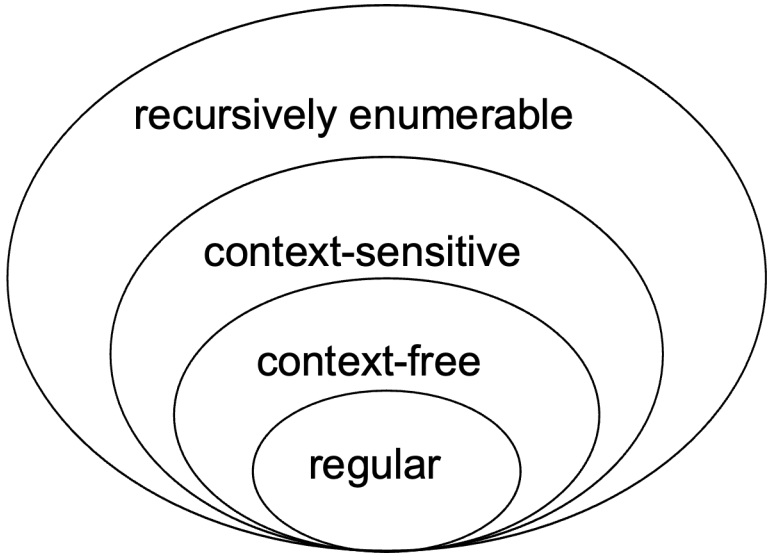
Type 0 grammars: Unrestricted

- Unrestricted grammars, called also recursively enumerable, do not have any restrictions in their production rules, and because of that they can generate any language.

- Therefore, they can produce any **natural language**.

$$\gamma \rightarrow \alpha$$

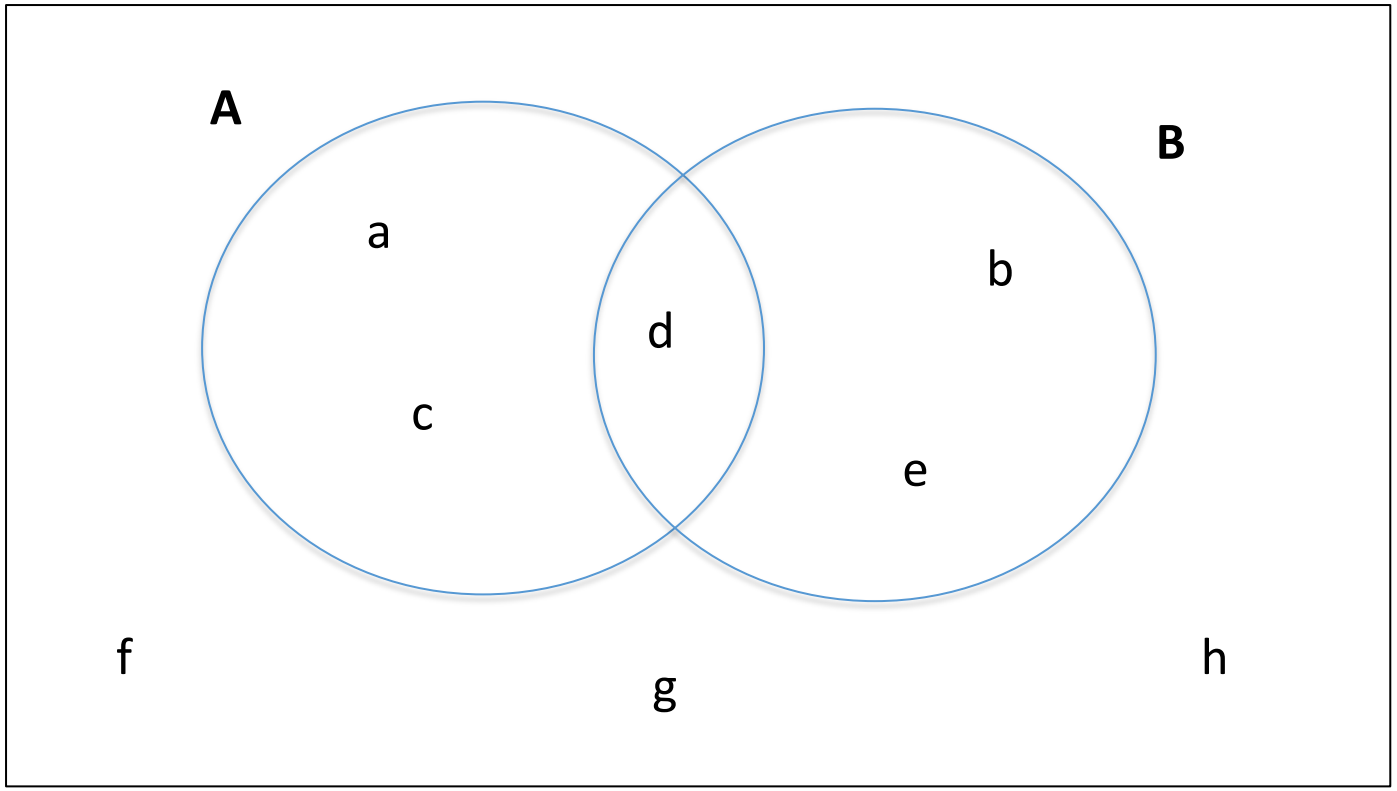
Where γ, α are strings of terminals and/or non-terminal symbols.
Only γ needs to be non-empty.



Index

- Language
- **Set theory**
- Graphs
- From Sets to Graphs

Intuition



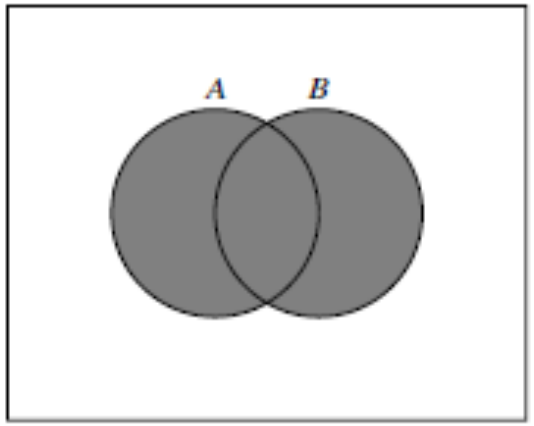
- We represent sets using Venn diagrams; each circle represents a set, and elements inside a circle represent elements that belong to the set it identifies.
- A set can be defined by listing all its elements or by providing their defining property.

$A = \{a, c, d\}$
$B = \{x \mid x\ appears\ in\ the\ word\ 'bed'\ \}$

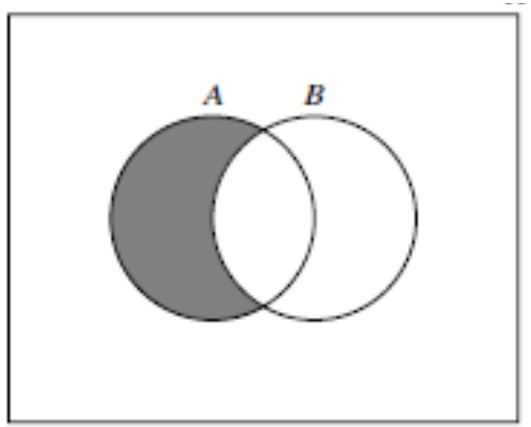
Basic notions

- **Alphabet**: $\{\emptyset, \in, \notin, =, \neq, \subset, \subseteq, \cup, \cap, \setminus, A, B, C, \dots, a, b, c, \dots, U\}$
- We use upper case letters to identify sets and lower case letters to identify elements.
- **Empty Set** $\emptyset$ is the set containing no elements.
- **Membership** $a \in A$ is when element a belongs to the set A.
- **Non-Membership** $a \notin A$ is when element a does not belong to the set A.
- **Equality** $A = B$ is true if and only if sets A and B contain the same elements.
- **Inequality** $A \neq B$ is true if sets A and B do not contain the same elements.
- **Subset** $A \subseteq B$ is when every element of A is also an element B.
- **Proper Subset** $A \subset B$ is when $A \subseteq B$ and $A \neq B$.
- **Universal Set**. The universal set is the set of all elements or members of all related sets and is denoted by the letter U.

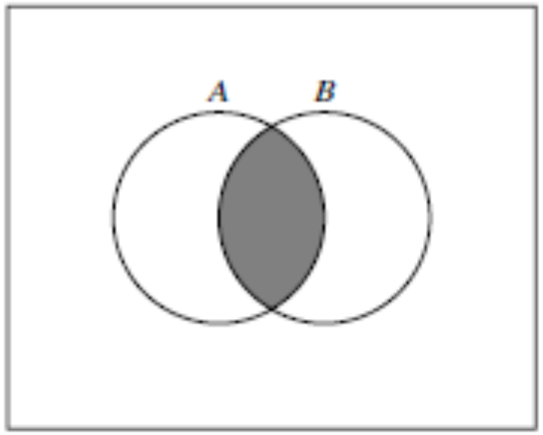
Basic notions



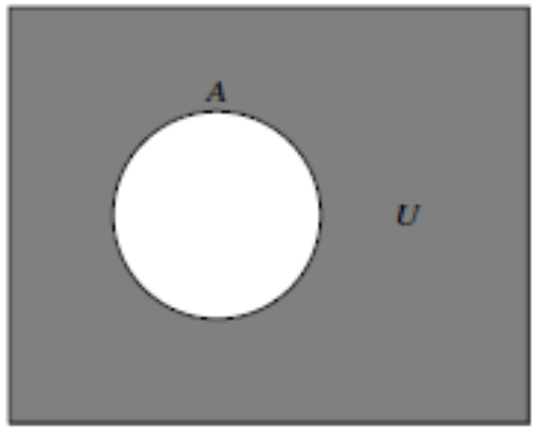
Union. Given two sets A and B, the union of A and B is the set containing the elements belonging to A or to B or to both, and is denoted with $A \cup B$.



Difference. Given two sets A and B, the difference of A and B is the set containing all the elements which are members of A, but not members of B, and is denoted with $A \setminus B$.



Intersection. Given two sets A and B, the intersection of A and B is the set containing the elements that belong both to A and B, and is denoted with $A \cap B$.



Complement. Given a universal set U and a set A, the complement of A in U is the set containing all the elements in U that do not belong to A, and is denoted with $U \setminus A$.

Relations

- **Cartesian product** $A \times B$ is defined as a set of ordered couples (a, b) where $a \in A$ and $b \in B$

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

Example:
given $A = \{1,2,3\}$ and $B = \{a, b\}$
$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$
$B \times A = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$

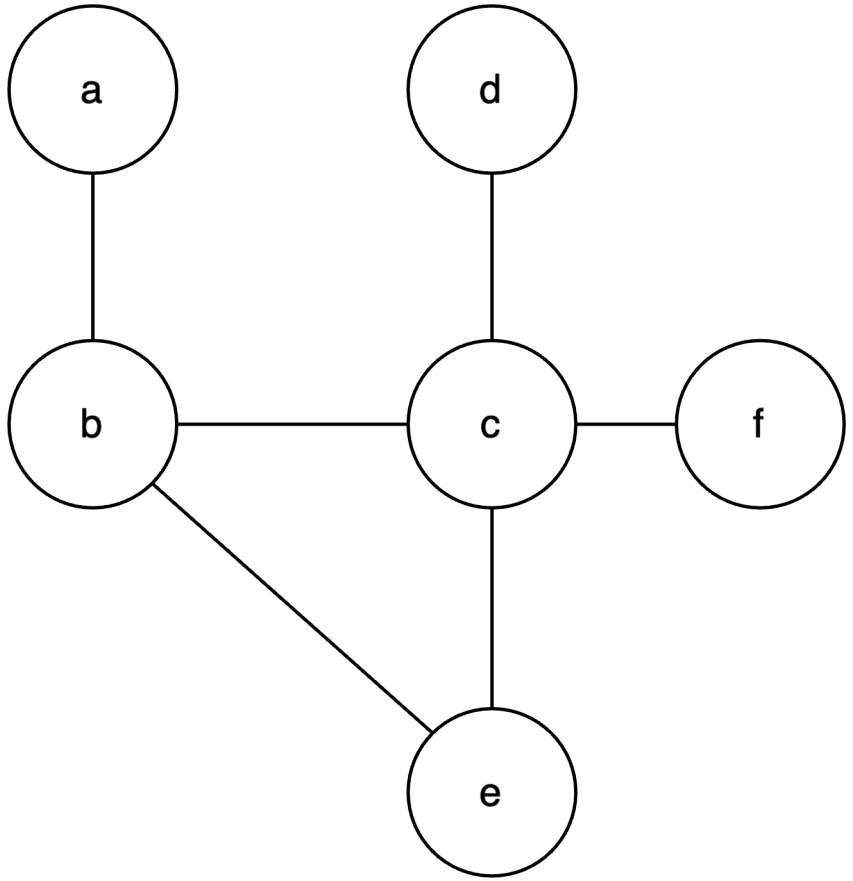
- **Relation** R from set A to set B is a subset of the cartesian product of A and B : $R \subseteq A \times B$.
  If $(x, y) \in R$ then we write xRy and we say 'x is R-related to y'.

  Given relation R from A to B:
  - the **domain** of R is the set $Dom(R) = \{a \in A \mid there\ exists\ a\ b \in B\ such\ that\ aRb\}$
  - the **co-domain** of R is the set $Cod(R) = \{b \in B \mid there\ exists\ an\ a \in A\ such\ that\ aRb\}$

Index

- Language
- Set theory
- **Graphs**
- From Sets to Graphs

Intuition



- Graphs are composed of nodes and arches connecting them.
- They are an intuitive way to represent relationships between objects.

- Formally, we define a graph as a pair $G = (V, E)$ where V is the set of **vertices** (or **nodes**) and E is the set of **edges** (or **links**), where an edge is a pair of nodes.

$$G = (V, E)$$
$$V = \{a, b, c, d, e, f\}$$
$$E = \{(a, b), (b, c), (e, b), (d, c), (c, e), (c, f)\}$$

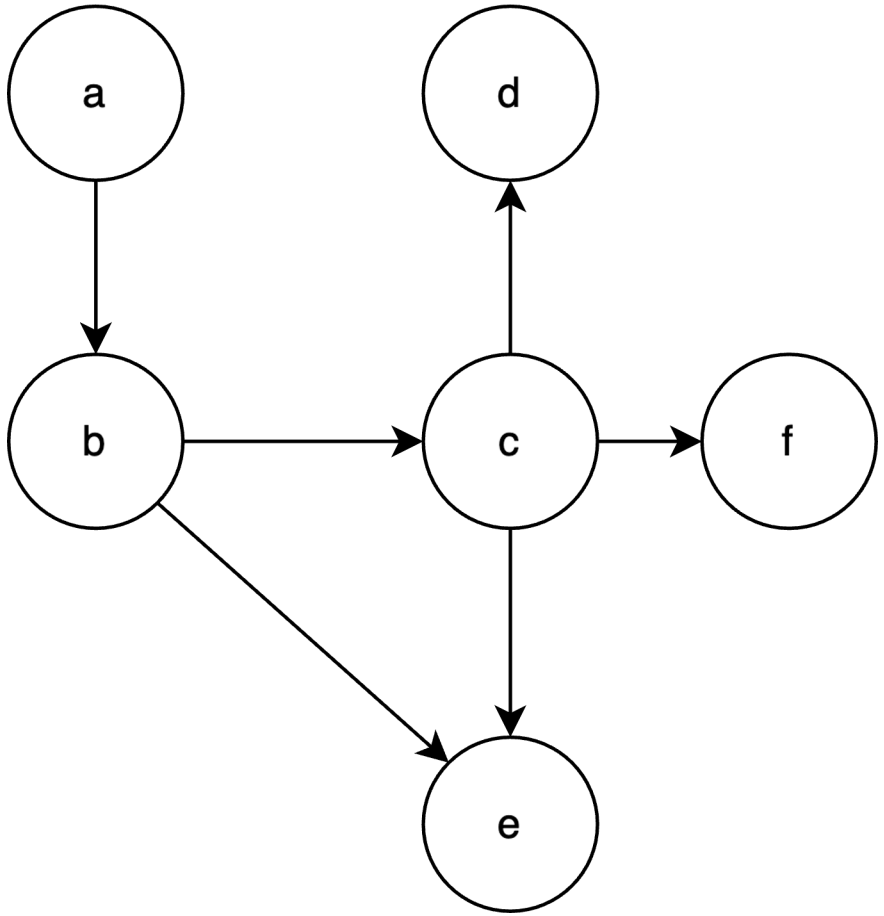
Basic notions

- The **order** of a graph is the number of its nodes.
- The **size** of a graph is the number of its edges.
- The **degree** of a node is the number of edges to or from it.

- A **directed graph** is a graph where edges are ordered pairs of distinct nodes (x, y). x and y are called **end points**, where x is the **tail** and y is the **head**.

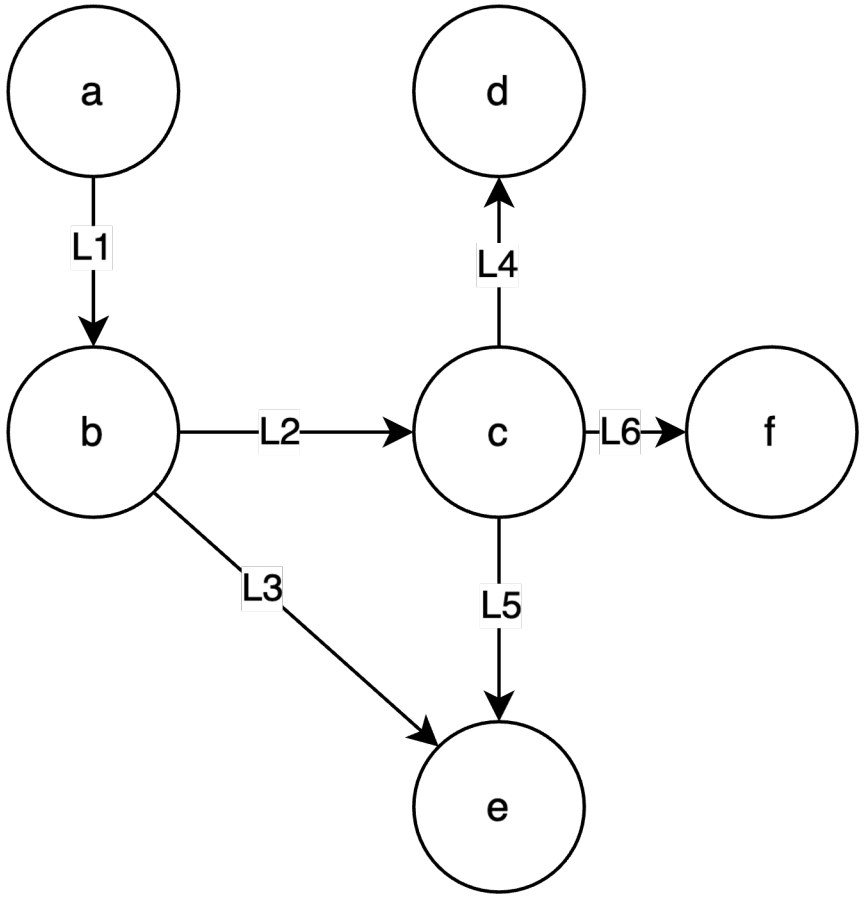
$G = (V, E)$
$V = \{a, b, c, d, e, f\}$
$E = \{(a, b), (b, c), (b, e), (c, d), (c, e), (c, f)\}$

- A **leaf** is a node in a directed graph with no outgoing edges.



Labeled Graphs

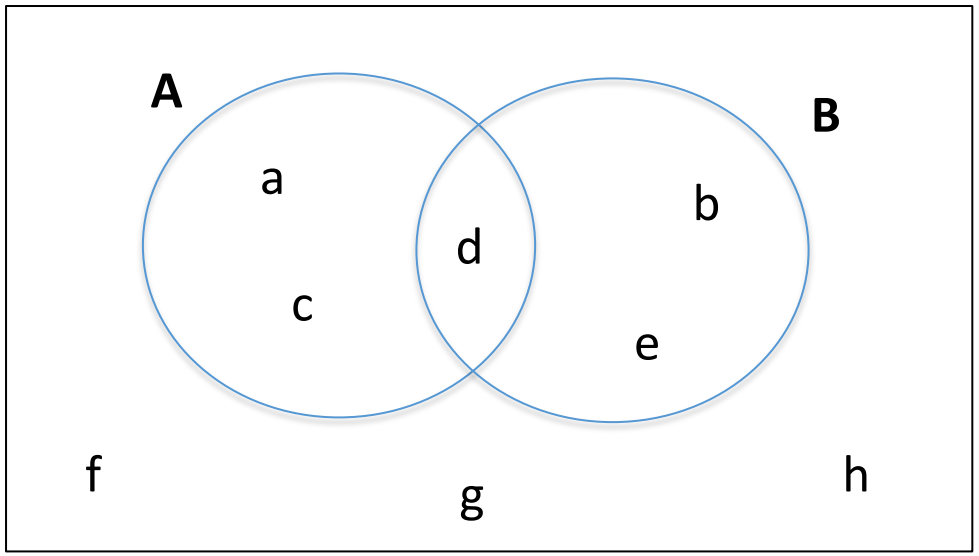
- A **labeled graph** is a graph where each node and edge is assigned a label. We are going to focus mostly on **directed labeled graphs.**

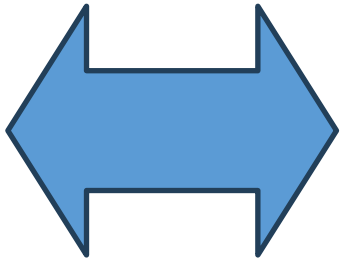


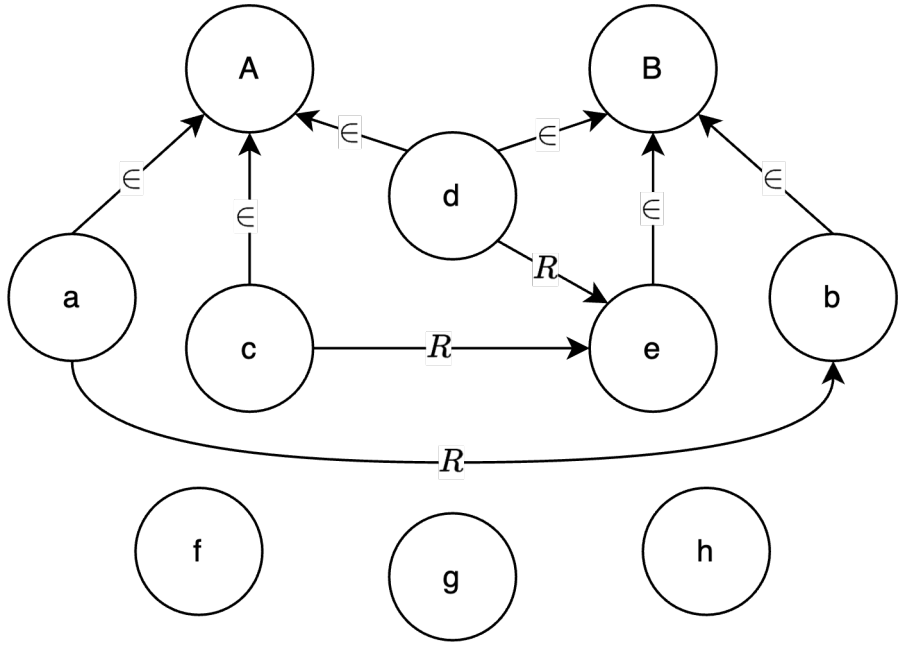
Index

- Language
- Set theory
- Graphs
- **From Sets to Graphs**

From sets to graphs





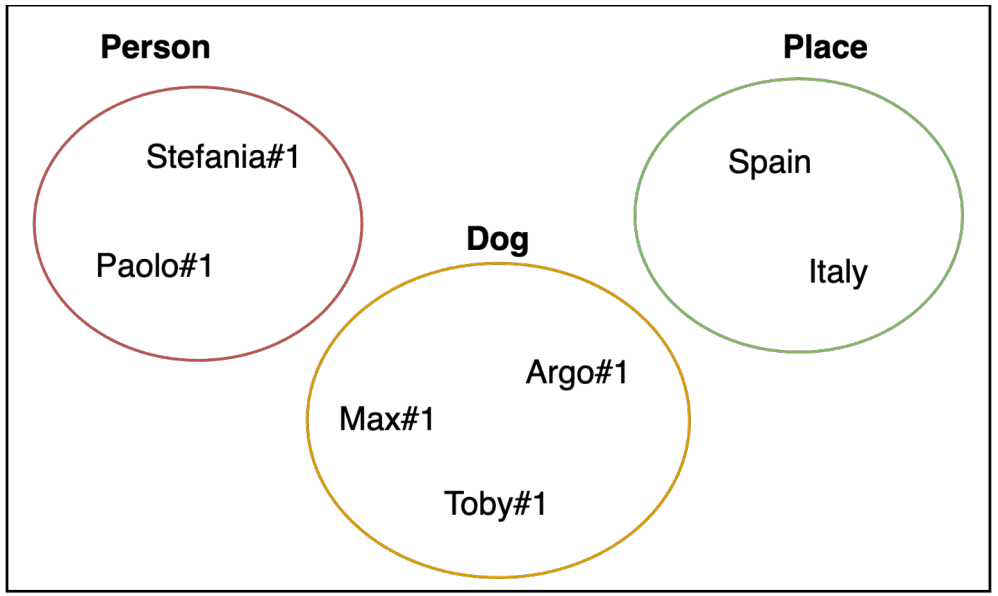


$R = \{(a,b),(c,e),(d,e)\}$
$A = \{a,d,c\}$
$B = \{d,e,b\}$

$V = \{A,B,a,b,c,d,e,f,g,h\}$
$E_{\in} = \{(a,A),(c,A),(d,A),(d,B),(e,B),(b,B)\}$
$E_R = \{(a,b),(c,e),(d,e)\}$

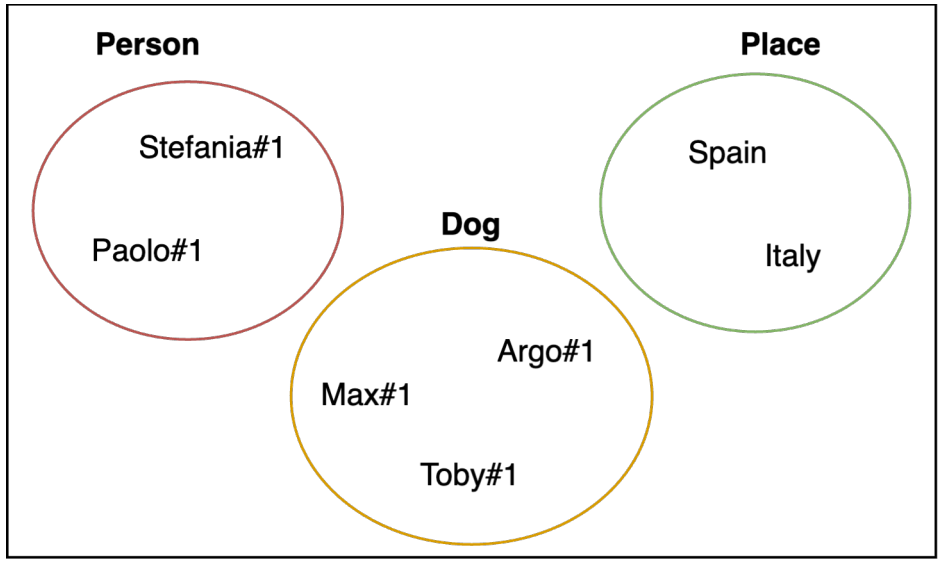
From sets to graphs

$Person(Stefania\#1)$
$Person(Paolo\#1)$
$Dog(Max\#1)$
$Dog(Argo\#1)$
$Dog(Toby\#1)$
$Place(Spain)$
$Place(Italy)$
$hasFriend(Stefania\#1, Paolo\#1)$
$hasPet(Paolo\#1, Max\#1)$
$hasPet(Paolo\#1, Argo\#1)$
$hasPet(Stefania\#1, Toby\#1)$
$hasBirthplace(Paolo\#1, Spain)$
$hasBirthplace(Stefania\#1, Italy)$

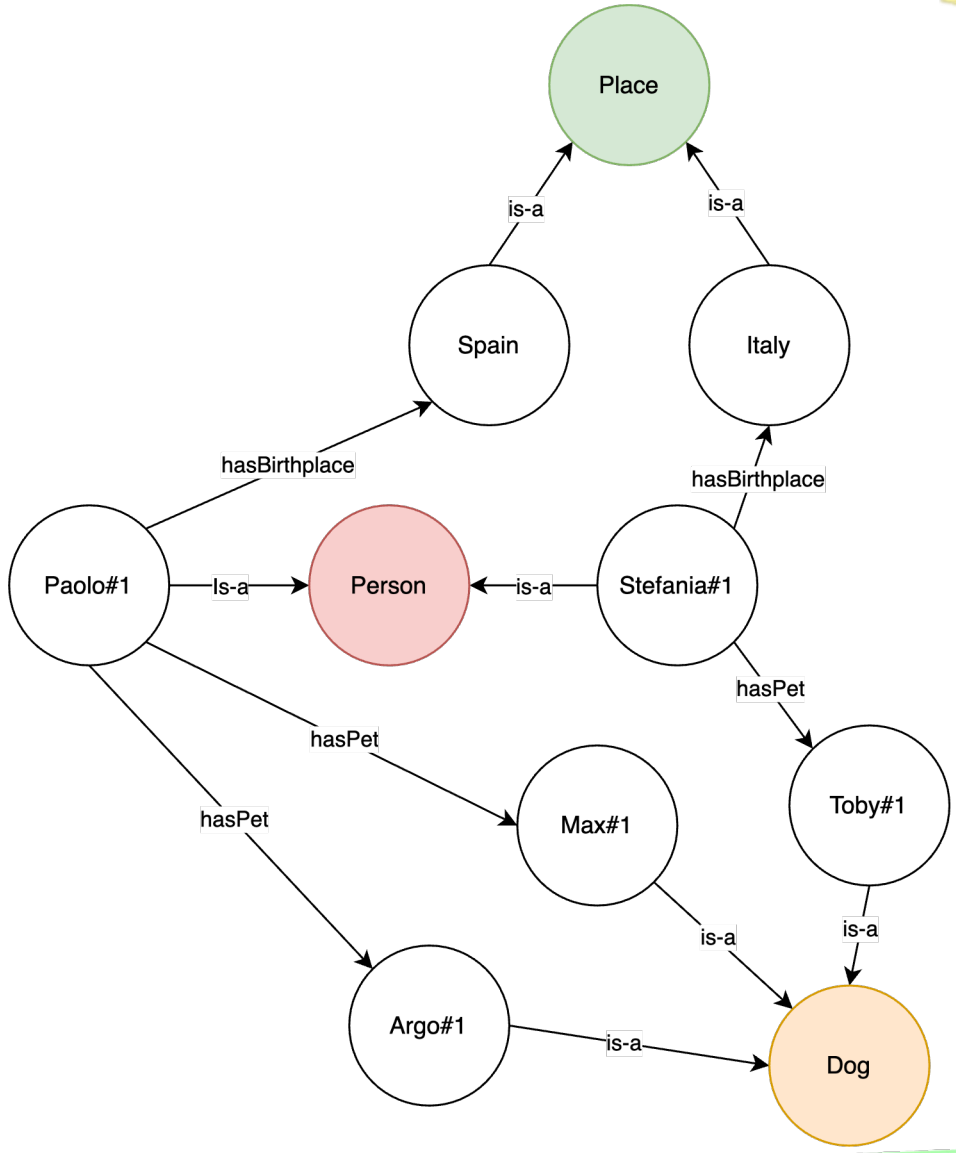


$$R = \{ (Stefania\#1, Paolo\#1), (Paolo\#1, Max\#1), (Paolo\#1, Argo\#1), (Stefania\#1, Toby\#1), (Paolo\#1, Spain), (Stefania\#1, Italy)\}$$

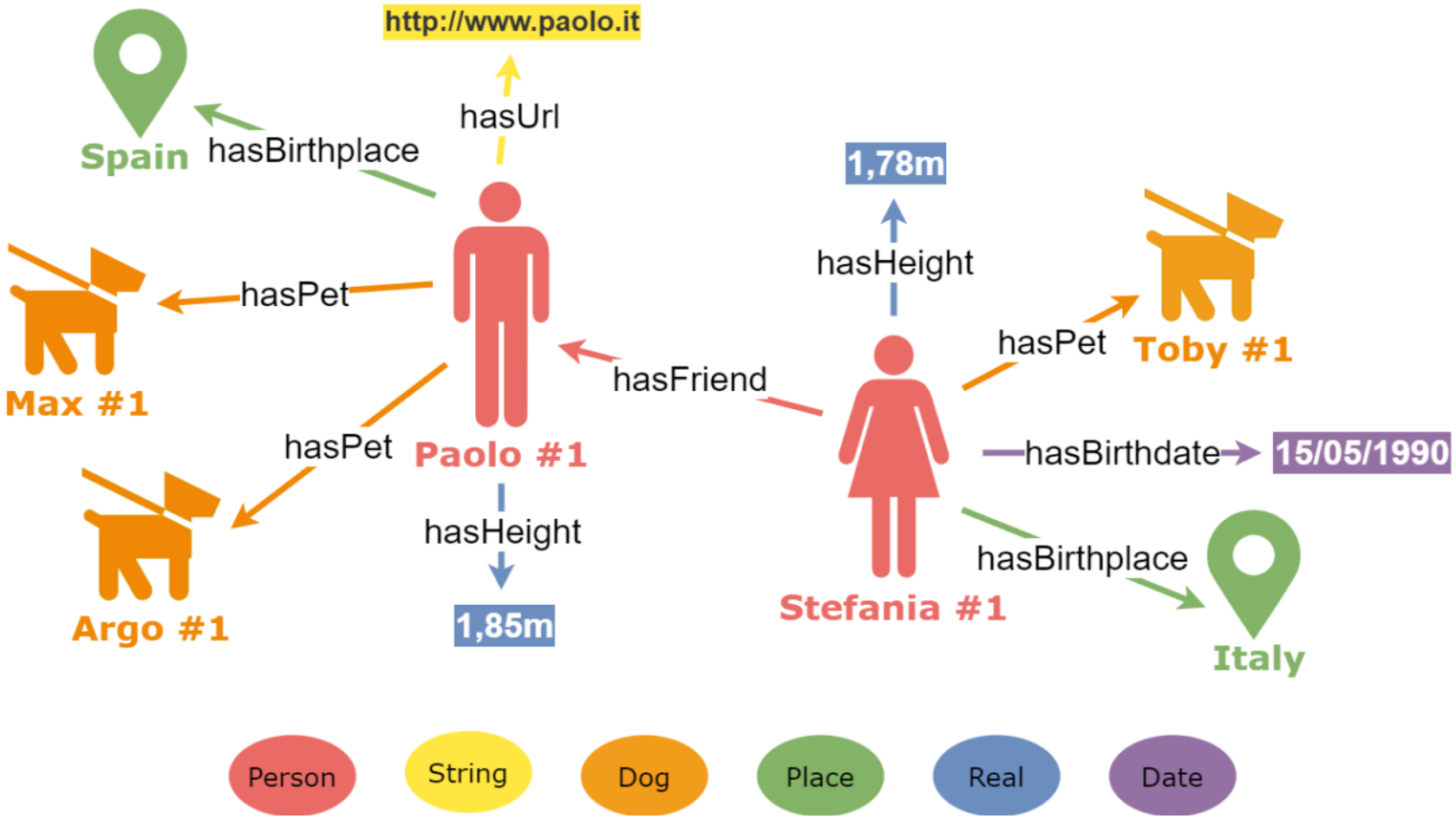
From sets to graphs



$$R = \{\ (Stefania\#1, Paolo\#1),\ (Paolo\#1, Max\#1),\ (Paolo\#1, Argo\#1),\ (Stefania\#1, Toby\#1),\ (Paolo\#1, Spain),\ (Stefania\#1, Italy)\}$$



From sets to graphs



Key Notions

- Formal grammar and formal languages.
- Chomsky hierarchy:
  - type 0 grammars
  - type 1 grammars
  - type 2 grammars
  - type 3 grammars
- Set theory basic notions and notation.
- Graphs basic notions and notation.
- Relations between elements of sets.
- Mapping of a set to a graph.

Reference formalisms (T2MP)