Lesson 1: First-order logic in database

Theme: Databases as first-order structures, and first-oder logic as their query language.

1 Databases as relational structures

Let $\tau = \{R_1, \ldots, R_k\}$ be a finite set of symbols, called *vocabulary*. Each symbol R_i is called a *relation* symbol, and is associated with a positive integer called its *arity*, denoted by $\text{ar}(R_i)$. We usually write $R, S, T, \ldots$, possibly indexed, to denote relation symbols.

A *database* (over vocabulary τ) is $\mathsf{DB} = (U, R_1^{\mathsf{DB}}, \ldots, R_m^{\mathsf{DB}})$ where

- U is a *finite* set of elements, called the *domain* of DB;
- each R_i^{DB} is a relation over U of arity $\text{ar}(R_i)$, i.e., $R_i^{\mathsf{DB}} \subseteq U^{\text{ar}(R_i)}$.

The superscripts DB in R_i^{DB} is to indicate that we are talking about R_i in the database DB. When DB is clear from the context, we will usually omit the superscript. Furthermore, in database context, every element $v \in U$ appears in at least one of the relations $R_1, \ldots, R_k$. So, when mentioning DB, it is not necessary to write the domain U. From now on, we will only write $\mathsf{DB} = (R_1, \ldots, R_k)$ to denote a database over vocabulary $\tau = (R_1, \ldots, R_k)$.

We will usually write $\bar{a}$ to denote $(a_1, \ldots, a_l)$ for some appropriate l. For example, we will simply write $R(\bar{a}) \in \mathsf{DB}$ to indicate that the tuple $\bar{a} = (a_1, \ldots, a_l)$ is in the relation R in DB, where l is the arity of R.

2 First-order logic for querying databases

We reserve a set VAR of *variables*: $x, x_1, x_2, \ldots, y, y_1, y_2, \ldots, z, z_1, z_2, \ldots$.

Syntax. First-order (FO) formulas over the vocabulary τ are defined inductively as follows.

- If $x, y \in Var$ are variables, then $x \approx y$ is an FO formula over τ.
- If $x_1, \ldots, x_n \in$ VAR are variables, and $R \in \tau$ is a relation symbol of arity n, then $R(x_1, \ldots, x_n)$ is an FO formula over τ.
- If α and β are FO formulas over τ, then so are $\neg\alpha$ and $\alpha \wedge \beta$.
- If α is an FO formula over τ, and $x \in$ VAR, then so is $\exists x(\alpha)$.

Formulas $x \approx y$ and $R(x_1, \ldots, x_n)$ are called atomic formulas. To avoid clutter, we will simply say "*formula*" to mean "FO *formula over* τ."

The set of *free variables* of a formula α, denoted by $\text{free}(\alpha)$, is defined inductively as follows.

- If α is an atomic formula $x \approx y$, then $\text{free}(\alpha) = \{x, y\}$.
- If α is an atomic formula $R(x_1, \ldots, x_n)$, then $\text{free}(\alpha) = \{x_1, \ldots, x_n\}$.
- $\text{free}(\neg\beta) = \text{free}(\beta)$.
- $\text{free}(\beta \wedge \gamma) = \text{free}(\beta) \cup \text{free}(\gamma)$.
- $\text{free}(\exists x\ \beta) = \text{free}(\beta) - \{x\}$.

Formulas without free variables are called *sentences*, or *closed formulas*. Otherwise, they are called *open formulas*. Sometimes, we will write $\varphi(x_1, \ldots, x_n)$ to indicate that the free variables in φ are $x_1, \ldots, x_n$. We will write $\varphi(\bar{x})$ to denote $\varphi(x_1, \ldots, x_n)$.

Semantics. Let DB be a database and U be its domain. An *assignment* over DB is a function $\mathrm{val} : \mathrm{VAR} \to U$. For $x \in \mathrm{VAR}$ and $a \in U$, we denote by $\mathrm{val}[z \mapsto a]$ the assignment obtained from s by changing the value of $\mathrm{val}(z)$ to a, while the values for the other variables are left untouched. Formally,

$$\mathrm{val}[z \mapsto a](x) \quad := \quad \begin{cases} a & \text{if } x \text{ is } z \\ \mathrm{val}(x) & \text{for any variable } x \text{ other than } z \end{cases}$$

Let φ be a formula. For an assignment val and a database DB, we define $\mathsf{DB}, \mathrm{val} \models \varphi$ (read: "φ *holds in database* $\mathsf{DB} = (R_1, \ldots, R_k)$ *under the assignment val*") inductively as follows.

- $(\mathsf{DB}, \mathrm{val}) \models x \approx y$, if and only if $\mathrm{val}(x) = \mathrm{val}(y)$.
- $(\mathsf{DB}, \mathrm{val}) \models R(\bar{x})$, if and only if $R(\mathrm{val}(\bar{x})) \in \mathsf{DB}$.
- $(\mathsf{DB}, \mathrm{val}) \models \neg\alpha$, if and only if it is *not true* that $(\mathsf{DB}, \mathrm{val}) \models \alpha$.
- $(\mathsf{DB}, \mathrm{val}) \models \alpha \wedge \beta$, if and only if $(\mathsf{DB}, \mathrm{val}) \models \alpha$ and $(\mathsf{DB}, \mathrm{val}) \models \beta$.
- $(\mathsf{DB}, \mathrm{val}) \models \exists x\ \alpha$, if and only if there is $a \in U$ such that $(\mathsf{DB}, \mathrm{val}[x \mapsto a]) \models \alpha$.

Note that the values of the assignment val on variables other than free variables are not important in deciding $(\mathsf{DB}, \mathrm{val}) \models \varphi$. So, we will usually omit val, and only indicate the values assigned to free variables. That is, we will simply write:

$$(\mathsf{DB}, x_1 \mapsto a_1, \ldots, x_n \mapsto a_n) \ \models\ \varphi(\bar{x}) \qquad \text{or, in short} \qquad (\mathsf{DB}, \bar{x} \mapsto \bar{a}) \ \models\ \varphi(\bar{x})$$

Querying database with FO formulas. Let $\varphi(\bar{x})$ be a formula. On database DB, we define:

$$\varphi(\bar{x})(\mathsf{DB}) \quad := \quad \{\ \bar{a} \mid \mathsf{DB}, \bar{x} \mapsto \bar{a} \models \varphi\ \}$$

When the free variables $\bar{x}$ are clear from the context, we will omit them and simply write $\varphi(\mathsf{DB})$. For a sentence φ, $\varphi(\mathsf{DB})$ is either `True` or `False`. Such query is called *Boolean* query.

Appendix

Database as a collection of tables. In our representation of database, we write $R(x, y, z)$ to access the tuples in a relation R by "denoting" the first, the second, and the third components with x, y and z, respectively. A more standard representation of databases is by viewing a relation R as a set of $\mathrm{ar}(R)$ number of attributes. For example, for a relation R of arity 3, $R.\mathrm{att}_1$, $R.\mathrm{att}_2$ and $R.\mathrm{att}_3$ are used to access the first, the second, and the third component of the tuples in relation R, respectively.