Database Theory in Action: Learning Logical Modelling with Iltis

Tristan Kneisel
Ruhr University Bochum, Germany

Fabian Vehlken
Ruhr University Bochum, Germany

Thomas Zeume
Ruhr University Bochum, Germany

Abstract

Important learning objectives in database education are to learn how to design database schemas and how to write queries to access data stored in a database adhering to a schema.

In this article we report on results from [7][1] where this learning objective is addressed for the related educational task of modelling with logical formalisms. Two key steps in logical modelling are to (a) choose a suitable *vocabulary*, that is, e.g., which first-order symbols to use and with which intended meaning, and then to (b) construct actual formal descriptions, i.e. first-order formulas over the chosen vocabulary. While (b) is addressed by several educational support systems for formal foundations of computer science, (a) is so far not addressed at all – likely because it involves specifying the intended meaning of symbols in natural language. We propose a conceptual framework for educational tasks where students choose a vocabulary and implement it for tasks for designing propositional and first-order vocabularies within the Iltis educational system.

2012 ACM Subject Classification Theory of computation → Logic; Social and professional topics → Computing education

Keywords and phrases Educational support systems, Logic, Database theory, Natural language processing

Digital Object Identifier 10.4230/LIPIcs.ICDT.2026.28

Related Version *Extended Version*: `https://doi.org/10.48550/arXiv.2504.21384` [7]

Funding Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant 532727578.

1 Introduction and Motivation

Modelling data and knowledge with mathematical formalisms and subsequently attacking the underlying problems with rigorous methods is an essential part of computer science education at the university level [6, 3]. This is taught, for instance, in introductory courses on *Databases* (where designing database schemata and querying data with query languages is an important part) and *Logic for computer science* (where formalizing knowledge with logical formulas and inference mechanisms play an important role).

Based on our recent work [7], we report on how learning to model natural language statements in first-order logic is supported by the Iltis educational support system [11, 4]. As first-order logic plays an important role in database theory research, we hope that our system is helpful for members of the community who teach foundations of logic. The connection between first-order logic and core-SQL (see, e.g, [1]) may allow to transfer our insights to educational support for learning schema design and SQL.

[1] A short variant of that article has appeared in AIED 2025 [8].

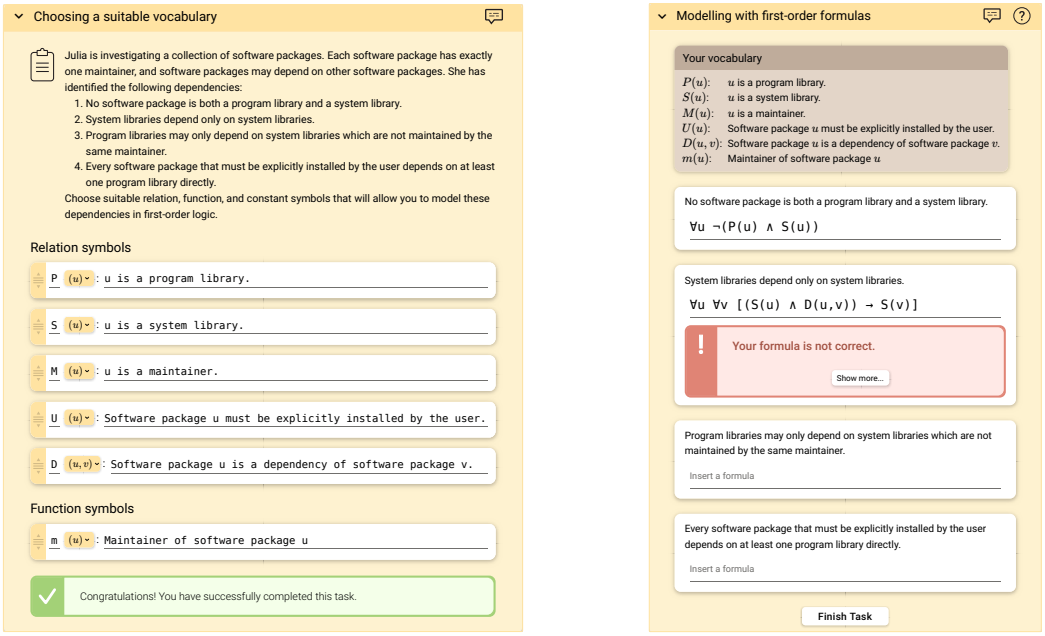

Figure 1 An assignment in the Iltis educational support system where students (a) design a first-order vocabulary (left), and then (b) write first-order formulas using their vocabulary (right).

Figure 1 depicts an exercise on first-order logic in the Iltis system[2] that students can typically solve after an introductory course on logic by executing the following steps:

Step 1: Design a suitable *vocabulary* to describe a real world scenario by identifying suitable relation and function symbols with their intended meaning (task on the left).

Step 2: Describe the scenario by logical formulas over this vocabulary (task on the right).

Step 3: Transform formulas into an adequate, more simple form.

Step 4: Infer new knowledge using an inference mechanism (e.g. resolution).

Technological support for Steps 2 – 4 has been present in state-of-the-art systems. We therefore focus on how we integrated Step 1 into the Iltis educational support system.

Related Work. Learning how to model with first-order logic is closely related to learning how to design schemas and write SQL queries. Systems like [10] have successfully leveraged database theory for teaching relational queries, e.g. by explaining incorrect queries via small counterexamples based on provenance. Similar aspects are integrated in Iltis, but for first-order logic instead of SQL. For testing correctness of vocabularies specified by students, we need to translate between different vocabularies. This task is similar to schema transformations [9] and schema mappings and data exchange [2].

[2] We invite interested readers to try this exercise in Iltis: [5]

2 A Framework for Vocabulary Design Tasks

When students design a first-order vocabulary with the intention of writing formulas, they must say (i) which symbols they want to include, and (ii) which meaning the symbols shall have. An educational task that allows to specify a vocabulary must then check student attempts for correctness. If an attempt is not correct, it must provide feedback; if it is correct, the vocabulary can subsequently be used in follow-up tasks (compare Figure 1).

For checking these conditions automatically, an instructor needs to specify (1) which symbols can in principal be used, and (2) which subsets of symbols constitute solutions. More formally, a *solution space* $(\mathcal{V}, \mathcal{S})$ consists of

- *potential vocabulary symbols* $v^* = (n^*, D^*) \in \mathcal{V}$ with a name n^* and a set $D^* = \{d_1^*, d_2^*, \dots\}$ of natural language descriptions;
- a *solution set* $\mathcal{S} \subseteq 2^{\mathcal{V}}$ of subsets of symbols that constitute solutions.

▶ **Example 1.** In the first-order modelling task from Figure 1, in addition to the names of symbols, information on whether the symbol is a function, relation or constant symbol and an arity, if applicable, is included. For example

- $\left(D(u,v), \left\{ \begin{array}{l} \text{Software package } u \text{ depends on software package } v, \\ \text{Software package } v \text{ is a dependency of software package } u \end{array} \right\}\right)$,
- $(m(u), \{\text{The maintainer of software package } u, \dots\})$,

where D is a binary relation symbol and m is a unary function symbol. Note that the descriptions may refer to parameters depending on the symbol's arity. ⌟

Scenarios can often be modelled with many different vocabularies, yet there is often a "canonical vocabulary" from which all other vocabularies can be derived by small modifications. Examples of such modifications are that (i) constant symbols can also be modelled by unary relation symbols; (ii) k-ary function symbols can be modelled by $(k+1)$-ary relation symbols; and (iii) for modelling k types of elements in a domain one can use either k or $k-1$ unary relation symbols. In order to simplify the specification of the solution set $\mathcal{S}$, our framework allows to specify proper solutions using Boolean formulas over potential vocabulary symbols. Such a formula, for instance, can state that a unary function symbol f or its corresponding binary relation symbol F must be included.

Our framework, when given a student attempt V containing *vocabulary symbols* $v = (n, d)$, works in two phases:

Phase 1: Mapping vocabulary symbols v to potential solution symbols v^*

(a) Construct a map $\mu\colon V \to \mathcal{V} \times \mathcal{C}$ that maps each $v \in V$ to a pair (v^*, c) where v^* is a potential vocabulary symbol in the solution space that is the best fit for v and c is a category from a finite set $\mathcal{C} = \mathcal{P} \uplus \mathcal{N}$ of positive categories $\mathcal{P}$ and negative categories $\mathcal{N}$ indicating how well v^* fits v.

(b) If μ assigns a negative category $c \in \mathcal{N}$ to some v, provide feedback according to the category c. If μ only assigns positive categories from $\mathcal{P}$, then continue with Phase 2.

Phase 2: Checking solutions

(a) Check that the vocabulary symbols V chosen by the student constitute a solution by testing that $\{v^* \mid (v^*, c) \in \mu(V)\} \in \mathcal{S}$.

(b) If the solution is not correct, provide feedback accordingly. Otherwise, the attempt V with its symbols and descriptions is the output of this educational task (and can be used in subsequent tasks).

Mapping a symbol $v = (n, d)$ from a student attempt into the solution space requires to determine whether, and to which degree, the description d corresponds to natural language descriptions of symbols in the solution space. This essentially boils down to the NLP task of determining the (semantic) similarity of pairs of strings. For first-order vocabulary symbols, it is slightly more intricate as descriptions may refer to parameters such as in Example 1. For reasons of resource efficiency and data sovereignty, instead of using LLMs, we use smaller similarity models that we fine-tuned with data sets specifically aimed at the respective assignment. Due to the lack of fine-tuning data at the time of implementation, we generated data sets for fine-tuning using a grammar-based approach that allowed us to compactly represent many different (artificial) student descriptions of different "quality".

Providing feedback. The most prominent reason for a student attempt V to be incorrect is that some necessary vocabulary symbol is not included in V. To provide feedback in such cases, our framework allows to specify feedback for different sets of potential vocabulary symbols given via Boolean formulas. For instance, when designing a vocabulary for the assignment in Figure 1, a student may have forgotten to provide a relation U expressing that a package must be explicitly installed by the user, because she did not read statement (4). To provide feedback in such cases, an instructor can specify that for student attempts V that satisfy $\neg U$ (i.e. U was omitted by the student) the system provides the following feedback

> "Did you make sure that the statement
>
> *Every software package that must be explicitly installed by the user depends on at least one program library directly.*
>
> can be modelled?"

Challenge: Flexibility in modelling. The compositional task model of Iltis allows instructors to flexibly combine small educational tasks into larger exercises such that student inputs can be used in later tasks. Providing flexibility in how students can design their vocabulary comes with the following challenge. In many assignments, we ask students to write formulas using the vocabulary they have designed in a previous step. Instructors specify solution formulas over some (canonical) vocabulary V^*; a student may have designed a different vocabulary V. To support this, our implementation supports translations of formulas over some vocabulary V^* into formulas over vocabulary V. This allows students to continue, for example, with a binary relation symbol F instead of a unary function symbol f used in V^*.

3 Summary

We summarized our article [7] and reported how the Iltis educational system integrates educational tasks for modelling with first-order logic. The full framework is described, implemented, and evaluated in that article. These educational tasks have been used in introductory logic courses for computer science students at Ruhr University Bochum and TU Dortmund University with > 300 students each.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

2 Ronald Fagin, Phokion G. Kolaitis, and Lucian Popa. Data exchange: getting to the core. *ACM Trans. Database Syst.*, 30(1):174–210, 2005. doi:10.1145/1061318.1061323.

3 Gesellschaft für Informatik e. V. Empfehlungen für Bachelor- und Master-Programme im Studienfach Informatik an Hochschulen. https://gi.de, 2016. URL: https://dl.gi.de/handle/20.500.12116/2351.

4 Iltis: Formal foundations of computer science online. URL: https://iltis.rub.de.

5 Iltis: Logical modelling – inference workflows. URL: https://iltis.rub.de/modelling-pipeline-showcase/.

6 Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, New York, NY, USA, 2013.

7 Tristan Kneisel, Fabian Vehlken, and Thomas Zeume. Logical modelling in cs education: Bridging the natural language gap, 2025. doi:10.48550/arXiv.2504.21384.

8 Tristan Kneisel, Fabian Vehlken, and Thomas Zeume. Logical modelling in cs education: Bridging the natural language gap. In Alexandra I. Cristea, Erin Walker, Yu Lu, Olga C. Santos, and Seiji Isotani, editors, *Artificial Intelligence in Education*, pages 457–463, Cham, 2025. Springer Nature Switzerland. doi:10.1007/978-3-031-98459-4_34.

9 Alexandra Poulovassilis and Peter McBrien. A general formal framework for schema transformation. *Data & Knowledge Engineering*, 28(1):47–71, 1998. 16th International Conference on Conceptual Modelling. doi:10.1016/S0169-023X(98)00013-5.

10 Sudeepa Roy, Amir Gilad, Yihao Hu, Hanze Meng, Zhengjie Miao, Kristin Stephens-Martinez, and Jun Yang. How database theory helps teach relational queries in database education (invited talk). In Graham Cormode and Michael Shekelyan, editors, *27th International Conference on Database Theory, ICDT 2024, March 25-28, 2024, Paestum, Italy*, volume 290 of *LIPIcs*, pages 2:1–2:9. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICDT.2024.2.

11 Marko Schmellenkamp, Fabian Vehlken, and Thomas Zeume. Teaching formal foundations of computer science with Iltis. *Educational Column of the Bulletin of EATCS*, 2024. URL: http://bulletin.eatcs.org/index.php/beatcs/article/download/797/842.