AKADÉMIAI KIADÓ

Evolving graph based knowledge space model for tutoring systems

László Csépányi-Fürjes* and László Kovács

Institute of Information Science, Faculty of Mechanical Engineering and Informatics, University of Miskolc, Miskolc-Egyetemváros, Hungary

Pollack Periodica • An International Journal for Engineering and Information Sciences

19 (2024) 3, 40–45

DOI: 10.1556/606.2024.01058

Received: January 3, 2024 • Revised manuscript received: April 25, 2024 • Accepted: May 28, 2024
Published online: July 31, 2024

ORIGINAL RESEARCH PAPER

ABSTRACT

An intelligent tutoring system is a computer-based educational tool designed to provide adaptive learning environment to learners, mimicking the role of a human tutor. Its most typical areas of application are language learning, mathematics education, programming courses and medical training. Intelligent Tutoring Systems are based on the knowledge-module that is holding the system's knowledge in a well-structured format. Considering the current state of the art knowledge-module representations, a model that can represent evolving information is lacking. Representing evolving information is needed for those tutoring systems that are working with dynamically changing domains, e.g., software science. In this paper a new combined model is presented that is based on the ontology model and the fundamentals of knowledge space theory. The proposed model introduces the term of abstract time to be able to formulate an evolving knowledge graph. This paper introduces the term of evoking-hooks that makes it possible to realize connections between external domain elements and the nodes of the proposed model.

KEYWORDS

evolving systems, time-dependent graph, knowledge space theory, intelligent tutoring system

1. INTRODUCTION

In recent years the phenomenon of fast technological and societal change has reached an unprecedented level. Learning does not end by leaving the educational institutions especially for Information Technology (IT) professionals and programmers. It is expected that Intelligent Tutoring Systems (ITS) will increasingly appear in workplaces and everyday situations [1] due to the constant need for adoption to business change [2]. There is a high demand for self-regulated learning experiences as well as transparent personalized learning environments [3]. Due to this high demand several research activities were initiated to realize ITS systems. An ITS system is considered intelligent because it is mimicking the role of a human tutor by recognizing and monitoring the actual knowledge state of the learner and by able to change the learning path according to that. Knowledge representation has been conceptualized by numerous theories, for instance knowledge spaces, ontologies, rough sets, fuzzy sets, formal concept analysis, evidence theory, granular computing, etc., [4]. For an educational system a knowledge representation model not only has to support structural knowledge storing, but also needs to enable determination of actual knowledge states of the learner [5]. One of the most fundamental theories of knowledge state determination is Knowledge Space Theory (KST) [6]. Although several tutoring systems were developed over the years, only few of them utilized the advantages of KST [7]. While KST conceptualizes the knowledge as a set of questions or tasks, the Competence-based Knowledge Space Theory (CbKST) focuses on skills and competencies. Probably this is why most of the tutoring systems are rather using ontology-like knowledge representation where knowledge elements are more like real-life entities or terms and their elaborated description [8]. It can also be stated that the currently existing tutoring systems are focusing on domains that can be considered unvarying, e.g.,

*Corresponding author.
E-mail: laszlo.csepanyi-furjes@uni-miskolc.hu

math, chemistry, physics, etc., and has not been much effort in exploring evolving fields, e.g., computer programming.

In this paper a novel knowledge representation model is presented that unifies the advantages of CbKST and ontologies. Also, this paper introduces the term of abstract-time to the model to be able to represent time-dependent, evolving knowledge. The proposed model is a knowledge-graph that also allows building connections between existing knowledge forms, like textual documents or programming source code and the graph nodes that are called knowledge-units.

2. RELATED WORK

2.1. Educational knowledge representation with ontologies

According to Gruber [9] ontology is an explicit specification of a conceptualization. Ontology and/or semantic web based educational systems are grounded on the principle of entities and their hierarchical relations. In the most common scenario, an entity represents a unit of study; the relation between the entities represents the hierarchical relation between the units of study. Moreover, ontology is good for representing teaching strategies, student profile, modeling competence and learning goals, etc., [10]. Using ontologies, it is also possible to simplify administration processes [11]. The process of building an educational ontology relies on domain experts who are extracting domain ontology from existing learning materials [12].

Domain concepts can be well modeled by ontologies. Studying the ontology based tutoring systems two conclusions can be drawn. First, the knowledge domain that is represented by the ontology is usually separated from the educational content. Second, the ontological structure is grounded on natural object hierarchy, where the relation between the elements is based on generalization/specialization of the concepts represented by the elements [10]. In other words, the relation between the elements is not representing prerequisite relations, which is a key factor in epistemology. According to another observation, there are no two learning ontologies, designed by different people, would be the same [8].

2.2. Competency based knowledge space theory

According to KST the knowledge domain is represented by a network of nodes, where the nodes are questions or problems, and the edges are surmise relations [5, 13]. Based on the assessment theory some knowledge elements normally precede, in time, other knowledge elements. In other words, a certain question can be answered, or a problem can be solved only if some other questions have already been answered or some other problems have already been solved [14].

One important extension of KST is CbKST. CbKST is grounded on the observation that the learner needs skills to solve specific problems. Using the CbKST representation, the prerequisite relation can be determined by identifying the skills needed to solve the problems or to answer the questions [15]. In KST the set of questions that are needed to be answered to master a specific field of knowledge is called *knowledge space*. The learner's *knowledge state* is a subset of knowledge space, meaning those questions that the learner can answer. If the actual knowledge state of a learner is known, it can be defined where the learner can proceed (*outer fringe*) and it can be specified where to step back in case there is an understanding problem (*inner fringe*) [5, 14]. The knowledge state problem was also generalized further to assess the state of any system [16].

It may happen that the learner fails to solve a task that is lower in the hierarchy but can solve another task that is at a higher level. This is the consequence of a certain degree of instability of knowledge [17]. According to the original theory the learner's answer to a question is dichotomously classified as correct or incorrect. This is well suited for the classical domains, but in other cases, like: psychological assessment, attitudes, and opinions the dichotomous case appears to be too restrictive. To solve this problem a polytomous generalization of the model was presented [18]. In one of the latest extensions of CbKST the Competence-based Concept-Cognitive Learning Model (CbCCLM) was introduced where CbCCLM can study the transformation relationship between skills and knowledge from the perspective of competences [19].

2.3. Time dependency

In a well-known and general sense, time is a temporal concept that represents one specific moment or duration. One moment can be also called one time-instance and a duration can be called a time-interval. The Web Ontology Language (OWL) is a language for defining Web ontologies. The temporal properties of a real-world object can be represented by the OWL-Time ontology [20].

3. THE PROPOSED EVOLVING KNOWLEDGE SPACE GRAPH

3.1. General description of the proposed model

In this paper the Evolving Knowledge Space Graph (EKSG) model is proposed. The network that represents prerequisite relations in CbKST can be displayed in a precedence diagram. The units of study in ontology can be the replacement of question/problem nodes of this precedence diagram. Thus, using prerequisite relations and units of study, a directed graph can be constructed. The resulted graph combines the benefits of ontologies and CbKST. Additionally; two important features are added to the proposed model. To cover the requirements of an evolving domain model, time dependency is introduced, and the term of abstract time is defined. Although the term of time dependent graph is well known and studied [21] it has not been

incorporated in educational knowledge representations yet. Second feature is the use of evoking-hooks that evokes certain knowledge elements by external triggers.

3.2. Abstract time

As opposed to the well-known and general meaning of time, in this paper time is considered as a higher-level abstraction. The reason why the model needs an abstract time property is to be able to represent evolving knowledge. There are domains where knowledge is not changing that fast, like mathematics, physics, or chemistry, etc. But in other fields, like the computer programming domain knowledge is constantly evolving.

From the perspective of change the term "time" does not have to mean Gregorian date-and-time. Software scientists invented the concept of "version" to represent change. A version behaves like a time instance. The main properties and functions of time concept, like "before", "after" and "interval" are also interpretable in the concept of version.

Definition 1: The proposed general *abstract time* in the given domain is an ordered finite set of instance values that are strongly related to the domain. Formally:

$$\mathbf{T}^{inst} = \left\{ t_1^{inst}, t_2^{inst}, \cdots, t_n^{inst} \right\}, \tag{1}$$

where t_i^{inst} is an instance value, and for every $1 \le i < n$: $t_i^{inst} < t_{i+1}^{inst}$. Here, n is the number of existing instance values.

Definition 2: Let $\mathbf{T}^{int}$ be a pairwise distinct set of *abstract time intervals* within the defined domain:

$$\mathbf{T}^{int} = \left\{ \mathbf{t}_1^{int}, \mathbf{t}_2^{int}, \cdots, \mathbf{t}_m^{int} \right\}, \tag{2}$$

where $\mathbf{t}_i^{int} = [t_{p_i}^{inst}, t_{k_i}^{inst}]$ be an abstract time interval where $1 \le p_i \le k_i \le n$ and $t_{p_i}^{inst}, t_{k_i}^{inst} \in \mathbf{T}^{inst}$ therefore $t_{p_i}^{inst} \le t_{k_i}^{inst}$. The following notation is introduced for the maximal interval: $\boldsymbol{t}_*^{int} = [t_1^{inst}, t_n^{inst}]$.

Definition 3: Let $\prec$ is the *matching relation* that works as follows:

$$\text{instance match} \prec \subseteq \mathbf{T}^{inst} \times \mathbf{T}^{inst}, \tau \prec t_i^{inst} \doteq \tau = t_i^{inst}, \tag{3}$$

$$\text{interval match} \prec \subseteq \mathbf{T}^{int} \times \mathbf{T}^{int}, \quad \tau \prec \boldsymbol{t}_j^{int} \doteq \exists l \left(p_j \le l \le k_j \right) : \tau \prec t_l^{inst}, \tag{4}$$

$$\text{set of instances match } \tau \prec \left\{ t_{l_1}^{inst}, t_{l_2}^{inst}, \cdots, t_{l_p}^{inst} \right\}, \quad (l_1 \le l_2 \cdots \le l_p) \doteq \exists l_i \left(\tau \prec t_{l_i}^{inst} \right), \tag{5}$$

$$\text{set of intervals match } \tau \prec \left\{ \boldsymbol{t}_{l_1}^{int}, \boldsymbol{t}_{l_2}^{int}, \cdots, \boldsymbol{t}_{l_k}^{int} \right\}, \quad (l_1 \le l_2 \cdots \le l_k) \doteq \exists l_j \left(\tau \prec \boldsymbol{t}_{l_j}^{int} \right), \tag{6}$$

where τ is the time instance that is being examined. The maximal interval $\boldsymbol{t}_*^{int}$ is used as default value when there is no time restriction in the model. Figure 1 is showing an example abstract timeline where all three definitions are visualized.

Time delay is not interpreted in the proposed model.

3.3. Evoking-hooks

An evoking-hook is a word or set of words that can be found in a domain document and is strongly related to a node of the knowledge graph. For example, in the python-programming-domain the keywords 'for' and 'in' are evoking the *ForLoop* knowledge-unit. A similar solution is mentioned in the definition of FrameNet, where frame-evoking words are used to evoke elements of FrameNet [22].

3.4. Formal description of the proposed EKSG model

Definition 4: The proposed *EKSG model* is a labeled acyclic directed graph that can be described in the following way: $\mathbf{G} = \{\mathbf{U}, \mathbf{R}\}$ where $\mathbf{U}$ is a set of unit nodes representing atomic knowledge, test, and material fragments, and $\mathbf{R}$ is a set of relations between the unit nodes. $\mathbf{U}$ is a triplet: $\mathbf{U} = \{\mathbf{U}^K, \mathbf{U}^T, \mathbf{U}^M\}$, where $\mathbf{U}^K$ is a set of knowledge-units, which are elements of a certain knowledge domain, represented by a set of four attributes:

$$\mathbf{U}^K \subseteq \left\{ (n, d, t, e) \mid n \in \mathbf{N}, d \in \mathbf{D}, t \subseteq \mathbf{T}^{int}, e \subseteq \mathbf{E} \right\}, \tag{7}$$

where $\mathbf{N}$ is the set of *non-null* and *unique* unit names. The name must reflect the meaning of the unit. The name acts as an external, human readable identifier; $\mathbf{D}$ is the set of *non-null* and *unique* textual descriptions. The aim of a description is to be able to better delimit the piece of knowledge represented by the unit; $\mathbf{T}^{int}$ is a set of abstract time intervals, t may be empty. If t is empty, then there is no time restriction in the model, meaning $\mathbf{U}^K$ is valid in all time intervals. If $\mathbf{T}^{int}$ contains no values the model reduces to a timeless graph; $\mathbf{E}$ is an evoking-hook set that may be empty.

$\mathbf{U}^T$ is a set of test-units which are items of an assessment set. An assessment item can be a question, a problem, or an exercise that is testing whether the learner has mastered a certain knowledge-unit. An assessment item is represented by a set of three attributes:

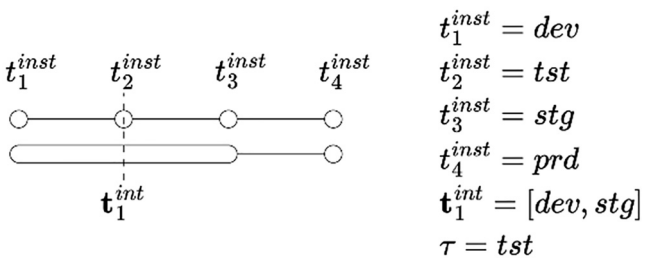

Fig. 1. An example abstract timeline that shows a software development process which consists of development, test, stage, and production time instances, where $\tau \prec t_2^{inst}$ and $\tau \prec \mathbf{t}_1^{int}$

$$\mathbf{U}^T \subseteq \{(d, m, t) \mid d \in \mathbf{D}, m : \mathbf{S} \to \{0, 1\}, t \subseteq \mathbf{T}^{int}\}, \quad (8)$$

where $\mathbf{D}$ is the set of *non-null* and *unique* textual descriptions; $\mathbf{S}$ is the set of learners who have completed the test-unit; m is a function that gives the information if the learner s_j has mastered the related knowledge-unit or not (dichotomy). The assessment item is shown to the learner who interacts with the system to provide information about her knowledge state; $\mathbf{T}^{int}$ is the set of time intervals as described in (7).

$\boldsymbol{U}^M$ is a set of material-units which are holding detailed notes about the knowledge-units. A material-unit is represented by a set of three attributes:

$$\mathbf{U}^M \subseteq \{(d, n, t) \mid d \in \mathbf{D}, n \in \mathbf{I}, t \subseteq \mathbf{T}^{int}\}, \quad (9)$$

where $\mathbf{D}$ is the set of textual descriptions. A material description d is *non-null* and *unique; n* is a detailed note about a specific knowledge-unit; $\mathbf{I}$ is the set of all information-material available in the domain in the form of notes. One note can be in many different formats e.g., a hypertext or a multimedia content, etc.; $\mathbf{T}^{int}$ is the set of time intervals as described in (7).

$\mathbf{R}$ is a set of directed edges, representing the relations between the units:

$$\mathbf{R} = \{\mathbf{R}^K, \mathbf{R}^T, \mathbf{R}^M\}, \quad (10)$$

where $\mathbf{R}^K \subseteq \mathbf{U}^K \times \mathbf{L}^K \times \mathbf{U}^K$ is a set of ordered triples $[u^K_{i,t}, l^K_t, u^K_{j,t}]$ in which $u^K_{i,t}$ is a knowledge-unit that is related to another knowledge-unit $u^K_{j,t}$ and l^K_t is a relation label. The direction of the relation edge is pointing to $u^K_{j,t}$. The relation type between the knowledge-units is determined by the label $l^K_t \in \mathbf{L}^K$. Between knowledge-units there are two relation types exist in the proposed model: $\mathbf{L}^K \subseteq \{PrerequisiteOf_t, IsA_t\}$. In case of $PrerequisiteOf_t$ relation the knowledge represented by $u^K_{i,t}$ needs to be acquired before $u^K_{j,t}$. In other words, $u^K_{j,t}$ depends on $u^K_{i,t}$. In case of IsA_t relation the knowledge represented by $u^K_{i,t}$ is generalized by $u^K_{j,t}$.

$\boldsymbol{R}^T \subseteq \boldsymbol{U}^T \times \boldsymbol{L}^T \times \boldsymbol{U}^K$ is a set of ordered triples $[u^T_{i,t}, l^T_t, u^K_{j,t}]$, where $u^T_{i,t}$ is a test-unit that belongs to one certain knowledge-unit $u^K_{j,t}$. The relation type between the knowledge-unit and the test-unit is determined by the label $l^T_t \in \boldsymbol{L}^T$, where $\boldsymbol{L}^T \subseteq \{Tests_t\}$ representing that the actual test-unit $u^T_{i,t}$ is testing the mastery level of learner in terms of the connected knowledge-unit $u^K_{j,t}$. The edge is pointing towards the knowledge-unit $u^K_{j,t}$.

$\mathbf{R}^M \subseteq \mathbf{U}^M \times \mathbf{L}^M \times \mathbf{U}^K$ is a set of ordered triples $[u^M_{i,t}, l^M_t, u^K_{j,t}]$, where $u^M_{i,t}$ is the material-unit that belongs to one certain knowledge-unit $u^K_{j,t}$. The relation type between the knowledge-unit and the material-unit is determined by the label $l^M_t \in \mathbf{L}^M$, where $\mathbf{L}^M \subseteq \{MaterialOf_t\}$ representing that the material-unit $u^M_{i,t}$ is a detailed note that explains and teaches the connected knowledge-unit $u^K_{j,t}$ to the learner. The edge is pointing towards the knowledge-unit $u^K_{j,t}$. Figure 2 is visualizing an example graph where the elements of the EKSG model can be observed.

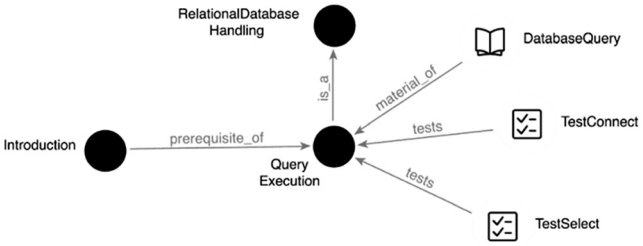

Fig. 2. An example graph from the databases knowledge domain, that shows the *QueryExecution* knowledge-unit which has *prerequisite_of* relation to the *Introduction* knowledge-unit as well as *is_a* relation to the more abstract *RelationalDatabaseHandling* knowledge-unit. The graph also shows two test-units and one material-unit connected to *QueryExecution*

Remark 1: If the given time interval is empty the default value steps in, which is $\boldsymbol{t}^{int}_{*}$, meaning the model element (node or relation) is valid in all time instances. In case all time restrictions of the model elements are empty the model reduces to a timeless graph. The fact that time intervals t may be empty gives the EKSG model great flexibility.

Definition 5: There are several ways to process an evolving, time dependent graph. Here the so-called *snapshot* solution is described using discrete time-dependent units and time independent relations. Given the snapshot $\mathbf{U}^K_\tau = \{u \in \mathbf{U}^K \mid \tau \prec u.t\}$ (where $u.t$ means: attribute t of object u) contains all knowledge-units in $\mathbf{G}$ at time τ. The same logic can be applied to material-units $\mathbf{U}^M_\tau = \{u \in \mathbf{U}^M \mid \tau \prec u.t\}$ and test-units $\mathbf{U}^T_\tau = \{u \in \mathbf{U}^T \mid \tau \prec u.t\}$. These representations are constituting the τ th instance of $\mathbf{G}$ graph therefore these instances can be called snapshots. A snapshot of $\mathbf{G}$ at τ is denoted by: $\mathbf{G}_\tau = \{\mathbf{U}^K_\tau, \mathbf{U}^M_\tau, \mathbf{U}^T_\tau, \mathbf{R}\}$.

Consequently, the described EKSG model can be considered as a sequence of static knowledge graphs $\mathbf{G} = \{\mathbf{G_1}, \mathbf{G}_2, \cdots, \mathbf{G}_\tau, \cdots, \mathbf{G}_k\}$.

4. EXPERIMENTS AND ANALYSIS

The proposed EKSG model is evaluated by implementing 31 knowledge-units from the Python programming language domain. During the evaluation it was examined whether the knowledge structure of the selected excerpt from the Python documentation can be represented by the given model. The experiment resulted a fine tuning in the model as the *is_a* relation was added to the model at this point. As an example the Python *str.removeprefix()* and *str.removesuffix()* methods are presented in Fig. 3. These methods were introduced in

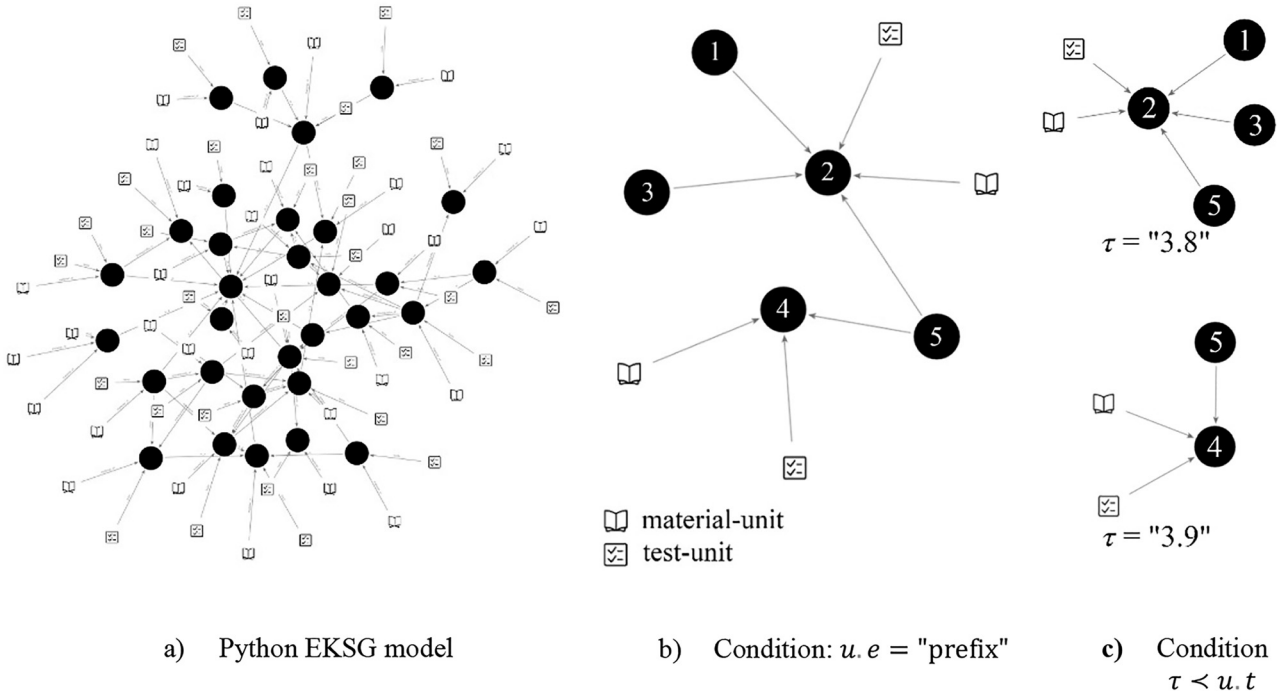

Fig. 3. Different knowledge structures in Python version 3.8 and 3.9 for removing prefix and suffix from a string. Black numbered nodes are knowledge-units as follows: 1: *NumberOfItems,* 2: *PrefixSuffixHandling,* 3: *Condition,* 4: *RemovePrefixSuffixFunction,* 5: *StringDataType*

Python version 3.9. After these methods became available the programmer only needs to know what a method is, and how to call it to achieve the expected result. According to (7) the related knowledge-unit since Python version 3.9 looks like (11), before Python version 3.9 looks like (12):

$$u_i = \left\{ \begin{array}{c} n=''\text{RemovePrefixSuffixFunction}'', \\ \text{d} = ''\text{Removing specified text from prefix or suffix…}'', \\ \text{t} = [''3.9'', ''3.10'', ''3.11'', ''3.12''], \\ \text{e} = [''\text{prefix}'', ''\text{suffix}'', ''\text{removeprefix}'', ''\text{removesuffix}''] \end{array} \right\}, \tag{11}$$

$$u_j = \left\{ \begin{array}{c} n=''\textit{PrefixSuffixHandling}'', \\ \text{d} = ''\text{You need to implement your own removal…}'', \\ \text{t} = [''3.6'', ''3.7'', ''3.8''], \\ \text{e} = [''\text{prefix}'', ''\text{suffix}'', ''\text{startswith}'', ''\text{endswith}''] \end{array} \right\}, \tag{12}$$

The knowledge that is needed to complete the *remove prefix or suffix* operation is different in version 3.8 and 3.9. The structural difference can be seen in Fig. 3c.

According to the domain expert before version 3.9 the following prerequisites existed for the resulted *PrefixSuffixHandling* knowledge unit: *NumberOfItems, Condition, StringDataType*. Since version 3.9, the *RemovePrefixSuffixFunction* has only one prerequisite relation: *StringDataType.*

In the implemented data model with a simple cypher query the required knowledge structure can be requested, *Code 1* is showing the query.

Code 1: Cypher query of determining the knowledge structure in version 3.8 for prefix and suffix handling.

```
MATCH (n) – [r] → (p:KnowledgeUnit)
WHERE ANY (evoker IN p.evokers WHERE evoker IN ["prefix", "suffix"])
AND ANY (version IN p.timeInstances WHERE version = "" OR version = "3.8" )
RETURN n, r, p
```

5. CONCLUSION

Expectedly, in the next decade ITS systems will be part of everyday life. In these systems it will be increasingly important to represent evolving knowledge while maintaining the possibilities of personalized teaching. The presented EKSG model provides enough flexibility to represent a regularly developing knowledge domain.

The presented EKSG model unites the advantages of CbKST and ontologies by realizing a knowledge graph that utilizes the prerequisite relation concept from CbKST and real-life entities (knowledge-units) from ontologies. With introducing the abstract time concept to the system, EKSG is able to model fast evolving knowledge, e.g., computer programming languages. Adding evoking-hooks to the system EKSG can realize connections between knowledge-units and existing external knowledge forms, e.g., pieces of program source code.

The knowledge-units within the EKSG model must be atomic to build correct prerequisite relations, which is crucial to define personalized learning paths. This study will be continued with the aim of defining knowledge-unit atomicity as well as how learning paths can be efficiently predicted using assessment results and pedagogical techniques.

REFERENCES

[1] L. Csépányi-Fürjes, "Controllable and explainable AI framework in the automatic assessment domain," in *Abstract book of 12th International Conference on Applied Informatics*, Eger, Hungary, May 2–4, 2023, pp. 1–3.

[2] A. Ghouse and C. Sipos, "RPA progression throughout years and futuristic aspects of RPA," *Pollack Period.*, vol. 17, no. 1, pp. 30–35, 2022.

[3] D. Albert,C. Hockemeyer,M. D. Kickmeier-Rust,A. Nussbaumer, and C. M. Steiner, "E-learning based on metadata, ontologies and competence-based knowledge space theory," in *Third Knowledge Technology Week*, D. Lukose, A. R. Ahmad, and A. Suliman, Eds, Kajang, Malaysia, July 18–22, 2011, *Communications in Computer and Information Science*, vol. 295, 2012, pp. 24–36.

[4] Q. Zhang, Q. Xie, and G. Wang, "A survey on rough set theory and its application," *CAAI Trans. Intelligence Technol.*, vol. 1, no. 4, pp. 323–333, 2016.

[5] E. Cosyn, H. Uzun, C. Doble, and J. Matayoshi, "A practical perspective on knowledge space theory: ALEKS and its data," *J. Math. Psychol.*, vol. 101, 2021, Art no. 102512.

[6] J. P. Doignon and J. C. Falmagne, "Spaces for the assessment of knowledge," *Int. J. Man-Machine Stud.*, vol. 23, no. 2, pp. 175–196, 1985.

[7] D. de Chiusole, L. Stefanutti, P. Anselmi, and E. Robusto, "Stat-Knowlab. assessment and learning of statistics with competence-based knowledge space theory," *Int. J. Artif. Intelligence Educ.*, vol. 30, pp. 668–700, 2020.

[8] D. Kanellopoulos, S. Kotsiantis, and P. Pintelas, "Ontology-based learning applications: A development methodology,", in *Proceedings of the IASTED International Conference on Software Engineering*, Innsbruck, Austria, February 14–16, 2006, pp. 27–32.

[9] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[10] C. Pierrakeas, G. Solomou, and A. Kameas, "An ontology-based approach in learning programming languages,", in *Proceedings of the 2012 16th Panhellenic Conference on Informatics*, Piraeus, Greece, October 5–7, 2012, pp. 393–398.

[11] E. Nyitrai and B. Varga, "Ontology-based higher educational information systems," *Pollack Period.*, vol. 7, no. 3, pp. 139–148, 2012.

[12] W. T. Sewunetie, G. Hussein, A. Ahmed, and L. Kovács, "The development and analysis of extended architecture model for intelligent tutoring systems," *Gradus*, vol. 6, no. 4, pp. 128–138, 2019.

[13] M. Koppen amd and J. P. Doignong, "How to build a knowledge space by querying an expert," *J. Math. Psychol.*, vol. 34, no. 3, pp. 311–331, 1990.

[14] J. C. Falmagne, E. Cosyn, J. P. Doignon, and N. Thiery, "The assessment of knowledge, in theory and in practice,", in *Proceedings of the 4th International Conference on Formal Concept Analysis*, Dresden, Germany, Feburary 13–17, 2006 pp. 61–79.

[15] D. Albert and J. Lukas, Eds, *Knowledge Spaces: Theories, Empirical Research, and Applications*, Lawrence Erlbaum Associates Publishers, 1999.

[16] J. C. Falmagne and J. P. Doignon, "A Markovian procedure for assessing the state of a system," *J. Math. Psychol.*, vol. 32, no. 3, pp. 232–258, 1988.

[17] Z. Tóth, "Examining knowledge structure and knowledge organization based on the knowledge space theory" (in Hungarian), *Magyar Pedagógia*, vol. 105, no. 1, pp. 59–82, 2005.

[18] L. Stefanutti, P. Anselmi, D. de Chiusole, and A. Spoto, "On the polytomous generalization of knowledge space theory," *J. Math. Psychol.*, vol. 94, 2020, Art no. 102306.

[19] X. Xie, W. Xu, and J. Li, "A novel concept-cognitive learning method: A perspective from competences," *Knowledge-Based Syst.*, vol. 265, 2023, Art no. 110382.

[20] Time Ontology in OWL, W3C Candidate Recommendation, Draft 15 November 2022. [Online]. Available: https://www.w3.org/TR/owl-time/. Accessed: Dec. 10, 2023.

[21] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," *Data Sci. Eng.*, vol. 4, pp. 352–366, 2019.

[22] J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, C. R. Johnson, and J. Scheffczyk. 2016. FrameNet II: extended theory and practice. [Online]. Available: https://framenet.icsi.berkeley.edu/fndrupal/index.php?q=the_book. Accessed: Dec. 10, 2023.