Extended Knowledge Graphs: A Conceptual Study*

Weronika T. Adrian[a], Marek Adrian[b], Krzysztof Kluza[c], Bernadetta Stachura-Terlecka[d] and Antoni Ligęza[e]
AGH University of Science and Technology, al. A.Mickiewicza 30, 30-059 Krakow, Poland

Keywords: Knowledge Graphs, Semantic Networks, Knowledge Representation, KRR, Rules, Ontologies, Reasoning, Logic Programming, Answer Set Programming, Prolog.

Abstract: The amount and variety of data that we produce every day pose a constant challenge for meaningful information processing. While *knowledge graphs* have gained a considerable attention in the recent years, due to their flexible and universal knowledge representation, they lack the mechanisms facilitating knowledge processing. In this paper, we propose an information system called *extended knowledge graphs (EKG)* that augments the concept of knowledge graphs with procedural attachments. We put forward the requirements and assumption of the EKG structure and present a categorisation of supported reasoning tasks.

1 INTRODUCTION

The amount of data produced and stored every day may be daunting for who wants to process it efficiently. The variety of data adds an additional difficulty layer for a meaningful analysis of the information encoded within. In fact, it seems that knowledge representation methods do not keep up with the ever-growing mass of data to process, leaving us with sub-optimal solutions, either in terms of possible tasks to perform or the amount of data that can be processed.

In the recent years, a considerable attention has been gained by so-called *knowledge graphs* (Hogan et al., 2020; Fensel et al., 2020; Ji et al., 2020) – graph-based knowledge representation systems that can organize information in a flexible and intuitive way. Although they are used in both academic (Paulheim, 2017) and business (Noy et al., 2019) communications, there is no commonly agreed upon definition of them (Ehrlinger and Wöß, 2016). In fact, some parties see them as an reincarnation of *ontologies* (Uschold et al., 1996; Staab and Studer, 2010; Ławrynowicz, 2017), other define them as *knowledge-based systems* (KBS) (Akerkar and Sajja, 2009; Ahmed et al., 2019) that consist of a knowledge base and a reasoner, or finally — a KBS with a knowledge integration component. In this last definition, a knowledge graph should integrate information from different sources into an ontology and provide means to reason over the integrated knowledge.

Regardless of the adopted definition, knowledge graphs still deal with only *static* knowledge representation, i.e., describing the properties of objects and classes (sets of objects) and relations between them. In practical use cases, one often is interested in associating also some dynamic *procedures* to static objects or classes. For instance, an editor in a journal would like to "process" a submitted paper which requiress, i.a., finding a suitable person to review the article. To this end, a candidate reviewer should be familiar with the topics considered in the paper, he or she should not have a conflict of interests with the paper's author (that e.g. may mean they should not be co-authors or work in the same department) etc. While theoretically we already have access to all the needed information e.g., in the ORCID and DBLP databases or Microsoft Academic Graph that describe the scientists and their work in a graph-like structure, it is not easy to solve such a problem automatically. To address such problems, we propose to extend the knowledge graph representation with procedural attachments, such that a procedure (understood as a process that contains some decisions) can be associated with a specific type of objects defined in the graph.

In the following sections, we present *extended knowledge graphs* (EKG) to encode both structural

[a] https://orcid.org/0000-0002-1860-6989
[b] https://orcid.org/0000-0002-0435-0994
[c] https://orcid.org/0000-0003-1876-9603
[d] https://orcid.org/0000-0003-2887-5936
[e] https://orcid.org/0000-0002-6573-4246
*This Paper Is Supported by AGH UST Grant.

Adrian, W., Adrian, M., Kluza, K., Stachura-Terlecka, B. and Ligęza, A.
Extended Knowledge Graphs: A Conceptual Study.
DOI: 10.5220/0010111601730180
In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020) - Volume 2: KEOD*, pages 173-180
ISBN: 978-989-758-474-9

(static) and procedural (dynamic) knowledge. We outline its rationale and main objectives, discuss main features and components and illustrate its use with exemplary use cases. We argue that a procedural extension to KG representation can be both beneficial for practical use cases and plausible to implement using existing logical reasoners.

The contributions of the paper are threefold:

- *Knowledge Representation*: we discuss a new language for representing both static graph-based knowledge and dynamic procedures;
- *Typology of Operations*: we consider different types of problems that extended knowledge graphs should address and organize them into groups
- *Applications*: we illustrate the use of our system on practical examples that describe academic community in terms of publications and collaboration.

2 BACKGROUND AND MOTIVATION

Knowledge Representation (KR) methods can be divided into descriptive (or declarative) and procedural (or imperative) ones (Torsun, 1995). The descriptive methods include tables, formulas or graphs and they are characterised by storing information explicitly. On the contrary, knowledge expressed in procedural manner is manifested in the actual execution of the program and is therefore stored implicitly. The declarative methods are easier to expand and modify. However, in various application, intelligent systems may need both kinds of representation.

"A semantic network or net is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs." (Sowa, 1987). Semantic nets have long been used in philosophy, psychology and linguistic before the computer implementation of them were developed. They have represented knowledge or supported automated reasoning. A declarative graphic representation characterizes semantic networks. There exist various sorts of semantic networks, designed for different purposes and of different levels of formalisation. In (Sowa, 1987), the following classification of semantic nets was introduced:

1. *Definitional Networks*. In these networks the *subtype* or *is-a* relation between a concept type and a newly defined subtype is emphasised. Their basic mechanisms are "as old as Aristotle", who worked on a classification of nature. Definitorial networks support the rule of inheritance and, usually, monotonic reasoning, in which inferred knowledge is added, but what is once known does not change.
2. *Assertional Networks* are designed to assert propositions. Some of the assertional (propositional) networks have been used to graphically represent structures underlying natural language semantics. C. S. Peirce introduced *relational graphs* and *existential graphs*, by means of which he intended to show "the atoms and molecules of logic". Propositional networks include also Tesnière's *dependency graphs* used to represent natural language sentences. A variety of semantic propositional networks called *Conceptual Graphs* have been developed by J. F. Sowa. *Core* conceptual graphs are successors of Peirce's existential graphs while *extended* and *research* conceptual graphs explored novel techniques for reasoning, knowledge representation and natural language semantics. For core and extended CG the semantics is defined by a mapping to ISO *Common Logic* standard using subset of First Order Logic.
3. *Implicational Networks*. These structures use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs (*belief networks*, *Bayesian networks*), causality (*causal networks*), or inferences (*truth-maintenance systems*). To one graph different interpretations and reasoning may apply. Both logical, forward- and backward-chaining, as well as probabilistic reasoning may be used.
4. *Executable Networks*. These networks contain mechanisms which can perform inferences, pass messages, or search for patterns and associations and cause some changes to the network itself. These mechanisms, which distinguish the networks from other, static ones include marker passing, attached procedures and graph transformations. Dataflow graphs and Petri Nets, although rarely classified as semantic networks, use similar techniques as procedural semantic networks.
5. *Learning networks* build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs (*rote memory*) or by modifying numerical values, called weights, associated with the nodes and arcs. Rote memory techniques find an application in case-base reasoning systems, whereas changing weights is often used in artificial neural nets.
6. *Hybrid networks* combine two or more of the previous techniques, either in a single network or in

separate, but closely interacting networks.

The approach to knowledge representation based on describing the world of interest in terms of objects enables to introduce hierarchy and structure into a knowledge base. Sentences or rules can be grouped by the concepts (objects) they refer to. One of the procedural knowledge representation formalism which is object-oriented is the one of frames, first proposed by Marvin Minsky in 1975. In a basic version of the formalism, there are two types of frames: *individual frames* or *instance-frames* for representing single objects and *general frames* or *class-frames* for representing classes or categories of objects. Frames have *attributes* or *slots*, which are filled by *fillers*. Two special distinguished slots were defined, *INSTANCE-OF* for individual frames and *IS-A* for general frames. By means of them one could specify that given object is an *instance* of a class or introduce hierarchy between two generic classes. The relations these slots specify are called *generalisation*. Other relations between frames include *aggregation*, in which several frames representing components are associated with another one, which represents a *whole* and association, which denotes other semantic relationship. Frame slots could also have *attached procedures*. Two basic sorts of such procedures are *IF-ADDED* and *IF-NEEDED* (Brachman et al., 2004) (or according to (Negnevitsky, 2005): *WHEN-CHANGED* and *WHEN-NEEDED*). The procedures are executed if a frame is added or a new value is placed in a slot or if some computation is needed to determine a slot value. In (Negnevitsky, 2005), a distinction is drawn between *methods* and *demons* which were both types of attached procedures. Demons usually are in a form of an IF-THEN rule and when the attribute specified in IF statement changes its value, then the consequent of the rule is fired. The term method is sometimes used as a synonym, but it usually denotes more complex procedure consisting of several actions.

Reasoning with frames mainly consists in discovering the relations between frames and creating individual instances of generic frames. It involves filling some slots with values and inferring other values. Of a particular significance are *INSTANCE-OF* and *IS-A* slots. They enable for instantiating individuals and triggering procedures, which were not stated explicitly in an individual frame. The process of "passing information from generic frames down through their specializations and eventually to their instances" (Brachman et al., 2004) is called *inheritance* and it is present both in frame formalism and in object-oriented programming. In fact, inheritance is an essential feature of frame-based systems. Frames were developed concurrently with object-oriented programming languages and share some of their intuition. According to (Negnevitsky, 2005), "frames are an application of object-oriented programming for expert systems". Although frames were applied in several fields (Brachman et al., 2004), they lack the reasoning support and are too inexpressive for many problems.

More advanced formalisms based on an idea to represent knowledge in terms of concepts and object are Description Logics (DL) (Baader et al., 2003). They enable for constructing complex *descriptions* which denote the classes of individuals. As opposed to the frame languages, Description Logics have formal, First Order Logic semantics. Therefore, they extend frames' capabilities and provide intrinsic reasoning support. Description Logics became the logical foundations of the Semantic Web ontology languages, and respective OWL subsets have their logical counterparts in DL languages. It is noteworthy that the catalogue of reasoning tasks in Description Logics is limited to several problems, such as *classification*, *consistency checking*, *realization* or *retrieval*.

3 RELATED WORK

Graph-based knowledge representation has been successfully used in almost every branch of Artificial Intelligence (Sowa, 1992). The newest incarnation of it are *knowledge graphs*. As a term, *Knowledge Graph* was "re-discovered" and popularized by Google in their communication about search based on objects, not words (Singhal, 2012). Since then, the term has been widely adopted in the Semantic Web community and beyond.

Basic notions of the knowledge graphs have been further extended in several directions. Yahya et al. considers extended knowledge graphs as graphs that combine relational facts with textual web contents (Yahya et al., 2016a; Yahya et al., 2016b). Zhang et al. (Zhang et al., 2020) extend knowledge graphs in order to integrate heterogeneous data resources and enhance knowledge-based services for health care systems. Thus, their Health Knowledge Graph Builder may be used to construct disease-specific and extensible health knowledge graphs from multiple sources. Another interpretation of *extending* knowledge graphs has been put forward in (Yoo and Jeong, 2020) in which the authors proposed a method for auto-extending knowledge graphs.

Combining static graph-based knowledge with a description of dynamics of a system is a recurring challenge. Depending on the domain and application, it appeared thorough years in a form of simple as-

sert/retract operations on conceptual graphs (Mineau, 1998), a range of proposals for integrating rules and ontologies (Eiter et al., 2008), associative graph data structures (Horzyk, 2013), and recently in complex processes integrated with "full-fledged data models" (van der Aalst, 2019). Specific ways to model the dynamics are *processes* and *rules*.

Business Process Model and Notation (BPMN) (OMG, 2011) is a notation created by OMG (Object Management Group) to graphically represent business processes using diagrams. It is now a well established standard not only for modeling, but also for managing process knowledge of organizations. Recently, the Decision Model and Notation (DMN) standard (OMG, 2014), also created by OMG, has been proposed. It works as a complementary notation for describing rules and decisions in organizations. DMN is especially useful for modeling decision logic for processes. As both processes and decisions constitute executable specifications (Janssens et al., 2016), it is possible to use process and decision engine for deployment and enactment of such models.

Integrating logic based on rules with static semantic representation of objects is another way to integrate declarative and procedural knowledge, both on the theoretical (Eiter et al., 2008; Nalepa and Furmańska, 2010a) and practical (Nalepa and Furmańska, 2010b) level. Embedding a rule engine into a semantic system (Adrian et al., 2011) allows to practically model and test such an integration.

This paper is a communication of a work-in-progress research on defining *yet another knowledge representation system* that would meet the challenges of processing large graph-based datasets with different types of data transformations, outlined and categorized in the following sections.

4 EXTENDED KNOWLEDGE GRAPHS

In this section, we put forward our rationale and main assumptions of *extended knowledge graphs* (EKG). We introduce the language syntax and semantics and outline the architecture of the systems adopting the language.

4.1 Main Assumptions

In the era of interconnected information systems, we recognize the value of graph-based knowledge representation and its visualization (Dudycz, 2017). It is both flexible and universal and so we assume that the data model rely on *graphs*. Moreover, we assume that it is convenient to distinguish different levels of abstraction in a knowledge based system. Thus, we assume that the following layers should be supported:

1. graph-based knowledge representation,
2. logic layer that allows for different kinds of data transformations,
3. visualization layer for clear knowledge summary.

While the layered architecture can be already observed in the Semantic Web in the form of the famous *semantic stack*, the intention of extended knowledge graphs is to encode also *dynamic* knowledge in the "logic layer" that could be related to Semantic Web's ontologies and rules. With use of rules and procedures, we would like to be able to design and implement a *dynamic* knowledge based system, that should be able to modify the knowledge if needed, rather than only query the static part of it.

Another important assumption is that we would like to be able to define *different kinds of objects* in EKG. The objects' classes (or categories) would determine what is possible to do upon the considered object.

Finally, we intend to express a range of tasks and problems to solve with our system. In particular, we would like to allow for tasks beyond simple querying (SELECT-like operations), such as information retrieval, path-finding, forward-chain and backward-chain reasoning, hard and weak constraints (optimization). To this end, we consider appropriate operators and envision the transformations they would induce.

4.2 Knowledge Representation in EKG

In principle, we propose to identify three main types of objects:

1. **Agents:** "active objects" that represent persons in the system that can *initiate* some actions or operations over the static objects, and who can have particular *permissions* over certain classes of objects and *roles* in the system, defined by these permissions;
2. **Objects:** "passive objects" that can be acted upon – they have *methods*, as in object-oriented programming, that define the operations in which they can participate
3. **Transformations:** "functional objects" that define the operations to be performed with active and passive objects; they are defined by the required *input* and expected *output*, a *trigger* that initiates them and the *task definition* that can be expressed in multiple forms such as a process, a set of rules and constraints or a process.

The transformations can be triggered by events. It is possible to distinguish different types and subtypes of them (the following specification of events is inspired by BPMN event types (OMG, 2011)):

- a *message event* indicates that a message is received,
- a *timer event* (Rademakers and van Liempd, 2012) usually provides some time specification which may denote:
  - a *specific time and date*, e.g in the format ISO 8601, when to trigger the event,
  - a *time duration* which sets some period of time that must elapse before the event is triggered, e.g. "for 3 days",
  - a *time cycle* which sets time intervals to trigger an event periodically, e.g. 5 repetitions, each for an hour,
- a *conditional* event which checks if a certain condition is true,
- a *signal event* which catches a global signal delivered to the process,
- an *exception event*, which in some cases may be realized by the abovementioned events, but usually is implemented as:
  - an *error event* in the case of occurring of a potential error,
  - an escalation event to escalate an exception from the sub-processes to the parent process,
  - a cancel event, used in the context of the transactions and invokes compensation events to compensate the specific tasks.

Agents, Objects and Transformations can be connected with each other with links that represent *relations* between them.

To put it into a slightly more formal perspective, we define an agent as:

$$A = \langle id, ATT, ROL, PRC \rangle$$

where:

- id is a string identifier
- ATT is a set of *attributes* such as for example a name, a surname, age, nationality etc.
- ROL is a set of *permissions* over the Objects
- PRC is a set of *procedures* the agent can initiate

We define a (passive) object as:

$$O = \langle id, ATT, PRC \rangle$$

where:

- id is a string identifier
- ATT is a set of *attributes*
- PRC is a set of *procedures* that can be performed *over* (using) the object

and a Transformation objects as:

$$T = \langle id, tr, ts, OBJ \rangle$$

where:

- id is a string identifier
- trg is a type of trigger that initiate the transformation
- tsk is a definition of the "task" to be executed
- OBJ is a set of objects that are transformed

The *EKG* structure is partially based on the entity network introduced in (Adrian and Manna, 2018) and is a tuple

$$Y = \langle C, I, P, R, rel, proc \rangle$$

where:

- C is a set of concept names, denoting both agents' and objects' classes
- I is a set of object instances
- P is a set of procedures (transformations)
- R is a set of relation names
- $rel : (I \cup C \cup P) \times (I \cup C) \rightarrow 2^R$ is a function that associated the relation names to pairs of entities in the network
- $proc : C \rightarrow 2^P$ is a function that associate procedures with classes of objects

5 REASONING WITH EKG

The language of the logic layer should cover different forms of rules, processes as well as hard and weak constraints. In fact, we envision that several different "levels" of tasks could be considered, from simple database-like operations, through more sophisticated logical reasoning up to full-fledged data analysis. To systematically analyze and define these operations in EKG, we propose the following categories of problems:

Level 0: CRUD (Create / Read / Update / Delete) operations – these are basic transformations of the underlying graph database

Level 1: SELECT operations, aggregation, creating views – these are the transformations available in classical relational databases. They "return" portions of data present in the knowledge graph or the results of simple operations such as sum or average of data.

Level 2: Deduction – these are operations that return data *inferred* from those present in the knowledge base via logical rules, possibly with use of recursion, chaining, transitive closure etc.

Level 3: Finding *similar* objects to the ones given – these are even higher level tasks that include an analysis of the given examples (with use of selected metrics of similarity), followed by search mechanisms to discover relevant objects in the knowledge base

Level 4: Induction – these are the operations that perform generalizations from given examples to more general assumptions about classes of objects; this level encompasses optimization problems based on data analysis, clustering and comparison.

Level 5: Merge – integration of knowledge from several graphs, understood not as a simple sum, but taking into consideration consistency analysis, induction and concept alignment

When it comes to the implementation perspectives, an EKG system is intended to work as a framework, with configurable elements: reasoners, knowledge graph databases and visualization component(s). As for the database layer, we consider both RDF standard based on triple representation and URIs for data identification, and *labeled property graph* paradigm in which both edges and vertices are represented with key-value structures. While the former is the base of Semantic Web standards and supported with different *triple stores*, the latter has been adopted in widely used graph databases such as Neo4j.

As far as reasoning layer is concerned, there are many reasoners that could realize particular tasks outlined above, such as Prolog implementations for deduction, constraint satisfaction problems and goal-driven inference, ASP engines (Adrian et al., 2018) for various graph problems, data-driven reasoning as well as constraint and optimization problems, ontology reasoners, process mining tools etc.

6 APPLICATIONS

The extended knowledge graphs could be used in various scenarios that include processes operating over a networked knowledge. In this section, we demonstrate an exemplary application of EKG over knowledge about researchers and their works. Let us consider a part of an *extended knowledge graph* designed for a simplified science database 1. We have three researchers $R1, R2, R3$, five topics/keywords $T1, T2, T3, T4, T5$ and seven papers ($P1 - P7$). In this knowledge graph, topics/keywords can only be connected to papers and so are the researchers, but researchers are also agents so they can initiate procedures. In the following sections, we present possible operations over this piece of knowledge.

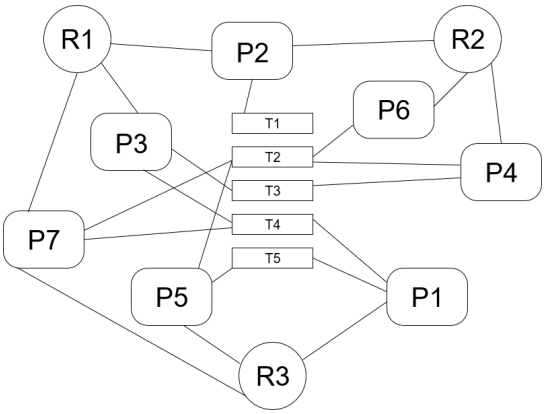


Figure 1: Exemplary extended knowledge graph about researchers and their work.

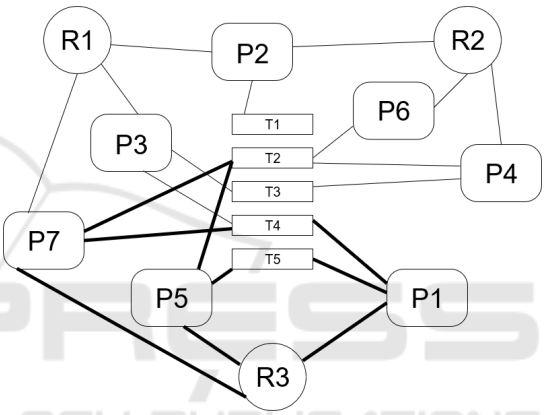


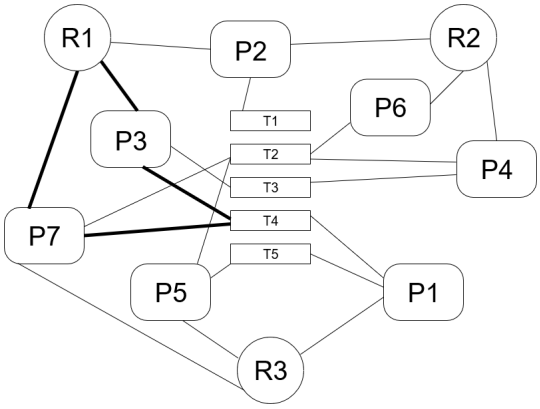
Figure 2: Research interests of $R3$.

6.1 Exemplary Use Cases

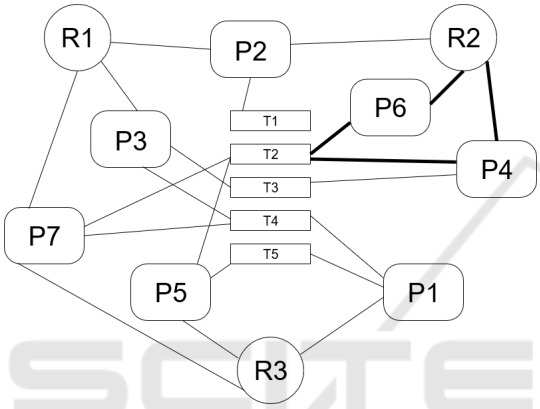
The most basic procedure would be recognizing in what topic a given researcher has a strong interest. In our case, we would define a strong interest as having written at least 2 papers related to a topic/keyword. So, $R3$ has a strong interest in $T2, T4, T5$ (see Fig. 2), $R1$ in $T4$, but has also written papers on $T1$, $T2$ and $T3$ (see Fig. 3(a)) and $R2$ has a strong interest in $T2$, but has also written on $T1$ and $T3$ (see Fig. 3(b)).

Another obvious procedure would be finding collaborators of a given researcher, namely researchers that have at least one co-authored paper with a given researcher. For instance, $R1$ is a collaborator of $R3$, but $R2$ is not.

We can broaden this operation to an "Erdös



(a) Research interests of $R1$.



(b) Research interests of $R2$.

Figure 3: Research interests recognition.

number"-style concept for any researcher. To recall, we define Erdös number recursively: if a researcher has a paper with Erdös, their Erdös number is 1; if the researcher has a paper with someone whose Erdös number is 1, but no paper with Erdös, their Erdös number is 2, and so on. In our case, the $R3$ number for $R1$ is 1 and is 2 for $R2$.

Finally, we can put some more sophisticated deduction using this knowledge representation. Consider a researcher who looks for people with similar research interests, but not too far away form him with respect to collaboration. To this end, we would look for all the researchers with at least one common strong interest and the "researcher number" within a given small boundary (for example 2). In our case, the potential collaborators of $R3$ are both $R1$ and $R2$. What is interesting is that while connection with $R1$ could be returned via simple SELECT operation, the connection between $R2$ and $R3$ is not that obvious, seeing that $R1$ and $R2$ have no common strong interest. However, $R2$ has a common strong interest with $R3$ and the $R3$-number of 2 as seen in Figure 4 which renders them a valid candidate.

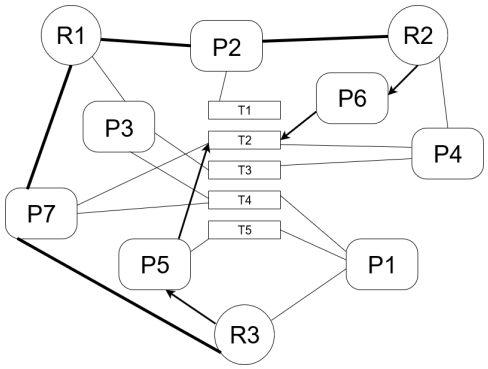


Figure 4: Looking for potential collaborators.

7 CONCLUSION

We have presented a concept of extending knowledge graphs with procedural attachments under a common knowledge representation. We have introduced *extended knowledge graphs* — a knowledge representation and reasoning system that integrates static knowledge from existing knowledge sources and can perform logical reasoning in forward and backward chaining mode.

For future work, we plan to fully implement the idea, define new use cases and problems over existing knowledge graphs and evaluate empirically the efficiency of the system. Moreover, we will continue to add new tasks to be possible to define and execute with the system.

REFERENCES

Adrian, W. T., Alviano, M., Calimeri, F., Cuteri, B., Dodaro, C., Faber, W., Fuscà, D., Leone, N., Manna, M., Perri, S., et al. (2018). The asp system dlv: advancements and applications. *KI-Künstliche Intelligenz*, 32(2-3):177–179.

Adrian, W. T., Bobek, S., Nalepa, G. J., Kaczor, K., and Kluza, K. (2011). How to reason by heart in a semantic knowledge-based wiki. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 438–441. IEEE.

Adrian, W. T. and Manna, M. (2018). Navigating online semantic resources for entity set expansion. In *Proc. of PADL'18*, pages 170–185.

Ahmed, A., Al-Masri, N., Abu Sultan, Y. S., Akkila, A. N., Almasri, A., Mahmoud, A. Y., Zaqout, I. S., and Abu-Naser, S. S. (2019). Knowledge-based systems survey.

Akerkar, R. and Sajja, P. (2009). *Knowledge-based systems*. Jones & Bartlett Publishers.

Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al. (2003). *The description*

logic handbook: Theory, implementation and applications. Cambridge university press.

Brachman, R., Levesque, H., and Pagnucco, M. (2004). *Knowledge Representation and Reasoning*. Morgan Kaufmann.

Dudycz, H. (2017). Application of semantic network visualization as a managerial support instrument in financial analyses. *Online Journal of Applied Knowledge Management A Publication of the International Institute for Applied Knowledge Management*, 5(1).

Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48.

Eiter, T., Ianni, G., Krennwallner, T., and Polleres, A. (2008). Rules and ontologies for the semantic web. In *Reasoning web*, pages 1–53. Springer.

Fensel, D., Şimşek, U., Angele, K., Huaman, E., Kärle, E., Panasiuk, O., Toma, I., Umbrich, J., and Wahler, A. (2020). *Knowledge Graphs*. Springer.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J. E. L., Kirrane, S., Neumaier, S., Polleres, A., et al. (2020). Knowledge graphs. *arXiv preprint arXiv:2003.02320*.

Horzyk, A. (2013). Artificial associative systems and associative artificial intelligence. *EXIT, Warsaw*, pages 1–276.

Janssens, L., De Smedt, J., and Vanthienen, J. (2016). Modeling and enacting enterprise decisions. In *International Conference on Advanced Information Systems Engineering*, pages 169–180. Springer.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. (2020). A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*.

Ławrynowicz, A. (2017). *Semantic data mining: an ontology-based approach*, volume 29. IOS Press.

Mineau, G. W. (1998). From actors to processes: The representation of dynamic knowledge using conceptual graphs. In *International Conference on Conceptual Structures*, pages 65–79. Springer.

Nalepa, G. J. and Furmańska, W. T. (2010a). Integration proposal for description logic and attributive logic–towards semantic web rules. In *Transactions on computational collective intelligence II*, pages 1–23. Springer, Berlin, Heidelberg.

Nalepa, G. J. and Furmańska, W. T. (2010b). Pellet-heart–proposal of an architecture for ontology systems with rules. In *Annual Conference on Artificial Intelligence*, pages 143–150. Springer, Berlin, Heidelberg.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems*. Pearson education.

Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., and Taylor, J. (2019). Industry-scale knowledge graphs: Lessons and challenges. *Queue*, 17(2):48–75.

OMG (2011). Business Process Model and Notation (BPMN): Version 2.0 specification. Technical Report formal/2011-01-03, Object Management Group.

OMG (2014). Decision model and notation. beta1. Technical Report dtc/2014-02-01, Object Management Group.

Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.

Rademakers, T. and van Liempd, R. (2012). *Activiti in Action: Executable business processes in BPMN 2.0*. Manning Publications.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings, may 2012. *URL http://googleblog. blogspot. ie/2012/05/introducing-knowledgegraph-things-not. html*.

Sowa, J. (1987). *Encyclopedia of Artificial Intelligence*, chapter Semantic Networks. Wiley, New York. revised and extended for the second edition, 1992.

Sowa, J. F. (1992). Conceptual graphs as a universal knowledge representation. *Computers & Mathematics with Applications*, 23(2-5):75–93.

Staab, S. and Studer, R. (2010). *Handbook on ontologies*. Springer Science & Business Media.

Torsun, I. S. (1995). Foundations of intelligent knowledge-based systems.

Uschold, M., Gruninger, M., et al. (1996). Ontologies: Principles, methods and applications. *Technical Report-University of Edinburgh Artificial Intelligence Applications Institute AIAI TR*.

van der Aalst, W. M. (2019). Modeling and reasoning over declarative data-aware processes with object-centric behavioral constraints. In *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings*, volume 11675, page 139. Springer.

Yahya, M., Barbosa, D., Berberich, K., Wang, Q., and Weikum, G. (2016a). Relationship queries on extended knowledge graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 605–614.

Yahya, M., Berberich, K., Ramanath, M., and Weikum, G. (2016b). Exploratory querying of extended knowledge graphs. *Proceedings of the VLDB Endowment*, 9(13):1521–1524.

Yoo, S. and Jeong, O. (2020). Automating the expansion of a knowledge graph. *Expert Systems with Applications*, 141:112965.

Zhang, Y., Sheng, M., Zhou, R., Wang, Y., Han, G., Zhang, H., Xing, C., and Dong, J. (2020). Hkgb: An inclusive, extensible, intelligent, semi-auto-constructed knowledge graph framework for healthcare with clinicians' expertise incorporated. *Information Processing & Management*, page 102324.