

Towards Conceptual Clustering in EL with Simulation Graphs

Ruud van Bakel, Michael Cochez and Patrick Koopmann

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands

Abstract

We introduce \mathcal{EL} clustering, a type of conceptual clustering in which each cluster is described by an \mathcal{EL} concept. The cluster then contains all instances of that concept. This contribution can be seen as a form of unsupervised learning of \mathcal{EL} concepts from data, useful for analyzing graph-based data as well as for ontology engineering. Towards a practical method for \mathcal{EL} clustering, we introduce complete simulation graphs, a structure from which all possible \mathcal{EL} clusterings of the given data can be extracted. From this structure, a good \mathcal{EL} clustering is then selected based on a utility function. We evaluate a first prototypical implementation of this idea on ABoxes of ontologies from a known benchmark, and show that bounded simulation graphs and \mathcal{EL} clusterings can often be computed in practice.


Keywords

unsupervised learning, conceptual clustering, summarization


1. Introduction

One of the central functions of an ontology is to specify the meaning of *concepts* from some domain of interest. In the case of ontologies formalized in description logics (DLs), these concepts are specified through axioms, usually producing a subsumption hierarchy of concepts from more general to more specific ones. Such an ontology can then be used in combination with a dataset (an ABox), and a reasoner can infer which objects in the data belong to which concepts defined in the ontology, thus organizing these objects into different categories of differing specificity. This makes ontologies useful for organizing and accessing data for many use cases. For example, consider a digital health record containing information about patients and what they are treated for in a hospital. Using a medical ontology, we can group patients into more high-level categories and e.g. find out how many patients are treated for heart diseases.

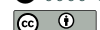
A major downside of ontologies is that they need to be formalized first, which requires the right expertise and can be a laborous process. As a solution, we propose to use *conceptual clustering* as an approach to mine DL concepts directly from data. This may be useful for different reasons: 1) the obtained concepts could be relevant concepts that should be added to an ontology, and thus support *ontology engineering*, and 2) they can help analyzing and exploring


 DL 2025: 38th International Workshop on Description Logics, September 3–6, 2025, Opole, Poland

 r.van.bakel@vu.nl (R. v. Bakel); m.cochez@vu.nl (M. Cochez); p.k.koopmann@vu.nl (P. Koopmann)

 <https://r-van-bakel.github.io/> (R. v. Bakel); <https://www.cochez.nl> (M. Cochez); pkoopmann.github.io (P. Koopmann)

 0000-0002-6891-5237 (R. v. Bakel); 0000-0001-5726-4638 (M. Cochez); 0000-0001-5999-2583 (P. Koopmann)

 © 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

data that are organized in graphs (e.g. knowledge graphs), by organizing the objects in the data into categories that can also be described in an intuitive way.

A plethora of research considers learning of DL descriptions, though not much considers the precise setting we have in mind. Approaches for *active learning*, e.g. following Angluin’s exact learning approach, assume that the learning system is interacting with a teacher (e.g. a domain expert) to whom it can send queries and current formalizations [1, 2]. Another line of research considers *concept learning*: here, we start with a set of individuals from the data grouped into positive and negative examples, and try to learn a DL concept that describes all of the positive and none of the negative examples [3, 4, 5, 6, 7]. Both types of learning are *supervised* in the sense that additional information from a domain expert is required.

In *unsupervised learning*, the goal is to learn axioms or concepts with less user input from the data. Unsupervised learning of GCIs is investigated in [8] using an induction-based approach. A limitation of this approach is that a set of interesting concepts has to be known beforehand, which arguably makes this approach less unsupervised. D’Amato et al. [9] developed a method for *conceptual clustering* to automatically generate hierarchies of \mathcal{ALC} concepts from data. The idea of conceptual clustering goes back to [10, 11]. Early systems such as COBWEB [12] cluster data represented as vectors with values for different features into a hierarchy of clusters that can be described based on their features. Clusters can then be described by “concepts” that associate to the different features a probability distribution of the weights. This is similar to *classical clustering*, where objects are typically clustered based on *distance measures*. However, instead of distance measures, usually a *utility criterion* is used that takes the quality of the cluster description and its predictive power into account. Moreover, the expressivity of the description language determines which clusters are possible, since every entity in the cluster must be described by the concept. D’Amato et al. follow this idea using the more expressive \mathcal{ALC} to describe the clusters. Unfortunately, the method does not scale well on larger data sets.

Moving to the more scalable DL \mathcal{EL} , a line of research considers unsupervised learning following the idea of *formal concept analysis* (FCA)[13, 14, 15, 16]. Those works focus on finding a finite *base* of axioms that fully characterize a given interpretation, i.e. cover all \mathcal{EL} axioms that hold in this interpretation. \mathcal{EL} provides nice model-theoretical properties that allow to use techniques such as products and simulations to more efficiently determine concepts that are relevant in a given interpretation. [17] uses the ideas from these FCA methods to apply a form of conceptual clustering for \mathcal{EL} with bounded role depth. The method was able to cluster 15 objects on a knowledge graph with 1,216 triples into \mathcal{EL} concepts with role depth two, but could not scale beyond. In this paper, we want to better understand the general feasibility of such an approach, using more closely the original idea of conceptual clustering used in [9], using an ad-hoc, but well-motivated, first utility measure.

Different from classical clustering approaches on graphs that first translate nodes into feature vectors, e.g. through embeddings [18, 19, 20], our conceptual clustering approach directly works on the graph structure, which adds to the transparency of the clustering result. The central idea is to use *simulation graphs*, a compressed representation of all possible clusterings of a given ABox. Similar structures were also used in works on FCA with \mathcal{EL} . To obtain scalability, we used an optimized algorithm to summarize ABox based on simulation—identifying objects that cannot be distinguished by \mathcal{EL} concepts. An extended version of the paper with additional details is provided together with the experimental data on Zenodo [21].

2. Preliminaries

2.1. The Description Logic \mathcal{EL}

Fix three countably infinite, pair-wise disjoint sets N_C , N_R and N_I of respectively *concept names*, *roles* and *individuals*. Concept names and roles correspond to unary and binary predicates in first-order logic, and individuals to constants. An *ABox* \mathcal{A} is a set of ground atoms called *assertions*, which are of the forms $A(a)$ and $r(a, b)$, where $A \in N_C$, $r \in N_R$ and $a, b \in N_I$. We use $N_I(\mathcal{A})/N_C(\mathcal{A})/N_R(\mathcal{A})$ to refer to the individuals/concept names/roles in \mathcal{A} . \mathcal{EL} concepts C are built based on the syntax rule $C ::= \top \mid A \mid C \sqcap C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. The *depth* of an \mathcal{EL} concept is the maximal nesting depth of role restrictions $\exists r.C$ occurring in it. The semantics of DLs is defined using first-order interpretations, e.g. tuples $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ with a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ assigning to every individual $a \in N_I$ an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, every concept name $A \in N_C$ a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and to every role $r \in N_R$ a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is lifted to \mathcal{EL} concepts as follows: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and $(\exists r.C)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \exists \langle d, e \rangle \in r^{\mathcal{I}} \text{ s.t. } e \in C^{\mathcal{I}}\}$. An individual $a \in N_I$ satisfies a concept C in \mathcal{I} , in symbols $\mathcal{I} \models C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. \mathcal{I} is a model of an ABox \mathcal{A} , in symbols $\mathcal{I} \models \mathcal{A}$, if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ for every $A(a) \in \mathcal{A}$, and $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for every $r(a, b) \in \mathcal{A}$. *TBoxes* or *ontologies* put additional constraints on an interpretation—an interpretation that satisfies these constraints is then a model of this TBox. The syntax is not relevant in this paper. A knowledge base (KB) is the union $\mathcal{T} \cup \mathcal{A}$ of a TBox and an ABox. A model of such a KB is an interpretation that is a model for both \mathcal{T} and \mathcal{A} . Note that a TBox and an ABox on their own is already a KB. If an assertion α is satisfied in every model of a KB \mathcal{K} , we write $\mathcal{K} \models \alpha$ and say α is *entailed by* \mathcal{K} . In the special case where $\alpha = C(a)$, we say that a is an *instance of* C in \mathcal{K} .

Every ABox has a unique *minimal model* that has one domain element per individual in the ABox which can be embedded into every other model of the ABox. For convenience, we may sometimes identify ABoxes with their minimal models.

2.2. Model Theory

Relevant to this work are also the notions of *simulations*, *bisimulations* and *products of interpretations*, which characterize the expressivity of DLs.

A *pointed interpretation* is a tuple $\langle \mathcal{I}, d \rangle$ of an interpretation \mathcal{I} and a domain element $d \in \Delta^{\mathcal{I}}$. Given two pointed interpretations $\langle \mathcal{I}, d \rangle$ and $\langle \mathcal{J}, e \rangle$, a *simulation from* $\langle \mathcal{I}, d \rangle$ *into* $\langle \mathcal{J}, e \rangle$ is a relation $\preceq \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{J}}$ that satisfies:

1. $d \preceq e$,
2. if $d' \in A^{\mathcal{I}}$ and $d' \preceq e'$, then $e' \in A^{\mathcal{J}}$,
3. if $\langle d_1, d_2 \rangle \in r^{\mathcal{I}}$ and $d_1 \preceq e_1$, then there must be $e_2 \in \Delta^{\mathcal{J}}$ s.t. $d_2 \preceq e_2$ and $\langle e_1, e_2 \rangle \in r^{\mathcal{J}}$.

We sometimes just write $\langle \mathcal{I}, d \rangle \preceq \langle \mathcal{J}, e \rangle$ to indicate that there exists such a simulation, without indicating the precise \preceq . If Conditions 2 and 3 also hold for the inverse relation, then \preceq is called a *bisimulation*. Simulations characterize the expressivity of \mathcal{EL} model-theoretically: for countable interpretations, if there is a simulation from $\langle \mathcal{I}, d \rangle$ into $\langle \mathcal{J}, e \rangle$, then for every \mathcal{EL} concept C , $d \in C^{\mathcal{I}}$ implies $e \in C^{\mathcal{J}}$. In the same way, bisimulations characterize the expressivity of the

classical DL \mathcal{ALC} , which plays only a minor role in this paper and is therefore not introduced beyond this fact. For elements d, e in an interpretation \mathcal{I} , we say that d is (bi-)similar to e if there exists a (bi-)simulation \preceq s.t. $\langle \mathcal{I}, d \rangle \preceq \langle \mathcal{I}, e \rangle$. (Note that similarity is in general not symmetric, contrary to what the name suggests. Bisimilarity on the other hand forms an equivalence relation.)

The *product* of two pointed interpretations $\langle \mathcal{I}_1, d \rangle$ and $\langle \mathcal{I}_2, e \rangle$ is a pointed interpretation $\langle \mathcal{J}, \langle d, e \rangle \rangle$ defined inductively as the smallest interpretation that satisfies the following:

1. $\langle d, e \rangle \in \Delta^{\mathcal{J}}$,
2. for every $\langle d', e' \rangle \in \Delta^{\mathcal{J}}$ s.t. $d' \in A^{\mathcal{I}_1}$ and $e' \in A^{\mathcal{I}_2}$, we have $\langle d', e' \rangle \in \Delta^{\mathcal{J}}$,
3. for every $\langle d_1, e_1 \rangle \in \Delta^{\mathcal{J}}$ s.t. $\langle d_1, d_2 \rangle \in r^{\mathcal{I}_1}$ and $\langle e_1, e_2 \rangle \in r^{\mathcal{I}_2}$, there is $\langle d_2, e_2 \rangle \in \Delta^{\mathcal{J}}$ and $\langle \langle d_1, e_1 \rangle, \langle d_2, e_2 \rangle \rangle \in r^{\mathcal{J}}$.

The construction of $\langle \mathcal{J}, \langle d, e \rangle \rangle$ ensures that $\langle \mathcal{J}, \langle d, e \rangle \rangle$ is a maximal interpretation such that $\langle \mathcal{J}, \langle d, e \rangle \rangle \preceq \langle \mathcal{I}_1, d \rangle$ and $\langle \mathcal{J}, \langle d, e \rangle \rangle \preceq \langle \mathcal{I}_2, e \rangle$. As a consequence, $\langle d, e \rangle$ satisfies exactly those \mathcal{EL} concepts that both d and e satisfy in their respective interpretations. If $\Delta^{\mathcal{I}_1}$ and $\Delta^{\mathcal{I}_2}$ are both finite, then $|\Delta^{\mathcal{J}}| \leq |\Delta^{\mathcal{I}_1}| \cdot |\Delta^{\mathcal{I}_2}|$.

Lemma 1. *Let $\langle \mathcal{J}, \langle d, e \rangle \rangle$ be the product of two pointed interpretations $\langle \mathcal{I}_1, d \rangle$ and $\langle \mathcal{I}_2, e \rangle$. Then, for every \mathcal{EL} -concept C such that $d \in C^{\mathcal{I}_1}$ and $e \in C^{\mathcal{I}_2}$, $\langle d, e \rangle \in C^{\mathcal{J}}$. Moreover, $\langle d, e \rangle$ satisfies no other concepts in \mathcal{J} .*

3. \mathcal{EL} Clusterings and Simulation Graphs

This paper is about computing \mathcal{EL} clusterings:

Definition 1. *Let \mathcal{K} be a KB. An \mathcal{EL} clustering for \mathcal{K} is a finite set of pairs $\langle C, \mathbf{A} \rangle$ where C is an \mathcal{EL} concept and \mathbf{A} is the set of instances of C in \mathcal{K} . We call the pairs $\langle C, \mathbf{A} \rangle$ \mathcal{EL} clusters.*

Note that this definition is highly unspecific, and even allows empty clusters or clusters that only differ in the concept but not in the set of individuals. For example, \mathcal{EL} clusters may be pair-wise disjoint, or they may form a (subsumption-)hierarchy. We also do not require the clustering to cover all individuals in the ABox, to allow for robustness against outliers. We will come to the question of what constitutes a good \mathcal{EL} clustering later.

At the center of our approach is the *simulation graph*, which is a structure that abstracts ABoxes into different sets of individuals in a way that preserves simulations. Fix an ABox \mathcal{A} . Given sets $\mathbf{A}, \mathbf{B} \subseteq \mathcal{A}$, we write $\mathbf{A} \xrightarrow{r} \mathbf{B}$ if for every $a \in \mathbf{A}$, there exists $b \in \mathbf{B}$ s.t. $r(a, b) \in \mathcal{A}$.

Definition 2 (Simulation graphs). *Let \mathcal{A} be an ABox. A simulation graph for \mathcal{A} is a directed labeled graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ with vertex-labeling λ_v and edge labeling λ_e s.t. $V \subseteq 2^{\mathbf{N}_I(\mathcal{A})}$, $\lambda_v : V \rightarrow 2^{\mathbf{N}_C}$ and $\lambda_e : V \rightarrow 2^{\mathbf{N}_R}$, and which satisfies the following conditions:*

1. $\lambda_v(\mathbf{A}) = \{A \in \mathbf{N}_C \mid A(a) \in \mathcal{A} \text{ for all } a \in \mathbf{A}\}$,
2. if $\mathbf{A} \xrightarrow{r} \mathbf{B}$, then $\langle \mathbf{A}, \mathbf{B} \rangle \in E$ with $\lambda_e(\langle \mathbf{A}, \mathbf{B} \rangle) = \{s \in \mathbf{N}_R \mid \mathbf{A} \xrightarrow{s} \mathbf{B}\}$,
3. for every $a \in \mathbf{N}_I(\mathcal{A})$ and $\mathbf{A} \in V$, if $A(a) \in \mathcal{A}$ for every $A \in \lambda_v(\mathbf{A})$ and $\{a\} \xrightarrow{r} \mathbf{B}$ for every $\langle \mathbf{A}, \mathbf{B} \rangle \in E$ s.t. $r \in \lambda_e(\langle \mathbf{A}, \mathbf{B} \rangle)$, then $a \in \mathbf{A}$.

The vertices in the simulation graphs correspond to sets of individuals from the ABoxes, with edges between vertices if there is a role connection between the corresponding individuals. The last condition ensures that the graph, intuitively, groups individuals based on simulations. Since simulation graphs label vertices with concept names and edges with roles, we can identify them with corresponding interpretations, and lift the definition of products and simulations to also include simulation graphs. Our definition then ensures that for all $\mathbf{A} \in V$, \mathbf{A} contains exactly those individuals $a \in N_1(\mathcal{A})$ for which $\langle \mathcal{S}, \mathbf{A} \rangle \preceq \langle \mathcal{A}, a \rangle$. This allows us to establish a connection between simulation graphs and \mathcal{EL} clusterings:

Lemma 2. *One can extract from every simulation graph \mathcal{S} for \mathcal{A} an \mathcal{EL} clustering \mathcal{C} for \mathcal{A} that has an \mathcal{EL} cluster $\langle C, \mathbf{A} \rangle$ for every node \mathbf{A} in \mathcal{S} and no other clusters. Moreover, for every \mathcal{EL} clustering \mathcal{C} for \mathcal{A} , there exists a simulation graph \mathcal{S} for \mathcal{A} that has a node \mathbf{A} for every cluster $\langle C, \mathbf{A} \rangle \in \mathcal{C}$.*

Note that the simulation graph corresponding to an \mathcal{EL} clustering may have more vertices than the clustering has clusters, in order to deal with role-successors.

We define some desirable properties of simulation graphs if we want to use them for computing \mathcal{EL} clusterings.

Definition 3 (Properties). *A simulation graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ for \mathcal{A} is*

- *complete if for every \mathcal{EL} concept C , there exists some $\mathbf{A} \in V$ that contains exactly the instances of C in \mathcal{A} ,*
- *n -complete, where $n \geq 0$, if for every \mathcal{EL} concept C of depth at most n , there exists some $\mathbf{A} \in V$ that contains exactly the instances of C in \mathcal{A} ,*
- *summarizing if for every $a \in N_1(\mathcal{A})$ there is some vertex $\mathbf{A} \in V$ that contains exactly the individuals in $N_1(\mathcal{A})$ that are similar to a .*
- *n -summarizing, where $n \geq 0$, if for every $a \in N_1(\mathcal{A})$, there is some vertex $\mathbf{A} \in V$ s.t. $\mathbf{A} = \{b \in N_1(\mathcal{A}) \mid \mathcal{A} \models C(b) \text{ for every } \mathcal{EL}\text{-concept } C \text{ of depth at most } n \text{ s.t. } \mathcal{A} \models C(b)\}$.*

Completeness implies all the other properties in this definition. Being n -summarizing is an approximation to being summarizing in the same way as n -completeness approximates completeness: for this, it is sufficient to recall that, if a is similar to b , then b is an instance of every \mathcal{EL} concept that a is an instance of. For $n \geq |N_1(\mathcal{A})|$, the notions n -complete and n -summarizing are equivalent to the non-approximative versions complete and summarizing.

3.1. Summarizing Simulation Graphs

As we will see later, summarizing simulation graphs can be computed efficiently. They have the potential to be useful not only for conceptual clustering in \mathcal{EL} , but also to improve reasoning times in general and to be used for other learning problems in DLs up to the expressivity of \mathcal{ALC} . The key property is that summarizing simulation graphs preserve all the relevant information about the individuals in an ABox, even if a TBox is added that is expressed in the more expressive DL \mathcal{ALC} . For a given simulation graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$, introduce a distinct individual $a_{\mathbf{A}}$ for every $\mathbf{A} \in V$ and define the *induced ABox* $\mathcal{A}_{\mathcal{S}}$ as

$$\mathcal{A}_{\mathcal{S}} = \{A(a_{\mathbf{A}}) \mid A \in \lambda_v(\mathbf{A})\} \cup \{r(a_{\mathbf{A}}, a_{\mathbf{B}}) \mid \langle \mathbf{A}, \mathbf{B} \rangle \in E \text{ and } r \in \lambda_e(\langle \mathbf{A}, \mathbf{B} \rangle)\}.$$

Theorem 1. Let $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ be a summarizing simulation graph for an ABox \mathcal{A} and \mathcal{T} an \mathcal{ALC} TBox. Let $a \in \mathbf{N}_I(\mathcal{A})$ and let $\mathbf{A} \in V$ be the smallest node in V (wrt. \subseteq) s.t. $a \in \mathbf{A}$. Then, for every \mathcal{ALC} concept C , $\mathcal{T} \cup \mathcal{A} \models C(a)$ iff $\mathcal{T} \cup \mathcal{A}_{\mathcal{S}} \models C(a_{\mathbf{A}})$.

Proof. The definition of summarizing simulation graphs makes sure that $a_{\mathbf{A}}$ is not only similar to a , but indeed bisimilar. Consequently, given a model \mathcal{I} of $\mathcal{T} \cup \mathcal{A}$, we can find a model \mathcal{J} of $\mathcal{T} \cup \mathcal{A}_{\mathcal{S}}$ s.t. $\langle \mathcal{I}, a^{\mathcal{I}} \rangle$ and $\langle \mathcal{J}, a_{\mathbf{A}}^{\mathcal{J}} \rangle$ are bisimilar, which implies that they satisfy the same \mathcal{ALC} concepts, and the same holds for the other direction. \square

Theorem 1 justifies why we define and compute simulation graphs only for ABoxes: adding a TBox, even if it is more expressive than \mathcal{EL} , cannot lead to new clusters being introduced. At the same time, reasoning is usually less computationally expensive without TBox. As we will see later, we are able to compute summarizing simulation graphs even for large ABoxes in very short time, and the number of nodes in such a graph is often significantly smaller than the size of the ABox. This indicates that simulation graphs also have the potential to speed up reasoning for KBs with large ABoxes. Theorem 1 also indicates potential for using simulation graphs for learning in more expressive logics:

Corollary 1. Let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ be an \mathcal{ALC} KB and $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ a summarizing simulation graph for \mathcal{A} . Then, for every \mathcal{ALC} concept C , the set of instances of C in \mathcal{K} is the union over some subset $V' \subseteq V$.

This means that even for supervised learning problems, that try to find a concept based on a given set of positive and negative examples, solutions can be obtained from summarizing simulation graphs. In fact, together with Lemma 2, we obtain that all solutions to concept learning problems in \mathcal{ALC} can be expressed as disjunction over \mathcal{EL} concepts. (Complement and value restrictions may however lead to more concise concepts.)

3.2. Complete Simulation Graphs

It is an easy consequence of our definitions that complete simulation graphs are unique.

Corollary 2. For any ABox \mathcal{A} , there is exactly one complete simulation graph.

We can get to this conclusion also by observing that complete simulation graphs are simply summarizing simulation graphs that are closed under *products*. For this, we first need to adapt the definition of products to simulation graphs. Given a simulation graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ and two elements $\mathbf{A}, \mathbf{B} \in V$, we can define the *product of \mathbf{A} and \mathbf{B}* as a node $\mathbf{A} \times \mathbf{B}$ in a new simulation graph $\mathcal{S}_{\mathbf{A} \times \mathbf{B}} = \langle V', E', \lambda'_v, \lambda'_e \rangle$ defined inductively as the smallest extension of \mathcal{S} with new vertices $\mathbf{A}' \times \mathbf{B}'$ that satisfies the following conditions:

1. $\mathbf{A} \times \mathbf{B} \in V'$,
2. for any $\mathbf{A}' \times \mathbf{B}' \in V'$, $\mathbf{A}' \cup \mathbf{B}' \subseteq \mathbf{A}' \times \mathbf{B}'$,
3. for any $\mathbf{A}' \times \mathbf{B}' \in V'$, $\lambda'_v(\mathbf{A}' \times \mathbf{B}') = \lambda(\mathbf{A}') \cap \lambda_v(\mathbf{B}')$,
4. for any $\mathbf{A}_1 \times \mathbf{B}_1 \in V'$, $\langle \mathbf{A}_1, \mathbf{A}_2 \rangle, \langle \mathbf{B}_1, \mathbf{B}_2 \rangle \in E$, there is $\mathbf{A}_2 \times \mathbf{B}_2 \in V'$ and $\langle \mathbf{A}_1 \times \mathbf{B}_1, \mathbf{A}_2 \times \mathbf{B}_2 \rangle \in E'$ with $\lambda'_e(\langle \mathbf{A}_1 \times \mathbf{B}_1, \mathbf{A}_2 \times \mathbf{B}_2 \rangle) = \lambda_e(\langle \mathbf{A}_1, \mathbf{A}_2 \rangle) \cap \lambda_2(\langle \mathbf{B}_1, \mathbf{B}_2 \rangle)$.

Note that as a consequence of Item 3 in Definition 2, $\mathbf{A}' \times \mathbf{B}'$ may contain more elements than $\mathbf{A} \cup \mathbf{B}$ —we require it to be the smallest extension that satisfies these criteria. The product is well-defined, since we can compute it using a fixpoint construction. We now say that a simulation graph is *closed under products* if applying the product on any pair of vertices produces again the same simulation graph. By using Lemma 1 and relating products in simulation graphs to products in pointed interpretations, we obtain the following theorem:

Theorem 2. *For all $n \geq 0$, any simulation graph that is (n) -summarizing and closed under products is (n) -complete.*

3.3. Extracting \mathcal{EL} -Clusterings

Because every \mathcal{EL} concept is represented with its set of instances, complete simulation graphs give an upper bound on what can be included in an \mathcal{EL} -clustering. If we are only interested in \mathcal{EL} concepts of bounded depth, n -complete simulation graphs are sufficient. Indeed, we can extract for each node in an n -complete simulation graph the corresponding \mathcal{EL} -concept of depth at most n using an inductive procedure. Fix a simulation graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$. We associate to every $\mathbf{A} \in V$ and $n \geq 0$ the \mathcal{EL} concept $C_{\mathbf{A}}^{(n)}$ defined inductively as follows:

1. $C_{\mathbf{A}}^{(0)} = \sqcap \lambda_v(\mathbf{A})$
2. For $n > 0$, $C_{\mathbf{A}}^{(n)} = \sqcap \lambda_v(\mathbf{A}) \sqcap \sqcap_{\langle \mathbf{A}, \mathbf{B} \rangle \in E} \sqcap_{r \in \lambda_e(\langle \mathbf{A}, \mathbf{B} \rangle)} \exists r. C_{\mathbf{B}}^{(n-1)}$

It follows directly from our definitions that $\mathcal{A} \models C_{\mathbf{A}}^{(n)}(a)$ for every $a \in \mathbf{A}$, and that for an n -complete simulation graph, we can obtain all possible \mathcal{EL} clusters with concepts of depth at most n . We can use those clusters also together with an \mathcal{EL} - or \mathcal{ALC} TBox. It can then be that we need less clusters: concepts for different vertices may become equivalent because of the TBox, and even capture the same set of individuals if they are not equivalent. The TBox can also be used to obtain more concise descriptions of the clusters: the method in [22] can compute for a given \mathcal{EL} concept a concept of minimal length that is equivalent wrt. the TBox. We may use this in future work to obtain more user-friendly representations of conceptual clusterings.

To determine which subset $V' \subseteq V$ constitutes a *good* \mathcal{EL} -clustering, we introduce *quality measures* for \mathcal{EL} -clusterings, which we here define in the most general way.

Definition 4. *An \mathcal{EL} -cluster quality measure is a function q mapping pairs of KBs and \mathcal{EL} clusters to real numbers. An \mathcal{EL} -clustering quality measure is a function q that assigns to every KB and set of \mathcal{EL} -clusters a real number.*

Quality measures may take into account different aspects such as the number of individuals in the cluster, the length of the concept, or try to minimize the overlap between different clusters. What constitutes a good quality measure will ultimately depend on the use case: for instance, if we want to use clusterings to support ontology engineering, we may also want to take into account whether introducing a concept name for a cluster will allow us to describe individuals more concisely. We keep a deeper investigation of quality measures as future work, and use for now a quality measure based on the idea of *category utility* already used in the conceptual clustering algorithm COBWEB [12]. Intuitively, the utility of a cluster captures its *predictive power*: how does knowing that an individual belongs to a cluster help us in predicting

which concepts it satisfies? This is in particular helpful if we want to use \mathcal{EL} clusterings to organize and explore the individuals in a KB: For this, we may organize the \mathcal{EL} clustering in a subsumption hierarchy, and navigate it starting from the most general concept. Assume we want to explore a dataset about employees in a company and the gender of an employee has no correlation to any of the other properties in the dataset. Then, organizing the data by gender does not bring any advantage, since it does not have any predictive power. Rather, we would like to organize the employees based on concepts that let us predict additional information.

Simulation graphs also help with defining this predictive power. Fix a KB $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$. Given two sets $\mathbf{A}, \mathbf{B} \subseteq \mathbf{N}_I(\mathcal{A})$, the conditional probability of \mathbf{A} given \mathbf{B} is defined as $P(\mathbf{A} \mid \mathbf{B}) = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{B}|}$, and the probability of \mathbf{A} as $P(\mathbf{A}) = P(\mathbf{A} \mid \mathbf{N}_I(\mathcal{A}))$. Given a concept name A , we use $A^\mathcal{K}$ to refer to the instances of A in \mathcal{K} . By abuse of notation, given $\mathbf{A} \subseteq \mathcal{A}$ and $r \in \mathbf{N}_R$, we use $\exists r.\mathbf{A}^\mathcal{K}$ to denote the individuals that have an r -successor in \mathbf{A} :

$$\exists r.\mathbf{A}^\mathcal{K} = \{a \in \mathbf{N}_I(\mathcal{A}) \mid r(a, b) \in \mathcal{A}, b \in \mathbf{A}\}$$

Definition 5. Let $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ be a KB and $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ a simulation graph for \mathcal{A} . The utility of a cluster $\langle C, \mathbf{A} \rangle$ in context $\mathbf{C} \subseteq \mathbf{N}_I(\mathcal{A})$, relative to \mathcal{S} , is defined as

$$\begin{aligned} ut_{\mathcal{S}}(\mathbf{A} \mid \mathbf{C}) = P(\mathbf{A} \mid \mathbf{C}) \cdot & \left(\sum_{A \in \mathbf{N}_C} (P(A^\mathcal{K} \mid \mathbf{A} \cap \mathbf{C})^2 - P(A^\mathcal{K} \mid \mathbf{C})^2) \right. \\ & \left. + \sum_{r \in \mathbf{N}_R} \sum_{\mathbf{B} \in V} (P(\exists r.\mathbf{B}^\mathcal{K} \mid \mathbf{A} \cap \mathbf{C})^2 - P(\exists r.\mathbf{B}^\mathcal{K} \mid \mathbf{C})^2) \right) \end{aligned}$$

The utility of $\langle C, \mathbf{A} \rangle \in V$ is defined as $ut_{\mathcal{S}}(\mathbf{A} \mid \mathbf{N}_I(\mathcal{A}))$.

Note that the concept C used to describe the cluster is not relevant here. \mathcal{S} determines which subsets of $\mathbf{N}_I(\mathcal{A})$ can be distinguished by the measure. In the case where \mathcal{S} is complete, these are indeed the subsets of $\mathbf{N}_I(\mathcal{A})$ that can be distinguished using \mathcal{EL} concepts. Individuals can be described based on which concept names they satisfy, and which \mathcal{EL} concepts their r -successors satisfy. The intuitive idea of the utility of a vertex \mathbf{A} is that *guessing* whether an individual a belongs to \mathbf{A} gives us an advantage in guessing any of these characteristics.

Looking at the utility of vertices in isolation is not sufficient for extracting a good \mathcal{EL} clustering, because there may be vertices with high utility that cover almost the same set of individuals. We therefore consider also the *relative utility* of one vertex to another: Assuming we know that an individual a belongs to \mathbf{A} or \mathbf{B} , does guessing which one it belongs to give any advantage in guessing its other properties? This corresponds to $ut_{\mathcal{S}}(\mathbf{A} \mid \mathbf{A} \cup \mathbf{B})$. If \mathbf{A} and \mathbf{B} help predicting the same things, then this value is going to be lower than if they predict different things, so that minimizing this value will lead to a more useful set of \mathcal{EL} clusters. We can thus define the *relative utility* of a cluster $\langle C, \mathbf{A} \rangle$ in a given \mathcal{EL} -clustering \mathcal{C} as

$$ut_{\mathcal{S}}(\mathbf{A} \mid \mathcal{C}) = \min\{ut_{\mathcal{S}}(\mathbf{A} \mid \mathbf{A} \cup \mathbf{B}) \mid \langle D, \mathbf{B} \rangle \in \mathcal{C}, D \neq C\}.$$

Intuitively, if the relative utility of a cluster is low, then dropping it from the clustering is not going to affect its overall predictive power, as the predictive power of that cluster is anyway low in presence of other clusters.

We now define the *utility of an \mathcal{EL} -clustering \mathcal{C}* (relative to a simulation graph \mathcal{S}) as

$$ut_{\mathcal{S}}(\mathcal{C}) = \sum_{\langle \mathcal{C}, \mathbf{A} \rangle \in \mathcal{C}} ut_{\mathcal{S}}(\mathbf{A} \mid \mathcal{C}).$$

4. Computing \mathcal{EL} -Clusterings in Practice

We first compute a summarizing simulation graph $\mathcal{S} = \langle V, E, \lambda_v, \lambda_e \rangle$ from the given ABox \mathcal{A} . For ease of presentation, we assume in the following that every concept assertion $A(a)$ is represented as a role assertion $r(a, d^*)$ with d^* some specific fresh individual. The backbone of our algorithm is the process of *partition refinement* [23]. The intuition here is that we cluster many individuals together at first, then as we consider deeper \mathcal{EL} concepts, we further refine the clusters in smaller ones. We use the following notation to refer to our partitions on the individuals:

Definition 6 (Partition on the individuals). $P^{(l)} := P^{(l)}(\mathcal{A})$ is the partition on the set of individuals \mathbb{N}_I , generated by \mathcal{EL} concepts of depth l .

We will now explain what it means for \mathcal{EL} concepts to generate a partition. The first partition always clusters all individual together in one set: $P^{(0)} = \{\mathbb{N}_I\}$. For any subsequent partition we have the following rule:

Definition 7 (Refinement). Any two concepts a and b are clustered together in $P^{(l)}$ (with $l \geq 1$) if all of the following conditions hold:

- a and b are clustered together in $P^{(l-1)}$,
- for every $r(a, c) \in \mathcal{A}$ there exists an $r(b, d) \in \mathcal{A}$, such that c and d are clustered together in $P^{(l-1)}$,
- for every $r(b, d) \in \mathcal{A}$ there exists an $r(a, c) \in \mathcal{A}$, such that c and d are clustered together in $P^{(l-1)}$.

If we assume a finite ABox, then when computing the partitions at some point the algorithm will find a fixed point t , i.e. $P^{(t+1)} = P^{(t)}$. From this point onward all the partitions will stay the same. Therefore when the fixed point is found, our algorithm stops. We can now get V as the union of all the partitions: $V = \bigcup_{i=0}^t P^{(i)}$.

For the edges E we have the following rule:

Definition 8 (Simulation graph edges). Any triple $(\mathbf{S}, r, \mathbf{O})$ is part of E if and only if there exists at least one level i that satisfies the following:

- $\mathbf{S} \in P^{(i)}$,
- $\mathbf{O} \in P^{(i-1)}$,
- there exist individuals $s \in \mathbf{S}$ and $o \in \mathbf{O}$, such that $r(s, o) \in \mathcal{A}$.

With the vertices and edges generated, we can now simply take the labelings as defined in Definition 2 to create \mathcal{S} . It is easy to verify that this results in a summarizing simulation graph. Moreover, if we replace V by $V_k = \bigcup_{i=0}^k P^{(i)}$, we obtain a k -summarizing simulation graph.

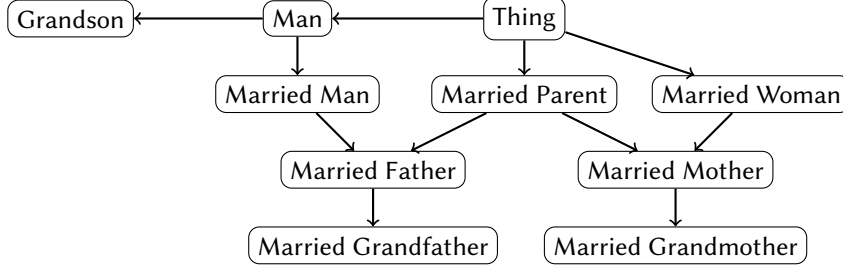


Figure 1: Clusters found in the family history ontology, with labels indicating their intuitive meaning.

To extend the summarizing simulation graph to a complete one, which at the same time serves as initial clustering, we add products of pairs of vertices in rounds until a fixpoint is reached. To optimize the utility, we follow a greedy approach, and remove the worst cluster according to $ut_S(\mathbf{A} \mid \mathbf{A} \cup \mathbf{B})$ until a termination criterion is met. In our experiments, we used the desired number of clusters as criterion.

5. Evaluation

We applied the implementation on the family-history ontology that was used in [5] to evaluate concept learning, which is an ontology with a very simple TBox (no complex concepts), and whose ABox contains family relations as roles and concepts for different types of family members. The ABox contained 1780 assertions with 202 individuals, which were summarized into 244 nodes in the summarizing simulation graph, and 30 in the 1-summarizing graph. We computed the 1-complete graph after 4 rounds of building products of all nodes, thus obtaining all relevant \mathcal{EL} concepts without nested role restrictions. This graph contained 53 nodes, from which we extracted 10 \mathcal{EL} clusters using the greedy algorithm. When computing the utility (here and in later experiments), we used the complete summarizing graph, and not just the 1-summarizing one. While the corresponding concepts were relatively complex, they mostly correspond to simple concepts in human language, e.g. “Person \sqcap Male” corresponding to “Man”—Figure 1 shows the clusters organized into the subsumption hierarchy, where we represent each \mathcal{EL} concept with the simple concept in natural language (the full list of concepts is in the appendix). We find more or less the concepts one would expect as central concepts in a family tree, with the exception of “Woman”, which one would have expected together with “Man”, and the fact that “Grandson” is the only cluster representing children. This might indicate a bias in the data, but also shows that more research is needed into utility measure and alternative methods for computing the clustering. Note that the most general cluster “Thing” is needed to allow for higher relative utility of the other clusters.

The next experiment was about getting a first idea of the feasibility of \mathcal{EL} clustering, and used ontologies from the OWL EL materialization track of the OWL reasoner competition 2015 [24, 25]. This corpus contains 110 ontologies in the OWL EL profile with non-empty ABoxes. We removed ABoxes axioms that were not supported by our method (e.g. equivalence assertions or assertions involving complex concepts), which resulted in 16 ontologies without

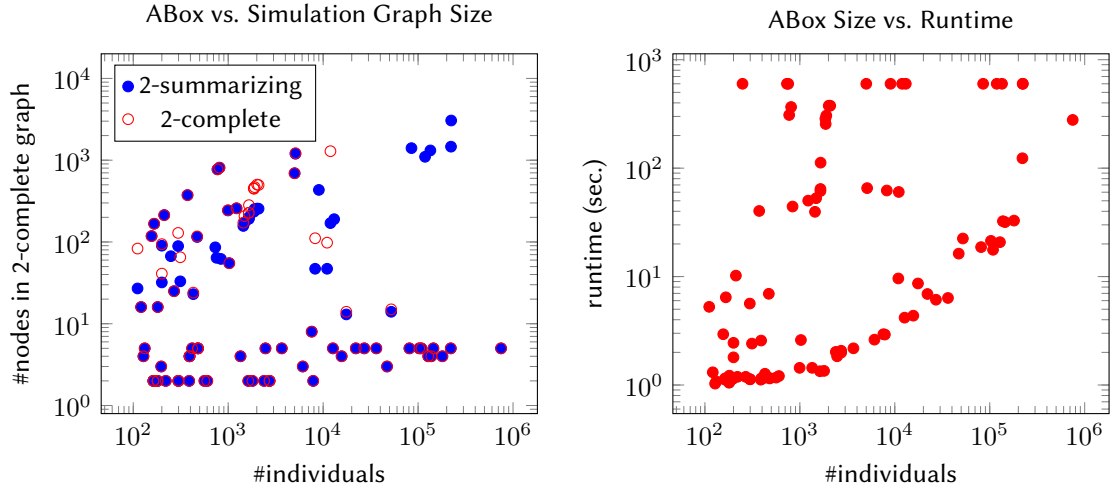


Figure 2: Comparing number of individuals to nodes in 2-complete graph (left) and runtime of computing the \mathcal{EL} clusterings (right).

ABox, leaving us with 94 ontologies to be used in the experiment. For each ontology, we computed the summarizing simulation graphs and attempted to compute 2-complete simulation graphs, from which we then extract \mathcal{EL} -clusterings of size 10 using the greedy algorithm. We used a time limit of 10 minutes for each ontology, which resulted in 12 timeouts. The detailed results are shown in the appendix. Section 5 visualizes how the ABox sizes relate to the sizes of the 2-complete graphs and to the computation times. The number of nodes in the summarizing simulation graph is often much smaller than the number of individuals, and that the computation of clusters, even though so far only the summarizing step is really optimized, is possible in short time in most cases. Quite remarkably, many very large ABoxes resulted in summarizing simulation graphs with under 10 nodes, including the largest one with almost 750,000 individuals which had only 5 nodes in the summarizing graph. This indicates the potential of summarizing simulation graphs also for speeding up ABox reasoning.

6. Conclusion

While our summarization method is already quite efficient, we want to investigate more dedicated algorithms for computing the clusterings. One option is to integrate the computation of products directly into the summarization implementation, to be able to compute products more efficiently on the different levels of the simulation graphs. Another way could be to use an incremental approach as done in the conceptual clustering algorithm for \mathcal{ALC} [9]. Our selection of good clusterings is currently done in a greedy fashion—we want to investigate whether encodings into answer set programming might allow us to compute globally optimal solutions. Future evaluations should look into different corpora than the ORE corpus, for instance knowledge graphs without an ontology. To understand whether our utility function makes sense and compare with other alternatives, user studies will be required.

References

- [1] B. Konev, C. Lutz, A. Ozaki, F. Wolter, Exact learning of lightweight description logic ontologies, *J. Mach. Learn. Res.* 18 (2017) 201:1–201:63. URL: <https://jmlr.org/papers/v18/16-256.html>.
- [2] M. R. C. Duarte, B. Konev, A. Ozaki, ExactLearner: A tool for exact learning of \mathcal{EL} ontologies, in: M. Thielscher, F. Toni, F. Wolter (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October - 2 November 2018*, AAAI Press, 2018, pp. 409–414. URL: <https://aaai.org/ocs/index.php/KR/KR18/paper/view/18006>.
- [3] J. Lehmann, DL-Learner: Learning concepts in description logics, *J. Mach. Learn. Res.* 10 (2009) 2639–2642. URL: <https://dl.acm.org/doi/10.5555/1577069.1755874>. doi:10.5555/1577069.1755874.
- [4] G. Rizzo, N. Fanizzi, C. d’Amato, Class expression induction as concept space exploration: From DL-Foil to DL-Focl, *Future Gener. Comput. Syst.* 108 (2020) 256–272. URL: <https://doi.org/10.1016/j.future.2020.02.071>. doi:10.1016/J.FUTURE.2020.02.071.
- [5] S. Heindorf, L. Blübaum, N. Düsterhus, T. Werner, V. N. Golani, C. Demir, A. N. Ngomo, Evolearner: Learning description logics with evolutionary algorithms, in: F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, L. Médini (Eds.), *WWW ’22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, ACM, 2022, pp. 818–828. URL: <https://doi.org/10.1145/3485447.3511925>. doi:10.1145/3485447.3511925.
- [6] A. N. Ngomo, C. Demir, N. J. Kouagou, S. Heindorf, N. Karalis, A. Bigerl, Class expression learning with multiple representations, in: P. Hitzler, M. K. Sarker, A. Eberhart (Eds.), *Compendium of Neurosymbolic Artificial Intelligence*, volume 369 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2023, pp. 272–286. URL: <https://doi.org/10.3233/FAIA230145>. doi:10.3233/FAIA230145.
- [7] B. ten Cate, M. Funk, J. C. Jung, C. Lutz, SAT-based PAC learning of description logic concepts, in: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, ijcai.org, 2023, pp. 3347–3355. URL: <https://doi.org/10.24963/ijcai.2023/373>. doi:10.24963/IJCAI.2023/373.
- [8] V. Sazonau, U. Sattler, G. Brown, General terminology induction in OWL, in: M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d’Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, K. Thirunarayan, S. Staab (Eds.), *The Semantic Web - ISWC 2015*, Springer International Publishing, Cham, 2015, pp. 533–550.
- [9] C. d’Amato, S. Staab, N. Fanizzi, F. Esposito, DL-Link: a conceptual clustering algorithm for indexing description logics knowledge bases, *Int. J. Semantic Comput.* 4 (2010) 453–486. URL: <https://doi.org/10.1142/S1793351X10001085>. doi:10.1142/S1793351X10001085.
- [10] R. S. Michalski, R. E. Stepp, Learning from observation: Conceptual clustering, *Machine learning: An artificial intelligence approach* (1983) 331–363.
- [11] A. P. Suárez, J. F. M. Trinidad, J. A. Carrasco-Ochoa, A review of conceptual clustering algorithms, *Artif. Intell. Rev.* 52 (2019) 1267–1296. URL: <https://doi.org/10.1007/s10462-018-9627-1>. doi:10.1007/S10462-018-9627-1.

- [12] D. H. Fisher, Knowledge acquisition via incremental conceptual clustering, *Machine learning* 2 (1987) 139–172.
- [13] F. Distel, Learning description logic knowledge bases from data using methods from formal concept analysis, Ph.D. thesis, Dresden University of Technology, 2011. URL: <https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-70199>.
- [14] D. Borchmann, F. Distel, F. Kriegel, Axiomatisation of general concept inclusions from finite interpretations, *J. Appl. Non Class. Logics* 26 (2016) 1–46. URL: <https://doi.org/10.1080/11663081.2016.1168230>. doi:10.1080/11663081.2016.1168230.
- [15] F. Kriegel, Efficient axiomatization of OWL 2 EL ontologies from data by means of formal concept analysis, in: M. J. Wooldridge, J. G. Dy, S. Natarajan (Eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, AAAI Press, 2024, pp. 10597–10606. URL: <https://doi.org/10.1609/aaai.v38i9.28930>. doi:10.1609/AAAI.V38I9.28930.
- [16] R. Guimarães, A. Ozaki, C. Persia, B. Sertkaya, Mining \mathcal{EL}_{\perp} bases with adaptable role depth, *J. Artif. Intell. Res.* 76 (2023) 883–924. URL: <https://doi.org/10.1613/jair.1.13777>. doi:10.1613/JAIR.1.13777.
- [17] F. Hodo, S. Pranav, B. Sertkaya, Clustering knowledge graphs using concept lattices (extended abstract), in: O. Kutz, C. Lutz, A. Ozaki (Eds.), *Proceedings of the 36th International Workshop on Description Logics (DL 2023) co-located with the 20th International Conference on Principles of Knowledge Representation and Reasoning and the 21st International Workshop on Non-Monotonic Reasoning (KR 2023 and NMR 2023)*, Rhodes, Greece, September 2-4, 2023, volume 3515 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: <https://ceur-ws.org/Vol-3515/abstract-12.pdf>.
- [18] F. Martel, A. Zouaq, Taxonomy extraction using knowledge graph embeddings and hierarchical clustering, in: C. Hung, J. Hong, A. Bechini, E. Song (Eds.), *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021, ACM, 2021*, pp. 836–844. URL: <https://doi.org/10.1145/3412841.3441959>. doi:10.1145/3412841.3441959.
- [19] L. Guo, Q. Dai, Graph clustering via variational graph embedding, *Pattern Recognit.* 122 (2022) 108334. URL: <https://doi.org/10.1016/j.patcog.2021.108334>. doi:10.1016/J.PATCOG.2021.108334.
- [20] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, *IEEE Trans. Knowl. Data Eng.* 29 (2017) 2724–2743. URL: <https://doi.org/10.1109/TKDE.2017.2754499>. doi:10.1109/TKDE.2017.2754499.
- [21] P. Koopmann, R. Bakel, M. Cochez, Towards conceptual clustering in \mathcal{EL} with simulation graphs — experimental data and extended version, 2025. URL: <https://doi.org/10.5281/zenodo.16794054>. doi:10.5281/zenodo.16794054.
- [22] N. Nikitina, P. Koopmann, Small is beautiful: Computing minimal equivalent \mathcal{EL} concepts, in: S. Singh, S. Markovitch (Eds.), *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, AAAI Press, 2017, pp. 1206–1212. URL: <https://doi.org/10.1609/aaai.v31i1.10684>. doi:10.1609/AAAI.V31I1.10684.

- [23] R. Paige, R. E. Tarjan, Three partition refinement algorithms, *SIAM Journal on Computing* 16 (1987) 973–989. URL: <https://doi.org/10.1137/0216062>. doi:10.1137/0216062.
- [24] B. Parsia, N. Matentzoglou, R. S. Gonçalves, B. Glimm, A. Steigmiller, The OWL reasoner evaluation (ORE) 2015 competition report, *J. Autom. Reason.* 59 (2017) 455–482. URL: <https://doi.org/10.1007/s10817-017-9406-8>. doi:10.1007/s10817-017-9406-8.
- [25] N. Matentzoglou, B. Parsia, ORE 2015 reasoner competition corpus, 2015. URL: <https://doi.org/10.5281/zenodo.18578>. doi:10.5281/zenodo.18578.