The GraphBRAIN Knowledge Graph Framework for XAI through Multistrategy Reasoning

Stefano Ferilli[1,*], Eleonora Bernasconi[1] and Domenico Redavid[1]

[1]University of Bari, Bari, Italy

Abstract

As long as Artificial Intelligence is pervading all aspects of our lives, it is becoming instrumental also for crucial aspects, impacting directly our existence and well-being. This means that some of its decisions must be checked and validated, which in turn requires them to be explainable. The explainable-by-design approach in Artificial Intelligence is the symbolic one, based on formal logics. It applies human-like automated reasoning and relies on suitable knowledge representations. The most widely known and adopted knowledge representation approach nowadays comes from the Semantic Web community, but due to its peculiarities it has some limitations and shortcomings. In this paper we propose an alternate framework that significantly expands the range of applicable automated reasoning strategies.

Keywords

Knowledge Graphs, Semantic Web, Multistrategy Reasoning, Labeled Property Graphs

1. Introduction & Motivations

In the last years, Artificial Intelligence (AI) started to be widely and pervasively adopted in all realms of human life. From more routinary issues, it started dealing with more and more delicate human activities, where its decisions and outcomes can directly and significantly affect human lives (e.g., in medicine, justice, economy, etc.). In these cases, we cannot afford to blindly trust the AI outcomes. A truly human-centric approach to AI must be based on a cooperation between the machines and their users, which in turn requires them to share a common representational and processing ground [1], so as to enforce trustworthiness and support accountability of the automated systems.

Key in this perspective are *transparency* (the possibility of understanding the internal machinery of an AI system) and *understandability* (the possibility for humans to understand AI outcomes). The latter, in particular, can take different forms, such as *interpretability* (helping the user by highlighting the components in the input that were relevant to determine the AI output) and *explainability* (reporting the full sequence of logic steps, connected by causal relationships, that were *actually* made to get to the output starting from the input). The latter is obviously the most appropriate level to have full control in the check and validation of AI outcomes, where the word 'actually' was emphasized on purpose to distinguish what we consider 'true' explanations from other kinds of 'simulated' explanations. E.g., post-hoc explanations are explanations that generate a reasonable sequence of reasoning steps that might generate the output from the input, with no guarantee that it is the actual reasoning originally followed by the system in producing its outcome. While a true explanation is mainly deductive (reasoning from causes to effects), a post-hoc explanation is abductive (a form of reasoning from effects to causes), meaning that it may be wrong, albeit sensible. This distinction is not just formal of philosophical: in some crucial situations one may need to know the real reasoning followed to obtain a given outcome, in order to check it and possibly refine it or turn it down [2].

XAI-KRKG@ECAI25: First International ECAI Workshop on eXplainable AI, Knowledge Representation and Knowledge Graphs, October 25–30, 2025, Bologna, Italy

*Corresponding author.

stefano.ferilli@uniba.it (S. Ferilli); eleonora.bernasconi@uniba.it (E. Bernasconi); domenico.redavid1@uniba.it (D. Redavid)

http://ara.di.uniba.it/~ferilli (S. Ferilli); https://www.uniba.it/it/docenti/eleonora-bernasconi (E. Bernasconi); http://ara.di.uniba.it/redavid.html (D. Redavid)

0000-0003-1118-0601 (S. Ferilli); 0000-0003-3142-3084 (E. Bernasconi); 0000-0003-2196-7598 (D. Redavid)

AI approaches can be distinguished into sub-symbolic ones [3], based on numbers for representation and on mathematical-statistical tools for processing, and symbolic ones [4], based on explicit concepts and on formal logic-based tools for processing [5]. The former simulate the brain; they are efficient, fit for reproducing perception or intuition, and Kahneman's 'fast thinking' [6]. The latter simulate human mind; they are quite effective, fit for reproducing reasoning, and 'slow thinking' (in terms of Kahneman's theory). The pros and cons of these two approaches are clearly complementary [7], and in humans they harmoniously cooperate to support the everyday behavior of single agents and of multi-agent systems. While the latter approach is *explainable by design* (i.e., its outcomes are inherently explainable), being based on a close reproduction of the conscious mechanisms in humans, the kind of AI that is nowadays becoming pervasive belongs to the sub-symbolic type, which is inherently non-explainable. Even worse, the complexity of the systems in wide adoption today is such that they can also barely be considered transparent. Due to the need for explainability, which in some countries is required by law for some applications (e.g., the AI Act in the EU [8]), much effort is being spent in research on such systems to make them explainable. However, our position is that, unfortunately, research in eXplainable Artificial Intelligence (XAI) [9, 10] is mostly proposing approaches that are just interpretable, not explainable. A true explainability should be obtained by a cooperation of the two approaches, just like it happens in humans: after perception happens from the outside world, and intuition immediately takes place, a process of rationalization and reasoning allows us to compensate for the biases and errors of intuition, and to get to a more conscious and controlled, possibly even shared, understanding and decision. Again, attempts in this direction (e.g., NeSy, the stream of research aimed at joining Neural with Symbolic approahes [11]) are often flawed: instead of striving for a true cooperation, they tend to overcharge the neural part so that it can also carry out something that can resemble reasoning. On the contrary, we strongly believe that a true cooperation must take place, and in this paper we propose a framework for Knowledge Representation and Reasoning (KRR) and associated tools.

Knowledge is the complex inter-relation of information items, where an information is an interpreted datum. By definition knowledge is *explicit*, so that it can be shared and communicated among different subjects. This was the basis of all human growth so far, and the way in which reliable theories were recognized, validated and adopted, by reaching consensus and superseding wrong ones. The current state-of-the-art for knowledge representation is based on the representation of knowledge bases (KBs) as graphs so-called *Knowledge Graphs* (KGs). Two parts can be distinguished in the content of a KG: the *ontology* and the *instances*. The former determines what can be described (entities, relationships and their attributes), how it can be described (the terminology for expressing the entities, relationships and attributes), and what properties or constraints must hold in the world that is being described; ontologies are important because they provide meaning and context to the symbols used in the KB. The latter are the specific objects of interest in the world that is being described, connected to the corresponding ontological elements and described by suitable values for their attributes).

The most widespread approach to KGs in the current research landscape is the Semantic Web (SW) one. Born for specific purposes (allowings machines —and humans— to share data by associating them with the very same interpretation, in order to support interoperability), it set its own standards for representation and reasoning, now adopted in a wide range of application fields. In the SW, both the ontology and the instances co-exist in the same graph, which is represented according to the RDF graph model, where a graph consists of a set of triples $< Subject, predicate, Object >$ whose constituents are atomic (i.e., simple values). The subject and object correspond to nodes in the graph, and the predicate corresponds to an arc. We believe that, while being a precious starting point for our work on XAI, the SW approach and standard are not fully satisfactory, for a number of reasons, and notably:

- the RDF representation is too scattered [12]: the description of an instance in the world corresponds to a subgraph, and it is not immediate to collect all the components of such a subgraph;
- even the representation of common knowledge items, such as attributes of relationships or the existence of several instances of one relationship between the same pair of objects, may be impossible or overly complex;
- the kind of reasoning that can be applied in the SW is mostly ontological (inheritance of properties,

completeness or consistency checks on the instances, etc.), leaving out the bulk of inference strategies that humans use everyday to carry out our practical activities.

This paper focuses on the latter limitation. Humans solve problems using combinations of inference strategies, and our thesis is that, to be fully explainable, also the AI approaches must support a similar behavior. To tackle this limitation, we also deal with the former one, adopting the LPG graph model. In addition to be more compact and readable than the RDF model, it is also less coupled with SW approaches, and thus amenable to broaden the set of inference strategies applicable to the knowledge, and is implemented by very efficient state-of-the-art DBMS adopted by big players in the industry.

In the following we will propose a framework for KRR that is aimed at overcoming these limitations, in order to provide a more extensive and useful approach to XAI. We will discuss its representation and reasoning strategies and formalisms, and its current implementation status and applications. Finally, we will conclude the paper and outline future work issues.

2. The GraphBRAIN Framework

GraphBRAIN is a general-purpose KRR framework, and a knowledge base management system aimed at covering all stages and tasks in the lifecycle of a KB, including knowledge acquisition, organization, and (personalized) fruition. It adopts the Labeled Property Graph (LPG) data model, where nodes (representing individuals) and arcs (representing relationships) may have labels (usually expressing their type) and associated attribute-value pairs, and uses the Neo4j [13] DBMS. Neo4j is schema-less, which ensures great flexibility but does not allow to associate a clear semantics to the graph items. For this reason, GraphBRAIN requires its users to work according to pre-specified data schemes, expressed in the form of ontologies. Thus, a characterizing feature of GraphBRAIN is its bringing to cooperation a database management system for efficiently handling, mining and browsing the individuals, with an ontology level that allows it to carry out formal reasoning on the knowledge.

The ontologies are expressed in a proprietary format[1], specifically tailored for LPGs [14]. It allows to define:

- data types (whose names start with a lowercase letter), as enumerations or tree-structured organization of values (whose names start with an uppercase letter);
- entities (whose names start with an uppercase letter);
- entity properties (e.g., being abstract, i.e. not allowing instances in the KB);
- relationships (whose names start with a lowercase letter);
- relationship properties (e.g., symmetricity, transitivity, etc.);
- entity or relationship attributes (whose names start with a lowercase letter – note that relationship attributes are not available in RDF);
- attribute properties (e.g., being mandatory or not);
- axioms (logic formulas or constraints that must be verified by the instances in the KB).

GraphBRAIN can apply several ontologies on the same graph, describing different domains and representing different perspectives on the same knowledge. The classes shared by different ontologies allow the system to connect knowledge across domains: their individuals act as bridges, allowing the users of a domain to reach information coming from other domains. In particular, GraphBRAIN comes with a top-level ontology defining very general and highly reusable concepts (e.g., Person, Place; Person.wasIn.Place; etc.). This top-level ontology plays a crucial role to interconnect the domain-specific ontologies, ensuring an overall connected KG. Using a suitable tool, GraphBRAIN administrators may create, build and maintain additional ontologies.

Information (instances) can be fed into, or retrieved from, the KB only according to the ontologies. Specifically, the following correspondence is established between the ontology and the LPG elements:

[1]Most other works tried to merge research on RDF and LPG knowledge representations, but always giving the RDF perspective priority and predominance. GraphBRAIN was the first to push for a native LPG-oriented approach [14]. After the publication of GraphBRAIN, other initiatives investigated the possibility of developing suitable schemas for LPGs specifically [15].

- nodes in the graph are entity instances (each node has a unique identifier, so that two nodes carrying exactly the same information can co-exist in the graph);
- each node is labeled with the name of the most specific entity it belongs to in the ontology; it is also labeled with all the domain names for which it is relevant;
- nodes include attribute-value maps expressing the values of the properties associated to their entity and to all of its superclasses (by inheritance);
- arcs in the graph are relationship instances (each arc has a unique identifier as well, so that two arcs carrying exactly the same information can co-exist in the graph);
- each arc is labeled with the name of the most specific relationship it belongs to in the ontology; it is also labeled with all the domain names for which it is relevant;
- arcs include attribute-value maps expressing the values of the properties associated to their relationship and to all of its generaizations (by inheritance).

Note that graphs can only represent binary relationships through arcs. For n-ary relationships ($n > 2$), reification is needed: the relationship is represented as an entity and its instances as nodes, and the arguments of the original relationship are connected by arcs to the node. n-ary relationships are expressed in the ontology as relationships that, in addition to the subject and object, also have additional attributes that are entity instances. Nodes representing reified relationships can be easily distinguished in GraphBRAIN since they get a label with the relationship name, which starts with a lowercase letter.

To make the KB self-contained, also the ontological items are described as instances in the KB, just like for the SW. Instances may also have attachments, making GraphBRAIN a digital library, whose content is organized according to formal ontologies, fostering interoperability with other systems. Users may add, display, or delete attachments.

In addition to basic KBMS functions, GraphBRAIN also provides its users with several advanced functionalities they can apply to the available knowledge. Currently included functions are:

- assess relevance of nodes and arcs in the graph, and extract the most relevant ones, using Network Analysis algorithms;
- extract a portion of the graph that is relevant to some specified starting nodes, using graph traversal algorithms;
- extract frequent patterns and associated sub-graphs, using Graph Mining algorithms;
- predict possible links between nodes;
- retrieve relevant knowledge using Information Retrieval and Extraction techniques;
- recommend relevant knowledge items;
- carry out high-level automated reasoning on the available knowledge (the main focus of this paper, see next section);
- interact bidirectionally with SW resources;
- express into natural language the information content of a portion of the graph.

If available, a user profile can be used to personalize the results of all these algorithms. This would ensure that each user obtains tailored information. The user profile takes the form of a set of weights, associated to the ontological elements (entity, relationship or attribute) or to specific nodes or arcs (i.e., entity or relationship instances). The weights are formed and continuously updated by taking into account both explicit preference indications by the users and implicit preferences computed on usage.

As said, both ontologies and instances may be imported from, or exported to, the standard SW format Ontology Web Language (OWL)[2], in order to support their interoperability and reuse as Linked Open Data (LOD) [16]. Still, the GraphBRAIN KG is not available in its entirety as LOD. Privacy is obtained by associating with each ontological element a *privacy* attribute, which can be set to True or False. When True, the owner of a knowledge item (an entity or relationship instance, i.e., a node or arc in the graph) can set its privacy value to Private (visible only to its owner), Restricted (only selected users

[2]http://www.w3c.org/owl

can access it), or Public (visible to all users and publishable as open data). When Restricted, he may specify, for each node or arc as a whole, or for its labels and attributes, which users may access it, and the specific kinds of access allowed.

Personalization and Privacy are handled in GraphBRAIN with a system of registered users. In a collaborative spirit, users may add comments on (to provide suggestions or add information), or approve/disapprove, each entity or relationship instance, each single attribute value thereof, and even the ontological items. This feedback is used to assign a trust value to the users which in turn may reflect the reliability of the knowledge they provide.

GraphBRAIN comes in the form of an API, that any interested application must use. Each function in the platform is exposed as a service by the API, and the API wraps the KG ensuring that every interaction is controlled and compliant with the specified ontology.

A general-purpose KG management interface for using the various features of GraphBRAIN was also developed[3]. It includes a form-based interface that allows users to manually insert/update/remove instances or to query the knowledge base for instances of entities and relationships: they must select one of the available domains/schemas, and the forms are automatically generated by the system starting from the corresponding ontologies. The knowledge base can also be fed by automatic knowledge extraction from documents and other kinds of resources (e.g., books or the Internet). It also includes another interface that allows users to display a portion of the graph, browse it interactively and display detailed information about entity and relationship instances. This allows the user to continue his search in a less structured way, by exploring the available knowledge without a predefined goal in mind, letting the data themselves drive the search, possibly finding relevant information in a serendipitous way.

3. Related Work: Automated Reasoning in Logic Programming

Research in AI investigated many approaches to simulate the inference strategies used by humans, proposing automated procedures that implement them, especially within the Logic Programming (LP) framework[4]. This makes LP a good candidate for their integration in an overall framework. Here we provide an overview of various inference strategies that have been investigated in connection with the LP framework (and thus are amenable for integration), and on their implementation.

Deduction Deduction is aimed at making explicit knowledge that is only implicit in the available knowledge, but is a strict consequence thereof. Deductive inference can be carried out using two strategies: *backward* (goal-driven) or *forward* (data-driven) . For the backward approach, we use the *refutation* procedure.

If the body of clauses may contain negated literals, the resulting programs are called *general* logic programs. In the backward approach, such negated literals are proved using the *Negation as Failure* (NAF) rule [17], that stems from the Closed World Assumption (only what is reported in the program is true; whatever is not reported in the program is considered as false).

Abstraction Abstraction reduces the amount of information conveyed by a set of facts, called the *reference set* [18]. This reduces the computational load needed to process the set of facts, provided that the information that is relevant to the achievement of a goal is preserved. we adopt the framework proposed in [19], where abstraction happens by means of a set of operators.

Abduction Abduction is devoted to cope with missing information, by guessing unknown facts when they are needed for a given purpose. The Abductive Logic Programming (ALP) [20, 21] framework extends deduction by allowing to guess some 'abducible' facts (abductive hypotheses) that are not stated in the available knowledge but are needed to explain given observations, provided that they

[3] A demo can be found at `http://193.204.187.73:8088/GraphBRAIN/`

[4] LP is a fragment of First-Order Logic based on *clauses*, i.e., implications of the form $C \Leftarrow P_1 \wedge \cdots \wedge P_n$. Full implications are called *rules*, clauses with no preconditions are called *facts*, and clauses with no conclusion are called *goals*.

are consistent with given Integrity Constraints (formulas that that must be satisfied by the abductive hypotheses). The set of guessed facts is called an 'explanation'. Of course, there may be many plausible explanations for a given observation, and thus abductive explanations are not conclusive, requiring strategies to filter and rank explanations. Among the various procedures proposed in the literature to obtain abductive explanations for abductive logic programs, also when negated literals are used in the body [22], we use the one proposed by [23]. We adopt *Expressive ALP* (EALP), an extended version of ALP allowing a wider variety of operators, proposed in [24].

Uncertain Reasoning Much research investigated how to combine logical and statistical inference, so that the former supports high-level reasoning strategies, and the latter improves flexibility and robustness. From an LP perspective, they resulted in the *Probabilistic Logic Programming* (PLP) setting [25]. A probabilistic logic program defines a probability distribution over a set of normal logic programs (called *worlds*). Different languages have been proposed, that differ in the way they define the distribution. Some allow to set probabilities only on facts; some allow two alternatives only (true or false)some offer a more general syntax than others. We use LPADs [26].

A different approach to uncertainty is based on an informal but quick ways of estimating the certainty of the information they handle. We opt for the approach aimed at simulating this behavior implemented in the famous expert system MYCIN [27], inspired by fuzzy set theory [28].

Argumentation Argumentation aims at dealing with inconsistent knowledge, in order to distinguish which of several contrasting positions in a dispute are justified. In a dispute, the participants make claims (the *arguments*) to support their own position, to attack the arguments for competing positions of the other participants, and to defend their position from the attacks of the others. Abstract argumentation, stemmed from ALP, focuses only on the inter-relationships among the arguments, neglecting their internal structure or interpretation.

Abstract Argumentation Frameworks (AFs) [29] can be represented as graphs, where nodes are the arguments and arcs represent the relationships between pairs of arguments. We adopt the *Generalized Argumentation Framework* (GAF) [30] extension of traditional AFs, a much more powerful model which provides bipolarity (the possibility of expressing both attacks and supports between pairs of arguments) and weights on both the arguments and the attack/support relationships (denoting their strength), and is compatible with the most prominent extensions proposed in the literature. In particular, the T-GAFs specialization of the GAF model introduces community and topics as a context of the arguments that can affect their reliability.

Induction The term *induction* refers to the inference of general rules or theories starting from specific instances. *Observations* are descriptions of objects or situations as 'perceived' from the world. *Examples* are labels assigned to observations to explicitly specify what are the concepts of interest (to be learned) in the observations. Examples can be positive (representing instances of the concepts) or negative (representing instances that do not belong to a concept). *Inductive Learning* aims, given a set of examples concerning some concept, at extracting a model (i.e., a characterization) of that concept.

Inductive Logic Programming (*ILP*) [31] is a branch of Machine Learning exploiting LP as a representation language. Some ILP systems work in a *batch* way: they start from an empty theory and stop the inductive process when the current set of hypotheses is able to explain all the available examples. When new evidence contradicts the learned theory, the whole process must be restarted from scratch, taking no advantage of the previously learned hypotheses. Other systems can revise and refine a theory in an *incremental* way: they try to change the previously generated hypotheses in such a way that the changed hypotheses explain both the old and the new examples. We perform induction using the incremental ILP system InTheLEx [32].

Ontological Typical ontology-based reasoning tasks of interest are satisfiability (checking if the described world may exist), instance checking (checking whether an instance belongs to a certain

concept), concept satisfiability (checking if a concept may exist in the described world), subsumption (checking if a concept is a subclass of another concept), equivalence (checking if two classes are the same), retrieval (of the set of instances that belong to a certain concept), extraction of super-/sub-classes, relationships and properties of a given concept. Particularly relevant is the relationship of generalization/specialization, on which inheritance can be applied.

The research on ontologies evolved separately from LP, and relied on the Description Logics [33] fragment of FOL. Different description logics can be defined, depending on the available operators. Adding more and different operators extends the expressive power of a DL, but may lead to indecidability. Unfortunately, DLs are only partially overlapping to LP, and the non-overlapping parts are incompatible, mainly due to the different fundamental assumption they make on unknown information (Open World Assumption in DLs vs Closed World Assumption in LP). Ontologies can be translated to default logic [34], one of the most famous formalisms for non-monotonic reasoning.

Similarity Similarity computation between FOL descriptions is complex due to *indeterminacy* (the possibility of mapping various portions of one description in many ways onto another description). For this reason, very few works in the literature tackled this problem. We adopt the approach proposed in [35]. It considers a set of parameters and defines a similarity function based on them, plus a set of criteria to assess the similarity for different clause components. The parameters it uses for comparing two objects i' and i'' are widely accepted in the literature [36]. The similarity criteria deal with increasingly complex clause components: terms, atoms, groups of atoms, clauses. The similarity of more complex components is based on the similarity of simpler components. In FOL formulæ, terms represent specific objects, while predicates express their properties and relationships. Accordingly, two levels of similarity can be defined for pairs of FOL descriptions: the *object* level, concerning similarities between the terms referred to in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

Analogy *Analogy* is the cognitive process of matching the characterizing features of two items (subjects, objects, situations, etc.). It allows one to reuse knowledge from a known item or domain (called the *base*) to an unknown one (called the *target*). While similarity is a syntactic task that looks for exactly the same features in two items, analogy maps 'roles', which has to do with semantics (i.e., the meaning). The mapping is bi-directional, while in metaphors it only holds in one direction. After finding an analogy on some roles, the association can be extended to further missing features. The analogy may depend on the context, goal or perspective, and its outcome might be inconsistent with previous knowledge [37].

Analogical reasoning consists of 5 steps [38]: (1) *Retrieval* finds the best base domain that may help to solve the problem in the target domain; (2) *Mapping* looks for a mapping between base and target domains; (3) *Evaluation* provides criteria to evaluate candidate mappings; (4) *Abstraction* shifts the representation of both domains to their roles' schema, converging to the same analogical pattern; (5) *Re-representation* adapts one or more pieces of the representation to improve the matching. A procedure that, applied to two descriptions, returns possible analogies between them is called an *analogy operator*. The analogy setting we adopt, specifically based on LP formalisms, was provided in [39].

4. Multistrategy Reasoning for Explainability in GraphBRAIN

As noted, single inference strategies have been typically studied in isolation or in combinations of very small sets (most often just pairs). This limitation motivated a new research direction, named *Multistrategy Reasoning* (MSR), as an extension of the *Inferential Theory of Learning* (ITL) [18] theoretical framework (developed from the specific perspective of learning agents) specifically aimed at combining as many approaches as possible, without giving priority to any of them. We now describe the MSR framework underlying GraphBRAIN's automated reasoning capabilities.

4.1. Combination of Inference Strategies

There are many interconnections among the inference strategies proposed above so that they can help each other in accomplishing their tasks. Here we describe the integration approaches and their implementations adopted by in GraphBRAIN.

Ontologies and Logic Programming While the realms of Logic Programming and Description Logics, used to specify ontologies, are partly incompatible, attempts to merge them have been made in the literature. We adopt $\mathcal{DL} + log$ [40], as the most powerful decidable combination of Description Logics and disjunctive Datalog rules (i.e., Datalog rules whose head may consist of a disjunction of atoms). This language allows to mix 'Datalog predicates', coming from the LP perspective, from 'DL predicates', coming from the ontological perspective, in a clause, but DL predicates cannot be negated.

Abduction and Deduction The integration of abduction and deduction has been defined in [20, 41], to allow reaching conclusions or making prediction when the available information is insufficient.

Similarity for Deduction Similarity can be used both to help a subsumption procedure to converge quickly towards the correct associations [35], and to weaken subsumption and obtain a *flexible matching* procedure that returns a degree of matching instead of a boolean decision.

Abstraction, Deduction, Similarity, Abduction and Argumentation for Induction A very interesting case of the use of several strategies in support of induction is provided by the incremental ILP system InTheLEx [32, 42].

Abstraction is carried out as a pre-processing step that removes useless information according to the framework in [19]. This is obtained by expressing abstraction operators as clauses, such that whenever the body is recognized in an observation, the involved facts are replaced by those in the head, suitably defined to hide the useless information.

Deduction is used in a *saturation* step that makes explicit facts that are implicit in the available description of the observations and that may be useful to correctly grasp the concept that is being learnt. To do this, deduction exploits the rules in the KB. Whenever the body of a clause in the theory is recognized in an observation, the head of the clause is added to the observation itself.

Abduction is used to check if an unexplained example/observation can be explained by assuming additional unseen information that is not present in the observations. In such a case, the guessed information is added to the example description. This prevents the refinement operators from being applied, and the theory from being changed.

Similarity is used to guide the generalization operator, by taking the paths univoquely determined according to the technique proposed in [35], and using a greedy techinque that adds the generalization of these paths by decreasing similarity, as long as they are compatible. Further generalizations can then be obtained through backtracking [43].

Recently, argumentation has been integrated to identify consistent portions of inconsistent observations and to choose the one to rely on, exploiting the same integrity constraints defined for abduction to identify attacks and supports (an example of a further integration of different strategies) [44].

Induction for Abduction, Abstraction and Deduction Abstraction operators, integrity constraints for abduction (and argumentation), and rules for saturation can be inductively learned from observations, as shown in [45, 46]. Combinations of facts that never occur generate integrity constraints for abduction (that can be used also for generating abstract argumentation frameworks [44]). Combinations that always occur generate abstraction operators. Concept definitions learned by an ILP system can be used to identify known concepts in observations when learning other concepts.

Argumentation and Induction for Analogy The analogy operator defined in [39] leverages argumentation to overcome the constraint that using the same descriptors in the two domains means that they necessarily denote the same roles. All possible analogical mappings between descriptors are considered, and mappings are inconsistent if they map one feature in one domain onto many features in the other. These inconsistencies are expressed as attacks in an argumentation framework, and abstract argumentation strategies are used to select only consistent associations.

In the same paper, the use of an inductive (generalization) operator to obtain more general knowledge structures that can be mapped onto several domains is also proposed.

Abduction and Probabilistic Reasoning While, from a logical standpoint, all consistent abductive explanations are equally good, in a probabilistic setting different explanations of a goal are associated to different possible worlds, and their validity depends on the validity of the rules, facts and integrity constraints used to obtain those worlds. PEALP takes into account all these items [24, 47].

A world that violates a probabilistic integrity constraint is not impossible, just differently probable. So, all (minimal) abductive explanations must be obtained to identify the most likely one, and whenever the abductive procedure has a choice, it must explore the worlds associated to all different options.

Cues for Further Cooperations As observed in the ITL, analogy is strictly connected to abstraction and deduction, since these two strategies are needed to go from a specific domain to its abstract structure and then from the latter to a new specific domain. It also has strict connections to abduction, that can be used to guess information in the target domain that is not observed but is analogous to information available in the source domain.

Argumentation can help deduction to resolve inconsistencies and determine (possibly different settings of) consistent information on which deduction can be carried out.

Uncertain reasoning allows to add flexibiity to all the others. Especially interesting is combining it with argumentation (to determine how reliable each consistent setting is and rank different settings) and induction (to assign a degree of reliability to the learned knowledge).

4.2. Implementation of Explainable Multistrategy Reasoning in GraphBRAIN

An implementation of the MSL framework has been started, resulting in the GEAR (acronym for 'General Engine for Automated Reasoning') inference engine [48]. GEAR tracks all reasoning steps, and can provide a full account/explanation of its outcomes, that can be analyzed and browsed by the users. GEAR is written in Prolog language, because it provides native support to deduction, unification, manipulation of logic representations, and Logic Programming. Knowledge bases handled by GEAR may include various kinds of knowledge items, including Facts, Rules, Integrity Constraints, Abstraction Operators, and Argumentative relationships. Uncertainty is expressed by values in $[0, 1]$, inspired by mathematical probability theory.

The main components of a KB are facts and rules. Facts are formalized as

$$\texttt{fact([}M\texttt{,}I\texttt{],}\ F\texttt{,}\ C\texttt{).}$$

while rules are formalized as

$$\texttt{rule([}M\texttt{,}I\texttt{],}\ H\texttt{,}\ B\texttt{,}\ P\texttt{,}\ C\texttt{).}$$

where I is the unique identifier of the fact or rule, and $C \in]0, 1]$ is the certainty value (1 meaning 'absolutely' true and 0 meaning 'absolutely' false). F is an atom, while H and B are the rule's head and body, respectively, and P is its priority (a number used to determine which rule should be executed first in case of conflicts).

B is a logistic expression built on the following operators:

and([C_1**,...,**C_n**])** representing the conjunction (AND) of the C_i's;

or([C_1**,...,**C_n**])** representing the disjunction (OR) of the C_i's;

no(C**)** representing a 'probabilistic' negation (NOT) of C;

not_exists(C**)** representing an 'existential' negation of C;

where the C's are atoms or nested operator applications, to express complex conditions. H is one of the following:

C an atom;

and([C_1**,...,**C_n**])** representing the conjunction (AND) of the atoms C_i;

or([C_1**,...,**C_n**])** representing the disjunction (OR) of the atoms C_i;

no(C**)** representing a 'probabilistic' negation (NOT) of the atom C.

Abducibles are formalized as

$$\texttt{abducible}(P/N).$$

where P is the predicate name and N is its arity, while integrity constraints for abduction and argumentation are formalized as

$$\texttt{ic}(I,\ O,\ C).$$

where I is the unique identifier of the constraint, C is its certainty value, and O is one of the following:

nand([l_1**,...,**l_n**])** at least one among literals l_1,...,l_n must be false;

xor([l_1**,...,**l_n**])** exactly one among literals l_1,...,l_n must be true;

or([l_1**,...,**l_n**])** at least one among literals l_1,...,l_n must be true;

if([l'_1**,...,**l'_n**],[**l''_1**,...,**l''_m**])** if all literals l'_1,...,l'_n are true, then all literals l''_1,...,l''_m must also be true (*modus ponens*); alternatively, if all literals l''_1,...,l''_m are false, then all literals l'_1,...,l'_n must also be false (*modus tollens*);

iff([l'_1**,...,**l'_n**],[**l''_1**,...,**l''_m**])** either all literals l'_1,...,l'_n and l''_1,...,l''_m are true, or all literals l'_1,...,l'_n and l''_1,...,l''_m are false;

and([l_1**,...,**l_n**])** all literals l_1,...,l_n must be true;

nor([l_1**,...,**l_n**])** all literals l_1,...,l_n must be false.

Abstraction operators are formalized as

$$\texttt{abstraction}([M,I],\ A,\ G).$$

where I is the unique identifier of the operator, and A is the abstracted set of atoms that replaces the ground set of atoms G whenever it is found in an observation.

Identifiers of knowledge items are in either of the following forms:

I a general unique identifier for the item in the overall KB;

[M**,**I**]** I is the unique identifier of the item within knowledge module M.

Finally, argumentation works on the following predicate:

$$\texttt{arg}(I,\ S).$$
$$\texttt{arg\_rel}(I',\ I'',\ S').$$

where I, I' and I'' are fact identifiers, $S \in [0, 1]$ is the strength of the argument and $S' \in [-1, +1]$ expresses the type (attack or support, based on the sign) and strength of the argumentative relationship.

Other predicates can be used to specify system settings (e.g., `gear_flag` allows to set flags that direct the system's behavior), information related to user interaction (e.g., `askable` specifies information that can be asked to the user if missing in the KB), calls to pre-defined procedures (e.g., `call` may call Prolog to carry out some computations), and others, but they are beyond the scope of this paper.

GEAR has been connected to GraphBRAIN via an export procedure of GraphBRAIN that generates facts from the instances in the graph and other knowledge items (rules, integrity constraints, etc.) from the axioms specified in the ontologies and from the ontological properties expressed in the definition of the ontological elements.

The predicates of the language are derived directly from the ontology, as follows:

- an instance with identifier I of entity E, which is associated with attributes $A_1, \ldots, A_n$, is expressed using a predicate $E(I, A_1, \ldots, A_n)$;
- an instance with identifier I of relationship R, which is associated with attributes $A_1, \ldots, A_n$, connecting subject instance (node) S to object instance (node) O is expressed using a predicate $R(I, S, O, A_1, \ldots, A_n)$;
- an instance with identifier I of an entity representing a reified relationship R, which connects entity instances $E_1, \ldots, E_m$ and is associated with attributes $A_1, \ldots, A_n$, is expressed using a predicate $R(I, E_1, \ldots, E_m, A_1, \ldots, A_n)$.

where the order of the instances and attributes in the arguments is as specified in the ontology.

GraphBRAIN can provide the facts to GEAR both in batches, by translating a relevant portion of the KG determined according to suitable algorithms, or on-demand, by generating single facts needed by the inference engine to carry on the current reasoning task.

Figure 1 shows two steps in the use of GEAR inside GraphBRAIN: on the top, the knowledge extraction algorithms have selected a (possibly personalized) subgraph of the KG starting from the nodes listed in the table on the left and with thicker borders in the graph displayed in the middle. After exporting this knowledge in GEAR format, GEAR can be applied. The bottom screenshot shows the interface from which the user may carry out several functions and get the results. Of interest here is the section on the top left of the window, where the explanation for the obtained outcome is displayed and can be interactively browsed by the user.

The proposed approach has been used so far in two application domains. The former is a porting to GEAR of an older version of the inference engine, called ReLay, and of the associated KB to GraphBRAIN representation. It was about the diagnosis of eating disorders (Figure 1 refers to this domain). Another, more recent application, is about Cultural Heritage (CH), for which GraphBRAIN is very suitable because it can enforce privacy, which is very important for some collections (especially private ones), and personalization, useful to support the needs of very different stakeholders (professionals, researchers, scholars, hobbyists, enthusiasts, tourists, curious people) with different background, culture, interests, aims, contexts, expectations, etc.

5. Conclusions and Future Work

Artificial Intelligence is nowadays pervading all aspects of our lives, starting to cover also applications that involve crucial aspects of human lives or well-being. In these cases, impacting directly our existence, we the humans need to stay in control, checking and validating its decisions. In turn, this requires them to be explainable. In spite of many attempts to simulate explainaility in sub-symbolic AI approaches, only the symbolic one, based on formal logics, is explainable-by-design. It applies human-like automated reasoning and relies on suitable knowledge representations, ensuring semantic and procedural interoperability with humans and human thought. The most widely known and adopted knowledge representation approach in the current research landscape is Knowledge Graphs, mostly

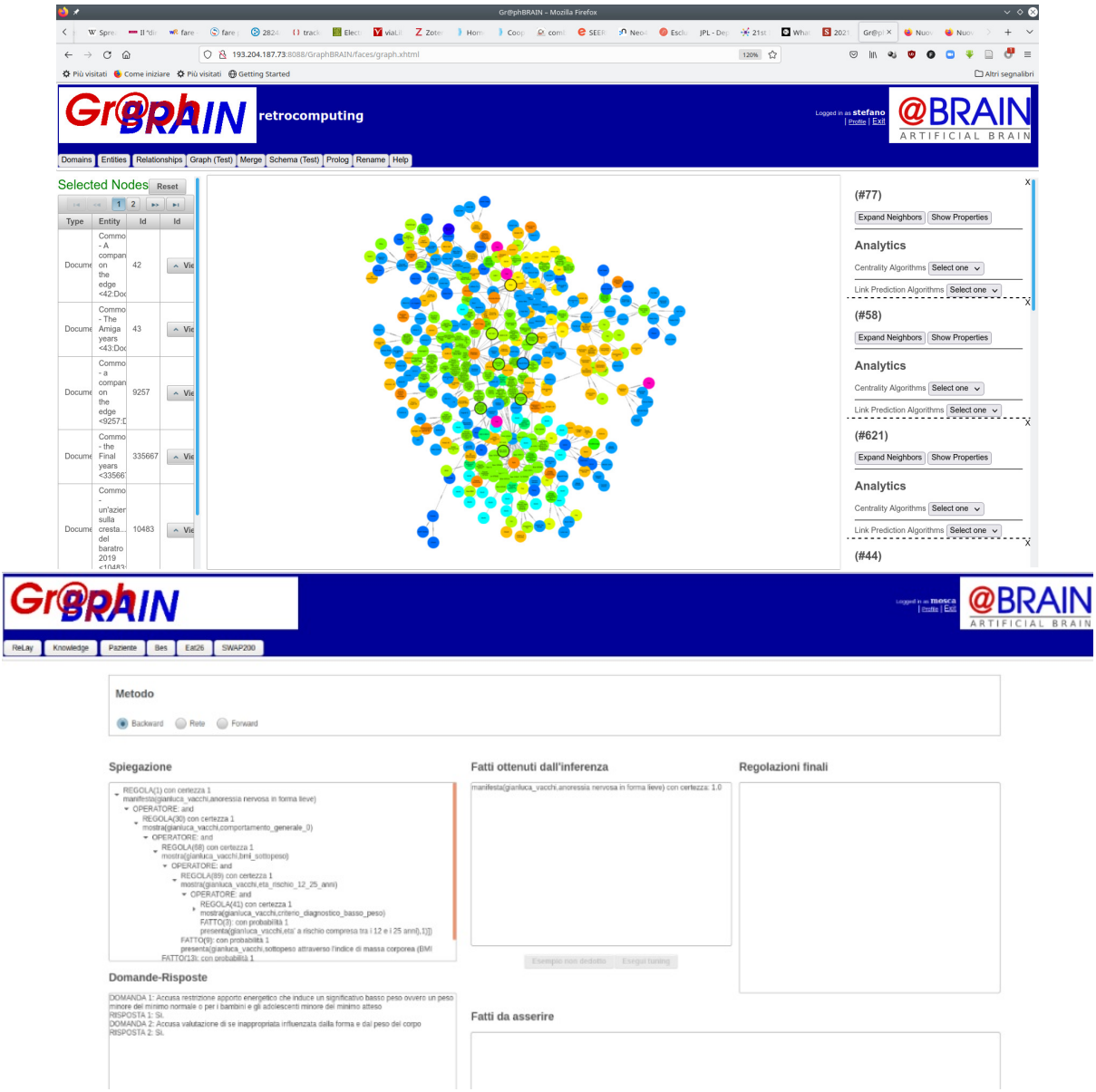

Figure 1: Connection between GraphBRAIN and GEAR: Knowledge extraction (top screenshot) and reasoning dashboard with an explanation (bottom screenshot)

investigated by the Semantic Web community, but due to its peculiarities the Semantic Web perspective and standards are affected by some limitations and shortcomings.

In this paper we proposed GraphBRAIN, an alternate KG framework, based on state-of-the-art DB technology and on a different graph model, allowing richer representations. By not being tightly coupled to the Semantic Web representations, this framework also expands the range of possible inference strategies to be used in automated reasoning, which in turn may better support explainability of the AI outcomes. In particular, we described how a Multistrategy Reasoning framework can be applied to GraphBRAIN, through the GEAR inference engine, that is specifically designed for supporting explainability of its outcomes.

Ongoing and future work is aimed at enriching the knowledge representation in GraphBRAIN, in order to support more advanced approaches to automated reasoning. Development of GEAR is also being carried on, to better integrate the different inference strategies and expand them. Finally, the theoretical and practical results of this effort are being applied to real-world tasks.

Acknowledgments

This research was partially supported by projects CHANGES "Cultural Heritage Active Innovation for Sustainable Society" (PE00000020), Spoke 3 "Digital Libraries, Archives and Philology" and FAIR "Future AI Research" (PE00000013), spoke 6 "Symbiotic AI", funded by the Italian Ministry of University and Research NRRP initiatives under the NextGenerationEU program.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] D. Michie, R. Johnston, The creative computer : machine intelligence and human knowledge, Viking, 1984.

[2] O. G. Yalçin, 5 significant reasons why explainable ai is an existential need for humanity https://towardsdatascience.com/5-significant-reasons-why-explainable-ai-is-an-existe 2020 (accessed 6 December 2022).

[3] M. Minsky, S. Papert, Perceptrons (2nd ed.), MIT Press, 1988.

[4] M. Genesereth, N. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann, 1987.

[5] O. G. Yalçin, Symbolic vs. subsymbolic ai paradigms for ai explainability https://towardsdatascience.com/symbolic-vs-subsymbolic-ai-paradigms-for-ai-explainab 2021 (accessed 6 December 2022).

[6] D. Kahneman, Thinking, Fast and Slow, Farrar, Straus and Giroux, 2011.

[7] M. Minsky, Logical versus analogical or symbolic versus connection or neat versus scruffy, AI Magazine 12 (1991) 34–51. URL: https://web.media.mit.edu/~minsky/papers/SymbolicVs.Connectionist.html.

[8] E. Parliament, Council, Regulation (eu) 2024/1689 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act), 2024.

[9] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, Information Fusion 58 (2020) 82–115. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103. doi:https://doi.org/10.1016/j.inffus.2019.12.012.

[10] P. J. Phillips, C. A. Hahn, P. C. Fontana, A. N. Yates, K. Greene, D. A. Broniatowski, M. A. Przybocki, Four Principles of Explainable Artificial Intelligence, Technical Report, ???? doi:10.6028/nist.ir.8312.

[11] M. K. Saker, L. Zhou, A. Eberhart, P. Hitzler, Neuro-symbolic artificial intelligence: Current trends, AI Communications 34 (2022) 197–209.

[12] S. Purohit, N. Van, G. Chin, Semantic property graph for scalable knowledge graph analytics, 2020. URL: https://arxiv.org/abs/2009.07410. arXiv:2009.07410.

[13] I. Robinson, J. Webber, E. Eifrem, Graph Databases, 2nd ed., O'Reilly Media, 2015.

[14] S. Ferilli, Integration strategy and tool between formal ontology and graph database technology, Electronics 10 (2021). URL: https://www.mdpi.com/2079-9292/10/21/2616.

[15] R. Angles, A. Bonifati, S. Dumbrava, G. Fletcher, A. Green, J. Hidders, B. Li, L. Libkin, V. Marsault, W. Martens, F. Murlak, S. Plantikow, O. Savkovic, M. Schmidt, J. Sequeda, S. Staworko, D. Tomaszuk, H. Voigt, D. Vrgoc, M. Wu, D. Zivkovic, Pg-schema: Schemas for property graphs, Proc. ACM Manag. Data 1 (2023).

[16] T. Heath, C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Synthesis Lectures on the Semantic Web, Morgan & Claypool Publishers, 2011.
[17] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Databases, Plenum Press, 1978, pp. 293–322.
[18] R. Michalski, Inferential theory of learning. developing foundations for multistrategy learning, in: R. Michalski, G. Tecuci (Eds.), Machine Learning. A Multistrategy Approach, volume IV, Morgan Kaufmann, San Mateo, CA, 1994, pp. 3–61.
[19] J.-D. Zucker, Semantic abstraction for concept representation and learning, in: R. S. Michalski, L. Saitta (Eds.), Proceedings of the 4th International Workshop on Multistrategy Learning, Desenzano d.G, Italy, 1998, pp. 157–164.
[20] A. Kakas, R. Kowalski, F. Toni, Abductive logic programming, Journal of Logic and Computation 2 (1993). 718–770.
[21] M. Denecker, A. Kakas, Abduction in logic programming, in: Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski, volume 2407 of *Lecture Notes in Computer Science*, Springer, 2002, pp. 402–437.
[22] M. Denecker, D. D. Schreye, Sldnfa: An abductive procedure for normal abductive programs, in: Proceedings of ICSLP, MIT Press, 1992, pp. 700–868.
[23] A. Kakas, P. Mancarella, On the relation of truth maintenance and abduction, in: Proceedings of the 1st Pacific Rim International Conference on Artificial Intelligence, Nagoya, Japan, 1990.
[24] S. Ferilli, Extending expressivity and flexibility of abductive logic programming, Journal of Intelligent Information Systems 3 (2018) 647–672.
[25] F. Riguzzi, Foundations of Probabilistic Logic Programming: Languages, semantics, inference and learning, River Publishers, Gistrup, Denmark, 2018.
[26] J. Vennekens, S. Verbaeten, M. Bruynooghe, Logic programs with annotated disjunctions, in: B. Demoen, V. Lifschitz (Eds.), 20th International Conference on Logic Programming (ICLP 2004), volume 3131 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 431–445. doi:10.1007/978-3-540-27775-0_30.
[27] E. H. Shortliffe, B. G. Buchanan, A model of inexact reasoning in medicine, Mathematical Biosciences 23 (1975) 351–379.
[28] L. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[29] P. M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial intelligence 77 (1995) 321–357.
[30] S. Ferilli, Introducing general argumentation frameworks and their use, in: AIxIA 2020 (reboot) - The 19th International Conference of the Italian Association for Artificial Intelligence, volume 12414 of *Lecture Notes in Artificial Intelligence*, Springer, 2021, pp. 136–153.
[31] S. Nienhuys-Cheng, R. de Wolf, Foundations of Inductive Logic Programming, volume 1228 of *Lecture Notes in Artificial Intelligence*, Springer, 1997.
[32] F. Esposito, G. Semeraro, N. Fanizzi, S. Ferilli, Multistrategy theory revision: Induction and abduction in inthelex, Machine Learning Journal 38 (2000) 133–156.
[33] The Description Logic Handbook: Theory, Implementation and Applications, 2 ed., Cambridge University Press, 2007. doi:10.1017/CBO9780511711787.
[34] Y. Sun, Y. Sui, Translating ontologies to default logic, in: AIAI, 2005.
[35] S. Ferilli, T. Basile, M. Biba, N. Di Mauro, F. Esposito, A general similarity framework for horn clause logic, Fundamenta Informaticae 90 (2009) 43–66.
[36] D. Lin, An information-theoretic definition of similarity, in: Proc. 15th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 296–304. URL: citeseer.ist.psu.edu/95071.html.
[37] D. O'Donoghue, M. T. Keane, A creative analogy machine: Results and challenges, in: Proceedings of the Third International Conference on Computational Creativity, Dublin, Ireland, 2012, pp. 17–24.
[38] D. Gentner, Analogy, A companion to cognitive science (1998) 107–113.
[39] F. Leuzzi, S. Ferilli, A multi-strategy approach to structural analogy making, J Intell Inf Syst 50

(2018) 1–28.
[40] R. Rosati, Dl+log: Tight integration of description logics and disjunctive datalog, in: Proc. of Tenth International Conference on Principles of Knowledge Representation and Reasoning, AAAI Press, 2006, pp. 68–78.
[41] A. Kakas, P. Mancarella, Generalized stable models: a semantics for abduction, in: Proceedings of the 9th European Conference on Artificial Intelligence, Pitman Publishing, 1990, pp. 385–391.
[42] F. Esposito, N. Fanizzi, S. Ferilli, T. M. Basile, N. Di Mauro, Multistrategy operators for relational learning and their cooperation, Fundamenta Informaticae 69 (2006) 389–409.
[43] S. Ferilli, T. M. A. Basile, N. Di Mauro, M. Biba, F. Esposito, Similarity-guided clause generalization, in: AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 278–289.
[44] A. Pazienza, S. Ferilli, Exploring abstract argumentation-based approaches to tackle inconsistent observations in inductive logic programming, in: AI*IA 2018 – Advances in Artificial Intelligence, Springer International Publishing, Cham, 2018, pp. 279–292.
[45] S. Ferilli, T. M. A. Basile, N. Di Mauro, F. Esposito, On the learnability of abstraction theories from observations for relational learning, in: Machine Learning: ECML 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 120–132.
[46] F. Esposito, S. Ferilli, T. Basile, N. Di Mauro, Inference of abduction theories for handling incompleteness in first-order learning, Knowledge and Information Systems 11 (2007) 217–242.
[47] D. Azzolini, E. Bellodi, S. Ferilli, F. Riguzzi, R. Zese, Abduction with probabilistic logic programming under the distribution semantics, International Journal of Approximate Reasoning (2022) 41–63.
[48] S. Ferilli, Gear: A general inference engine for automated multistrategy reasoning, Electronics 12 (2023).