Reasoning Web 2023, Oslo, Norway - Summary

Session 1: Declarative AI for Industry: methods, applications, trends

Speaker: Evgeny Kharlamov, Bosch Center for AI & University of Oslo

Topics:

- Evolution of declarative AI in industry
- Semantics in industry in a nutshell
- What, where and how of current hype for semantics in the industry
- Semantic Digital Twins
- Neurosymbolic AI

Summary:

Data Silos challenge: Companies (mostly refers to manufacturing companies) don’t know what they know because they can’t process the data and get the most out of it. Also many companies that provide industrial applications require proper industrial data but industrial companies are not able to provide it. Because the data they have is big, scattered and heterogeneous.

Solution: Data Silos → Extraction, Transformation and Loading (ETL) → Integrated Info + Industrial KGs → Insights using AI → Value Generation. (Declarative AI)

Major bottleneck of Declarative AI is that there is not enough advanced semantic tools, and also people with PhD or high level of understanding to use or develop those tools. Knowledge engineering is usually hard for the industry.

KGs in industry is replacing relational data warehouses. Because KGs are user-oriented, makes it easy to pass data around, has flexible schema, and can be used to enhance data intensive tasks.

Semantic Digital Twins: Interest in semantics is declining in research, but increasing in the industry. Semantics was too futuristic for many companies, until it became impossible to manage complexity of working with industrial data, so that physical and digital space had to be separated (digital twins, semantic digital twins, or cognitive twins). Factories became fully computerized software-driven systems and 40% of the overall cost comes from software (for Bosch).

Neurosymbolic AI: For industry experts and systems that has been working with ontology-based reasoners, DL is a black box that no one understand why it does what it does. Domain expertise can be incorporated into ML/DL via KG, Semantic DT, and rules, which is referred as neurosymbolic integration. **After ChatGPT**, neural part is being replaced by LLMs, at least there is heavy research to achieve this.

Session 2: Ontologies and Constraints

Speaker: Martin Giese, University of Oslo

Topics:

- What is Validation?
- Shapes Constraint Language (SHACL)
- Epistemic Approaches to Validation

Summary:

Constraints represent knowledge about the representations of ontologies **and** knowledge graphs. Validation can be achieved via constraints. Constraints are written using: i) OWL as constraint language, ii) Epistemic Description Logic: adds a knowledge operator K, which allows us to talk about knowledge that we have on the data. It can get highly complex mathematically, and does not use triples which are very useful computationally in many cases, iii) SPARQL queries to check constraints, iv) W3C RDF Data Shapes Working Group: create a language for defining structural constraints on RDF graphs. What must be in the graph → Shape Expressions (ShEx). Which violations to check for → Shape Constraint Language (SHACL).

Session 3: Compact Rewritings for Existential Rules

Speaker: Micheal Thomazo, National Institute for Research in Digital Science and Technology (Inria)

Topics:

- Ontology-based Query Answering
- Query Rewritability
- (Union of Conjunctive Query) UCQ-Rewriting Algorithms
- Compacting UCQ-Rewritings

Summary:

Query rewriting is a subproblem of ontology-based data access, which is categorized under data integration. Ontology plays the role of a vocabulary across different

databases and eases the data integration. Ontology-based query answering is an undecidable problem and the goal of query rewriting is to make it decidable by applying some restrictions to the ontology language.

The naive approach is to incrementally apply queries to instances in a DB. It is a well-defined but non-terminating approach. The idea is to rewrite to query using already existing relations in the relational database, which entails the initial query, but terminates. After rewriting, a compacting step is applied which utilizes union operations to shorten the query.

Session 4: Termination of Reasoning

Speaker: Andreas Pieris, University of Edinburgh, UK and University of Cyprus, Cyprus

Topics:

- Rule-based Ontologies
- Reasoning with Rule-based Ontologies
- Checking Termination Conditions in Rule-based Reasoning
- Main Techniques for Termination of Reasoning

Summary:

“If some facts are true, then some other facts are also true”. Such statements/rules can be created using ontology-based reasoning. The rules allow for inferring new implicit information via reasoning that is not explicitly available in data. One of the challenges to rule-based reasoning is the termination problem. Two things can prevent ontology-based reasoning from terminating; i) recursive nature of ontological rules, e.g. definition of a predicate may depend on itself, ii) ability of rules to infer new unknown objects that are not mentioned in the data.

Uniform termination of reasoning: given a rule-based ontology O, does reasoning with O terminates with every set of possible inputs? This problem is undecidable, and the proof for undecidabliity is based on non-realistic languages that goes beyond existing “well-behaved” ontologies. Focusing on languages of certain form namely languages that consists of linear existential rules (closed formulas), can make the problem decidable under certain restrictions. A basic approach to reason on linear existential rules is shown during the session. It is based on a concept named “trigger”, a trigger to run a reasoning step which refers to satisfying conditions of left side of a rule. There are two types of triggers; naive triggers and smart triggers. Naive ones are always run as long as the condition is satisfied, while the smart one

also checks for homomorphism of the rules, meaning that each rule should be unique.

Session 5: Graph Queries and Description Logics

Speaker: Filip Murlak, University of Warsaw, Poland

Topics:

- Unrestricted Entailment of Conjunctive Regular Path Queries (CRPQs)
- Finite Query Entailment
- Finite entailment of simple CRPQs and CRPQs

Summary:

Problem: Given a query Q, a graph G and TBox T, how to decide whether G, T → Q hold or not in finite duration? Due to the finiteness paradox, the problem is undecidable. An example to this paradox is the following: if there is a person, that person is a child of another person!?

We reduce the problem to Union of Conjunctive Regular Path Queries (UCRPQs). CRPQs are an extension of conjunctive queries (primitive positive formulas) in which regular expressions over binary predicates can be used as atoms; they are the core of practical query languages used in the Semantic Web and in graph databases. And then we utilize starlike countermodels and factorization method to check for countermodels. Starlike countermodels refers to creating a “star-shaped” extension of a graph by applying homomorphism on its nodes and then applying factorization, i.e. if Q holds in a starlike graph iff it holds in any of its parts. The proof continues with unravelling the extended graph and applying least fixed point. The same problem can also be solved using automata theory.

Some of the current research areas given by the speaker are as follows: i) there are many description logics which we can add new features to in order to solve such problems, e.g. query entailment or reachability problem, and this is not yet well studied, ii) applications of description logics on knowledge graphs, iii) finite query entailment of UCRPQ, in ALCOIF (a description logic).

Session 6: Controlled Query Evaluation in Description Logic Ontologies

Speaker: Riccardo Rosati, Sapienza, University of Rome

Topics:

- Controlled Query Evaluation in DL

- Tractability via: Intersecting the Censors, Adding Preferences, Maximally Cooperative Approach
- Towards Practical Systems: Extension/Reduction to OBDA

Summary:

The main problem tackled in this session is called confidentiality-preserving query answering. Given a set of data access restrictions, e.g. due to confidentiality, how to maximize the amount of information given to the user without revealing any “secret” info/data?

A policy tells us what not to reveal, and it can be stated in the form of logical formulas. Enforcement of policies are formalized through the notion of censor. A censor models the answers that the query answering engine should provide. An optimal censor maximizes query answers still guaranteeing that the policies are not violated.

Two approaches to satisfy a confidentiality-preserving policy in a knowledge base (Assumption: TBox can not be hidden/censored (users know the TBox):

- Materialized approach: modify the knowledge base, without changing the query answering technique
- Virtual approach: modify the query answering technique, without changing the knowledge base

One of the challenges is that user(s) can query multiple times, however answers must be given in a way that none of the restricted information can be re-constructed using answers to different queries.

Prioritized Conjunctive Query Evaluation Framework: If we “have to” give up some restricted info, which ones to give up first? Preferences are taken as input to the system, and they make the problem easier. There is a trade-off between computational requirements and giving correct and tractable answers.

Longest honeymoon approach: Which censor to apply is progressively selected based on user queries. The idea is that how long the engine can continue giving as much information as possible, without starting to give up confidential preferred/prioritized info.

Some of the open problems: i) extend the results to different knowledge bases/databases beyond DLs, ii) improve user/attacker model, policy language, and query language, iii) more powerful notions of censor (not only tuple-deletion based, meaning what can we do besides not giving any info at a certain point)

Session 7: Learning from Neural Networks with Queries and Counter Examples

Speaker: Ana Ozaki, University of Oslo

Topics:

- Exact Learning, NN as Oracles
- Learning Automata with Queries and Counterexamples
- Learning Horn Expressions with Queries and Counterexamples

Summary:

Motivation is the explainability of AI. Two goals with explanation:

- Explain the output of the model given a particular input
- Or the general behavior of the model, independent from the input and output

Approaches are based on Angluin's exact learning framework.

Learner ↔ Queries and Answers ↔ Teacher(Oracle)

Two types of queries:

- Membership query: Does the input belong to a target set
- Equivalence query: Does the input matches to the target set. If not, teacher gives a counterexample.

Learning automata (L* algorithm, assumption: learner knows the nature of the target): Learn an automata from the NN that recognizes a target language L. The main idea of the algorithm is to iteratively ask NN whether a specific input is recognized by the target language. Based on the negative responses and counterexamples, refine the query and form an automata. Keep track of the queries and responses with an observation table.

Learn horn expressions: Iteratively create hypothesis in order to finally satisfy some certain target horn formulas, and try to find the horn formulas in this way.

Problems to learning horn expressions from NNs:

- NN may not generalize to satisfy horn rules. If the target is not in the form of horn formulas then the algorithm does not terminate. In that case we can use positive and negative counterexamples to learn Horn approximations. If the intersection of two positive examples results in a negative result, that means the target is not in the form of horn.

- Equivalence queries does not exists when using LLMs and learning horn formulas.
- Format of data. Speaker used lookup tables and a template sentence.

Session 8: Proof-Theoretic Approaches in Logical Argumentation (pure theory, I had to look at an empty wall in a dark room for an hour after the session)

Speaker: Christian Straßer and Kees van Berkel, Ruhr University Bochum

Topics:

- Defeasible reasoning
- Formal argumentation
- Logical Argumentation in Sequent-calculi
- Applications to normative reasoning
- Deontic logic and explanations

Summary:

Deductive reasoning is about certain inferences while defeasible reasoning is to retain the option to retract an inference, e.g., as we obtain new information. Defeasible reasoning can also be used if a given information is uncertain. Most of our everyday reasoning is defeasible. Formal approach to use defeasible reasoning is non-monotonic logic where monotonicity delibaretely does not hold to work with incomplete, uncertain or inconsistent knowledge bases. Classical logic is not able to reason defeasibly.

Formal argumentation is used to structure knowledge, and indicates when to retract inferences.

Argumentative Knowledge Representation and Reasoning (ArgKRR): Arguments are represented by nodes, and edges represent whether an argument “attacks” another argument. Some types of arguments:

- Conflict-free argument: 2 arguments that are not attacking each other
- Admissible argument: if it defends every of its arguments and it is conflict-free
- Complete: if admissible and contains all the arguments it defends
- Grounded: refers to a unique smallest complete set of arguments
- Stable: conflict-free set of arguments that attacks every argument that is not in the set

Sequent calculi: Distinguish between reliable statements and general statements. We can choose to trust one set of arguments and see the outcome, or based on the chosen arguments to trust, we get different results and the results are defeasible.

How to select arguments: one approach is to find stable arguments, find the consequents of those arguments and take the intersection.

Normative reasoning: drawing conclusions from and about obligations, prohibitions, permissions, rights, violations, …. Important to law, ethics, AI, business protocols, social customs and many more. It is conflict-sensitive and defeasible.

Deontic logic: formal field that deals with normative reasoning.

Some open problems:

- Formal argumentation: bridging the gap between symbolic and non-symbolic AI.
- Automated reasoning: heuristics to work with finite AFs even when infinitely many arguments are available.