# Decision Support Systems based on Answer Set Programing

Fadi Al Machot and Kyandoghere Kyamakya

Description logics is a group of logics for knowledge modeling, which are derived from semantic networks. They are to be understood essentially as accessible fragments of predicate logic of first order, to allow strong expressions to be expressed through a complexity class with restrictions.

Other description logics provide such strong expressions in polynomial complexity. For description logics, there are a special inputs and intuitive notation that facilitates the handling of them substantially.

Modeling in expert systems is very important, especially systems are having a complex domain where the spatial and temporal data are needed to be modeled. The ontology model permits the use of a reasoner that can check definitions of the statements in the ontology, if they are consistent or not. It can also recognize which concepts are the best for which definitions. The reasoner can therefore help to maintain the hierarchy correctly. This is particularly helpful when dealing with multiple classes hierarchies, where for an expert system complex concepts can therefore be built up in definitions out of simpler concepts. So, An ontology is an engineering artefact or is a formal representation of knowledge as a set of concepts within a domain, it often includes classification based information and constraints capturing the knowledge about the domain [13]. Despite of the powerful semantic of ontology languages, they lead to high querying execution time and cause a not real time inferring on the defined ontology. In this lecture, we show the efficiency of Answer Set Programing (ASP) to represent our knowledge database and then how to define the rules between all individuals of the desired ontology. Finally, we compare the execution time of an ASP-based reasoner and other existing reasoning paradigms.

## I. Answer Set programming

A logic program in the language of AnsProlog (also known as A-Prolog) [6, 3] is a set of rules of the form:

$$a_0 \leftarrow a_1, ..., a_m, \neg a_{m+1}, ..., \neg a_n \tag{1}$$

where $0 \leqslant m \leqslant n$,each $a_i$ is an atom of $a_i$ propositional language and $not$ represents $negation - as - failure$. A negations- failure literal (or naf-literal) has the form not a, where a is an atom. Given a rule of this form, the left and right hand sides are called the $head$ and $body$, respectively. A rule may have either an empty head or an empty body, but not both. Rules with an empty head are called constraints, while those with an empty body are known as $facts$. A definite rule is a rule which does not contain naf-literals, and a definite program is composed solely of definite rules [3].

F. Al Machot and K. Kyamakya are with Institute for Smart System Technologies, Transportation Informatics, Alpen-Adria-University Klagenfurt, 9020 Klagenfurt, Universitätsstrasse 65-67, Austria (e-mail: forename.surname@uni-klu.ac.at).

Let $X$ be a set of ground atoms. The body of a rule of the form (1) is satisfied by $X$ if $\{a_{m+1}, ..., a_n\} \bigcap X = \phi$ and $\{a_1, ..., a_m\} \subseteq X$. A rule with a non-empty head is satisfied by $X$ if either its body is not satisfied by $X$, or$a_0 \in X$. A constraint is satisfied by $X$ if its body is not satisfied by $X$. Given an arbitrary program, $\Pi$ and a set of ground atoms, $X$, the reduct of $\Pi$ w.r.t. $X$, $\Pi^X$, is the definite program obtained from the set of all ground instances of $\Pi$ by:

1) deleting all the rules that have a naf-literal not a in the body where $a \in X$, and
2) removing all naf-literals in the bodies of the remaining rules.

A set of ground atoms $X$ is an answer set of a program $\Pi$ if it satisfies the following conditions:

1) If $\Pi$ is a definite program, then $X$ is a minimal set of atoms that satisfies all the rules in $\Pi$.
2) If $\Pi$ is not a definite program, then $X$ is the answer set of $\Pi^X$. (Recall that $\Pi^X$ is a definite program, and its answer set is defined in the first item [3].

The advantage of ASP that the order of program rules does not a matter and the order of subgoals in a rule does not a matter also.

### A. Logic programming with Ordered Disjunction

Logic programming can be extended to allow us to represent new options for problems in the head of the rules. ASP gives this ability through ordered disjunctions. Using ASP under specific conditions reasoning from most preferred answer sets gives optimal problem solutions. Through LPODs like normal logic programs we are able to express incomplete and defeasible knowledge through the use of default negation, they allow us to represent performances among intended properties of problem solutions which depend on the current context. As in real life there are situations where the best options simply do not work out, there for LPODs are very well suited for representing problems where a certain choice has to be made . In general using ASP we can optimize the solution we want to generate, we can improve the rules and define the constraints we are using to get the maximum optimization of the desired answer sets (solutions).

### B. NP problems in Answer Set Programming

Answer set programming (ASP) is widespread used expressing properties in NP, where answer sets of normal logic programs can be generated through solutions and polynomial time proofs for such properties. The solution of such problems can be through two steps:

- Generate a candidate solution through a logic program
- Check the solution by another logic program.

However, it is often not clear how to combine $\Pi_{guess}$ and $\Pi_{check}$ into a single program $\Pi_{solve}$ which solves the overall problem. If we take the Simply the union $\Pi_{guess} \vee \Pi_{solv}e$ does not work, so we have to rewrite the program [5].

Theoretical results proof that for problems with $\Sigma_2^P$ complexity, it is required that $\Pi_{check}$ is rewritten into a disjunctive logic program $\grave{\Pi}_{check}$ such that the answer sets of $\Pi_{solve} = \Pi_{guess} \vee \grave{\Pi}_{check}$ yield the solutions of the problem, where $\grave{\Pi}_{check}$ emulates the inconsistency check for $\grave{\Pi}_{check}$ as a minimal model check, which is co-NP-complete for disjunctive programs. This becomes even more complicated by the fact that $\grave{\Pi}_{check}$ must not crucially rely the use of negation, since it is essentially determined by the $\Pi_{guess}$ part. These difficulties can make rewriting $\Pi_{check}$ to $\grave{\Pi}_{check}$ a formidable and challenging task [5].

### C. query optimization based on ASP

There are algorithms that can be used for an effective optimization strategy for queries on knowledge databases query optimization.

One of the hardest problems in query optimization is the accurate estimation of the costs of alternative query plans. Cardinality estimation depend on estimates of the selection factor of predicates in the query. The estimation of the cardinality of a query usually is used to approximate the data transfer times of the result set, as part of the estimation of the total cost of executing a query. As an example a maximal complete query pattern path consists of a maximal query pattern path and a set of value constraints. The total cardinality of such a path is obtained by calculating the product of the cardinality estimate of its maximal query path with all the value ratios for every variable in the query pattern path.

In a highly distributed architecture where data in different locations connected through the Internet, this is the most critical aspect of query execution time and the speed of the connections and the amount of data play an important role to retrieve the data from texts, images or videos.

A query optimization strategy is crucial to obtain reasonable performance over queries against ontology data models, especially if they are done over a highly distributed architecture.

But using Answer set programming paradigm, we are able to model the spatial and temporal context models and define the relationships between classes and individuals (rules) and generating the answer sets using advanced solvers that support binder splitting, back jumping and all other features. The statements can be optimized to find the maximal or the minimal answer set of the logic program. The statement can be weighted or not, therefore weights can be omitted. If there are several minimize maximize statements in the logic program the latter will be preferenced.

The difference between the performance of ASP solvers and other existed ontology query languages is clearly high.The estimation of the cardinality of a query usually is used to approximate the data transfer times of the result set, as part of the estimation of the total cost of executing a query.

### D. ASP as an automatic synthesis for Flexible Multiprocessor Systems

Configurable on chip multiprocessor systems are resource limited due to memory bandwidth and power consumption. Because of the huge size of the design space of such systems, it is important to automatically optimize design parameters in order to facilitate wide and disciplined explorations. Answer Set Programming (ASP) as a solution for such systems can be effectively employed and has a great potential for solving difficult system design problems[15].

## II. Conclusion

ASP provides meaning to logic programs with default negation $not$, supports problem solving paradigm where models represent solutions, many interesting applications in planning, reasoning about action, configuration, diagnosis, space shuttle control,..etc, has several useful extensions, disjunctive LPs, cardinality constraints and weight constraints. Answer set programming (ASP) is widespread used expressing properties in NP, where answer sets of normal logic programs can be generated through solutions and polynomial time proofs for such properties beside that using ASP we can also apply spatial and temporal Reasoning, constraints, and incomplete information in the ontology to get the best and the optimal solution.

## References

[1] Fernyhough J., Cohn A.G., and Hogg D.C., "Constructing qualitative event models automatically from video input" *3rd ed. IEEE Conference on Advanced Video and Signal Based Surveillanc, Representing and recognizing,, 3rd ed. Image and Vision Computing, 18(2):81 103, 2000*

[2] Brewka, Gerhard and Niemela, Ilkka and Truszczynski, Miroslaw, "Answer set optimization," *Proceedings of the 18th international joint conference on Artificial intelligence, lAcapulco, Mexico, pp. 867–872, 2003*

[3] Baral, C. and Gelfond, G. and Son, T.C. and Pontelli, E., "Using answer set programming to model multi-agent scenarios involving agents' knowledge about other's knowledge," *Proceedings of the 18th international joint conference on Artificial intelligence, Toronto, Canada, pp. 259–266, 2010*

[4] Brewka, G., "Logic programming with ordered disjunction," *proceedings of the national conference on artificial intelligence, USA, pp. 100–105, 2002*

[5] Eiter, T. and Polleres, A., "Towards automated integration of guess and check programs in answer set programming: a meta-interpreter and applications," fsoun dcutfnormungsprozess *Theory and Practice of Logic Programming, Cambridge Univ Press, USA, pp. 23–60, 2006*

[6] Shironoshita, E.P. and Ryan, M.T. and Kabuka, M.R., "Cardinality estimation for the optimization of queries on ontologies," *ACM SIGMOD Recordg, Miami, USA, pp. 13–18, 2007*

[7] Micheloni C., Snidaro L., and Foresti G. L, "Complex events in surveillance applications" *3rd ed. IEEE Conference on Advanced Video and Signal Based Surveillanc, Representing and recognizing, AVSS, 2007*

[8] Wang X.H., Zhang D.Q., Gu T., and Pung H.K. , "Ontology based context modeling and reasoning using owl" *In, 3rd ed. Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications, pp. 18–22, March 2004*

[9] Tao, F, Shadbolt, N.R., Chen, L, Xu., F. and Cox, S.J. , “Semantic Web based Content Enrichment and Knowledge Reuse in e-Science.” *I3rd International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE), Larnaca, Cyprus, 25-29 Oct, 2004*

[10] Gebser, M, Kaminiski, R, Kaufmann, B,Ostrowsky, M, Schaub, T, Thiele, S, “Using gringo, clingo and iclingo” *September 25, 2008*

[11] Fuchs, S “A komprehensive Knowledge Base for Context-Aware Tactical Driver Assistance Systems” *ISBN 978-3-8322-8078-9, Shaker Verlag, Aachen, 2008*

[12] Matheus, C. and Baclawski, K. and Kokar, M. and Letkowski, J. “Using SWRL and OWL to capture domain knowledge for a situation awareness application applied to a supply logistics scenario” *pp. 130–144, Springer, 2005*

[13] Kohler, J. and Philippi, S. and Lange, M. “Uontology based semantic integration of biological databases” *Bioinformatics journal, volume 19, issn 1367-4803, pp. 2420, Oxford Univ Press, 2003*

[14] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. “Nonmonotonic Description Logic Programs: Implementation and Experiments” *In F. Baader and A. Voronkov, editors, Proceedings 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004), number 3452 in LNCS, pages 511-517. Springer, 2005*

[15] Harold Ishebabi, Philipp Mahr, Christophe Bobda, Martin Gebser, and Torsten Schaub. “Application of ASP for Automatic Synthesis of Flexible Multiprocessor Systems from Parallel Programs” *Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, Berlin, 2009*