Explanation Generation Systems for Answer Set Programming

Ly Ly Trieu[1], Tran Cao Son[1]

[1] New Mexico State University, New Mexico, USA
lytrieu@nmsu.edu, tson@cs.nmsu.edu

My research focuses on developing explanation generation systems for *Answer Set Programming (ASP)*. In this abstract, we present three different systems, `exp(ASP)`, `exp(ASP`^{c}`)` and `xASP`, which generate explanation graphs for a literal ℓ—an atom a or its default negation $\sim a$—given an answer set A of a normal logic program P, which explain why ℓ is true (or false) given A and P.

ASP is a popular paradigm for decision-making and problem-solving in Knowledge Representation and Reasoning. It is an attractive programming paradigm as it is declarative where programmers focus on the representation of a specific problem as a set of rules, and then leave the computational solutions of that problem to an answer set solver. It has been applied in several applications such as planning, diagnosis, robotics, etc. However, this mechanism typically gives little insight into why a set of literals is a solution and why some proposed set of literals is not a solution. This type of reasoning falls within the scope of explainable Artificial Intelligence and is useful to enhance the understanding of solutions as well as for debugging programs. `exp(ASP)` produces the explanation graph for normal logic programs. It is used to interpret the responses produced by a diagnosis system modeling the Three Mile Island disaster. `exp(ASP`^{c}`)` is an extended system from `exp(ASP)`. Different from `exp(ASP)`, it supports choice rules and utilizes constraint rules to provide explanation graphs that include information about choices and constraints. In `exp(ASP)` and `exp(ASP`^{c}`)`, grounding a program before computing an explanation eliminates various pieces of information, resulting in explanations that are no longer faithful to the original program, or are unable to even provide an explanation. Therefore, we propose `xASP` which works directly with programs including variables. Thus, it generates explanation graphs that are faithful to the program, thus distinguishing itself from other existing systems. In addition, it generates all full explanation graphs for an atom given an answer set, and its behavior is not affected by semantics-preserving changes in the program.

Research on explainable AI will not only contribute to the development of computer science in general but also brings significant benefits to society in critical fields such as health care, power systems, etc. The applications in these fields require high accuracy and understandable results for the users. With the explanation, the users can decide whether they should trust the results. Our future goal is to test `xASP` on realistic debugging tasks and extend the system so that it can support the full language of `clingo`.