Design and Implementation of a Modern ASP Grounder

Jessica Zangari

Abstract

Answer Set Programming (ASP) is a declarative programming paradigm proposed in the area of non-monotonic reasoning and logic programming in the late '80 and early '90. Thanks to its expressivity and capability of dealing with incomplete knowledge, ASP became widely used in AI and recognized as a powerful tool for Knowledge Representation and Reasoning (KRR). On the other hand, its high expressivity comes at the price of a high computational cost, thus requiring reliable and high-performance implementations. Throughout the years, a significant effort has been spent in order to define techniques for an efficient computation of its semantics. In turn, the availability of efficient ASP systems made ASP a powerful tool for developing advanced applications in many research areas as well as in industrial contexts. Furthermore, a significant amount of work has been carried out in order to extend the "basic" language and ease knowledge representation tasks with ASP, and recently a standard input language, namely ASP-Core-2, has been defined, also with the aim of fostering interoperability among ASP systems.

Although different approaches for the evaluation of ASP logic programs have been proposed, the canonical approach, which is adopted in mainstream ASP systems, mimics the definition of answer set semantics by relying on a grounding module (*grounder*), that generates a propositional theory semantically equivalent to the input program, coupled with a subsequent module (*solver*) that applies propositional techniques for generating its answer sets.

The former phase, called grounding or instantiation, plays a key role for the successful deployment in real-world contexts, as in general the produced ground program is potentially of exponential size with respect to the input program, and therefore the subsequent solving step, in the worst case, takes exponential time in the size of the input. To mitigate these issues, modern grounders employ smart procedures to obtain ground programs significantly smaller than the theoretical instantiation, in general.

This thesis focuses on the ex-novo design and implementation of a new modern and efficient ASP instantiator. To this end, we study a series of techniques geared towards the optimization of the grounding process, questioning the techniques employed by modern grounders with the aim of improving them and introducing further optimization strategies, which lend themselves to the integration into a generic grounder module of a traditional ASP system following a ground & solve approach. In particular, we herein present the novel system *I-DLV* that incorporates all these techniques leveraging on their synergy to perform an efficient instantiation. The system features full support to ASP-Core-2 standard language, advanced flexibility and customizability mechanisms, and is endowed with extensible design that eases the incorporation of language updates and optimization techniques. Moreover, its usage is twofold: besides being a stand-alone grounder, it is also a full-fledged deductive database engine. In addition, along with the solver *wasp* it has been integrated in the new version of the widespread ASP system *DLV* recently released.