Artificial Intelligence 207 (2014) 1–22

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Relating constraint answer set programming languages and algorithms

Yuliya Lierler

Department of Computer Science, The University of Nebraska at Omaha, 6001 Dodge Street, Omaha, NE 68182, United States

ARTICLE INFO

Article history:
Received 9 October 2012
Received in revised form 28 October 2013
Accepted 29 October 2013
Available online 11 November 2013

Keywords:
(Constraint) answer set programming
Constraint satisfaction processing
Satisfiability modulo theories

ABSTRACT

Recently a logic programming language *AC* was proposed by Mellarkod et al. [1] to integrate answer set programming and constraint logic programming. Soon after that, a CLINGCON language integrating answer set programming and finite domain constraints, as well as an EZCSP language integrating answer set programming and constraint logic programming were introduced. The development of these languages and systems constitutes the appearance of a new AI subarea called constraint answer set programming. All these languages have something in common. In particular, they aim at developing new efficient inference algorithms that combine traditional answer set programming procedures and other methods in constraint programming. Yet, the exact relation between the constraint answer set programming languages and the underlying systems is not well understood. In this paper we address this issue by formally stating the precise relation between several constraint answer set programming languages – *AC*, CLINGCON, EZCSP – as well as the underlying systems.

1. Introduction

Constraint answer set programming (CASP) is a novel, promising direction of research whose roots can be traced back to propositional satisfiability (SAT). SAT solvers are efficient tools for solving Boolean constraint satisfaction problems that arise in different areas of computer science, including software and hardware verification. Answer set programming (ASP) extends computational methods of SAT using ideas from knowledge representation, logic programming, and nonmonotonic reasoning. As a declarative programming paradigm, it provides a rich, and yet simple modeling language that, among other features, incorporates recursive definitions. Satisfiability modulo theories (SMT) extends computational methods of SAT by integrating non-Boolean symbols defined via a background theory in other formalisms, such as first order theory or a constraint processing language. The key ideas behind such integration are that (a) some constraints are more naturally expressed by non-Boolean constructs and (b) computational methods developed in other areas of automated reasoning than SAT may complement its technology in an effective manner processing these constraints.

Constraint answer set programming draws on both of these extensions of SAT technology: it integrates answer set programming with constraint processing. This new area has already demonstrated promising results, including the development of the CASP solvers ACSOLVER [1] (Texas Tech University), CLINGCON[1] [2,3] (Potsdam University, Germany), EZCSP[2] [4] (KODAK), IDP[3] [5] (KU Leuven). These systems provide new horizons to knowledge representation as a field by broadening the applicability of its computational tools. CASP not only provides new modeling features for answer set programming but

[1] http://www.cs.uni-potsdam.de/clingcon/.
[2] http://marcy.cjb.net/ezcsp/index.html.
[3] http://dtai.cs.kuleuven.be/krr/software/idp.

also improves grounding and solving performance by delegating processing of constraints over large and possibly infinite domains to specialized systems. The origins of this work go back to [6,7].

Drescher and Walsh [8,9] (INCA, NICTA, Australia), Liu et al. [10] (MINGO, Aalto University, Finland) took an alternative approach to tackling CASP languages – a translational approach. In the former case, the CASP programs are translated into ASP programs (Drescher and Walsh proposed a number of translations). In the latter, the program is translated into integer linear programming formalism. The empirical results demonstrate that this is also a viable approach towards tackling CASP programs.

The general interest towards CASP paradigms illustrates the importance of developing synergistic approaches in the automated reasoning community. To do so effectively one requires a clear understanding of the important features of the CASP-like languages and underlying systems. Current CASP languages are based on the same principal ideas yet relating them is not a straightforward task. One difficulty lies in the fact that these languages are introduced together with a specific system architecture in mind that rely on various answer set programming, constraint satisfaction processing, constraint logic programming, and integer linear programming technologies. The syntactic differences stand in the way of clear understanding of the key features of the languages. For example, the only CASP language that was compared to its earlier sibling was the language EZCSP. Balduccini [4] formally stated that the EZCSP language is a special case of *AC*. Relating CASP systems formally is an even more complex task. The variations in underlying technologies complicate clear articulation of their similarities and differences. For instance, the main building blocks of the CASP solver ACSOLVER [1] are the ASP system SMODELS [11] and SICSTUS Prolog.[4] The technology behind CLINGCON [2,3] is developed from the ASP solver CLASP [12] and the constraint solver GECODE [13]. In addition, the CASP solvers adopt different communication schemes among their heterogeneous solving components. For instance, the system EZCSP relies on *blackbox* integration of ASP and CSP tools in order to process the EZCSP language [4]. Systems ACSOLVER and CLINGCON promote tighter integration of multiple automated reasoning methods.

The broad attention to CASP suggests a need for a principled and general study of methods to develop unifying terminology and formalisms suitable to capture variants of the languages and solvers. This work can be seen as a step in this direction. First, it presents a formal account that illustrates a precise relationship between the languages of ACSOLVER, CLINGCON, and EZCSP. Second, it formally relates the systems that take a hybrid approach to solving in CASP. In particular, it accounts for systems ACSOLVER, CLINGCON, and EZCSP.

Usually backtrack search procedures (Davis–Putnam–Logemann–Loveland (DPLL)-like procedures [14]), the backbone of CASP computational methods are described in terms of pseudocode. In [15], the authors proposed an alternative approach to describing DPLL-like algorithms. They introduced an abstract graph-based framework that captures what the “states of computation” are and what transitions between states are allowed. This approach allows us to model a DPLL-like algorithm by a mathematically simple and elegant object, a graph, rather than a collection of pseudocode statements. We develop a similar abstract framework for performing precise formal analysis on relating the constraint answer set solvers ACSOLVER, CLINGCON, and EZCSP. Furthermore, this framework allows an alternative proof of correctness of these systems. This work clarifies and extends state-of-the-art developments in the area of constraint answer set programming and, we believe, will promote further progress in the area.

More on related work Another direction of work related to the developments in CASP is research on HEX-programs [16]. These programs integrate logic programs under answer set semantics with external computation sources via *external atoms*. They were motivated by the need to interface ASP with external computation sources, for example, to allow the synergy of ASP and description logic computations within the context of the semantic web. CASP has a lot in common with HEX-programs. System DLVHEX[5] [17] computes models of such programs. It allows defining plug-ins for inference on external atoms and as such can be used as a general framework for developing CASP solvers (but it does not provide any specific computational mechanism by default).

Heterogeneous nonmonotonic multi-context systems [18] is another formalism related both to CASP and HEX-programs. CASP and HEX-programs can be seen as one of the possible incarnations of a special case of multi-context systems. Multi-context systems provide a more general formalism where “contexts” written in different logics relate with each other via bridge rules. Intuitively, CASP provides two contexts: one in the language of answer set programming and another one in the language of constraint programming. Yet, the bridge rules are of extremely simplistic nature in CASP, in particular, they relate atoms in a logic program to constraints of constraint processing.

Paper structure We start by reviewing *AC* programs introduced by Mellarkod et al. [1] and the notion of an answer set for such programs. In the subsequent section we introduce the CLINGCON language and formally state its relation to the *AC* language. We then define a new class of weakly-simple programs and demonstrate that the ACSOLVER algorithm is applicable also to such programs. We review a transition system introduced by Lierler [19,20] to model SMODELS. We extend this transition system to model the ACSOLVER algorithm and show how the newly defined graph can characterize the computation behind the system ACSOLVER. We define a graph suitable for modeling the system CLINGCON and state a formal result on the

[4] http://www.sics.se/isl/sicstuswww/site/index.html.
[5] http://www.kr.tuwien.ac.at/research/systems/dlvhex/.