Some Recent Advances in Answer Set Programming (from the Perspective of NLP)

Marcello Balduccini

College of Computing and Informatics
Drexel University
`marcello.balduccini@gmail.com`

1 Introduction

Answer Set Programming (ASP) [12, 13, 15, 8] is a logical language for knowledge representation and reasoning that combines a non-monotonic nature, strong theoretical foundations, an intuitive semantics, and substantial expressive power. The language has been successfully used for modeling a number of very diverse domains (e.g. [7, 14]) and for capturing key reasoning tasks such as planning, diagnostics, learning and scheduling (e.g. [10, 5, 1]). All of this makes ASP a prime candidate for use in the sophisticated knowledge representation and reasoning tasks involved in Natural Language Processing (NLP).

In this note I will give an overview of some of my recent work on ASP that I believe may be useful in the context of NLP.

2 Motivation

Reasoning about natural language involves various knowledge-intensive tasks. Particularly challenging from this perspective are the extraction of semantic content from phrases (e.g. anaphora resolution) and the disambiguation of phrases using world knowledge and commonsense knowledge. These two tasks are not only challenging, but also heavily interconnected.

Consider the following collection of passages and proposed corresponding reasoning:

- "John was walking his dog. He said hi."
  To conclude that "he" refers John, we can use knowledge about grammar, stating that "he" normally refers to a human male. Additionally, the fact that saying hi is a capability proper of humans, confirms the correctness of the association.
- "John was walking his dog. He ran away after a rabbit"
  This type of sentence is quite common especially in spoken English. Commonsense tells us that "he" here refers to John's dog. A justification for this is the everyday knowledge that running away after a rabbit is a behavior common of dogs and other animals with hunting habits. Humans typically

do not run away after rabbits (although carefully capturing this last statement appears to be a rather interesting and intricate modeling task in itself). Although according to grammar rules "he" should be associated with John, it is also typical for people, and especially pet owners, to refer to their pets by "he" or "she."

- "John and Frank entered the room. Frank left right away. He came out two minutes later."
  In this case the difficulty in finding which object "he" refers to derives from the fact that two human males are mentioned in approximately the same locations of the passage. To properly link this occurrence of "he" to John, one needs to follow the evolution of the domain described by the passage. The phrase "came out two minutes later" appears to refer to the room that John and Frank had initially entered. The second sentence states that Frank has already left the room. So, John is the only other person *of interest in the passage* who is left in the room, and thus it is reasonable to assume that "he" refers to him.
- "Andrea and Frank entered the room, but he left empty-handed."
  To reason about this sentence, it is useful to recall that, in English, Andrea is both a male and a female name. Reasoning by cases, one can observe that, if Andrea is a man, then the occurrence of "he" in the sentence is ambiguous. On the other hand, if Andrea is a female, then it can be concluded without ambiguity that "he" refers to Frank. Under the assumption that the speaker or writer crafted the sentence in such a way as to convey the relevant information in an unambiguous way, then it is reasonable to assume that "he" refers to Frank. Moreover, one can conclude that Andrea is a woman. This information can be stored and used later in reasoning about other parts of the passage.
- "Andrea cannot be the one who took the computer from that room. Andrea and Frank did enter the room, but he left empty-handed."
  Let us suppose that this passage is in the context of an investigation aimed at determining who stole a computer from a room. The first sentence focuses the discussion on Andrea and on Andrea's innocence. Let us reason again by cases on the possible associations of "he" – Andrea and Frank. Under both possible associations, no grammar rules are violated, as long as Andrea is a man. If "he" refers to Frank, however, the first and the second sentences appear to have no logical connection, while their construction suggests that indeed some link exists. On the other hand, if "he" refers to Andrea, then the link is clear: the first sentence claims Andrea's innocence, and the second sentence offers evidence in support of the claim. Similarly to the previous example, the second case appears to be preferred based on the commonsensical assumption that the speaker or writer crafted the passage in such a way as to convey the relevant information in an unambiguous *and economical* way.

Whereas carefully crafted, written-language passages may require relatively limited reasoning, everyday, colloquial language such as the one exemplified here requires substantial reasoning for a proper understanding. For the success of practical systems with natural language interfaces, I argue that everyday, colloquial language must be supported.

Overall, it appears that successfully reasoning about the semantic content of sentences such as the ones shown above requires a sophisticated combination of world knowledge, commonsense, and (commonsensical) information about speaker's/writer's behavior and intentions. It is my belief that ASP and its extensions can be useful in tackling such a task.

In the rest of this note I describe some extensions of ASP I authored or co-authored, and which may be useful in capturing certain aspects of the reasoning about natural language. For a thorough discussion on ASP and on its use for knowledge representation, the reader is referred to the existing literature (e.g. [8]).

3 CR-Prolog

CR-Prolog [6] is an extension of ASP that adds to the language constructs, called consistency-restoring rules (cr-rules), designed to capture certain advanced aspects of non-monotonic reasoning.

A central, well-known feature of languages for non-monotonic reasoning such as ASP is that the the programmer can write "defeasible statements," which are normally true, but may not apply to certain cases, called exceptions. A well-known example is that of the statement "birds normally fly." While true for most birds, this statement has exceptions, such as penguins and birds with broken wings, and hence the use of the word "normally."

In most languages for non-monotonic reasoning the exceptions must be explicitly listed. In the example above, if a new type of bird is discovered that does not fly, suitable statements must be added to the system, saying that that type of bird is an exception. If the exceptions are not added, the systems will apply the default statement and conclude that the birds of the new type fly. From a practical perspective, having to know in advance all the exceptions may be a limiting factor in the development of autonomous systems, since there may not be sufficient understanding of the problem domain for such a complete list. It is worth observing that, *in everyday reasoning, humans are typically capable of postulating exceptions to defaults, especially when they observe phenomena that contradict such defaults.*

Cr-rules are an attempt to capture this capability, allowing a reasoner to postulate exceptions to default statements, but only when strictly necessary. Making such assumptions is considered strictly necessary when the reasoning process is otherwise inconsistent. This is the case, for example, of a system given observations that contradict its knowledge base. For instance, the cr-rule:

$$exception(H, A) \overset{+}{\leftarrow} human(H), animal(A), small(A).$$

can be used in combination with a default "normally, humans do not chase small animals" to state that, under exceptional, *unknown*, circumstances, the default can be violated. When inconsistencies arise in the knowledge base, the system can then use the cr-rule to postulate exceptions to the default.

My co-authors and I demonstrated that cr-rules allow for elegantly capturing types of non-monotonic reasoning that are otherwise difficult or impossible to capture. We also showed that they can be used to formalize concisely diagnostic reasoning and certain types of planning.

4 EZCSP

EZCSP [2] is an extension of ASP aimed at increasing performance and scalability in certain – rather large – application domains.

To see how the ability to reason about numbers is important in reasoning about natural language, consider the following passage: "The train left at 10. A couple of hours later, we were having lunch in Paris." To determine if "10" refers to 10am or 10pm, one may reason as follows. Normally, "a couple of hours" means two hours. Moreover, let us assume that it is common knowledge that in Paris people have lunch between noon and 2pm. Hence, if we reason by cases, the interpretation that "10" refers to "10am" sets the time of the speaker's lunch in Paris to noon. The interpretation that "10" refers to "10pm" sets the time of the lunch to midnight. The second interpretation contradicts the custom of having lunch between noon and 2pm, and thus the former interpretation is preferred.

This reasoning process relies on the ability to process effectively numerical information. Although ASP allows in principle for natural and concise formalizations of many kinds of knowledge, in practical applications efficiency often degrades quickly when dealing with numerical information and variables with large domains. To overcome this limitation, I designed EZCSP to allow for the use of constructs from constraint programming within ASP programs. For example, the rule:

$$\mathit{required}(\mathit{hour}(T) \geq 12) \leftarrow \mathit{lunchtime}(T).$$

states that, if timestamp T refers to lunch time, then the hour of T must be greater than or equal to 12.

The new language makes it possible to represent and reason about numerical information efficiently, while at the same time keeping the representations elegant, concise and elaboration tolerant, as usual in ASP. Differently from other languages that combine ASP and constraint programming (e.g. [11]), EZCSP includes support for global constraints (a powerful type of construct from constraint programming), and both language and solver are designed to be independent from the underlying ASP and constraint solvers chosen for the computation.

5 ASP{f}

One shortcoming of EZCSP is that it does not allow one to perform full-fledged non-monotonic reasoning on numerical quantities. For example, one cannot easily state that "in Paris, people *normally* have lunch between noon and 2pm" and reason with evidence that "John had lunch at 3pm."

This is due to the monotonic nature of the underlying constraint programming constructs. A further shortcoming of EZCSP is that performance in the presence of variables with large non-numerical domains still tends to be limited, because constraint programming constructs mainly apply to numerical quantities. In fact, these limitations are shared by all the research attempts aimed at hybridizing ASP and constraint programming. To overcome these issues, I have developed a new language, called ASP{f} [3, 4], which adds to ASP the ability to represent, and reason about, arbitrary (non-Herbrand) functions, including but not limited to numerical functions. With ASP{f}, the default about lunch time in Paris can be captured in a simple way by statements such as:

$$hour(T) < 14 \leftarrow lunchtime(T),\ \text{not}\ hour(T) \geq 14.$$

which intuitively states that "lunch ends at 2pm unless otherwise specified." The observation about John's lunch time can be encoded by $\{lunchtime(t_1), hour(t_1) = 15\}$. In ASP{f}, this observation is sufficient to defeat the default.

In ASP{f} it is also possible to capture rather complex numerical calculations, such as:

$$\begin{array}{l} financially\_sound(C) \leftarrow \\ \qquad revenue(C) > sum[employee(E, C) = salary(E)] + investments(C). \end{array}$$

This rule states that company C is financially sound if its revenue is greater than the sum of the salaries paid to the employees and of the investments made by the company. Another example, demonstrating ASP{f}'s ability to deal with non-numerical information, is the rule:

$$\leftarrow siblings(P1, P2), first\_name(P1) = first\_name(P2), \text{not}\ exception(P1, P2).$$

which captures the commonsensical statement that two siblings shouldn't have the same first name. Rather than relying on constraint programming, the new language includes "native" support for functions, and, differently from other attempts in this direction (e.g. [9]), is crafted in such a way that state-of-the-art inference engines for ASP can be extended to support ASP{f} with relatively simple modifications.

6 Conclusions

In this note I have described some challenges of the task of reasoning about natural language that are relevant to ASP and to commonsense and non-monotonic reasoning in general. I have also discussed recent extensions of ASP that I have developed, and which I believe may be useful in tackling these tasks. Of course, many other extensions of ASP exist, which can be useful for this endeavour. For the reader's convenience, a small selection of relevant works was cited in this note.

References

1. Balduccini, M.: Learning Action Descriptions with A-Prolog: Action Language C. In: Amir, E., Lifschitz, V., Miller, R. (eds.) Procs of Logical Formalizations of Commonsense Reasoning, 2007 AAAI Spring Symposium (Mar 2007)
2. Balduccini, M.: Representing Constraint Satisfaction Problems in Answer Set Programming. In: ICLP09 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP09) (Jul 2009)
3. Balduccini, M.: Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz, chap. 3. A "Conservative" Approach to Extending Answer Set Programming with Non-Herbrand Functions, pp. 23–39. Lecture Notes in Artificial Intelligence (LNCS), Springer Verlag, Berlin (Jun 2012)
4. Balduccini, M.: ASP with non-Herbrand Partial Functions: a Language and System for Practical Use. Journal of Theory and Practice of Logic Programming (TPLP) (2013)
5. Balduccini, M., Gelfond, M.: Diagnostic reasoning with A-Prolog. Journal of Theory and Practice of Logic Programming (TPLP) 3(4–5), 425–461 (Jul 2003)
6. Balduccini, M., Gelfond, M.: Logic Programs with Consistency-Restoring Rules. In: Doherty, P., McCarthy, J., Williams, M.A. (eds.) International Symposium on Logical Formalization of Commonsense Reasoning. pp. 9–18. AAAI 2003 Spring Symposium Series (Mar 2003)
7. Balduccini, M., Gelfond, M., Nogueira, M.: Answer Set Based Design of Knowledge Systems. Annals of Mathematics and Artificial Intelligence 47(1–2), 183–219 (2006)
8. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press (Jan 2003)
9. Cabalar, P.: Functional Answer Set Programming. Journal of Theory and Practice of Logic Programming (TPLP) 11, 203–234 (2011)
10. Erdem, E.: Application of Logic Programming to Planning: Computational Experiments. In: Proceedings of the 5th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR-99). No. 1730 in Lecture Notes in Artificial Intelligence (LNCS), Springer Verlag, Berlin (1999)
11. Gebser, M., Ostrowski, M., Schaub, T.: Constraint Answer Set Solving. In: 25th International Conference on Logic Programming (ICLP09). vol. 5649 (2009)
12. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of ICLP-88. pp. 1070–1080 (1988)
13. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
14. Grasso, G., Leone, N., Manna, M., Ricca, F.: ASP at Work: Spin-off and Applications of the DLV System. In: Balduccini, M., Son, T.C. (eds.) Symposium on Constructive Mathematics in Computer Science (Oct 2010)
15. Marek, V.W., Truszczynski, M.: The Logic Programming Paradigm: a 25-Year Perspective, chap. Stable Models and an Alternative Logic Programming Paradigm, pp. 375–398. Springer Verlag, Berlin (1999)