Answer Set Programming for Computational Psychological Models

Sara Girotto (sara.girotto@ttu.edu)
Department of Psychology, Texas Tech University
Lubbock, TX 79404

Marcello Balducchini (marcello.balduccini@gmail.com)
Intelligent Systems, KRL, Eastman Kodak Company
Rochester, NY 14650

Keywords: formalization of psychological knowledge; answer set programming; short term memory.

Abstract: Our work explores the use of Answer Set Programming (ASP) to formalize and reason about psychological knowledge. To demonstrate the viability of ASP for this task, in this paper we discuss an ASP-based formalization of the mechanisms of Short Term Memory.

Introduction

Our work explores the use of Answer Set Programming (ASP) (Gelfond & Lifschitz, 1991; Marek & Truszczynski, 1999) to formalize and reason about psychological knowledge. Whereas some psychological models have a clear quantitative nature, which allows modeling e.g. with neural networks or Bayesian networks, other models have a more logical or qualitative nature and are not suitable for formalization with these techniques. ASP is a knowledge representation formalism allowing for concise and simple representations of defaults, uncertainty, common-sense and evolving domains, and has been demonstrated to be a useful paradigm for the formalization of knowledge of various kinds (e.g., Baral & Gelfond, 2005; Son & Sakama, 2009). For this reason, we believe that ASP can be used successfully for the formalization of psychological knowledge that is of qualitative nature. ASP is also directly executable, in the sense that the consequences of collections of ASP statements can be directly computed using computer programs. Hence, ASP-based formalizations of psychological knowledge can be viewed as computational models of the underlying psychological theories.

Answer Set Programming

In ASP, terms and atoms are formed according to the standard rules of first-order logic. A literal is either an atom a or its strong (also called classical or epistemic) negation $\neg a$. In its simplest form, a *rule* is a statement:

$$h \leftarrow l_1, l_2, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where h and l_i's are literals and *not* is the so-called *default negation*. The intuitive meaning of the rule is that a reasoner who believes $\{l_1, \ldots, l_m\}$ and has no reason to believe $\{l_{m+1}, \ldots, l_n\}$, must believe h. The availability of two types of negation is one important feature of ASP, allowing for great flexibility in knowledge representation. In particular, the way default negation is treated in ASP allows to easily encode defaults (such as "an action is allowed unless it is explicitly stated that it is not") and also to represent uncertainty and alternative, different views of the world (e.g. "either symbol *a* or symbol *b*, but not both, will be forgotten, but we do not know which one). The precise definition of the meaning of sets of ASP rules (called an *ASP program*) is given by the *answer set semantics* (Gelfond & Lifschitz, 1991), which characterizes a suitable notion of logical consequence. We omit further details due to space constraints; rather, in the rest of the paper we rely on the informal meaning of rules given above. From a practical perspective, the logical consequences of sets of ASP rules can be computed automatically, and rather efficiently, using computer programs called *ASP solvers*. These solvers can of course be also interfaced to pre-processors, post-processors and user interfaces, to build sophisticated end-to-end systems (e.g., Balduccini, Gelfond & Nogueira, 2006).

ASP-Based Formalization of STM

To demonstrate that ASP is suitable for and successful at formalizing psychological knowledge, we have developed an ASP-based formalization of the mechanisms of operation of Short-Term Memory (STM), as described by Atkinson & Shiffrin (1971). This theory was selected because it reflects the type of psychological knowledge that we aim at formalizing: it is mostly of qualitative nature and is expressed in the literature at a rather high level of abstraction. Moreover, its formalization is challenging because it involves modeling of a sophisticated dynamic domain, involving non-determinism, fixed-capacity storage, and decay over time. In order to show that the use of ASP is not limited to a single theory, we have formalized not only the traditional theory of STM (Atkinson & Shiffrin, 1971), but also an alternative STM model (e.g., Card, Moran & Newell, 1983) in which decay is influenced not only by elapsed time but also by other variables such as the number of chunks a user is trying to remember and retrieval interference with similar chunks activated in working memory. It is worth stressing that the accounts of STM (Atkinson & Shiffrin, 1971; Card, Moran & Newell, 1983) that have been formalized by means of ASP are relatively well-established models from the existing literature. Our purpose in this study is not to modify the models but to show that they can indeed be formalized using ASP.

Using a common methodology in ASP-based knowledge representation, the formalization process starts by condensing the description of STM in a number of *precisely formulated* statements in natural language. For example, the set of statements for the traditional theory of STM contains 16 items, including the following: (1) STM is a collection of symbols; (2) the size of STM is limited to ω elements (Cowan, 2000); (3) each symbol has an expiration time; (4) symbols can be added to STM; (5) if a new symbol is added to STM when ω elements are already in it, the symbol that is closest to expiring is removed from STM (i.e. forgotten).

Then, the logical representation of the formalization is created by choosing suitable relations and functions, and using them to encode the natural language statements and the underlying knowledge. Because we are interested in describing how the contents of STM change over time, we use two special relations, $holds(f,i)$, saying that property f holds at step i in the evolution of the contents of STM, and $occurs(a,i)$, saying that action a occurs at step i. For instance, statement (4) is encoded by a rule:

$$holds(in\_stm(S), I+1) \leftarrow occurs(store(S), I)$$

whose informal reading is: if the action of storing some symbol S (by convention an uppercase initial denotes a variable) occurs at some step I, then S will be in STM at the next step $I+1$. The rule is based on the stipulation that $store(S)$ represents the action of adding a symbol to STM, and that $in\_stm(S)$ encodes the fact that S is in STM.

Statement (5) is formalized by the following rule, as well as the definition (omitted to save space) of the auxiliary relations used in it:

$$\begin{aligned}\neg holds(in\_stm(S'), I{+}1) \leftarrow\ & S \neq S', occurs(store(S), I),\\ & stm\_max\_size(MX),\\ & curr\_stm\_size(MX, I)\\ & \text{not } some\_symbol\_expiring(I),\\ & oldest\_in\_stm(S', I)\end{aligned}$$

The informal reading of the rule is: when S is stored in STM, if the current size of STM equals its maximum size (represented by relations $curr\_stm\_size(MX,I)$ and $stm\_max\_size(MX)$, respectively) and no symbol is due to expire (written as $\text{not } some\_symbol\_expiring(I)$), then the symbol which is closest to expiring is removed from STM (encoded by means of the classical negation of $holds(in\_stm(S'), I+1)$, saying that S' will not be in STM at the next step).

Using terminology from the literature on ASP, the set of all rules formalizing STM is called *action description*, and denoted by Π_{STM}. To use the action description in order to predict the behavior of STM in a particular situation, one writes additional rules, say Π_{sit}, describing the situation, and then uses an ASP solver to compute the logical consequences of the ASP program consisting of Π_{STM} and Π_{sit}. For example, given the encoding of a memory-span test in which the subject is required to remember the sequence 1-7-3-2-6-5 and the maximum size of STM is of 4 items, the output of the ASP solver for $\Pi_{STM} \cup \Pi_{sit}$ contains statements such as $holds(in\_stm(digit\_1), 1)$ and $\neg holds(in\_stm(digit\_1), 5)$, showing that digit 1 was correctly stored initially, but forgotten at step 5 (because digit 6 was stored when STM was already at its full capacity).

In our study we also demonstrate the computational aspects of our modeling technique by creating an application of our formalization to the task of predicting a user's performance in the interaction with a graphical user interface. We developed an ASP-based representation of a scenario in which a user is told a sequence of tasks and is expected to execute it relying only on memory of the sequence. The prediction of the corresponding ASP program is in line with what the STM model (Atkinson & Shiffrin, 1971) predicts**,** correctly determining (1) if the user will or will not be able to remember and execute the sequence, and, in case the user forgets part of the sequence, (2) which pieces of information are (likely to be) forgotten and when. Execution of the ASP program is also rather fast, with most predictions computed in less than a second.

Conclusions

The ASP-based formalization appears promising in terms of producing accurate predictions of performance from mostly qualitative models of behavior. It also allows analysis and comparison of different psychological theories, as well as prediction of the outcome of experiments, thus making it possible to design better experiments and diminishing the need for prototyping and user testing. This success opens the door to the use of ASP for the formalization of other psychological knowledge and models, as well as for its practical use in HCI-oriented applications.

References

Atkinson, R. C., & Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American, 225,* 82-90.

Balduccini, M., Gelfond, M., & Nogueira, M. (2006). Answer set based design of knowledge systems. *Annals of Mathematics and Artificial Intelligence*, *47(1-2),* 183-219.

Baral, C., & Gelfond, M. (2005). Reasoning about intended actions. *Proceedings of the 20th International Conference on Artificial Intelligence* (pp. 689-694). AAAI Press.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Mahwah, NJ: Lawrence Erlbaum Associates.

Cowan, N. (2000). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24,* 87-185.

Gelfond, M., & Lifschitz, V. (1991). Classical negotiation in logic programs and disjunctive databases. *New Generation Computing*, *9*, 365-385.

Marek, V. W., & Truszczynski, M. (1999). Stable models and an alternative logic programming paradigm. In Apt, K., Marek, V., Truszczynski, M., Warren, D. (Eds), *The logic programming paradigm: A 25-year perspective*. Berlin, Germany: Springer Verlag.

Son, T. C. & Sakama, C. (2009). Negotiation using logic programming with consistency restoring rules. *21st International Joint Conferences on Artificial Intelligence (IJCAI).* Morgan Kaufmann Publishers Inc.