Reasoning in Highly Reactive Environments

Francesco Pacenza[1][0000−0001−6632−3492]
Advisor: Giovambattista Ianni[1][0000−0003−0534−6425]

University of Calabria, Arcavacata di Rende 87036, Italy
pacenza@mat.unical.it
https://www.mat.unical.it/ComputerScience

1 Introduction

The aim of my research project concerns **Reasoning in Highly Reactive Environments**.

As *reasoning in highly reactive environments*, we identify the setting in which a knowledge-based agent,with given goals, is deployed in an environment subject to repeated, sudden and possibly unknown changes. This is for instance the typical setting in which, e.g., artificial agents for video-games (the so called "bots"), cleaning robots, bomb clearing robots, and so on are deployed. In all these settings one can follow the classical approach in which the operations of the agent are distinguished in "sensing" the environment with proper interface devices, "thinking", and then behaving accordingly using proper actuators [18].

In order to operate in an highly reactive environment, an artificial agent needs to be:

- *Responsive* $\rightarrow$ The agent must be able to react repeatedly and in a reasonable amount of time;
- *Elastic* $\rightarrow$ The agent must stay reactive also under varying workload;
- *Resilient* $\rightarrow$ The agent must stay responsive also in case of internal failure or failure of one of the programmed actions in the environment.

Nowadays, thanks to new technologies in the field of Artificial Intelligence, it is already technically possible to create AI agents that are able to operate in reactive environments. Nevertheless, several issues stay unsolved, and are subject of ongoing research.

2 Motivation

The increasing availability of mobile or light terminals, characterized by good computational power and by the accessibility to sensors of various kind, paves the way to distributed and widespread artificial intelligence applications. In this respect, it is necessary to have special tools ensuring that these terminals can autonomously perform intelligent tasks, and not only communicate with a centralized "remote intelligence" [21,22]. Moreover, the availability of data streams has accelerated advances in information processing tools that do not necessarily require to store data for static querying, but push query answers to consumers as soon as they become available. In this setting, stream reasoning aims at providing languages and tools for data that changes at high rate. Notably, in the field of stream reasoning, some work aim at introducing advanced reasoning capabilities under a well understood and formalized framework [2, 3]. We aim at extending this work by considering explicitly some important issues that come into play when assuming that the agent at hand is deployed in an highly reactive environment.

3 State of the Art

During the last years, some different approaches have been explored in order to create responsive agents that are able to operate in highly reactive environments. Most of them make use of *deep learning techniques* in order to let the agent be able to emulate human learning. These techniques are applied in the field of video-games for automatic game playing accordingly to [16]. The same technique can be also used for predicting human behavior as discussed in [15]. This kind of approaches have their "limits" especially when one aims to, *a)* generalize reasoning (indeed, deep learning techniques are usually built ad-hoc for a specific operating environment); *b)*, to create development and reasoning tools that can be "used" as middleware for the production of the so called "bots", in which "intelligent" capabilities can be tuned, refined and prototyped; and *c)*, to provide intelligible explanations of choices taken by the agent.

In the *Answer Set Programming (ASP)* community there have been some recent studies about reasoning in reactive environments; in particular, researchers recently proposed ASP-based forms of reactive reasoning. One of these (called "*oclingo*") allows to implement real-time dynamic systems running online in changing environments (as described in [14]). This new technology paves the way for applying ASP in many areas like robots, bots, and video-games due to the fact that now an ASP solver can be also adopted for online usage. However *oclingo* cannot be defined as totally "declarative". Indeed, the user has to specify how the environment will evolve and the operational design of some *base*, *cumulative* and *volatile* rules, that should be respectively evaluated only one time, at each iteration and only in one iteration and then discarded.

A second promising ASP extensions like *HEX* [1] provides the possibility of a bidirectional access to external sources of knowledge and/or computation using the concept of external atoms; the extension *ActHEX* [13] of HEX programs introduces the notion of action atoms, which are associated to corresponding functions capable of actually changing the state of external environments. Using this ASP extension it is in practice possible to realize artificial agents able to operate in many different settings (for instance, an HEX extension has been successfully used for implementing an AI agent operating in video-games scenarios as described in [5]). However, ActHex does not make explicitly possible to check and act if the planned actions in the schedule have failed at each shot. Moreover, in very complex scenarios, the implementation of ActHex knowledge bases could become very hard and unhandy due to the complex ways for programming the order of actions composing a scheduled plan. Nevertheless, we found in literature a large variety of languages designed for programming logical Agents and Multi-Agent Systems. One of these is Jason [4]; Jason is a platform for developing agents based on the Agent-Speak logic language. A Jason agent consists of a set of plans, each of which has a triggering event, a context, and a body of actions. Moreover also other languages like IMPACT [12], DALI [8], ALP [11] and ASTRA [10] can be used to design logical Agents and Multi-Agent Systems.

4 Problem Statement and Contributions

We decided to focus our attention on the following open issues which are clearly related to **Reasoning in Highly Reactive Environments**:

– Estimating the time required for the execution of logic programs;
– Handling the correct execution of all the planned actions and recovering from failure (both because of internal or external failure);
– Handling the order of the planned actions in a simpler and quicker way, possibly declarative.

Studying the first issue is important in order to decide if a program is able to produce a complete output in a specific amount of time. If not, one could decide at runtime to use another (”easier”) program to obtain a result (answer set) that can be executed by the agent in the allotted time [19, 20]. In this way the agent will always have some action to perform, although possibly suboptimal. Furthermore, it is also very important to find a way to check if all the actions in a schedule are correctly executed after each shot. This way, one could instruct the agent to react in the cases of failure [9, 17, 28]. Finally, handling the order of the planned actions in a simpler and quicker way, would make the implementation of bots easier and more immediate, especially when the schedule is composed of a lot of different actions.

The aim of our research is to explore the possibility to overcome the limits of the currently available systems. Some of these tools could also be extended in order to make them easier to be used. Moreover, we would like to produce a *middleware* that will be useful for the creation and deployment of bots in real environments. This middleware should have the following properties:

– *Simple* → It has to be easy to use;
– *Generalizable* → It has to be as general as possible, so it can be applied in a variety of contexts;
– *Extensible* → The end user has to be able to extend the middleware easily and in a very few time.

In other words, we would like to create a *reactive, declarative* and *generalizable* scripting language that could be potentially applied not only in a single context, but in more than one, starting from the video-games to robotics systems and so on.

In our intentions there is also the aim to overcome the issues regarding the responsiveness of the AI agents in reactive environments. This could be achieved, for instance by using a partial evaluated answer sets strategy or by preparing many different programs, estimating the execution time for each of them, and finally running the best program that will end in the bounds of the available time. We aim to explore both possibilities.

Another possible research field is the usage of *behavioral models* for the automatic selection of an agent behavior depending on the current context (mathematically speaking, we can say that an agent's behavior is described by the ***agent function*** that maps any given percept sequence to an action [25]).

After a thorough study of previous literature, we plan to focus on one of the goals above, while exploring a second one as a secondary objective.

5 Evaluation Plan

The theoretical part of my work will undergo the traditional review process aimed to check soundness and correctness. As for the practical aspects, these will undergo an experimental evaluation focusing on performance and usability of the framework. In particular we aim to obtain a good performance both in terms of execution time and optimal results by reusing as much as possible the previously computed plans and then replanning and rescheduling the tasks with the new information gained from the current state of the world.

6 Preliminary or Intermediate Results

In order to achieve our goal, we started to investigate the literature [1–3, 5, 13–17, 20]. Then, we focused our research on artificial agents for video-games (also called "bots"). We embedded the rule-based *Reasoning Module* into the well-known Unity[1] game development engine. To this end, we presented an extension of EmbASP[2], a framework to ease the integration of declarative formalisms with generic applications. Finally, we proved the viability of our approach [3] by developing a proof-of-concept Unity game that makes use of ASP-based AI modules [6].

As a side research topic we investigated to what extent ASP-based approach is scalable enough for industrial contexts in the field of video games, by proposing a Unity extension capable to automatically generate dungeons maps [4]. In

[1]`https://unity3d.com/unity`

[2]`https://www.mat.unical.it/calimeri/projects/embasp`

[3]All the development material, including logic programs, source code and a fully playable version of the game are available at
`https://github.com/DeMaCS-UNICAL/Pac-Man-Unity-EmbASP`.

[4]Both versions of our prototypes, together with logic program specifications and source code are fully available online at

this context we first investigated over the usage of a partition-based generation technique [26], then we proposed a multiple step-generation approach, set in the context of the 2-D caves generation domain, where each step is declaratively controlled by an ASP specification in ASP. With respect to existing literature [23, 27], our approach promises to be better scalable to real contexts with higher size mazes; experiments aimed at confirming that are currently ongoing. Finally we developed two plugins based on our generation technique, which were respectively deployed as an asset available in the Unity development and in the GVGAI [24] frameworks, respectively [7].

7 Conclusions

In this work we would like to investigate in depth in the field of stream reasoning. Our goal is to create a framework capable to ease the generation of responsive, elastic and resilient artificial agents. In order to make it possible, it is necessary to explore the possibility to restart at runtime the computation of a logic program reusing all the already done computations. We have presented some of the steps forward made and the research themes that we would like to deepen (restartability of computation at runtime, reusability of content, general language for AI and so on).

References

1. Basol, S., Erdem, O., Fink, M., Ianni, G.: HEX Programs with Action Atoms. Technical Communications of the 26th International Conference on Logic Programming **7**, 24–33 (January 2010)
2. Beck, H., Dao-Tran, M., Eiter, T.: LARS: A Logic-based framework for Analytic Reasoning over Streams. Artif. Intell. **261**, 16–70 (2018). https://doi.org/10.1016/j.artint.2018.04.003, `https://doi.org/10.1016/j.artint.2018.04.003`
3. Beck, H., Eiter, T., Folie, C.: Ticker: A system for incremental ASP-based stream reasoning. TPLP **17**(5-6), 744–763 (2017). https://doi.org/10.1017/S1471068417000370, `https://doi.org/10.1017/S1471068417000370`
4. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology). John Wiley &#38; Sons, Inc., USA (2007)
5. Calimeri, F., Fink, M., Germano, S., Humenberger, A., Ianni, G., Christoph Redl, D.S., Tucci, A., Wimmer, A.: AngryHEX: an Artificial Player for Angry Birds Based on Declarative Knowledge Bases. IEEE Transactions on Computational Intelligence and AI in Games **8**(2), 128–139 (June 2016)
6. Calimeri, F., Germano, S., Ianni, G., Pacenza, F., Perri, S., Zangari, J.: Integrating Rule-Based AI Tools into Mainstream Game Development. In: Benzmüller, C., Ricca, F., Parent, X., Roman, D. (eds.) Rules and Reasoning. pp. 310–317. Springer International Publishing, Cham (2018)

`https://github.com/DeMaCS-UNICAL/DCS-Maze_Generator-GVGAI` and
`https://github.com/DeMaCS-UNICAL/DCS-Maze_Generator-Unity`

7. Calimeri, F., Germano, S., Ianni, G., Pacenza, F., Pezzimenti, A., Tucci, A.: Answer Set Programming for Declarative Content Specification: a scalable partitioning-based approach. In: Ghidini, C., Magnini, B., Passerini, A., Traverso, P. (eds.) 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018). Springer International Publishing (2018)
8. Costantini, S., Tocchio, A.: The DALI Logic Programming Agent-Oriented Language. In: Alferes, J.J., Leite, J. (eds.) Logics in Artificial Intelligence. pp. 685–688. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
9. Crestani, D., Godary-Dejean, K.: Fault Tolerance in Control Architectures for Mobile Robots: Fantasy or Reality? 7th National Conference on Control Architectures of Robots (2012)
10. Dhaon, A., Collier, R.W.: Multiple inheritance in agentspeak(l)-style programming languages. In: Proceedings of the 4th International Workshop on Programming Based on Actors Agents &#38; Decentralized Control. pp. 109–120. AGERE! '14, ACM, New York, NY, USA (2014). https://doi.org/10.1145/2687357.2687362, `http://doi.acm.org/10.1145/2687357.2687362`
11. Drescher, C., Schiffel, S., Thielscher, M.: A declarative agent programming language based on action theories. In: Ghilardi, S., Sebastiani, R. (eds.) Frontiers of Combining Systems. pp. 230–245. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
12. Eiter, T., Subrahmanian, V., Rogers, T.: Heterogeneous active agents, iii: Polynomially implementable agents. Artificial Intelligence **117**(1), 107 – 167 (2000). https://doi.org/https://doi.org/10.1016/S0004-3702(99)00104-6, `http://www.sciencedirect.com/science/article/pii/S0004370299001046`
13. Fink, M., Germano, S., Ianni, G., Redl, C., Schller, P.: ActHEX: Implementing HEX Programs with Action Atoms. International Conference on Logic Programming and Nonmonotonic Reasoning **8148**, 317–322 (2013)
14. Gebser, M., Grote, T., Kaminski, R., Schaub, T.: Reactive answer set programming. In: Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning. pp. 54–66. LPNMR'11, Springer-Verlag, Berlin, Heidelberg (2011), `http://dl.acm.org/citation.cfm?id=2010192.2010201`
15. Hartford, J.S., Wright, J.R., Leyton-Brown, K.: Deep Learning for Predicting Human Strategic Behavior. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2424–2432. Curran Associates, Inc. (2016), `http://papers.nips.cc/paper/6509-deep-learning-for-predicting-human-strategic-behavior.pdf`
16. Justesen, N., Bontrager, P., Togelius, J., Risi, S.: Deep Learning for Video Game Playing. Cornell University Library (August 2018)
17. Koenig, S., Likhachev, M.: Improved fast replanning for robot navigation in unknown terrain. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. vol. 1, pp. 968–975. IEEE (2002)
18. Kowalski, R., Sadri, F.: From logic programming towards multiagent systems. Annals of Mathematics and Artificial Intelligence **25**(3), 391–419 (Nov 1999). https://doi.org/10.1023/A:1018934223383, `https://doi.org/10.1023/A:1018934223383`
19. Liang, S., Kifer, M.: Deriving Predicate Statistics for Logic Rules. In: Proceedings of the 6th International Conference on Web Reasoning and Rule Systems. pp. 139–155. RR'12, Springer-Verlag, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33203-6_11, `http://dx.doi.org/10.1007/978-3-642-33203-6_11`

20. Maratea, M., Pulina, L., Ricca, F.: Automated Selection of Grounding Algorithm in Answer Set Programming. In: Proceeding of the XIIIth International Conference on AI*IA 2013: Advances in Artificial Intelligence - Volume 8249. pp. 73–84. Springer-Verlag New York, Inc., New York, NY, USA (2013). https://doi.org/10.1007/978-3-319-03524-6_7, `http://dx.doi.org/10.1007/978-3-319-03524-6_7`
21. Mileo, A., Merico, D., Bisiani, R.: Non-monotonic reasoning supporting wireless sensor networks for intelligent monitoring: The sindi system. In: Erdem, E., Lin, F., Schaub, T. (eds.) Logic Programming and Nonmonotonic Reasoning. pp. 585–590. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
22. Mileo, A., Merico, D., Pinardi, S., Bisiani, R.: A logical approach to home health-care with intelligent sensor-network support. Comput. J. **53**, 1257–1276 (2010)
23. Nelson, M.J., Smith, A.M.: ASP with Applications to Mazes and Levels, pp. 143–157. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-42716-4_8, `https://doi.org/10.1007/978-3-319-42716-4_8`
24. Pérez-Liébana, D., Liu, J., Khalifa, A., Gaina, R.D., Togelius, J., Lucas, S.M.: General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms. CoRR **abs/1802.10363** (2018)
25. Russell Stuart J. and Norvig Peter: Artificial Intelligence: A Modern Approach. Pearson Education, 3 edn. (2010)
26. Shaker, N., Liapis, A., Togelius, J., Lopes, R., Bidarra, R.: Constructive generation methods for dungeons and levels, pp. 31–55. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-42716-4_3, `https://doi.org/10.1007/978-3-319-42716-4_3`
27. Smith, A.M., Mateas, M.: Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 187–200 (Sept 2011). https://doi.org/10.1109/TCIAIG.2011.2158545
28. Stentz, A.: The Focussed D* Algorithm for Real-time Replanning. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2. pp. 1652–1659. IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995), `http://dl.acm.org/citation.cfm?id=1643031.1643113`