AI and videogames: a "drosophila" for declarative methods

Denise Angilica[1], Giovambattista Ianni[1], Francesca A. Lisi[2] and Luca Pulina[3]

[1]University of Calabria
[2]University of Bari Aldo Moro
[3]University of Sassari

Abstract

Videogames and research in artificial intelligence (AI) techniques share a fruitful past of reciprocal knowledge exchange. On the one hand, videogames offer unsolved challenges to the academy: these challenges are as hard as those offered by what we can call "serious" applications, yet they can be faced in a controlled and reproducible setting. This characteristic makes videogames a kind of advanced "drosophila" for the researcher in AI: they are the ideal controlled ground in which to invent, experiment, and test new AI paradigms, methodologies and techniques. It is not uncommon to resort to simulated game environments as lesser expensive, yet comparably complex, digital twins. On the other hand, the videogame industry (VI) itself is exemplary of typical demands that AI research cannot meet yet. The VI looks for reduced development costs, better integration with AI tools and real-time performance. Especially, knowledge transfer between developers and AI tools must be as fast and smooth as possible: this is one of the reasons why the machine learning (ML) revolution had so far a controversial reception in the VI, in that ML provides black-box AI modules which are not easily "tunable" and configurable at will, and have non-negligible design-time costs. In order to overcome the above limits, one can consider declarative knowledge representation techniques (DKR). However, some of the shortcomings of DKR methods are fairly challenging to be addressed: performance, ease of use, integration, mining of reusable deductive knowledge are not up to the par yet. All these limitations prevent the real adoption of declarative paradigms in many highly-demanding applicative settings of which the videogame development field is an exemplary generalized testbed. Narrowing the above gaps is nontrivial challenge. In this paper we identify some of the key issues which we deem important for the research community and outline our current research progress.

Keywords

Declarative logic, Answer Set Programming, Knowledge Representation, Games and Videogames, Stream Reasoning,

1. Context and motivation

Videogames and the AI research. Many examples of the usage of declarative languages in the industrial videogame realm exists, starting from the pioneer F.E.A.R. game [1], which

IPS-RiCeRcA-SPIRIT 2022: 10th Italian Workshop on Planning and Scheduling, RiCeRcA Italian Workshop, and SPIRIT Workshop on Strategies, Prediction, Interaction, and Reasoning in Italy.

denise.angilica@unical.it (D. Angilica); giovambattista.ianni@unical.it (G. Ianni); francesca.lisi@uniba.it (F. A. Lisi); lpulina@uniss.it (L. Pulina)

0000-0002-4069-978X (D. Angilica); 0000-0003-0534-6425 (G. Ianni); 0000-0001-5414-5844 (F. A. Lisi); 0000-0003-0258-3222 (L. Pulina)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3345/paper8_RiCeRCa1.pdf

used STRIPS-based planning [2]. Other remarkable examples are the games Halo [3] and Black & White [4]. If we look at videogames from the basic research perspective, it must be noted the longstanding interest in using (video-)games as a controllable and reproducible setting in which to face open issues: one might cite the GDL [5], VGDL [6] and Ludocore [7] languages adopted for declaratively describing General Game Playing [8]. The Planning Domain Definition Language (PDDL) found natural usage in the videogame realm [9, 10, 11]; among its sister languages, we will herein focus particularly on Answer Set Programming (ASP), the known declarative paradigm with a tradition in modeling planning problems, robotics, computational biology as well as other industrial applications [12]. ASP does not come last in its experimental usage in videogames: it has been used to various extents, e.g., for declaratively generating level maps [13] and artificial architectural buildings [14]; it has been used as an alternative for specifying general game playing [15], for defining artificial players [16] in the Angry Birds AI competition [17], and for modelling resource production in real-time strategy games [18], to cite a few.

ASP specifications are composed of set of rules, hard and soft constraints, by means of which it is fairly easy to express qualitative and quantitative statements. In general, a set of input values F (called *facts*), describing the current state of the world, are fed together with an ASP specification S to a *solver*. Solvers in turn produce sets of outputs $AS(S \cup F)$ called *answer sets*. Answer sets contain the result of a decision-making process in terms of logical assertions which, depending on the application domain at hand, might encode actions to be made, employee shifts to be scheduled, protein sequences, and so on.

When evaluating S and F a traditional solver performs two consecutive steps: grounding (also called instantiation) and solving (also called answer set search). Instantiating a program consists in generating, rule by rule, substitutions of variables with constants, thus obtaining an equivalent propositional program.

Despite their potential advantages, the widespread adoption of methods like ASP in games is still prevented by a number of shortcomings, which one roughly categorize in *i)* integration issues and *ii)* performance issues.

Concerning the first, ASP-based solutions have been coupled with industrial applications using many schemes which were proposed in the last decade. These differ on how the so-called "procedural" side and "declarative" side are coupled, and on which of the sides is at the center of the development picture [19, 20, 21, 22].

However, during a videogame, decision-making processes are continuously repeated within the so called *game loop*. The game loop is, in a way, a sense-think-act cycle in disguise, where the *think* step might be occupied by a logic-based reasoner. This context makes the integration of reasoners/ASP solvers non-obvious: how to cope with fast changing sensor readings? how to accommodate long running reasoning tasks? What if a reasoning task is no longer needed because of a sudden change in the game scene? How to reuse at least part of the wasted computational effort?

The ThinkEngine [23], developed by our team, allows programmers to deal with AI modules inside the popular Unity game and industrial development engine [24] and is a good starting point for getting closer to the above goals. As for performance, ASP solvers greatly improved in the last years [25]. Effort has been made on incremental, online and stream computing of ASP inputs [26, 27, 28]. Fast, repeated reasoning tasks required by real-time applications

seem, however, out of reach unless manual tuning and optimizations are introduced on a per application basis [29]. The ASP-based player appearing in the Angry Birds AI Competition [16] and other benchmarks on games show that the performance gap can be reduced with more research [30].

In this paper, we overview the key issues that prevent declarative paradigms being adopted in the VI. In section 2 we briefly illustrate our current research progress, introducing the ThinkEngine Asset; in section 3 we outline some selected open issues in the context of AI and videogames that we deem relevant; in section 4 we discuss the general advantages of videogames and declarative paradigms being coupled together; in section 5 we overview how this line of research could bridge videogames to academic research and to society in general.

When evaluating S and F a traditional solver performs two consecutive steps: grounding (also called instantiation) and solving (also called answer set search). Instantiating a program consists in generating, rule by rule, substitutions of variables with constants, thus obtaining an equivalent propositional program.

2. The ThinkEngine Asset

Figure 1, adapted from the description of our ThinkEngine tool [23] shows our proposal of an integration scheme between a reasoner based on declarative specifications and a game engine. A number of so-called *brains* are able to interact with a videogame scene.

An AI developer can use the ThinkEngine by first identifying objects and/or parts of the game logic that are to be connected to brains; *sensor readers* are wired from the game scene to brains, and *actuators* from brains to the game scene. A brain consists of a simple high-level specification that describes decision criteria: one can add rules, constraints, soft constraints and other declarative statement types. A brain reasoning task can be then triggered on specific events or on a cyclic basis. *Reactive brains* produce immediate modifications on the game, while *Planning brains* can be used to synthesize complex execution plans or, in principles, even arbitrarily reprogram reactive behaviors of artificial players.

In the case of our ThinkEngine tool, reasoning modules are implemented using declarative specifications written using ASP; an ASP solver executes such specifications, then its outputs are converted into actions or plans to be executed on the game world. Thinkengine is implemented respecting a paradigm-agnostic stance: a wiring infrastructure for connecting other solvers based on other declarative paradigms (PDDL, SMT or others) is provided. In the specific setting of Thinkengine, most of the interoperability burden is implemented within the EmbASP library [31].

The tool is available as a plugin for the known game development engine Unity [24]. Unity owns 45% of the market share as a game development engine: it is not however limited to the development of ludic products as it is appreciated also for the development of simulations and industrial applications.

From the software development perspective, the presence of brains based on declarative tools implies several potential benefits, like:

- declarative tools can be used for defining, in a very short time, different aspects of game

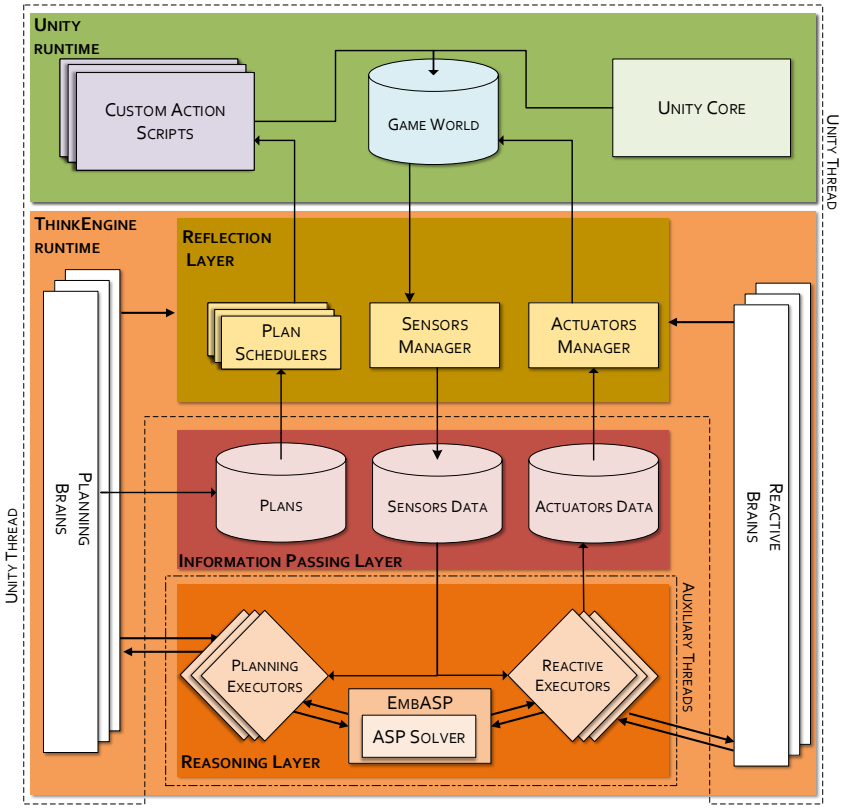

Figure 1: General run-time architecture of the ThinkEngine framework

AI, like: modelling of non-player characters (NPC) with spatio-temporal reasoning based AIs, automated game resource management, path planning, declarative (and not more "procedural") content generation, automated narrative generation based on qualitative descriptions, dynamic dialogue systems, etc.

- non programmers can participate in the implementation of decision making modules;
- the strong decoupling between declarative decision making and action execution makes much easier to separate and parallelize the development of both;
- the decoupling from the game engine itself allows standalone testing, isolated debugging and optimization.

However, the above advantages pair up with the known barriers to the adoption of declarative tools in demanding applications and in the videogame application settings in particular. Such barriers are mostly related to the actual declarativity of the formalism at hand, its evaluation efficiency, and the integration methodology with applications.

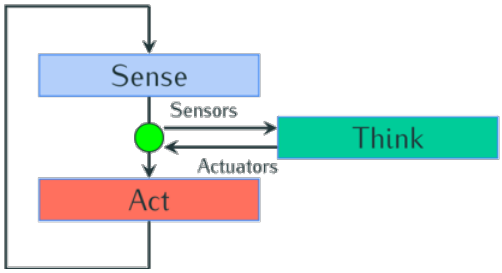

Figure 2: Hybrid Deliberative/Reactive (HDR) schema.

3. Some selected open issues

Integration. When one aims to integrate reasoning modules in industrial applications, and especially videogames, two main technical obstacles arise. One concerns the information flow and the coordination between reasoning tasks and other parts of the software; the second obstacle is related to the symbol grounding, i.e. the longstanding issue of abstracting raw data to a symbolic synthesis thereof (not to be confused with the close technical concept of "grounding" meant in the sense of "instantiation" of an ASP program).

Concerning the coordination of the reasoning tasks, recall that a videogame is typically executed by running the so called game-loop routine [32]. The game loop consists in a single-threaded repeated execution of update operations to the current game scene: within each step of this cycle, user input is processed, AI decision-making operations are made, and the game scene is modified accordingly. A physics simulation of the world and the general game logic is also taken into account when changing the game environment. The operations performed within the game loop strictly mimic the reactive sense-think-act cycle typical of agents and robotic systems [33], in which sensing, thinking and acting steps are repeatedly performed.

Similarly to real life applications, in videogames reactive AI modules coexist with non-reactive portions of the game logic: reactive modules are usually implemented using relatively fast techniques, such as behavior trees, finite state automata, rule sets or combinations thereof [34], and can run in the main game-loop. This type of module comes into play when quick "instinctive" decisions need to be taken by artificial characters: think, e.g., at programmed fight-or-flight responses when an artificial opponent is attacked.

On the other hand, strategic decision-making processes are likewise necessary. They are realized using long term, goal-oriented techniques, which often involve utility-based, planning, and simulation techniques. Longer running reasoning tasks are for instance required for common videogame AI modules such as path-planning, hierarchical task planning and resource management. As these require longer processing time, their execution within the default game-loop is troublesome. It is thus natural to generalize this reactive/non-reactive duality to the so-called hybrid deliberative/reactive paradigm [33] (HDR, Figure 2), in which fast "instinctive" behaviors are combined in a proper way with long term reasoning tasks such as plan generation.

This model has in turn been recently generalized both in economics and social science [35] and in the wider AI community [36], by proposing to model rational agents in terms of two

interacting sides: *System 1*, the fast, instinctive, reactive, often non-symbolic subsystem; and *System 2*, a subsystem capable of better strategic decisions, yet requiring longer reasoning times and a proper symbolic preprocessing and abstraction layer. Both System 1 and System 2 decision-making activities can be deemed important and need to co-exist.

The interaction between both types of reasoning calls for the introduction of a scheme similar to the general HDR one, in which long-running reasoning tasks are made possible, but do not enforce heavy computational loads on the main game loop. The outcome of non-reactive tasks can be used to act directly on the game logic, or to re-program reactive behaviors of game characters. Although prototypical systems, like the ThinkEngine developed by our research group, offer a form of HDR scheme in which ASP reasoning tasks can be run asynchronously, many aspects are not properly systematized yet. Especially, there is no clear semantics on how deliberate modules can re-program reactive modules and execute plans. A good starting point in this respect can be hybrid ASP languages like ActHEX [21] that are meant to synthesize actions to be executed in an external environment.

Abstraction. Concerning the second integration issue, abstraction has been studied in ASP [37] especially aiming at reducing the number of symbols processed by solvers, thus reducing computing times yet preserving some form of semantic equivalence. An exploration of how fine-grained knowledge, such as raw spatial and temporal floating point data, can be systematically abstracted to a symbolic layer suitable for ASP reasoning has not been done yet. In this respect interval reasoning technique[38] look promising and could be beneficial on at least two aspects: performance-wise, abstraction can often make the difference in reaching the desired performance cutoff threshold. Abstraction is common practice in the videogame realm, such as when pixel-grained game maps are abstracted to a few strategic waypoints. As a second advantage, the availability of an automated abstraction layer can cut down design times, as this operation usually requires time-consuming manual design decisions.

Performance. Automated decision making in dynamic environments, where the world description is continuously and fastly changing, enforces strict reasoning time limits. Let us consider a decision-making reasoning task T, which can be either reactive or nonreactive. Specifically, if T is represented using ASP syntax, it is fed to an ASP solver composed of two separate computational steps: instantiation (also called "grounding", not to be confused with the related notion of symbol grounding), and resolution (also called "solving" in the proper sense). Both steps have their impact in performance, and especially grounding is crucial when big data flows and reactive tasks come into play. Rapid environment changes are typical of videogames, which are thus an ideal controlled environment for introducing *stop & restart techniques* capable of stopping no more valid reasoning tasks and resuming them under updated input data: this is needed, e.g., in case of events that trigger replanning. Also, no matter whether reactive or not, a time consuming "think" step may constitute a great performance bottleneck. The execution times of reasoning tasks can be greatly improved if *incremental techniques* are introduced. An incremental solver makes only quick differential updates to its internal state when some of the input data are changed. Videogames are exemplary also in this respect and could greatly benefit from AIs implemented using fast incremental solving techniques.

Whenever it is not desirable or possible to execute long, offline, reasoning tasks like in hybrid reactive/deliberative models, the introduction of incremental reasoning techniques becomes even more strategic. Much effort has been done recently in this direction: some ASP systems allow full incrementality, at the price of manually programming which and how parts of an ASP specifications should be (re-)evaluated [27], thus loosing some declarativity; other contributions enable a form of incrementality which is transparently performed "under-the-hood" from the perspective of the application designer, but are limited on the language expressivity [26]; the recent work of our group, based on the notion of "overgrounding" is also fully transparent to specification designers, yet it limits differential computations to the so called "grounding" stage only [30, 28, 39]. This choice limits performance improvements mostly to reactive tasks. None of the above work addressed so far the issue of computation restarts triggered by external input changes. We nonetheless foresee that our recently developed overgrounding techniques could be lifted to all the stages of the ASP solving pipeline, and can be generalized to introduce forms of computation restart techniques. New extensions should inherit the support of all the linguistic features of ASP, and keep the incremental evaluation out of the necessity of manual programming.

Whitebox AI procedures for content generation. Machine learning techniques are often used in the videogames industry for content generation of videogame levels that are aesthetically like human-authored examples (see, e.g., [40]). For this purpose, Generative Adversarial Networks (GANs) have been shown particularly effective on this task because they enable the creation of levels that are stylistically like human examples [41, 42]. However, it has been recently shown that GANs can often fail to produce videogame levels with playability criteria [43], making tiresome human intervention necessary to "repair" a machine-generated level. In order to automate this process, several techniques have been proposed, most of them focused on a generate-then-repair approach for first generating levels from models trained on human-authored examples and then repairing them with minimum cost edits to render them playable. Such techniques leverage on Mixed Integer Linear Programming (MILP) to encode the playability constraints for the repair procedure (see, e.g., [44]). It is appealing to investigate the effectiveness of symbolic methods, namely ASP, Pseudo-Boolean, and Satisfiability Modulo Theory (SMT), in combination with subsymbolic methods (GANs), to encode playability constraints for automated repair of generated content.

Declarative pattern mining. The detection of changes in the scenes or in the game characters is relevant e.g. for triggering NPC behavior changes, or for simply proposing different background music or elements. Pattern mining traditionally provides a bunch of techniques for the discovery of regularities in a data set, e.g. so-called frequent patterns are statistically significant regularities in a set of transactions. The classic problem of frequent pattern mining [45] concerns indeed the enumeration of all the patterns whose absolute support exceeds a user-defined minimum support threshold minSup.

A complementary problem is the discovery of statistically significant *differences* (or *contrast*) between two disjoint sets of transactions [46]. This requires the enumeration of all the patterns whose absolute support difference exceeds the user-defined minimum support threshold. The

discovery of the pattern types such as contrast/difference patterns, as well as emerging patterns and change patterns, can be viewed as special cases of the same pattern mining problem, whose aim is to detect patterns relating two or more given datasets. These patterns have been shown to be a powerful method for constructing accurate classifiers, since they can describe emerging behavior with respect to a property of interest. One such feature is desirable also for the modeling of non-player characters.

An interesting direction of work is the investigation of some of these variants of the frequent pattern mining problem within a declarative framework, notably ASP. A driving feature of ASP solvers is indeed the possibility of enumerating all solutions, in the form of answer sets. This appears particularly suitable for pattern mining problems. Both contrast, emerging and change pattern mining fall into the broader category of constraint-based pattern mining [47], and can therefore be effectively encoded in declarative formalisms such as ASP, like in [48]. The problem specifications will naturally combine different constraints without having to devise new solving algorithms for specific mining tasks, thanks to the declarative nature of ASP. The challenges here are the identification of the most appropriate level of abstraction in the representation of a scene or a player, and the definition of change underlying the discovery process.

A second direction to pursue concerns the use of background knowledge related to the specific domain for the videogame in hand. Such prior domain knowledge might cover some aspects deemed of interest from either the methodological viewpoint or the applicative one. Particularly relevant in the videogame context is the spatial dimension. So, the background knowledge might come in the form of hierarchies of spatial objects, and patterns could then highlight spatial relations among the entities and offer descriptions of the scenes at multiple levels of granularity, as done in [49]. This direction of research would bridge the gap between the declarative approach and the object-based representations that are typical in the videogame context. Hybrid knowledge representation formalisms could be considered to the purpose, notably those formalisms which integrate ASP and DLs [50].

4. Impact for applicative research

The Videogame industry is an emblematic representative of a real-world application field where practical barriers prevent the wide adoption of AI methodologies, both symbolic/deductive and sub-symbolic/inductive. Elevated standards of efficiency and ease of use set a hard to reach cutoff threshold for the introduction of promising paradigms such as ASP in the above contexts. Today, ASP applications can be made performant, yet at the price of much "under-the-hood" optimization burden of which only ASP specialists are capable. This compromises ease of use and declarativity and is not welcome from the industrial perspective.

One might thus look at *redefined adherence to declarativity and a transparent efficiency*, confining back under-the-hood the technicalities of the ASP semantics yet ensuring efficient evaluation. Besides the widely accepted interest in explainability, a general goal for the AI community would be to regain the ability of *knowledge transfer* and *elaboration tolerance*: the first, not to be confused with transfer learning, is intended as achieving rapid transfer of knowledge from AI designers to AI modules. *Elaboration tolerance* [51] represents the capability of easily configuring and modifying the behavior of an AI module whenever new desiderata need to be

added. It is thus important to offer whitebox AI components featuring "visible knobs" which can be used for instant tuning and configuration. This is in contrast with subsymbolic techniques that, although very useful in several respects, currently lack explainability, a fundamental requirement for achieving knowledge transfer and elaboration tolerance.

Besides these general goals, progress in videogames AI broadly overlaps with many applicative research fields. A punctual, yet incomplete list of possible technological applications of videogames AI includes:

Virtual environments and augmented reality. Serious games and virtual environment exploration tools are customarily built with game development tools, and share many requirements with videogames. In this respect declarative path-planning and declarative content generation are naturally appealing. Early research lines are moving in this direction [52].

Simulation and predictive maintenance. When realizing a simulation task in the automotive field, controlled content generation of images is of paramount importance in order to train ML algorithms. Concerning predictive maintenance, advances in integrating declarative AI in games could offer new digital twin-based solutions in order to test predictive maintenance techniques.

Robot guidance and robotic surgery. Among many industrial and research usages of the Unity engine it is worth mentioning robotic surgery [53], a field in which automated motion planning and spatial reasoning are seen with keen interest. In the same direction goes the research investigating the relationship between fine-grained motion planning and declarative methods [54].

5. Bridging videogames to research: a Drosophila for AI

There are countless research application fields where it is very expensive or difficult to let researchers access real data or physical resources and for which videogames represent the natural *digital twin*: traffic control, smart cities, autonomous driving, robotic surgery, digital forensics, are only a few examples. Basic KR research areas such as spatio-temporal reasoning, deontic and legal reasoning, qualitative physics, planning, to cite a few, can potentially have strong beneficial fall-out on applicative research like the above. This impact can be boosted if reproducible controlled environments were available and made of the same complexity of real contexts. Enabling easy-to-wire declarative modules in one of the most used game and simulation engines goes in this direction, also considering the role of ASP as a middle-ware for implementing many of the abovementioned KR formalisms.

An incomplete list of hot research topics can include:

Digital Forensics. Evidence Analysis, a crucial phase in Digital Forensics, ranges from the analysis of fragmented incomplete knowledge, to the reconstruction and aggregation of complex scenarios involving time, uncertainty, causality and alternate possibilities. The Scientific Investigation experts usually proceed by relying on their experience and intuition as no

established methodology exists today. Games can be useful and powerful means for supporting digital forensics. They can provide digital twins where hypotheses, made using temporal reasoning and qualitative physics, can be evaluated in a comparative way. The automated discovery of regularities in crime scenes provides precious hints to investigators [55, 56, 57]: when empowered with reasoning modules, games show an added value with respect to other simulation environments. In this respect it is worth mentioning the COST Action "Digital forensics: evidence analysis via intelligent systems and practices" (DigForASP). DigForASP has set up an international network for exploring the potential of AI techniques (in particular, KR and Automated Reasoning) in the Digital Forensics field, and for creating synergies between these two fields.

Agent interaction, machine ethics. Among AI research areas possibly benefitting from the availability of a tool like ThinkEngine playing the role of a controlled environment for experimentation, one may consider using games for simulating the interaction between agents, e.g., in order to study their compliance with the ethical guidelines for a Trustworthy AI. In this context, a major obstacle to the operationalization and implementation of these requirements in AI systems is just the lack of datasets and benchmarks. Games might compensate for this lack, might help to get better insights into the dynamics of a corresponding real-world system, and can assess the practical challenges of building such a system. ASP, and more generally non-monotonic reasoning, is particularly suitable for dealing with ethical principles as testified by several proposals [58].

Stream reasoning. Besides the technical and scientific interest of connecting videogames to research, it is worth mentioning the very close connection with stream reasoning. The potential impact on applications like smart cities, power grid management, urban traffic control, where fast decision-making on big and dynamic data flows has immediate implications on the green economy, is fairly straightforward. All these applications fall under the stream reasoning umbrella where declarative techniques find a natural application. Stream reasoning also shares with videogames the need of performing AI reasoning under fast-paced input event flows.

Intrinsic value of videogames. The videogame market, the highest selling category in the entertainment industry [59], has seen a high increase in demand in the COVID-19 era, with new job openings and an even larger portfolio of games. Videogames, no matter if played with family, with friends, online in massive groups, or alone, were the place in which many of us found a form of sociality and consolation in the difficult times of COVID-19. One should recall that videogames are not just the place many scientists dream to leverage their ideas on, but also a way of living engaging stories, worlds and atmospheres. The realization of these latter necessitates innovative AI tools.

6. Acknowledgments

This article has been partially supported by the Italian MIUR Ministry and the Presidency of the Council of Ministers under the project "Declarative Reasoning over Streams" under the

"PRIN" 2017 call (Project 2017M9C25L_001), and includes work from COST Action 17124 "Digital forensics: evidence analysis via intelligent systems and practices (DigForASP)", supported by COST (European Cooperation in Science and Technology).

References

[1] J. Orkin, Three states and a plan: the AI of F.E.A.R., in: Game Developers Conference, 2006.
[2] N. Nilsson, STRIPS planning systems, Artificial Intelligence: A New Synthesis (1998) 373–400.
[3] Halo, 2001. URL: https://www.xbox.com/en-US/games/halo.
[4] Black & White, 2001. URL: https://www.ea.com/games/black-and-white.
[5] M. R. Genesereth, N. Love, B. Pell, General game playing: Overview of the AAAI competition, AI Mag. 26 (2005) 62–72.
[6] T. Schaul, A video game description language for model-based or interactive learning, in: CIG, IEEE, 2013, pp. 1–8.
[7] A. M. Smith, M. J. Nelson, M. Mateas, LUDOCORE: A logical game engine for modeling videogames, in: CIG, IEEE, 2010, pp. 91–98.
[8] D. P. Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, General video game AI: competition, challenges and opportunities, in: AAAI, 2016, pp. 4335–4337.
[9] O. Bartheye, E. Jacopin, A real-time PDDL-based planning component for video games, in: AIIDE, The AAAI Press, 2009.
[10] J. Robertson, R. M. Young, The general mediation engine, Experimental AI in Games: Papers from the 2014 AIIDE Workshop. AAAI Technical Report WS-14-16 10 (2014) 65–66.
[11] J. Robertson, R. M. Young, Automated gameplay generation from declarative world representations, in: AIIDE, AAAI Press, 2015, pp. 72–78.
[12] E. Erdem, M. Gelfond, N. Leone, Applications of answer set programming, AI Mag. 37 (2016) 53–68.
[13] A. M. Smith, M. Mateas, Answer set programming for procedural content generation: A design space approach, IEEE Trans. Comput. Intell. AI Games 3 (2011) 187–200.
[14] L. van Aanholt, R. Bidarra, Declarative procedural generation of architecture with semantic architectural profiles, in: CoG, IEEE, 2020, pp. 351–358.
[15] M. Thielscher, Answer set programming for single-player games in general game playing, in: ICLP, volume 5649 of *LNCS*, Springer, 2009, pp. 327–341.
[16] F. Calimeri, M. Fink, S. Germano, A. Humenberger, G. Ianni, C. Redl, D. Stepanova, A. Tucci, A. Wimmer, Angry-hex: An artificial player for angry birds based on declarative knowledge bases, IEEE Trans. Comput. Intell. AI Games 8 (2016) 128–139.
[17] J. Renz, X. Ge, S. Gould, P. Zhang, The angry birds AI competition, AI Mag. 36 (2015) 85–87.
[18] M. Stanescu, M. Certický, Predicting opponent's production in real-time strategy games with answer set programming, IEEE Trans. Comput. Intell. AI Games 8 (2016) 89–94.
[19] F. Calimeri, D. Fuscà, S. Germano, S. Perri, J. Zangari, Fostering the use of declarative

formalisms for real-world applications: The EmbASP framework, New Gener. Comput. 37 (2019) 29–65.
[20] O. Febbraro, N. Leone, G. Grasso, F. Ricca, JASP: A framework for integrating answer set programming with Java, in: KR, AAAI Press, 2012.
[21] M. Fink, S. Germano, G. Ianni, C. Redl, P. Schüller, Acthex: Implementing HEX programs with action atoms, in: LPNMR, volume 8148, Springer, 2013, pp. 317–322.
[22] J. Rath, C. Redl, Integrating answer set programming with object-oriented languages, in: PADL, 2017, pp. 50–67.
[23] D. Angilica, G. Ianni, F. Pacenza, Declarative AI design in unity using answer set programming, in: CoG, IEEE, 2022, pp. 417–424.
[24] Unity 3D game engine, Last accessed: Jan 2023. URL: https://unity3d.com/unity.
[25] M. Gebser, M. Maratea, F. Ricca, The seventh answer set programming competition: Design and results, Theory Pract. Log. Program. 20 (2020) 176–204.
[26] H. Beck, T. Eiter, C. Folie, Ticker: A system for incremental ASP-based stream reasoning, Theory Pract. Log. Program. 17 (2017) 744–763.
[27] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, Theory Pract. Log. Program. 19 (2019) 27–82.
[28] G. Ianni, F. Pacenza, J. Zangari, Incremental maintenance of overgrounded logic programs with tailored simplifications, Theory Pract. Log. Program. 20 (2020) 719–734.
[29] A. A. Falkner, G. Friedrich, K. Schekotihin, R. Taupe, E. C. Teppan, Industrial applications of answer set programming, Künstliche Intell. (2018) 165–176.
[30] F. Calimeri, G. Ianni, F. Pacenza, S. Perri, J. Zangari, Incremental answer set programming with overgrounding, Theory Pract. Log. Program. 19 (2019) 957–973.
[31] F. Calimeri, S. Germano, G. Ianni, F. Pacenza, S. Perri, J. Zangari, Integrating rule-based AI tools into mainstream game development, in: RuleML+RR, volume 11092, Springer, 2018, pp. 310–317.
[32] M. Joselli, et al., An adaptative game loop architecture with automatic distribution of tasks between CPU and GPU, Comput. Entertain. 7 (2009) 50:1–50:15.
[33] R. R. Murphy, Introduction to AI Robotics, MIT Press, 2000.
[34] I. Millington, Artificial Intelligence for Games, Third Edition, CRC Press, 2019.
[35] D. Kahneman, Thinking, fast and slow, Farrar, Straus and Giroux. New York, 2011.
[36] G. Booch, F. Fabiano, L. Horesh, K. Kate, J. Lenchner, N. Linck, A. Loreggia, K. Murugesan, N. Mattei, F. Rossi, B. Srivastava, Thinking fast and slow in AI, in: AAAI, AAAI Press, 2021, pp. 15042–15046.
[37] Z. G. Saribatur, T. Eiter, Omission-based abstraction for answer set programs, in: KR, AAAI Press, 2018, pp. 42–51.
[38] J. F. Allen, G. Ferguson, Actions and events in interval temporal logic, J. Log. Comput. 4 (1994) 531–579.
[39] F. Calimeri, G. Ianni, F. Pacenza, S. Perri, J. Zangari, ASP-based multi-shot reasoning via DLV2 with incremental grounding, in: PPDP, ACM, 2022, pp. 2:1–2:9.
[40] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, J. Togelius, Procedural content generation via machine learning (PCGML), IEEE Trans. Games 10 (2018) 257–270.
[41] E. Giacomello, P. L. Lanzi, D. Loiacono, DOOM level generation using generative adversarial

networks, in: GEM, IEEE, 2018, pp. 316–323.
[42] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. M. Smith, S. Risi, Evolving Mario levels in the latent space of a deep convolutional generative adversarial network, in: GECCO, ACM, 2018, pp. 221–228.
[43] R. R. Torrado, A. Khalifa, M. C. Green, N. Justesen, S. Risi, J. Togelius, Bootstrapping conditional GANs for video game level generation, in: CoG, IEEE, 2020, pp. 41–48.
[44] H. Zhang, M. C. Fontaine, A. K. Hoover, J. Togelius, B. Dilkina, S. Nikolaidis, Video game level repair via mixed integer linear programming, in: L. Lelis, D. Thue (Eds.), AIIDE, AAAI Press, 2020, pp. 151–158.
[45] C. C. Aggarwal, J. Han (Eds.), Frequent Pattern Mining, Springer, 2014.
[46] G. Dong, J. Bailey (Eds.), Contrast Data Mining: Concepts, Algorithms, and Applications, CRC Press, 2013.
[47] S. Nijssen, A. Zimmermann, Constraint-based pattern mining, in: C. C. Aggarwal, J. Han (Eds.), Frequent Pattern Mining, Springer, 2014, pp. 147–163.
[48] F. A. Lisi, G. Sterlicchio, A declarative approach to contrast pattern mining, in: AIxIA. To appear., 2023.
[49] F. A. Lisi, D. Malerba, Inducing multi-level association rules from multiple relations, Mach. Learn. 55 (2004) 175–210.
[50] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, Artif. Intell. 172 (2008) 1495–1539.
[51] J. McCarthy, Elaboration tolerance, https://stanford.io/3UIlwsM, 1998.
[52] A. Brännström, J. C. Nieves, A framework for developing interactive intelligent systems in unity, in: 10th EMAS workshop at AAMAS, 2022.
[53] A. Segato, M. D. Marzo, S. Zucchelli, S. Galvan, R. Secoli, E. D. Momi, Inverse reinforcement learning intra-operative path planning for steerable needle, IEEE Trans. Biomed. Eng. 69 (2022) 1995–2005.
[54] Y. Izmirlioglu, E. Erdem, Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming, Theory Pract. Log. Program. 20 (2020) 942–957.
[55] F. A. Lisi, Combining knowledge representation and machine learning in forensics, Applications of AI to Forensics 2020 (AI2Forensics 2020) (2020) 1.
[56] F. A. Lisi, G. Sterlicchio, Declarative pattern mining in digital forensics: Preliminary results, in: CILC, volume 3204, CEUR-WS.org, 2022, pp. 232–246.
[57] F. A. Lisi, G. Sterlicchio, Mining sequences in phone recordings with answer set programming, in: P. Bruno, F. Calimeri, F. Cauteruccio, M. Maratea, G. Terracina, M. Vallati (Eds.), Joint Proceedings of (HYDRA 2022) and (RCRA 2022) at (LPNMR 2022), volume 3281 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 34–50. URL: http://ceur-ws.org/Vol-3281/paper4.pdf.
[58] A. Dyoub, S. Costantini, F. A. Lisi, Logic programming and machine ethics, in: ICLP, volume 325, 2020, pp. 6–17.
[59] Newzoo global games market report 2022, https://newzoo.com/insights/trend-reports/newzoo-global-games-market-report-2022-free-version, Last accessed: Jan 2023.