Towards General Game-Playing Robots: Models, Architecture and Game Controller

David Rajaratnam and Michael Thielscher

The University of New South Wales, Sydney, NSW 2052, Australia
{daver,mit}@cse.unsw.edu.au

Abstract. General Game Playing aims at AI systems that can understand the rules of new games and learn to play them effectively without human intervention. Our paper takes the first step towards *general game-playing robots*, which extend this capability to AI systems that play games in the real world. We develop a formal model for general games in physical environments and provide a systems architecture that allows the embedding of existing general game players as the "brain" and suitable robotic systems as the "body" of a general game-playing robot. We also report on an initial robot prototype that can understand the rules of arbitrary games and learns to play them in a fixed physical game environment.

1 Introduction

General game playing is the attempt to create a new generation of AI systems that can understand the rules of new games and then learn to play these games without human intervention [5]. Unlike specialised systems such as the chess program Deep Blue, a general game player cannot rely on algorithms that have been designed in advance for specific games. Rather, it requires a form of general intelligence that enables the player to autonomously adapt to new and possibly radically different problems. General game-playing systems therefore are a quintessential example of a new generation of software that end users can customise for their own specific tasks and special needs.

This paper makes the first step towards *general game-playing robots*, which we define to be autonomous systems that can understand descriptions of new games and learn to play them in a physical game environment. Ultimately, a truly general game-playing robot must be able not only to accommodate new game rules but also adapt to unknown game environments. However, as a first step we aim at robots that learn to play new games within a fixed physical world. These robots should accept arbitrary game descriptions for such environments and then play these games effectively by manipulating the actual objects, e.g. pieces, of the real game.

In seeking to build a novel system such as a general game-playing robot, it is necessary to provide some justification for why such a system is of interest. We briefly highlight three disparate reasons. Firstly, interaction with a game-playing robot requires both mental and physical activity from a human opponent and could therefore be of interest in the area of rehabilitation and elderly care [3]. Secondly, games have a wide variety of physical requirements, and consequently offer a vast array of well-defined and controlled environments in which to benchmark techniques for robot recognition

S. Cranefield and A. Nayak (Eds.): AI 2013, LNAI 8272, pp. 271–276, 2013.

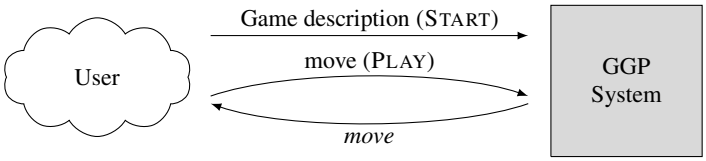


Fig. 1. Functionality of a general game player

and manipulation. Finally, this research has applications beyond games. Many tasks have game-like properties. For example, a domestic robot fetching an item has to adapt to changes in the environment; the item may not be where the robot expected, or the operator may change locations. Viewing such a task as a game can provide a natural framework for controlling the robot's behaviour.

Our specific contribution to general game-playing robots in this paper is three-fold. First, we develop a formal model to identify essential requirements for game descriptions when translating gameplay moves from a purely virtual to a physical environment (Section 3). Second, we provide a systems architecture that integrates research in General Game Playing with research in Robotics (Section 4). This architecture is generic in that it allows the embedding of both existing general game players and suitable robotic systems. Third, to evaluate the feasibility of our framework, we have built a first general game-playing robot and tested it on different games of variable difficulty played on a physical chess-like board with moving pieces (Section 5).

2 Background

General Game Playing. The annual AAAI GGP Competition [5] defines a general game player as a system that can understand the rules of any n-player game given in the general Game Description Language (GDL) and is able to play those games effectively. The functionality is illustrated in Fig. 1: A player must first accept any communicated game description (START message). After a given time period, and without human intervention, the player then makes his opening move, accepts legal moves by other players (PLAY message) and continues to play until the game terminates.

Since the first AAAI competition in 2005, General Game Playing has evolved into a thriving AI research area. Established methods include Monte Carlo tree search [2], the automatic generation of evaluation functions [4] and knowledge acquisition [7]. Several complete general game-playing systems also have been described [12,2].

Robotics. Building a robot that can recognise and manipulate objects in a physical environment is a complex and challenging problem. While basic robot kinematics is well understood, the ability to recognise and manipulate arbitrary objects in a complex environment remains an active area of research [9]. However, dealing with a predetermined set of rigid objects, such as common household items or game boards and pieces, is more amenable to known techniques and technologies [10,6]. Consequently, the research in domestic robotics provides an excellent platform for the development of general game playing robots that can play a wide variety of games.

3 GDL Descriptions for Physical Game Environments

Game descriptions must satisfy certain basic requirements to ensure that the game is effectively playable; for example, players should always have at least one legal move in nonterminal positions [5]. In bringing gameplay from mere virtual into physical environments, general game-playing robots add a new class of desirable properties that concern the manifestation of the game rules in the real world. Notably, a good GDL description requires all moves deemed legal by the rules to be actually physically possible. In this section we develop a framework for mathematically describing the application of game descriptions to physical environments that allows us to formalise such properties.

Environment Models. Consider the robotic game environment shown in Fig. 2(a). It features a 4×4 chess-like board with an additional row of 4 marked positions on the right. Tin cans are the only type of objects and can be moved between the marked positions (but cannot be stacked). To formally analyse the use of physical environments like this to play a game, we model them by *state machines* (Σ, S, s_0, δ), where

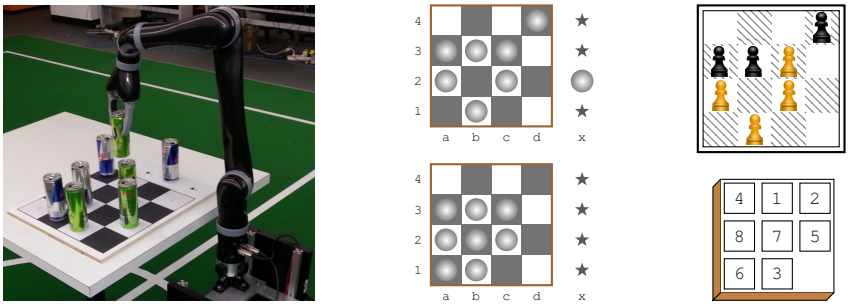
- Σ denotes the *actions* that can be performed in the environment;
- S are the *states* the environment can be in, including the special *failure* $\in S$;
- $s_0 \in S$ is the *starting state*;
- $\delta : S \times \Sigma \to S$ is the *transition function*.

Example. The following is a model for our cans-on-a-chessboard environment.

- We want to allow players two actions: doing nothing and moving a can. Formally, $\Sigma = \{\texttt{noop}\} \cup \{\texttt{move}(u,v,x,y) \colon u,x \in \{\texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{x}\};\ v,y \in \{1,2,3,4\}\}$.
- Any location can either be empty or house a can. Hence, the environment can be in any of 2^{20} different states plus *failure*. Fig. 2(b) illustrates two example states.
- Any configurations of cans can be set up as the starting state s_0.
- For the transition function δ, action `noop` has no effect, hence $\delta(s, \texttt{noop}) = s$. Action $\texttt{move}(u,v,x,y)$ maps any state with a can at (u,v) and no can at (x,y) to the same state except that now there is a can at (x,y) and none at (u,v). If no can is at (u,v) or there already is one at (x,y), then $\delta(s, \texttt{move}(u,v,x,y)) = \textit{failure}$. E.g., $s_2 = \delta(\texttt{move}(\texttt{d},4,\texttt{b},2), \delta(\texttt{move}(\texttt{x},2,\texttt{a},1), s_1))$ where s_1 and s_2 respectively denote the top and bottom states depicted in Fig. 2(b).

Projecting Games onto Physical Environments. When we use a physical environment to play a game, the real objects become representatives of entities in the abstract game. A pawn in chess, for example, is typically manifested by an actual wooden piece of a certain shape and colour. But any other physical object, including a tin can, can serve the same purpose. Conversely, any game environment like our 4×4 board with cans can be interpreted in countless ways as physical manifestation of a game. Thus our example board can not only be used for all kinds of mini chess-like games (cf. top of Fig. 2(c)) but also to play, say, the single-player 8-PUZZLE (cf. bottom of Fig. 2(c)). For this the cans represent numbered tiles that need to be brought in the right order.

The manifestation of a game in a physical environment can be mathematically captured by *projecting* the positions from the abstract game onto actual states of the game



(a) (b) (c)

Fig. 2. Game environment with robot, and games projected onto this environment

environment. Formally, if $\texttt{Positions}$ is the set of all possible positions in the game, then a *projection function* is of the form $\pi : \texttt{Positions} \to S \setminus \{\mathit{failure}\}$, where S is the set of states in the environment model as above.

Example. Fig. 2(c) shows two positions from two different games, a mini-variant of Breakthrough and the 8-puzzle. We can view the states depicted to the left of each position (Fig. 2(b)) as their projection onto our example physical game environment, in which the extra row can be used to park captured pieces. For the sake of simplicity, the robot in our environment is not required to distinguish between different types of cans. Hence, it is only through the projection function that the robotic player knows whether a can stands for a white or a black pawn, say. The reader should note that a similar feature is found in many games humans play; for example, the pieces on a chessboard alone not telling us whose move it is or which side still has castling rights.

Requirements for Game Descriptions for Physical Environments. The standard semantics for GDL [13] defines precisely how to interpret a given set of rules as a game. In particular, every valid game description determines the following.

- The set $\texttt{Positions}$ of all possible positions in the game.
- The initial position, $\texttt{Init} \in \texttt{Positions}$.
- The set $\texttt{LegalMoves}(p)$ of legal moves in each position p.
- The function $\texttt{Update}(p, m)$ to compute the new position after move m in p.

All of these elements of a general, abstract game can be related to gameplay in a physical game environment via the projection function that maps game positions onto physical states. Specifically, we can formalise fundamental requirements concerning the *playability* of an arbitrary game in a given environment (Σ, S, s_0, δ) as follows.

1. All legal moves must correspond to actions in the physical environment. Formally, $\forall p \in \texttt{Positions}. \forall m \in \texttt{LegalMoves}(p). m \in \Sigma$.
2. Initial position and initial physical state correspond. Formally, $\pi[\texttt{Init}] = s_0$.
3. All legal position updates correspond to a possible physical state update. Formally, $\forall p \in \texttt{Positions}. \forall m \in \texttt{LegalMoves}(p). \pi[\texttt{Update}(p, m)] = \delta(\pi[p], m)$.

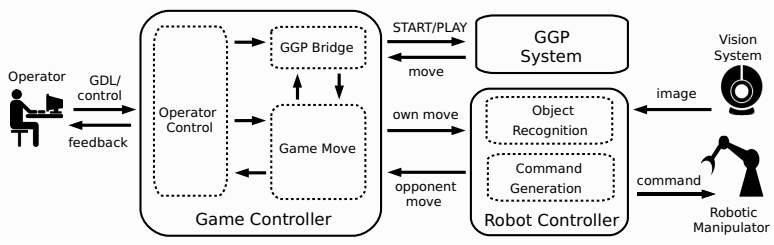


Fig. 3. Systems architecture

4 Systems Architecture

In the following, we describe a general systems architecture for general game-playing robots (Fig. 3). Its defining feature is to allow for the seamless integration of any existing general game player as the "brain" and any suitable robotic system as the "body."

The **Robot Controller** serves as the low-level executor of the robot. The required functionality is, (1) to process sensor data in order to recognise moves by detecting changes in the environment, and (2) to command the manipulator to execute any instance of the defined actions in the environment. Performing a single action may require a complex series of operations; for example, the sequence of operations required to move a can from one location to another. The robot controller needs to monitor the execution of such an action and, if possible, recover from failures whenever they occur.

The **Game Controller** provides the link between the low-level executor and the high-level decision making. It accepts any new GDL description sent through the interface and transmits these game rules to the general game-playing system in the form of a standard START message. During gameplay, the game controller passes on any move decided upon by the high-level controller to the low-level controller, and it converts opponent moves reported by the robot controller into standard PLAY messages for the general game player. The latter involves validating the legality of all opponent moves.

The systems architecture allows for plugging in as the high-level control module any **General Game-Playing System** that follows the standard specification [5].

5 Prototype

We constructed a prototype system using a Kinova Jaco arm, pictured in Fig. 2(a), and the ROS robotic software [11]. The highly successful CadiaPlayer [2] was used as the underlying GGP reasoner. While the basic system architecture has been implemented as described in Sections 4, the prototype simulates the vision module through operator input. For this reason the robot relies on the pieces being in the correct game position.

We ran tests with three games, each repeated 10 times. Each round of the short single-player 4-QUEENS game lasted approximately 2.5 minutes. At no point was operator intervention required. 8-PUZZLE, a single-player game with a cluttered physical environment (cf. Fig. 2(b), bottom), lasted approximately 16-17 minutes with 30 moves. Two out of 10 games required operator intervention to pick up a can that had fallen

after being bumped by the robot gripper. Being more interactive, the 2-player MINI-BREAKTHROUGH games (cf. Fig. 2(b)-(c), top) varied in length from 5 joint moves in 4 minutes to 14 joint moves taking 10 minutes. Two games required operator intervention to correct fallen cans caused by interference to the gripper by adjacent cans.

6 Conclusion and Future Work

In this paper we take a first step towards building general game-playing robots; robots that can understand and play arbitrary games in a physical environment. The system was realised by drawing together two topical yet disparate areas of artificial intelligence research: general game playing and robotics. A prototype system was built and experiments performed to validate the general architecture and feasibility of the project.

Beyond the necessary task of implementing the object recognition system, there are a number of directions for future work. Firstly, given some GDL game it should be possible to automatically construct the rules of a meta-game that incorporates recovery from known error states. The recovery behaviour would then simply be a result of the robot playing the larger meta-game. A second broad area for future work is that of learning. Instead of having to describe new games in GDL, it would be beneficial if the robot were able to learn how to play a game by being shown how by a user, as in [8,1].

References

1. Barbu, A., Narayanaswamy, S., Siskind, J.: Learning physically-instantiated game play through visual observation. In: Proc. of ICRA, pp. 1879–1886. IEEE Press (2010)
2. Björnsson, Y., Finnsson, H.: CADIAPLAYER: A simulation-based general game player. IEEE Transactions on Computational Intelligence and AI in Games 1(1), 4–15 (2009)
3. Broekens, J., Heerink, M., Rosendal, H.: Assistive social robots in elderly care: a review. Gerontechnology 8(2) (2009)
4. Clune, J.: Heuristic evaluation functions for general game playing. In: Proc. of AAAI, pp. 1134–1139 (2007)
5. Genesereth, M., Love, N., Pell, B.: General game playing: Overview of the AAAI competition. AI Magazine 26(2), 62–72 (2005)
6. Goldfeder, C., Ciocarlie, M.T., Dang, H., Allen, P.K.: The columbia grasp database. In: Proc. of ICRA, pp. 1710–1716. IEEE Press (2009)
7. Haufe, S., Schiffel, S., Thielscher, M.: Automated verification of state sequence invariants in general game playing. Artificial Intelligence 187-188, 1–30 (2012)
8. Kaiser, Ł.: Learning games from videos guided by descriptive complexity. In: Proc. of AAAI, pp. 963–969 (2012)
9. Kemp, C.C., Edsinger, A., Torres-Jara, E.: Challenges for robot manipulation in human environments. IEEE Robotics & Automation Magazine 14(1), 20–29 (2007)
10. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: Proc. of ICRA, pp. 1817–1824. IEEE Press (2011)
11. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
12. Schiffel, S., Thielscher, M.: Fluxplayer: A successful general game player. In: Proc. of AAAI, pp. 1191–1196 (2007)
13. Schiffel, S., Thielscher, M.: A Multiagent Semantics for the Game Description Language. In: Filipe, J., Fred, A., Sharp, B. (eds.) ICAART 2009. CCIS, vol. 67, pp. 44–55. Springer, Heidelberg (2010)