Investigating the Application of Action Model Learning for Player Modeling

Abhijeet Krishnan
Principles of Expressive Machines (POEM) Lab
North Carolina State University
Raleigh, NC 27606
akrish13@ncsu.edu

Abstract

Player modeling attempts to create a computational model which accurately approximates a player's behavior in a game. Most player modeling techniques rely on domain knowledge and are not transferable across games. Additionally, player models do not currently yield any explanatory insight about a player's cognitive processes, such as the creation and refinement of mental models. In this paper, we present our findings with using *action model learning* (AML), in which an action model is learned given data in the form of a play trace, to learn a player model in a domain-agnostic manner. We demonstrate the utility of this model by introducing a technique to quantitatively estimate how well a player understands the mechanics of a game, and present a hypothetical evaluation method to assess a model's accuracy. We evaluate existing AML algorithms in terms of their suitability for player modeling. We find that current AML algorithms are performant but are unable to use prior models or failed actions to learn.

Introduction

Player modeling is the study of computational models of players in games (Yannakakis et al. 2013). It sees varied and widespread usage in today's video game industry. Imangi Studios, developers of the popular *Temple Run* (Imangi Studios 2011) series of mobile games, collect player telemetry in order to analyse player behavior and provide customized gameplay experiences (Studios 2019). *Forza Motorsport 5* (Turn 10 Studios 2013) implements a *Drivatar* (Reynolds 2019) system which learns to mimic the player's behavior in the game and simulate the player in other races.

Despite the multitude of techniques used to build player models, such as self-organizing maps (Drachen, Canossa, and Yannakakis 2009), Bayesian networks (Yannakakis and Maragoudakis 2005), and multi-layer perceptrons (Pedersen, Togelius, and Yannakakis 2010), many of them rely on features extracted from domain knowledge of the game's rules and as such cannot be generalized easily, except perhaps to games of the same genre. The inability to easily train new models for different games using the same technique presents a barrier to any single technique's adoption.

Submitted to the Department of Computer Science at North Carolina State University in partial fulfillment for the Doctoral Written Preliminary Qualifier Exam

Furthermore, while current techniques aim to *predict* player actions, we argue that there is a corresponding need to *explain* their underlying cognitive processes. There is empirical evidence in cognitive science that game players build *mental models*, knowledge structures capable of simulating the system they are interacting with and predicting and explaining the outcomes of scenarios while playing games (Boyan, McGloin, and Wasserman 2018). Mental models may start out erroneous, but improve over time as the player learns from which of their attempted actions succeed and fail. We believe that mental model alignment is one of the most important aspects of using games for impactful applications such as education and training, and that player modeling techniques should be designed to yield insight into this cognitive process.

To address these needs, we propose *action model learning* (AML), in which an action model is learned from play traces, as a viable technique. Action models can be used to learn player models in any game which can be represented in a planning formalism like PDDL. There exists a rich body of literature on learning action models from action traces which we can leverage to learn action models.

In this paper, we describe the results of our investigation into the application of action model learning to player modeling. More generally, we argue that AML is a *tractable* and *domain-agnostic* approach to player modeling, and that the learned action model is a *useful* player model. We justify the tractability claim by successfully applying FAMA (Aineto, Celorrio, and Onaindia 2019) to the task of learning a player model from action traces, measuring their efficiency on problems of various sizes as well as precision and recall. We justify the domain-agnosticity claim by successfully learning action models for two additional domains: Hanoi (the Tower of Hanoi puzzle) and N-puzzle (a sliding tile puzzle), using publicly available domain files[1] and trajectories from FAMA's evaluation dataset. We justify the usefulness claim by presenting a technique to quantify a player's mechanical mastery of a game given their action model-based player model. We evaluate a number of other AML algorithms for the purpose of player modeling. We discuss the techniques' advantages and limitations and suggest

[1]http://planning.domains/

avenues for future work.

Related Work

Player modeling is a relatively new field, previously studied under the umbrella of HCI in the form of user modeling (Biswas and Springett 2018) and student modeling (Chrysafiadi and Virvou 2013). Surveys on player modeling provide useful taxonomies (Smith et al. 2011), an overview of techniques that have been used for player modeling generally (Machado, Fantini, and Chaimowicz 2011) or for MMORPGs (Harrison and Roberts 2011), and challenges commonly encountered in the field, including the reliance on knowledge engineering (Hooshyar, Yousefi, and Lim 2018) that motivates our work.

Domain-agnostic approaches to player modeling have been attempted before. Snodgrass, Mohaddesi, and Harteveld (2019) describe a general player model using the PEAS framework, which presents a theoretical framework for providing game recommendations, but does not account for in-game player behavior. Nogueira et al. (2014) use physiological readings of players to model their emotional response to game events. While domain-agnostic, this technique relies on physiological data from sensors, which is difficult to acquire. Our approach requires only gameplay traces, which can be easily done by making minor modifications to the game engine code. Deep neural networks have been used to simulate the actions of human players in interactive fiction games (Wang et al. 2018) and MMORPGs (Pfau, Smeddinck, and Malaka 2018) and to generate new levels in a platformer game based on learned player preferences (Summerville et al. 2016). These techniques rely only on easily obtainable input like gameplay logs or video recordings, and do not use any knowledge engineering to train the model. We believe our approach has two advantages: a) that it does not require as much data to learn, and, b) by generating a rule-based model, offers more explanatory power for player behavior.

To the best of our knowledge, this is the first-known application of AML to player modeling. The literature on AML has primarily focused on learning sound action models for use by an automated planner, with little attention paid to modeling the player's cognitive processes. *Human-aware planning* attempts to create planning agents which can take human mental models into account when planning (Chakraborti et al. 2017), while we treat a human as a planning agent and attempt to learn a domain model which mirrors their mental model as closely as possible. Serafini and Traverso (2019) introduce a perception function which maps sensor data to state variables in order to learn action models. We believe investigating player perception functions for cases like player disability to learn action models would be fruitful to pursue.

In this paper, we implicitly assume an action model as an appropriate rule-based model with which to represent a player's mental model. There are many other rule-based models and the field of inductive rule-learning is dedicated to extracting these rules from a set of observations. Angluin and Smith (1983) provides a theoretical background for inductive inference and mentions context-free grammars and regular expressions as examples of hypotheses that can be inferred. (Novak, Lavrač, and Webb 2009) surveys associative rule-mining methods which model rules as a decision tree or if-then rules. The field of inductive logic programming (ILP) models rules as first-order logic statements (Muggleton and De Raedt 1994). SimStudent (Matsuda, Cohen, and Koedinger 2015) is a system which uses ILP to learn rules for solving simple algebra problems by expert demonstration. We choose action models to represent a player's mental model since we believe it is the more appropriate choice to represent knowledge of game mechanics.

Background

Action Model Learning

Action model learning (AML) is situated within the tradition of automated planning, in which an *action model* describes the preconditions and effects of each action an agent may take in a world. Preconditions and effects are propositional formulas over fluents (predicates whose truth changes with time). A *plan* is a sequence of actions, each instantiated with terms for each parameter position, such that the effects of each action entail the preconditions of the action following it. AML, then, is the problem of discovering an action model from a set of observed plans.

Formally, an AML algorithm takes as input a number of *action traces*, each a sequence $\tau = \langle s_0, a_1, s_1, a_2, s_2, \cdots, a_n, s_n \rangle$, where s_i are states and a_i are actions, and returns as output an action model, which is a specification of the preconditions and effects of every action in the domain. These action traces are usually assumed to be fully observed, i.e. every fluent of a state is present, but various developments in AML algorithms allow action models to be learned from action traces with partially observed or noisy states as well. Theoretical work in AML tries to bound the number of trajectories required to learn a solution model (Walsh and Littman 2008; Stern and Juba 2017). An action model is represented as a STRIPS-style domain, with the most common representation format being PDDL. Evaluation of an action model is usually done by calculating the number of missing or extra predicates in the preconditions and effects of each action in the learned model.

Sokoban

The primary domain used in development and testing is the classic Japanese block-pushing puzzle game Sokoban, which we use to demonstrate the feasibility of our approach. We select it based on its relative minimalism and existing formalizations in PDDL as part of the International Planning Competition (IPC) benchmark suite (Coles et al. 2012). In Sokoban, there are stones (or crates) scattered around a warehouse floor, which the player must push to certain specified locations around the warehouse, but can only push the stones in cardinal directions, cannot climb over the stones or other obstacles, and can only push one stone at a time. For an overview of Sokoban's rules and its characteristics as a search problem, we refer readers to Junghanns and Schaeffer (1997). The specific representation of Sokoban we use is

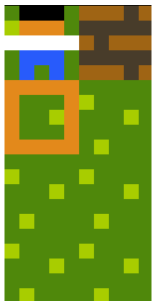

Figure 1: A simple Sokoban level (the goal is on the same cell as the player) rendered using PuzzleScript[4]

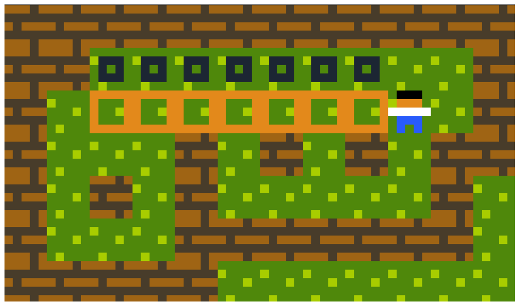

Figure 2: A more complex level, taken from IPC 2011

a domain that appeared in the 2011 International Planning Competition,[2] modified to remove action costs.

Figures 1 and 2 describe two Sokoban levels of differing complexity. The level in Figure 1 can be solved by moving the player down, right, down, down, left, up, and finally up to push the crate onto the goal.

Player Modeling with AML

Learning a model

We attempt to solve the unsupervised learning problem of inferring an action model from a set of input trajectories which most closely mirrors the player's mental model of a game's mechanics. We choose to use AML algorithms for this task.

An AML algorithm takes as input a set of action traces or trajectories T, and outputs an action model A, which is a specification of the preconditions and effects of each action in the domain. Algorithms may differ in the assumptions they make, with many algorithms assuming STRIPS-style domains but improvements accounting for other features of the PDDL specification such as quantifiers (Zhuo et al. 2010), conditional effects (Oates and Cohen 1996) or type hierarchies (Gregory and Cresswell 2015).

For player modeling, we assume that the input set of trajectories are generated by a single player's interaction with a game. The game must be representable as a PDDL domain. The domain representation is constrained by the AML algorithm being used, since not all AML algorithms can support the full set of PDDL features available.

The player is presented with an abstract interface to carry out all of the possible actions in the game, irrespective of whether they are applicable or not. If a chosen action can be successfully performed in the current state, it is applied and the new state is generated, adding a tuple $\langle s_{pre} - a - s_{post} \rangle$ to the trajectory. If the player selects an action which cannot be carried out (the action's preconditions are not met), then no change is made to the current state and the action selected is recorded as a *failed action* in the trajectory. Formally, these are recorded as a tuple $\langle s - a_f - s \rangle$ in the trajectory. Certain AML algorithms (Amir and Chang 2008; Wang 1995) make use of failed actions to improve their action model. Those which do not can ignore the failed actions and use the rest of the trajectory to discover an action model. Trajectories can be discretized in any meaningful way, such as per level, or per game session.

The AML algorithm is given the set of trajectories as input and learns an action model. The action model we learn over the input trajectories represents the player's model of the mechanics of the game given their interaction with it. This assumes that every action in the domain of a game defined in a planning formalism corresponds to a mechanic.

Using the model

Player models in literature have been used to perform a variety of functions. Karpinskyj, Zambetta, and Cavedon (2014) describe five ways in which players differ from each other, namely by preferences, personality, experience, performance and in-game behavior. These suggest avenues for what a player model might be used to predict. Harrison and Roberts (2011) explore how various player modeling techniques might be applied to MMORPGs, using the specific examples of interactive tutorials, targeted skill improvement, quest offering and dynamic quests.

We present a technique to derive a quantitative estimate of the player's mastery of the game's mechanics by comparing the learned player model with the ground truth model. We assume a ground truth domain model of the game is available in PDDL. We believe this is a reasonable assumption since our task is to learn the *player's* representation of the game mechanics, and not the actual game mechanics themselves. Our technique is based on a proposed evaluation technique for action models by Aineto, Celorrio, and Onaindia (2019), which uses precision and recall as metrics. We compare the learned action model to the ground truth model and *for each action* count the number of predicates in the learned model's preconditions and effects which are correct (true positives), extra (false positives) and missing (false negatives). We use this confusion matrix to compute the F_1-score for every action in the model and report it as the player's proficiency score for a particular mechanic given the model. We use the F_1-score since it is a meaningful way to combine precision and recall into a single number.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{1}$$

[2]https://github.com/potassco/pddl-instances/blob/master/ipc-2011/domains/sokoban-sequential-satisficing/domain.pddl

[4]https://www.puzzlescript.net/index.html

The learned action model can also be treated as a game itself, with its rules based on the modeled player's knowledge of the actual game's rules. This learned game represents a way to meaningfully interact with a player's mental model and we believe has artistic merit.

Evaluating the model

Evaluation of a learned player model is typically done by measuring how well the model's predictions match ground truth data. We have been unable to find a documented method to directly elicit an action model from a human player. Instead, we present a hypothetical method to evaluate action models based on comparison between predictions regarding game state changes made by human players and the learned action model. The human player is presented with a quiz post their play session with questions consisting of two types - 1) to assess their knowledge of action preconditions, and 2) to assess their knowledge of action effects. Questions of type 1 present the quiz taker with a game state and an action, and ask them to predict whether the action can be successfully performed in the shown state. Questions of type 2 present the quiz taker with a game state, an action possible in that state, and a post-state, and asks them to predict whether the action would transform the pre-state into the shown post-state. The game states are snapshots of the game as presented to the player. The learned action model is used to generate predictions for each question in the quiz, and its responses are compared with the player responses to measure the accuracy of the model.

Evaluation

To determine baseline feasibility of action model learning as an approach to player modeling, we start with an out-of-the-box algorithm known as FAMA (Aineto, Celorrio, and Onaindia 2019).

We use traces generated manually from two hand-crafted levels (L_1 and L_2) and an instance from the IPC 2011 collection (L_3). The levels differ in complexity, with L_1 having 7 actions in the optimal solution and L_3 having 129 actions. We use FAMA with full state observability to successfully learn two action models M_1 and M_2 from the first two traces. Figure 3 shows the `move` action learned by both models. Table 1 shows the player proficiency scores predicted by our model for each mechanic.

Mechanic name	F_1-score	
	M_1	M_2
`move`	0.267	0.250
`push-to-nongoal`	0.308	0.216
`push-to-goal`	0.222	0.240

Table 1: Proficiency scores for each mechanic

To determine whether AML is a domain-agnostic player modeling technique, we attempt to use it to learn player models in two different domains, namely Hanoi and N-puzzle. We source trajectories for both domains from FAMA's evaluation dataset. We successfully learn two action models for each domain (D_h for Hanoi and D_n for N-puzzle), from which we report the `move` action in Figure 4.

```
(:action move
  :parameters
    (?p - player
     ?from -
    location
     ?to - location
     ?dir -
    direction)
  :precondition (
    and
    (clear ?to)
    (at ?p ?from)
    (is-nongoal ?
    from))
  :effect (and
    (clear ?from)
    (at ?p ?to)
    (move-dir ?from
     ?to ?dir)
    (not (at ?p ?
    from))
    (not (clear ?to
    ))))
```

```
(:action move
  :parameters
    (?p - player
     ?from -
    location
     ?to - location
     ?dir -
    direction)
  :precondition (
    and
    (clear ?to)
    (at ?p ?from)
    (move-dir ?from
     ?to ?dir))
  :effect (and
    (clear ?from)
    (at ?p ?to)
    (is-nongoal ?to
    )
    (is-nongoal ?
    from)
    (not (at ?p ?
    from))
    (not (clear ?to
    ))))
```

Figure 3: A comparison between the `move` action learned by M_1 and M_2

Comparison of AML Algorithms

We compare five AML algorithms along metrics we believe are relevant to player modeling, namely scalability (as measured by computation time and memory usage), ability to use previously learned action models, ability to use failed actions and accuracy. Scalability is important because we anticipate player trajectories to contain hundreds of states and actions and having an algorithm which can learn in near real-time is valuable for providing real-time skill assessments using the model. The ability to use previously learned action models is valuable because it allows us to take into account trajectories from previous play sessions when learning new models, thus making the models a closer representation of the player's mental model. We denote this by $f_{AML}(\tau, \alpha)$, which indicates the ability of the AML algorithm f to use as input α, which is a prior action model. The ability to use failed actions is valuable since we hypothesize that it shares similarities with the cognitive process by which players form models of games and thus would help learn models which are more accurate representations of players' mental models. We denote this by $\langle s - a_f - s \rangle \in \tau$ which indicates that failed action tuples are part of the input trajectory for the AML algorithm. Lastly, accuracy is valuable because we want our model to be representative of the player's mental model for the usefulness of its skill predictions.

The five algorithms we use are ARMS (Wu, Yang, and Jiang 2005), LOCM (Cresswell, McCluskey, and West

```
(:action move
  :parameters (?o1
   - object ?o2 -
   object ?o3 -
   object)
  :precondition (
   and (clear ?o3)
   (clear ?o1) (on
   ?o1 ?o2))
  :effect (and
   (not (clear ?o3
   ))
   (not (on ?o1 ?
   o2))
   (clear ?o2)
   (on ?o1 ?o3)
   (smaller ?o1 ?
   o1)
   (smaller ?o2 ?
   o1))
```

Listing 1: M_h

```
(:action move
  :parameters (?o1
   - tile ?o2 -
   position ?o3 -
   position)
  :precondition (
   and (at ?o1 ?o2)
    (empty ?o3) (
   neighbor ?o3 ?o2
   ) (neighbor ?o2
   ?o3))
  :effect (and
   (not (at ?o1 ?
   o2))
   (at ?o1 ?o3)
   (not (empty ?o3
   ))
   (empty ?o2))
```

Listing 2: M_n

Figure 4: A comparison between the `move` action learned by D_h and D_n

2013), LOCM2 (Gregory and Cresswell 2015), LOUGA (Kučera and Barták 2018) and FAMA (Aineto, Celorrio, and Onaindia 2019). The implementation for ARMS is sourced from the authors' website [5]. The source code is for the HTNML (Zhuo et al. 2010; 2009) algorithm which behaves as ARMS when not using hierarchical tasks. The implementations for LOCM, LOCM2 and LOUGA are sourced from the LOUGA implementation and comparison tool for Windows provided by the authors. We run LOUGA for 100 generations. The implementation for FAMA is sourced from the author's GitHub repository [6] We use a machine with an Intel Core i7-9750H CPU and 16 GB RAM to compare the elapsed time and memory usage of for each algorithm. We attempt to learn a single action model for each of L_1, L_2 and L_3 using each AML algorithm. We measure elapsed time and memory usage via *wall clock time* and *maximum resident set size* as output by the Unix `time` utility. On Windows, we measure these using the self-reported elapsed time and the maximum private bytes as reported by the Performance Monitor utility. $f_{AML}(\tau, \alpha)$ and $\langle s - a_f - s \rangle \in \tau$ are manually entered based on the algorithm's properties. Accuracy is measured by conducting the evaluation procedure described earlier, however we have not done so in this work, hence we do not report it. Our findings are listed in Table 2. For reproducibility, we share all the code used to obtain our results in a GitHub repository[7]

Computation time and memory usage for most AML algorithms evaluated seem to scale well with the trajectory size. However, none of them are able to use a previously learned action model or failed actions to augment their learning process. Algorithms which *are* able to account for previously learned action models (Zhuo, Nguyen, and Kambhampati 2013) and failed actions (Wang 1995) were not readily available to us for testing. We do not report data for FAMA on L_3 since it ran out of memory on our test machine.

[5]http://xplan-lab.org/#SourceCode

[6]https://github.com/daineto/meta-planning

[7]https://github.com/AbhijeetKrishnan/expertise-estimation

Discussion and Future Work

To the best of our knowledge, our work is the first application of AML to player modeling. We demonstrated the feasibility of AML as a domain-agnostic player modeling approach by evaluating an existing AML algorithm for the task of player modeling, and we presented a technique to measure the player's mechanical proficiency using the learned model. Based on these findings, we believe action model learning has the potential to serve as useful player modeling technique in a wide range of games.

Our claim that action models can serve as general player models rests on the assumption that the game be representable in a planning formalism such as PDDL. This is a strong assumption and it might not be practically possible to represent large categories of games which operate in real-time (although there is some prior work is using PDDL for real-time domains (Bartheye and Jacopin 2009)). We conjecture that games which fall under the genres of turn-based strategy (e.g. *Pokémon* series (Game Freak 1996-2019), *Chrono Trigger* (Square 1995)), tile-based board games (e.g. *CATAN*® (CATAN GmbH 2002), chess) and tile-based puzzle games (e.g. Sokoban, *Baba is You* (Hempuli 2019)) are most amenable to representation in a planning formalism. There remain concerns regarding the PDDL authoring burden in creating such games (Strobel and Kirsch 2020).

Our proposed measure of player competency will also benefit from refinement in future work. Quantitatively estimating a player's mechanical proficiency leaves out higher-order skills learned by combining the use of mechanics that players need to learn in order to become proficient at a game (Cook 2007). Future work will attempt to model and measure the acquisition of higher-order skills.

We were motivated in this project by a hypothesis that the cognitive process of mental model formation (Wasserman and Koban 2019) could help design better AML algorithms for player modeling. AML algorithms which update action models based on failed actions, which are used similarly in mental model alignment (Boyan, McGloin, and Wasserman 2018), provide a theoretical basis for correlation of the learned action models with human mental models. However, we have not yet conducted an experiment to measure how well our learned models match human players' mental models; we plan to do this in future work.

Our proposed evaluation method measures similarity between the learned player model and the player's mental model along one narrowly defined function i.e. predicting the validity of an action or next state. It is desirable to have a more fine-grained evaluation procedure which compares directly against a ground-truth action model obtained from the player. This could be done by providing an interface to humans for directly manipulating pieces of an action model in order to construct an action model closest to their mental model. Existing research into providing intuitive interfaces

Algorithm	Time (s)			Memory (MB)			$f_{AML}(\tau, \alpha)$	$\langle s - a_f - s \rangle \in \tau$
	L_1	L_2	L_3	L_1	L_2	L_3		
ARMS	0.05	0.01	0.67	17.06	16.66	91.75	✗	✗
LOCM	0.21	0.19	0.15	21.50	34.35	74.64	✗	✗
LOCM2	0.25	0.20	0.10	20.93	33.72	70.97	✗	✗
LOUGA	0.92	0.77	3.28	25.06	37.74	64.04	✗	✗
FAMA	10.59	1572.60	—	21.19	4632.52	—	✗	✗

Table 2: Results of AML algorithm evaluation

for rule-based systems (Card and Martens 2019) could prove useful.

Even then, we run into the issue that certain predicates in the domain are not represented in the game as shown to the player. For example, in the Sokoban domain we use, there exists a predicate `(clear ?location - location)` to denote that a location does not contain the player or a stone. However, this is not communicated to the player visually, and the predicate's existence must be inferred by exploration of the game's mechanics. Therefore, any evaluation method which asks the player to construct an action model of their player model must account for this mismatch of representation.

Conclusion

We have proposed and justified the use of AML as a feasible, domain-agnostic and useful technique for player modeling. We have described how to use any existing AML algorithm to learn a player model, how to use it to quantify player skill and how to evaluate the learned player model. We have evaluated a number of existing AML algorithms for their suitability for player modeling. We found that most AML algorithms meet the performance requirements for player modeling, but lack the ability to utilize failed actions or take prior action models into account; properties which are useful for player modeling. We believe new AML algorithms with these properties, along with heuristics derived from mental model formation, will provide better results for player modeling.

References

Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning action models with minimal observability. *Artificial Intelligence* 275:104 – 137.

Amir, E., and Chang, A. 2008. Learning partially observable deterministic action models. *Journal of Artificial Intelligence Research* 33:349–402.

Angluin, D., and Smith, C. H. 1983. Inductive inference: Theory and methods. *ACM Comput. Surv.* 15(3):237–269.

Bartheye, O., and Jacopin, É. 2009. A real-time pddl-based planning component for video games. In *AIIDE*.

Biswas, P., and Springett, M. 2018. User modeling. *The Wiley Handbook of Human Computer Interaction Volume* 143.

Boyan, A.; McGloin, R.; and Wasserman, J. A. 2018. Model matching theory: A framework for examining the alignment between game mechanics and mental models. *Media and Communication* 6(2):126–136.

Card, A., and Martens, C. 2019. The ceptre editor: A structure editor for rule-based system simulation. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 133–137.

Chakraborti, T.; Kambhampati, S.; Scheutz, M.; and Zhang, Y. 2017. AI challenges in human-robot cognitive teaming. *CoRR* abs/1707.04775.

Chrysafiadi, K., and Virvou, M. 2013. Student modeling approaches: A literature review for the last decade. *Expert Systems with Applications* 40(11):4715 – 4729.

Coles, A.; Coles, A.; Olaya, A. G.; Jiménez, S.; López, C. L.; Sanner, S.; and Yoon, S. 2012. A survey of the seventh international planning competition. *AI Magazine* 33(1):83–88. Copyright - Copyright Association for the Advancement of Artificial Intelligence Spring 2012; Document feature - ; Diagrams; Last updated - 2018-10-06.

Cook, D. 2007. The chemistry of game design. Accessed May 15, 2020 from https://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using locm. *Knowledge Engineering Review* 28(2):195–213.

Drachen, A.; Canossa, A.; and Yannakakis, G. N. 2009. Player modeling using self-organization in tomb raider: Underworld. In *2009 IEEE Symposium on Computational Intelligence and Games*, 1–8.

Gregory, P., and Cresswell, S. 2015. Domain model acquisition in the presence of static relations in the lop system. In *Twenty-Fifth International Conference on Automated Planning and Scheduling*.

Harrison, B., and Roberts, D. L. 2011. Using sequential observations to model and predict player behavior. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, 91–98.

Hooshyar, D.; Yousefi, M.; and Lim, H. 2018. Data-driven approaches to game player modeling: A systematic literature review. *ACM Comput. Surv.* 50(6).

Junghanns, A., and Schaeffer, J. 1997. Sokoban: A challenging single-agent search problem. In *In IJCAI Workshop on Using Games as an Experimental Testbed for AI Reasearch*. Citeseer.

Karpinskyj, S.; Zambetta, F.; and Cavedon, L. 2014. Video game personalisation techniques: A comprehensive survey. *Entertainment Computing* 5(4):211 – 218.

Kučera, J., and Barták, R. 2018. Louga: learning planning operators using genetic algorithms. In *Pacific Rim Knowledge Acquisition Workshop*, 124–138. Springer.

Machado, M. C.; Fantini, E. P.; and Chaimowicz, L. 2011. Player modeling: Towards a common taxonomy. In *2011 16th international conference on computer games (CGAMES)*, 50–57. IEEE.

Matsuda, N.; Cohen, W. W.; and Koedinger, K. R. 2015. Teaching the teacher: Tutoring simstudent leads to more effective cognitive tutor authoring. *International Journal of Artificial Intelligence in Education* 25(1):1–34.

Muggleton, S., and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming* 19:629–679.

Nogueira, P. A.; Aguiar, R.; Rodrigues, R. A.; Oliveira, E. C.; and Nacke, L. 2014. Fuzzy affective player models: A physiology-based hierarchical clustering method. In *Tenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Novak, P. K.; Lavrač, N.; and Webb, G. I. 2009. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research* 10(2).

Oates, T., and Cohen, P. R. 1996. Learning planning operators with conditional and probabilistic effects. In *Proceedings of the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*, 86–94.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1):54–67.

Pfau, J.; Smeddinck, J. D.; and Malaka, R. 2018. Towards deep player behavior models in mmorpgs. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '18, 381–392. New York, NY, USA: Association for Computing Machinery.

Reynolds, M. 2019. How forza 5's cloud-based drivatars work. https://www.digitalspy.com/videogames/xbox-one/a530468/forza-motorsport-5-how-the-xbox-one-racers-drivatar-system-works. Accessed: January 23, 2020.

Serafini, L., and Traverso, P. 2019. Incremental learning of discrete planning domains from continuous perceptions. *arXiv preprint arXiv:1903.05937*.

Smith, A. M.; Lewis, C.; Hullet, K.; Smith, G.; and Sullivan, A. 2011. An inclusive view of player modeling. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, 301–303.

Snodgrass, S.; Mohaddesi, O.; and Harteveld, C. 2019. Towards a generalized player model through the peas framework. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, FDG '19. New York, NY, USA: Association for Computing Machinery.

Stern, R., and Juba, B. 2017. Efficient, safe, and probably approximately complete learning of action models. *CoRR* abs/1705.08961.

Strobel, V., and Kirsch, A. 2020. Mypddl: Tools for efficiently creating pddl domains and problems. In *Knowledge Engineering Tools and Techniques for AI Planning*. Springer. 67–90.

Studios, I. 2019. Imangi studios - privacy policy. https://www.imangistudios.com/privacy/privacy-policy.html. (Accessed: January 23, 2020).

Summerville, A.; Guzdial, M.; Mateas, M.; and Riedl, M. O. 2016. Learning player tailored content from observation: Platformer level generation from video traces using lstms. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Walsh, T. J., and Littman, M. L. 2008. Efficient learning of action schemas and web-service descriptions. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, 714–719. AAAI Press.

Wang, P.; Rowe, J.; Min, W.; Mott, B.; and Lester, J. 2018. High-fidelity simulated players for interactive narrative planning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, 3884–3890. AAAI Press.

Wang, X. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Machine Learning Proceedings 1995*. Elsevier. 549–557.

Wasserman, J. A., and Koban, K. 2019. Bugs on the brain: A mental model matching approach to cognitive skill acquisition in a strategy game. *Journal of Expertise/June* 2(2).

Wu, K.; Yang, Q.; and Jiang, Y. 2005. Arms: Action-relation modelling system for learning action models. *Proceedings of ICKEPS*.

Yannakakis, G. N., and Maragoudakis, M. 2005. Player modeling impact on player's entertainment in computer games. In Ardissono, L.; Brna, P.; and Mitrovic, A., eds., *User Modeling 2005*, 74–78. Berlin, Heidelberg: Springer Berlin Heidelberg.

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and André, E. 2013. Player modeling.

Zhuo, H. H.; Hu, D. H.; Hogg, C.; Yang, Q.; and Munoz-Avila, H. 2009. Learning htn method preconditions and action models from partial observations. In *IJCAI*, 1804–1810.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence* 174(18):1540 – 1569.

Zhuo, H. H.; Nguyen, T.; and Kambhampati, S. 2013. Refining incomplete planning domain models through plan traces. In *Twenty-Third International Joint Conference on Artificial Intelligence*.