Learning Physically-Instantiated Game Play Through Visual Observation

Andrei Barbu, Siddharth Narayanaswamy, and Jeffrey Mark Siskind

Abstract—We present an integrated vision and robotic system that plays, and learns to play, simple physically-instantiated board games that are variants of Tic-Tac-Toe and Hexapawn. We employ novel custom vision and robotic hardware designed specifically for this learning task. The game rules can be parametrically specified. Two independent computational agents alternate playing the two opponents with the shared vision and robotic hardware, using pre-specified rule sets. A third independent computational agent, sharing the same hardware, learns the game rules solely by observing the physical play, without access to the pre-specified rule set, using inductive logic programming with minimal background knowledge possessed by human children. The vision component of our integrated system reliably detects the position of the board in the image and reconstructs the game state after every move, from a single image. The robotic component reliably moves pieces both between board positions and to and from off-board positions as needed by an arbitrary parametrically-specified legal-move generator. Thus the rules of games learned solely by observing physical play can drive further physical play. We demonstrate our system learning to play six different games.

I. INTRODUCTION

Children learn to play games by watching the play of others. While both formal board games, like Chess, Checkers, and Backgammon, and less formal play like Hopscotch, Tag, and Dodgeball all have well defined rules that children ultimately come to know, they are rarely told those rules explicitly. Knowledge of how to play many classic board games is largely passed down culturally, with children never reading, or even explicitly ignoring, the formally specified rules (e.g., as in the case of Monopoly®). We are engaged in a long-term research effort to emulate on robots this ability to learn to play games by observing others play. The work presented here is in turn part of a larger effort to ground learning, reasoning, and language in visual perception and motor control. Physical instantiation is crucial to our effort of situating robust learning, visual perception, and manipulation in the real world. We want physical robots to play a physical game where knowledge of game play allows their vision systems to determine game progress and motor systems to effect game progress. In addition, we want a physical learner to visually observe that play to learn the game rules and ultimately be able to use the learned rules to support physical game play.

Our long-term vision for this overall task is depicted in Fig. 1. In this task, two robotic agents, the protagonist and antagonist, play a board game like Chess. A third robotic agent, the wannabe, does not know the rules of the game but must infer the rules by visually observing the play of the protagonist and antagonist. The wannabe must then use these rules for further physically-instantiated play. In the long term, we wish to be able to do this for a wide variety of off-the-shelf game hardware for a wide variety of common physically-instantiated board games. Our efforts are similar to those of [1], [2], and [3], differing in four crucial ways. Our learned rules support:

a) play by physical robots,
b) learning by observing fully autonomous robotic play rather than human play,
c) learning by observing play with off-the-shelf game hardware, and
d) a significantly expanded space of two-player board games.

Our objective is to learn to play legally, not necessarily well. Expert computer game play is one of the most extensively studied and successful sub-disciplines of AI. Our goal is orthogonal to that enterprise.

We have constructed a novel custom robotic testbed, containing three separate robotic agents to support this enterprise, and have used this testbed to successfully learn six different physically-instantiated games. While one long-term goal is to learn a wide variety of common board games, like Chess, Checkers, Backgammon, and Go, with differing physical game hardware, the work presented in this paper is limited to games which share the same game hardware; a restriction which we are in the process of eliminating. And while another long-term goal is to use three separate robotic agents to play the roles of protagonist, antagonist, and wannabe, the work presented here uses a single robot to play all three roles.\footnote{We have however, run our system on three robotic agents and can demonstrate the ability of the virtual wannabe to learn by observing play on one or more different physical robots and then drive play on a yet another different physical robot.}

The remainder of this paper is organized as follows. Section II discusses our custom robotic testbed. Section III discusses the space of games learned. Section IV discusses the techniques used by the robotic vision system to reconstruct the game state. Section V discusses the techniques used to control the robotic manipulator to effect game play. Section VI discusses the techniques used to learn game rules. Section VII demonstrates our system’s effectiveness in learning physically-instantiated games. Section VIII concludes with a discussion of future work.

The authors are with the School of Electrical Engineering, Purdue University, West Lafayette, IN, 47907, USA \{abarbu, snarayan, qobi\}@purdue.edu
Fig. 1. Learning physically-instantiated game play through visual observation. Two robotic agents, the protagonist and antagonist, play a board game like Chess. A third robotic agent, the wannabe, does not know the rules of the game but must infer the rules by visually observing the play of the protagonist and antagonist. The wannabe must then use these rules for further physically-instantiated play.

II. OUR CUSTOM ROBOTIC TESTBED

We have designed a custom robotic testbed consisting of three identical agents, one of which is shown in Fig. 2. While much of our robot is constructed with off-the-shelf parts from vendors like Lynxmotion and Logitech, many crucial parts were custom designed, milled, or repurposed to meet the particular needs of the game-playing task. The two most novel parts are the overall housing and camera-mount assembly. The overall housing consists of a two-level wood platform, where the upper level constitutes the game-play surface and the lower level serves as the mounting point for the camera assembly.

A 4 DOF arm with two independently controllable fingers, constructed from servo motors and parts from Lynxmotion, is bolted to the upper level. The camera assembly consists of a pair of pan-tilt USB webcams mounted on a 1 DOF pendulum arm which is in turn mounted on a Lynxmotion servo base bolted to the lower level. We found Logitech QuickCam Orbit cameras well-suited to our task, as we were able to strip them down to a lightweight assembly containing the camera, pan-tilt motors, and electronics. The assembly was then mounted on the pendulum arm using a custom milled bracket. Mounting the cameras in this fashion on a pendulum arm allows them to pivot, under computer control, 180° around the center of the game-play surface. However, due to the excessive forces on the pendulum arm and camera assembly, we needed to replace the central mounting bracket in the servo base with a stronger custom-milled variant.

The size of the overall housing and the arm link lengths were designed to support game play with off-the-shelf game hardware. Further, the design of the pivotable camera mount, and in particular the position of the cameras relative to the center of the game-play surface, allows the entire game board to be in the binocular field of view, through the entire 180° pivot range. The experiments reported in this submission image the game-play surface from a single camera pivot angle. In the future, we plan to use the ability to pivot the camera under computer control to have the protagonist, antagonist, and wannabe view the game from different camera poses.

III. THE SPACE OF GAMES CONSIDERED

For reasons to be discussed momentarily, our games share common physical game hardware consisting of an off-the-shelf Tic-Tac-Toe set (see Fig. 2). This particular game hardware simplifies the necessary robotic manipulation in several ways. First, the fact that the board positions are depressions makes piece placement somewhat self-correcting. Second, the piece size is well matched to our manipulator. Finally, the shape of the X’s makes the grasping process and resulting piece orientation somewhat self-correcting. The game hardware also simplifies the necessary visual processing by allowing one to find the board positions with an ellipse detector. In addition to the board, we have staging areas for storing off-board game pieces that are not in play. Since the off-the-shelf Tic-Tac-Toe set did not include such, we constructed our own that contain self-correcting circular depressions of the same size as the board.

Our long-term objective is to be able to learn any typical board game such as Chess, Checkers, Backgammon, Go,

Stratego, etc. Such a wide variety of games would require more general perceptual and motor abilities than we have implemented. We wish to leverage our implemented perceptual and motor abilities as much as possible yet verify the generality of our overall approach by evaluating its ability to learn a variety of games. Thus we have chosen a collection of six simple games that can all be played with the same physical game hardware, robotic testbed hardware, and perceptual and motor software. Two, Tic-Tac-Toe and Hexapawn\(^3\) [4], are commonly-known games, while the remaining four are minor variants of Hexapawn. We summarize these variants below:

**Variant A** Non-capturing moves are forward along the diagonal instead of straight ahead.

**Variant B** Variant A augmented with backward diagonal non-capturing moves.

**Variant C** Hexapawn allowing backwards vertical non-capturing moves.

**Variant D** Variant C augmented with sideways non-capturing moves.

Our system contains a generic game-playing engine that is parameterized by a game specification containing an initial board configuration, legal-move generator, and outcome predicate. It accepts game specifications coded either in SCHEME or PROLOG. We have hand-coded game specifications for all of the above six games. These can be used to drive physical game play where the robot (Fig. 2) alternates play between X and O and determines the game state by visual observation of the physical game between each move. The learning component to be described in Section VI can learn game specifications in PROLOG for all of the above six games from visual observation of physical game play which can then be used to drive new physical game play. While our game-playing engine also supports optimal play via minimax game-tree search with alpha-beta pruning, optimal play does not facilitate learning of games rules as many situations necessary to learn the complete rule set do not arise, particularly for learning the outcome predicate in deterministic games. Thus for learning we employ random legal play.

### IV. RECONSTRUCTING GAME STATES FROM IMAGES

Physically-instantiated game play requires recovery of the game state from visual information, which nominally is a two-stage process. First, one must determine the world state, i.e., the shapes and positions of various pieces and board regions. Then, one must map this world-state information into game states. The former process is nominally a game-independent general vision task of scene reconstruction and may incorporate camera calibration, segmentation, object recognition, metric reconstruction, and pose estimation. The latter process requires game-specific knowledge to determine which features of the reconstructed scene are relevant to the particular game being played. For example, in some games, like Chess and Checkers, pieces are placed between edges, while in others, like Go, pieces are placed on edges. In the longer term, we intend to learn such game-specific knowledge of how world states map to game states along with the initial board configuration, legal-move generator, and outcome predicate. In this paper, we restrict consideration to games played with the particular game hardware discussed in Section III where all games use the same hard-coded mapping from world state to game state. Thus we conflate the two-stage process into a direct mapping from images to game states.

This direct mapping requires prior camera calibration. Under a two-stage process, camera calibration would support metric reconstruction, in part, by determining the camera pose relative to the board. With the direct mapping, there is no such metric reconstruction and no need for camera pose information. Instead, our calibration process finds image coordinates that correspond to game-state board positions. A Tic-Tac-Toe board consists of a 3 \(\times\) 3 grid of board positions. With our particular game hardware, these board positions are circular depressions which, when imaged projectively by our robot, manifest as ellipses. Thus we can identify the board positions in images taken from a particular camera pivot angle by searching an image of an empty board taken at that camera pivot angle for a 3 \(\times\) 3 grid of ellipses. We use the ellipse detector (cvFitEllipse) in OpenCV [5] for this purpose. We make this process more robust by increasing the contrast of the circular depressions relative to the remainder of the game board to highlight the edges.

The above calibration process needs to be done only once at the beginning of a game or training session, provided that the camera does not move. Given the known image locations of target game-board positions the process of game-state reconstruction reduces to the problem of determining whether a given board position is empty or contains an X or an O. This three-way distinction is made in a fashion that is particular to our game hardware. The presence of any piece in a board position is detected by the fact that it obscures any ellipse that would correspond to a depression at that board position. The distinction between pieces at a board position rests on the fact that Os manifest themselves as smaller ellipses at a given board position whereas Xs do not. As the error rate of the ellipse detector depends on various threshold settings, we make this process highly robust by sampling at multiple thresholds and voting on the outcome.

Our game-state reconstruction process is sufficiently robust that it made only two errors in the approximately 2000 reconstructions performed during the 62 games played autonomously for the experiments reported in this paper. As both occurred during the cleanup at the end of the training sequence, they did not impact the correctness of learning or subsequent physical play with the learned rules.

While the cognitive portion of game play relies solely on the game state represented in the 3 \(\times\) 3 grid of board positions, the robotic portion of game play requires knowing

\(^3\)On a 3 \(\times\) 3 board, three white pieces start on one edge and three black pieces start on the opposite edge. Pieces move and capture like chess pawns without en passant or initial two-square moves. Players win by queening and lose when unable to move.
the positions of off-board pieces in the staging areas. We currently do not determine those positions from visual input and rely on properties of the particular robotic manipulation strategy discussed in Section V that allow inferring the positions of off-board pieces from the observed game state.

V. GAME-RULE-INDEPENDENT ROBOTIC CONTROL

Physically-instantiated game play also requires robotic ability to effect the desired changes in the physical game state. This also nominally can be divided into a two-stage process. First, one must determine the necessary changes in the world state that correspond to the desired changes in the game state, i.e., a legal move. Then, one must effect that change to the world state. The latter process is again nominally a game-independent robotic manipulation task which may incorporate forward and inverse kinematics, grasp planning, and path planning. The former process, however, requires game-specific knowledge, essentially the inverse of the game-specific knowledge needed for game-state reconstruction from visual input. Like before, we restrict consideration to games played with the particular game hardware discussed in Section III but do not conflate this into a single-stage process.

Our legal-move generator is formulated as a mapping from old game states to new game states. We formulate a generic method, particular to our class of games played on a $3 \times 3$ grid but applicable to any game in that class, that finds a minimal number of pick up and put down operations to effect the target change in the physical game state. Such operations may move pieces between two board positions, or between the staging area and the board. In the case of the latter, we assume that game rules do not constrain the choice of staging-area location for any particular legal move and treat the staging areas as a last-in-first-out stack, one on one side for Xs and on the other for Os. This stack behavior is what allows indirect inference of the positions of off-board pieces from the observed game state without direct visual observation. We also have a generic “clean up” capability that can return all pieces to a state that corresponds to an arbitrary (but learned) initial board configuration. This allows completely autonomous robotic play of a sequence of games to provide training data for the learner and evaluate autonomous play with the learned rules.

The above constitutes the first stage of the two-stage process, namely mapping from target game-state changes to world-state changes. The second stage currently uses an open-loop dead-reckoning process. We have hard-coded the 3D world coordinates of the board and staging-area positions for each of our three robots. We then employ inverse kinematics to determine a sequence of joint-angle configurations to effect a desired pick up or put down operation, parameterized by a specific board or staging-area position. The nature of board-game play allows straightforward planning of a collision-free path by approaching board and staging-area positions from above. Restriction of the game hardware to a particular piece set means that we can hard-code the grasp planning for each piece type. This is implemented by providing the pick up or put down operations with piece type as an additional parameter, derived visually. Overall, the self-correcting nature of the position depressions in the board and staging areas, combined with the self-correcting character of picking up Xs with two fingers, allows very robust game play with fairly simple methods. This is crucial when training a learner on long sequences of autonomous game play.

We make the above process even more robust by using visual feedback to confirm the success of a desired pick up or put down operation and compensating upon failure. Our combined vision and robotic-manipulation systems are sufficiently robust that the approximately 2000 pick up and put down operations during autonomous play of the 62 games for the experiments reported in this paper required fewer than 20 human interventions to correct robotic errors.

VI. LEARNING GAME RULES BY ILP

Interestingly, [6] formulates the task of learning a legal-move generator for Tic-Tac-Toe and Hexapawn as reinforcement learning, but it unfortunately does not usually converge to the correct result. Furthermore, the legal-move generator is not represented in a perspicuous human-readable format. It also does not learn the initial board configuration or outcome predicate. [7] have employed inductive logic programming (ILP) to learn a static evaluator for Chess endgames but this prior effort differs from our work in that it does not learn game rules.

We employ ILP with PROGOL [8] to learn perspicuous PROLOG specifications for the game rules from autonomous physically-instantiated game play. We currently do this on a single robot that plays all three roles of protagonist, antagonist, and wannabe, taking care not to allow information flow that could not happen if this were done on three distinct robots. The system is seeded with hand-coded rules for a game in either SCHEME or PROLOG. We start by taking a single image of an empty board to perform camera calibration. Then two agents autonomously play a sequence of random but legal games, at least six for Tic-Tac-Toe and at least ten for Hexapawn and its variants. As discussed earlier, we do not train on optimal play because this does not provide sufficient information to infer the game rules. Since games ending in a draw are not required to learn the outcome predicate, we ignore such randomly occurring games, and thus more games might be played to obtain sufficient training data.

Each game starts with the robot autonomously setting up the physical game hardware to correspond to the initial board configuration, given an image of the current world state, which for any game in the training set but the first, contains pieces remaining on the board from the previous game. The virtual protagonist and antagonist alternate play by taking an image to determine the current game state (which is not stored from the previous turn), selecting a random desired next state from the legal-move generator, planning a sequence of pick up and put down operations to effect that move, and executing those operations on the
robot. Independent from this, the wannabe takes an image between each turn to determine the sequence of game states corresponding to a game and is given the outcome of the game at each turn, which may indicate that the game is not yet over. The only communication between the three agents is this labeling of the outcome at each turn as well as the turn-taking coordination that informs each agent of the transition time between game states.

Upon completion of the autonomous training play, the wannabe formulates the training set as input to PROGOL in the following form. A game state is formulated as a 3×3 matrix. Each game is formulated as a fact

\[ \text{initial_board}(G,P), \text{legal_move}(P,G1,G2) \text{ and } \text{outcome}(P,G,O) \]

for each turn, where \( G \) denotes a game-state matrix, \( P \) denotes a player, either \( X \) or \( O \), and \( O \) denotes a win by either \( X \) or \( O \). Note that we provide an outcome training fact for all moves, negated for moves that do not end in a win, which provides negative evidence for learning the outcome predicate. We then use PROGOL to learn definitions for \( \text{initial_board}/2, \text{legal_move}/3 \), and \( \text{outcome}/3 \). To facilitate learning, we augment the training data with the background knowledge in Fig. 3. This background knowledge is the same for all six games discussed in Section III. Much of it encodes general elementary physical and mathematical properties known by almost all children, such as arithmetic (\( \text{inc}/2, \text{dec}/2, \text{row_to_int}/2 \), and \( \text{col_to_int}/2 \)), the concept of a line (\( \text{linear_test}/7 \) and \( \text{linear_obj}/3 \)), and the frame axiom (\( \text{replace}/4, \text{frame}/4 \), and \( \text{frame_obj}/6 \)). Some (\( \text{player}/1, \text{opponent}/2, \text{piece}/1, \text{empty}/1, \text{owns}/2, \text{win_outcome}/1, \text{owns_outcome}/2, \text{owns_piece}/2, \text{row}/1, \text{col}/1, \text{board}/1, \text{ref}/3, \text{at}/4, \text{and}/5 \)) encode knowledge about the mapping between world state and game state which we currently do not learn but anticipate learning in the future. The remainder (\( \text{forward}/3 \) and \( \text{sideways}/2 \)) encode combinations of that mapping with general geometric knowledge known by almost all children. To drastically reduce the learning time, we use a non-generative version of \( \text{at}/4 \) (with cuts) while learning the legal-move generator.

During training, we first learn both the initial board configuration and the legal-move generator with the background knowledge from Fig. 3. Then, to learn the outcome predicate, we augment the training set and the background knowledge with the learned legal-move generator, along with two predicates, \( \text{has}_\text{move}/2 \) and \( \text{has}_\text{no}_\text{move}/2 \), that access that learned legal-move generator. Furthermore, due to significant overlap in coverage of the clauses generated by CPROGOL for the outcome predicate, we search for a minimal subset of the candidate clauses produced by CPROGOL before its internal redundancy algorithm that covers the training set. To drastically reduce the learning time, we replace the definition of the frame axiom with one that is vacuously true while training the outcome predicate.

VII. RESULTS

Our system can learn the rules of all six games discussed in Section III from autonomous physically-instantiated play by robotic agents using the methods discussed. The learned game rules are given in Fig. 4. From simulated non-robotic play, we have determined that six training examples are almost always sufficient in practice to correctly determine the game rules for Tic-Tac-Toe (ten for Hexapawn and its variants). Due to the fact that we train on random legal play, in rare occurrences the training set can be pathological and contain a skewed mix of the possible game situations leading to incorrect generalization. Furthermore, random play can make certain situations rare, such as the requirement in Hexapawn that players lose when unable to move. We have also determined, from simulated non-robotic play, that this situation occurs with high probability among a sample of ten games but not six.

We have set up a website\(^4\) that contains unedited screen-dump videos that demonstrate the full autonomously-played training set for each of the six games as well as subsequent autonomous play using each of the six sets of learned game rules. This website also contains the full source code for our system as well as engineering drawings for the design of our custom robotic testbed and staging areas allowing others to replicate and build upon our work.

VIII. DISCUSSION AND FUTURE WORK

While the early vision for “greater AI” integrated low-level perceptual and motor processing with high-level reasoning and language, the field fractured into disjoint disciplines long ago. We wish to advocate the \( \text{ab initio} \) learning of the rules of common board games from autonomous physically-instantiated play on off-the-shelf game hardware through visual observation as a touchstone task to motivate the field to return to its roots. The task of physically-instantiated board-game play is an extremely rich yet circumscribed metaphor for much of the real-world experience of children.

This task requires linkage between low-level representations of image-level characteristics, such as edges and regions derived by segmentation, through mid-level representations of objects and actions, such as players picking up and putting down the playing pieces, to high-level representations of game notions, such as game states and legal moves. By \( \text{ab initio} \), we mean starting out with no knowledge about the states or rules of particular games and making as few assumptions as possible about the nature of games and game playing in general, except for general background knowledge that children possess such as physical reasoning, perceptual organization, social behavior and reasoning, along with the ability to form metaphoric analogies between games and the real world.

Physical reasoning can determine stability and support relations and aid the task of learning the rules of board games. Physical metaphors greatly constrain the space of possible game definitions that an observer must consider.

\(^4\)\url{http://tinyurl.com/oxt2f9}
when learning a game from visual input. This is what allows children to learn games so effectively and quickly, in contrast to machine-learning and data-mining systems that attempt to find arbitrary structure in uninterpreted datasets. Most board games share a common generic ontology that is constrained by physics. Game states typically correspond to physical states, i.e., positions of pieces on the board. Since game states must persist without human intervention, they must correspond to physically stable configurations of the playing pieces. This implies that boards are typically horizontal and planar with playing pieces typically resting on the board or on top of other pieces. Some games, like Checkers and Backgammon, allow pieces to stack, while others, like Chess, Parcheesi, and Go, do not. This tends to correlate with whether the piece shape allows stable stacking.

Gestalt rules of perceptual organization can aid learning of the mapping between world states and game states. Boards tend to be divided into regions that are differentiated by salient edges. As stated earlier, most game states correlate to pieces being placed inside regions, as in Checkers, Chess, and Parcheesi, or being placed on the edges, as in Go. The relative size of the pieces and regions correlates with whether game states allow multiple pieces per region. Games like Backgammon and Parcheesi do, while games like Checkers, Chess, and Go do not. The topology of the board correlates with the nature of play. Games where the region graph contains a one-dimensional path, like Backgammon and Parcheesi, are typically race games, while games where the region graph has two-dimensional connectivity, like Checkers, Chess, and Go, are typically war games.

Social reasoning and metaphoric analogy to social behavior can aid learning of the legal-move generator. Games are primarily played by picking up pieces and putting them down elsewhere. The legality of game-state transitions typically correlates with physical constraints. Some games only allow moves to adjacent regions. Others allow moves to nonadjacent regions, but only when such moves are along linear trajectories where pieces do not collide. Just as physical collisions injure the impacted party, collisions in games usually negatively affect the impacted piece, capturing or killing the piece, in war games, or increasing its distance from the goal, in race games. The speed of movement is relevant to race games but not war games. Thus the allowed distance that a piece can move in a race game is usually governed by a random-number generator, such as the roll of a die, while in war games, legal moves usually correlate with piece rank, which is usually indicated by piece shape and correlated with piece size. Larger pieces typically are more powerful and have greater freedom of movement. Players can only move pieces that they own, where ownership is usually indicated by color.

Incorporating physical reasoning, perceptual organization, and social and behavioral reasoning into the background knowledge is part of our long-term research plan. Our medium-term focus is on more robust and integrated vision, robotic, and learning components. We foresee using a probabilistic programming language like CHURCH [9] and basing robotic manipulation on closed-loop visual servoing instead of open-loop dead reckoning. One can even imagine integrated stochastic reasoning across different games within a training set to support incremental learning and across different training sets for different games to support metaphorical knowledge transfer between different game rules. Finally, one can imagine using social knowledge in the form of estimations of the relative expertise of the players to allow learning using negative evidence from play of differing levels of optimality ranging from random to informed to near optimal.

The richness of this task allows for an open-ended research program yet the circumscribed nature of the task allows for robust incremental progress. Our experience reported in this paper demonstrates the feasibility of the enterprise.

ACKNOWLEDGMENTS

This work was supported, in part, by NSF grant CCF-0438806. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

REFERENCES

inc(X,Y):- Y is X+1.
dec(X,Y):- Y is X-1.
player(player_x).
player(player_o).
opponent(player_x,player_o).
opponent(player_o,player_x).
piece(x).
piece(o).
piece(none).
empty(none).
owns(player_x,x).
owns(player_o,o).
owns_outcome(player_x,x_wins).
owns_outcome(player_o,o_wins).
owns_piece(x_wins,x).
owns_piece(o_wins,o).
row(r0).
row(r1).
row(r2).
col(c0).
col(c1).
col(c2).
row_to_int(r0,0).
row_to_int(r1,1).
row_to_int(r2,2).
col_to_int(c0,0).
col_to_int(c1,1).
col_to_int(c2,2).
board([[A,B,C],[D,E,F],[G,H,I]]):-
piece(A),piece(B),piece(C),
piece(D),piece(E),piece(F),
piece(G),piece(H),piece(I).
fig(3).

Tic Tac Toe

- initial_board([[none,none,none],
  [none,none,none],
  [none,none,none]],player_x).
- legal_move(A,B,C) :- owns(A,D),
  row(E),
  col(F),
  at(E,F,B,none,G),
  at(E,F,C,D,H),
  frame_obj(G,B,B,G,B,B,C).
- outcome(A,B,C) :- owns_piece(C,D),
  at(E,F,B,D),
  at(G,H,B,D),
  linear(B,F,G,H,I,J).

Hexapawn

- initial_board([[x,x,x],
  [none,none,none],
  [o,o,o]],player_x).
- legal_move(A,B,C) :- row(D),
  col(E),
  owns(A,F),
  empty(G),
  forward(A,B,D),
  at(k,E,B,F,D),
  frame_obj(i,H,D,G,H,C,B).
- outcome(A,B,C) :- row(D),
  frame_obj(i,j,K,J,B,C).

Fig. 3. Background knowledge encoded in Prolog. Prolog-specific settings and mode, type, and pruning declarations have been omitted.

Fig. 4. Rules for two of the six games discussed in Section III learned automatically from visual observation of autonomous physically-instantiated game play.