Advances in Cognitive Systems 3 (2014) 31-46 Submitted 9/2013; published 7/2014

X Goes First: Teaching Simple Games through Multimodal Interaction

Thomas R. Hinrichs T-HINRICHS@NORTHWESTERN.EDU
Kenneth D. Forbus FORBUS@NORTHWESTERN.EDU
Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA

Abstract

What would it take to teach a computer to play a game entirely through language and sketching? In this paper, we present an implemented program through which an instructor can teach the rules of simple board games using such input. We describe the architecture, information flow, and vocabulary of instructional events and walk through an annotated example. In our approach, the instructional and communication events guide abductive reasoning for language interpretation and help to integrate information from sketching and language. Having a general target representation enables the learning process to be viewed more as translation and problem solving than as induction. Lastly, learning by demonstration complements and extends instruction, resulting in concrete, operational rules.

1. Introduction

A long-term promise of cognitive systems is that they should be able to learn simple tasks by instruction, rather than requiring detailed programming, yet formidable obstacles remain. For tasks that are structured, conditional, iterative, and context sensitive, a learner must construct fairly sophisticated representations. Resolving the imprecision and ambiguities of natural interaction depends critically on plausible inference and the expectations of the learner. What kinds of expectations can be brought to bear when learning from such interaction?

We have been exploring these issues through the multimodal instruction of simple games. The teacher and learner communicate via natural language textual dialog and sketching. From this interaction, the learner acquires the definitional rules of a game sufficient to play it. The benefit of this is that the simplicity of the learning task allows us to focus on instructional interaction patterns. It is clear when the rules have been learned correctly, incorrectly, or incompletely, whereas higher-level strategies are more subjective and harder to evaluate. Board games, in particular, are ideal because they are simple, highly spatial, and have explicit concrete rules.

This paper presents our initial analysis and experiments in teaching Tic Tac Toe. Although an extremely simple game, the instruction runs headlong into many of the difficulties to which we have alluded. In fact, the title of this paper suggests some of the issues: “X goes first.” By itself, this has no clear meaning. Here, X is a metonymic reference to a player based on the form of his or her mark. “Goes” in this case, means takes a turn, and “first” means that turn is first in an implicit sequence of turns. The input sentence and the intended meaning are both simple, but the interpretation process is not. In our model, expectations are built up progressively by recognizing implicit instructional events in the dialog. This contextual bootstrapping is complemented by multimodal constraints that ground reference and guide generalization. We aim to identify patterns of interaction by which depiction and verbal instruction mutually constrain each other to help build a coherent task representation.

We make four claims about teaching such content through multimodal interaction:

1. *A small set of instructional events can suffice to constrain interpretation.* These need not be strictly sequenced or form any kind of high-level grammar, but they serve as strong contextual hints for resolving ambiguity, in combination with temporal and spatial proximity clues.
2. *Natural interaction is facilitated by multiple levels of representation.* The content of natural instructional dialog primarily concerns concepts and examples. To produce an executable program, this content must be translated into operational rules. By translating incrementally through different levels of representation, appropriate expectations, such as instructional events, can help disambiguate and interpret input.
3. *Spatial depiction grounds verbal abstractions.* Because a sketch is intrinsically propositional, it can resolve ambiguous abstractions, while verbal descriptions help lift and generalize overly specific examples.
4. *Learning by demonstration complements instruction.* Verbal instruction can most easily express abstract taxonomic classifications, associative relations, and parameters, while demonstration can often present procedures and configurations more concisely. Even a simple game like Tic Tac Toe benefits from integrating these two learning strategies.

The next section describes the concrete task of teaching Tic Tac Toe, the target representation, and the general cognitive architecture. Sections 3 and 4 present the language and sketch understanding components respectively. Section 5 describes the multimodal fusion, learning, and game-playing processes. Section 6 presents an annotated transcript of learning Tic Tac Toe. Section 7 analyses the interaction patterns and their generality, briefly describing experiments in learning the piece-moving game Hexapawn. Lastly, we describe related and future work and summarize the contributions of our research.

2. Teaching Tic Tac Toe

Most of us have some experience teaching Tic Tac Toe to a child or remember learning it ourselves. The extreme simplicity of the game is deceptive – learning it assumes familiarity with a number of concepts such as making physical marks, taking turns, spatial relations between marks, learning from an instructor, rule-defined behavior, and winning and losing, as well as facility with language. Just as a child is not a blank slate, our program also builds on prior knowledge.

Our approach treats learning as a kind of problem solving, rather than as inducing rules over data sets. If you know what you are trying to learn, i.e., the form of the target representation, then learning becomes a matter of filling gaps in this model. Moreover, if that model is operational – it can be employed in a task – then instruction can be augmented by demonstration, practice, and feedback. We therefore define our task as introducing concepts and rules of the game through natural language and sketching, followed by interactively playing the learned game in the same interface with the same sketch.

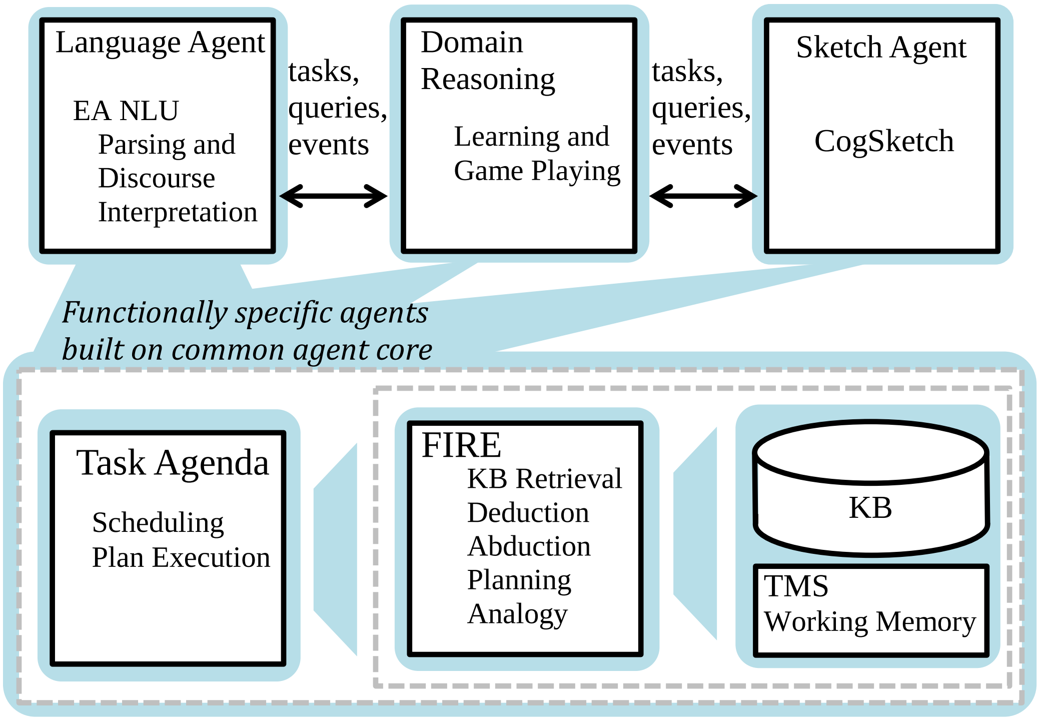


Figure 1. Overview of the Companions agent architecture. Three functionally specific agents are built on a common core of scheduling, reasoning, and retrieval.

2.1 Target Representation

The target representation to be learned must be concrete enough to support interactive play, yet general enough to cover a wide range of possible games. Primarily for this reason, we use an implementation of the Game Definition Language (GDL) (Genesereth & Thielscher, 2014), which represents games as a set of Horn clauses that define a finite state machine. The predicates *initial*, *legal*, *next*, and *terminal* support the definition of a game, augmented by game-specific predicates *role* and *goal*. By fixing the form of the target representation in this way, we are able to support a wide variety of games with an interpreter that can translate between the logical state of a game and the sketch-based presentation.

2.2 Architecture

Our game learner is implemented within the Companions cognitive architecture (Forbus, Klenk, & Hinrichs, 2009), as diagrammed in Figure 1. The learner consists of plans and rules that coordinate the interactions of three primary agents: a language interaction agent that processes natural language, a sketch agent that supports sketch interaction and game play, and a domain reasoning agent that assimilates events from the other two agents and incrementally builds the domain rules of the game. Like nesting dolls, these functionally-specific agents are built on a common reasoning substrate that encapsulates progressively more primitive capabilities. Each agent iteratively plans and executes actions from its own agenda of tasks. This agenda lets game-specific control strategies be declaratively specified without requiring a custom interpreter. Planning and reasoning in agents is supported by the FIRE reasoning engine (Forbus et al., 2010), which performs deductive retrieval and backward chaining, and/or solving, and hierarchical task network planning. FIRE, in turn, employs a truth maintenance system (TMS) which serves as its working memory, and a knowledge base containing the ResearchCyc ontology.[1] This provides a starting vocabulary of many thousands of interrelated concepts and predicates, denotations and semantic frames for mapping from English lexical tokens to symbolic concepts, and a mechanism for partitioning knowledge into *microtheories*, an inheritance lattice of collections of mutually consistent assertions.

3. Language Interaction

The interaction manager is built on top of the Explanation Agent natural language understanding (EA NLU) system (Tomai, 2009), which integrates several off-the-shelf components and resources to produce, given a sentence and discourse context, representations of the possible meanings of that sentence, expressed in the ResearchCyc ontology. The system preserves syntactic and semantic ambiguities in explicitly represented choice sets. Ambiguous choices are resolved using controlled abductive inference (Hobbs et al., 1990), constrained by high-level recognition rules that seek contextually meaningful interpretations.

As we instruct the learner, it applies these general rules to look for instructional events that

Table 1. Example sentence, interpretive rule skeleton, and resulting predicate calculus interpretation.

"The first player to mark three squares in a row wins"

```
(<== (definesGoal ?sent ?role ?goal)
   (winning ?sent ?role ?win)
   (winningAction ?sent ?win ?role
    ?act)
   (winningState ?sent ?role ?act
    ?goal)
   (goalStateConfiguration ?sent
    ?output))

(<== (isa win5927 Winning)
   (abductiveAssumption
   (selectedChoice
    (and (isa win5927 Winning)
       (performedBy win5927
         player5762))))
         …
```

```
(and
 (isa win5927 Winning)
 (performedBy win5927 player5762)
 (isa player5762 GameRole)
 (nthInSeries player5762 series5758 1)
 (doneBy mark5769 player5762)
 (isa mark5769 MarkingOnASurface)
 (objectMarked
  mark5769 group-of-square5801)
 (isa group-of-square5801
  Set-Mathematical)
 (cardinality group-of-square5801 3)
 (elementOf square5801
  group-of-square5801)
 (isa square5801 Square)
 (in-UnderspecifiedContainer
  square5801 row5880)
 (isa row5880 RowOfObjects))
```

1. http://www.cyc.com/platform/researchcyc

could explain an utterance. For example, could a given utterance be interpreted as introducing or naming an entity in the game? When such a query can be satisfied by collapsing an ambiguous choice to a particular semantic interpretation, it will do so. This relies on inference bottoming out in propositional abductive rules that are compiled by the parser for each sentence. By querying for individual conjuncts of an interpretation, it identifies the assumptions that could make it true and selects the reified semantic choice. Collectively, these semantic choices entail a coherent representation of the sentence. Table 1 illustrates an example sentence, a high-level interpretation rule for recognizing goal expressions, a portion of a low-level abductive rule, and the resulting interpretation. In this case, the rule identifies the utterance as a statement of a goal based on the presence of a Winning concept. It further looks for the action performed by the winner and the outcome state of the action. If the outcome state can be interpreted as a spatial configuration, it will prefer that interpretation.

The resulting sentence representation is a predicate calculus description that may contain tokens representing discourse variables (essentially skolems) and embedded microtheories. This representation, although formal, is not an operational specification of a game rule. Consequently, this logical form undergoes at least two additional levels of expectation-based translation. It is first lifted to a more language-neutral form that replaces embedded microtheories and discourse variables with second-order predicates. Based on whether the utterance is declarative, interrogative, or imperative, it is then passed to the domain reasoning agent as the arguments of a *communication event.* The domain reasoning agent assimilates communication events and further translates the content into explicit game rules. This process of progressive translation is illustrated in Figure 2. The vertical arrows show the direction of translation from input to the target game representation.

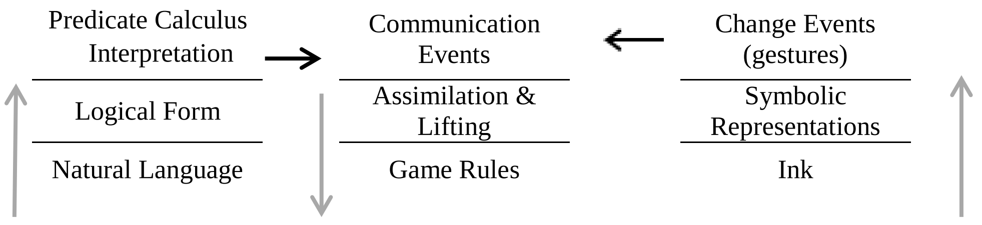
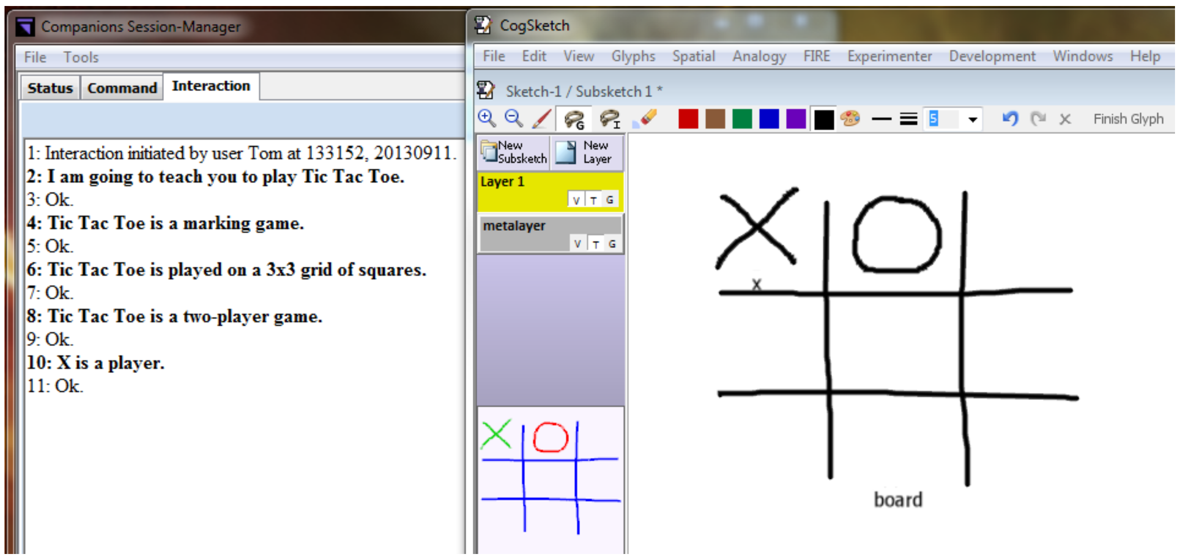


Figure 2. Representational levels used in language, domain reasoning, and sketch agents.

4. Sketch Interaction

Sketch interaction is supported through a Companions agent wrapper around the CogSketch sketch understanding system (Forbus et al., 2011). One of the major functions of CogSketch is to convert between a visual depiction and a structured, symbolic representation of the ink. It also captures and represents mouse and/or pen gestures in a way that can be communicated to a reasoning system and supports action primitives for modifying, duplicating, and deleting elements in a sketch. Supporting such programmatic control lets the sketch interface serve as the user interface for playing games.

Structurally, a sketch may contain a number of subsketches that can represent alternative views or different states in time of a depicted scene or object. The subsketches in turn consist of aligned



1: Interaction initiated by user Tom at 133152, 20130911.
2: I am going to teach you to play Tic Tac Toe.
3: Ok.
4: Tic Tac Toe is a marking game.
5: Ok.
6: Tic Tac Toe is played on a 3x3 grid of squares.
7: Ok.
8: Tic Tac Toe is a two-player game.
9: Ok.
10: X is a player.
11: Ok.

Figure 3. State of sketch and instructions prior to introducing second player 'O'.

layers (like tracing paper). Layers contain entities called *glyphs* that consist of ink segments. Glyphs also have semantic content, expressing either an entity in the depicted world or a relationship or elaboration between such entities.

When a human instructor teaches or plays a board game, he or she draws glyphs to represent the pieces or marks and the board. As these pieces are introduced and identified, the system adds them to a subsketch that we can think of as a catalog. The catalog holds prototypes of pieces or marks, but does not preserve any particular spatial arrangement between them. When objects are named, the system associates the prototype name with the glyph. Figure 3 shows the state of a sketch and instructions when the teacher is about to introduce the second player, O.

Later, when the system starts to play a learned game, it generates another subsketch to represent the current state of the game. It marks or populates the board by instantiating a prototype glyph from the catalog into a legal location on the game board.

5. Learning and Generalizing

User interactions are passed from the language and sketch agents to the reasoning agent in the form of reified *communication events*. This does two things: It preserves the relative ordering of utterances and gestures, and it supports flexible control for interactive learning and game play. The learner can respond to a request or declarative statement, or the creation, destruction, movement, or selection of a sketch entity. Moreover, it can wait for such a typed event, and keep track of the last entity created, moved, and so forth. This enables flexibility in presentation ordering and it allows implicit turn taking. There is no need for the human opponent to say "your turn", because the agent is waiting for an entity creation event from which to update the state and start its move.

The reasoning agent is responsible for translating (possibly incomplete) predicate calculus statements into operational game rules. This is a translation from one formal representation to another, so it is much more straightforward than language interpretation.

Not all game rules can be conveniently expressed through language. In our example transcript, the goal is expressed as three marks in a row. This leads to a single rule that recognizes a horizontal row. When we play the game, the instructor can introduce new rules on the fly after a player wins through a vertical or diagonal line. “I win” or “you win” signifies that a new rule should be produced by lifting the game configuration for the winning role. It consults the existing goal rule and determines that the state is a spatial configuration of the winner’s marks. Given that, it lifts the current winning state in a similar manner. In the case where there are extraneous marks that do not contribute to the goal, the instructor may explicitly select the relevant glyphs in the sketch, to avoid overfitting. Of course, many games have goals that cannot be defined so simply, checkmate in Chess being an obvious example. In such cases, a more sophisticated mixed initiative dialog will be required.

6. Annotated Example

We clarify the system’s operation by showing how Tic Tac Toe can be learned in a very small number of interchanges.[2] The instructor starts by introducing the topic:

I am going to teach you to play Tic Tac Toe.

The purpose of this is to bootstrap the interpretation context. Because TicTacToe is already a concept in the ontology, the learner knows that it is a game, so it enables the game-learning interpretation rules and instructional events. Since there is no natural language generation, the system responds with "*Ok*." to signal that it understood. We omit these responses from here on.

Next, the instructor creates a new blank sketch in which to draw the elements such as marks and the board. He then classifies the game:

Tic Tac Toe is a marking game.

This distinguishes it from a piece-moving game or a construction game and imports some general rules about marking, such as the property that marks accumulate monotonically.

Instructor draws the game board.

The system does not recognize the ink or have any expectations about what it could be, so it merely records the fact that an entity was created.

TicTacToe is played on a 3×3 grid of squares.

Now the meaning of the hash mark in the sketch becomes clear. The utterance describes a spatial configuration that is inferred to be a game board. The most recent un-interpreted sketch entity is assumed to be that board and is implicitly divided into a 3×3 Cartesian coordinate system.

Tic Tac Toe is a two-player game.

This classifies the game again and imports basic rules for turn-taking.

X is a player.

Because this is a game domain, ‘player’ is understood to mean a game role, rather than an athlete. This is another example of the abductive mechanism at work. Each domain has a set of abductive

2. A video demonstration can be accessed at https://www.youtube.com/watch?v=WX8Zo3soWSw.

preferences specified in a microtheory associated with that domain. Thus, the preferences are expressed declaratively, making them potentially easier to learn.

Instructor draws an X.

The most recently introduced role name is assigned to the new sketch entity. Tic Tac Toe is a slightly special case in that the player's roles and their marks are the same, whereas in a game like Chess, entities would correspond to pieces.

Instructor draws an O.

The O can be drawn either before or after naming it.

O is a player.

Again, this sentence defines the role and associates the mark entity with it.

X and O take turns marking empty squares.

Now that the roles have been defined, the instructor has elaborated the marking action precondition to require empty squares and turn taking.

The first player to mark three squares in a row wins.

This sentence describes one of the winning conditions – a horizontal row of marks. Rather than extend the definition of "row" to cover columns and diagonals, we leave those remaining conditions to be learned by demonstration.

X goes first.

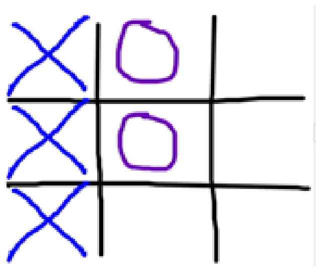
X is now understood to be one of the players, and although the sequence of turns is not explicit in the language, the expectation of turn taking primes the interpretation.

Start a game.

The imperative utterance becomes a request to play a game. Enough information has been conveyed to play a legal game.

You win.

The learner plays with one-step lookahead, but, beyond that, its actions are almost entirely random legal moves. By letting it win with a vertical column, the instructor conveys a new winning condition from which it must lift a new rule. Figure 4 shows the state of the board game



```
(<== (gdl-goal ?player 100)
   (isa ?player GameRole)
   (entityLabel ?player ?mark)
   (gdl-true (cell 3 ?y ?mark))
   (gdl-true (cell 2 ?y ?mark))
   (gdl-true (cell 1 ?y ?mark)))
```

Figure 4. Learning to win with a vertical column plus an induced rule representation.

at this point of the game, plus the rule extracted as a consequence of being told that this state is an example of a win condition.

7. Analysis

Learning by instruction presents some challenges for evaluation, because it does not happen incrementally over many trials, so there are no learning curves to show effectiveness. This is known as *one-shot* learning. Instead, we will analyze how the program and its behavior illustrates our theoretical claims. Recall that these claims are: the sufficiency of a small set of instructional events, the utility of decomposing the understanding process to multiple phases and representations, the synergy of multiple modalities and the complementarity of instruction and demonstration.

7.1 Instructional Events

Most of the expectations about games are manifested as *instructional events* in the language interpretation. These are similar to dialog acts, except that they do not drive the learner from one explicit high-level state to another, but, instead, a side effect of recognizing them is that they make abductive choices that collapse ambiguity.

Table 2 presents our catalog of instructional events for game learning indexed by the example utterance that triggers them. Note that Entity Introduction is not used in this example because Tic Tac Toe conflates roles with marks. Ambiguity reduction is a measure of the number of possible semantic interpretations of the input sentence. This can vary widely for different sentences and grammatical constructions, but shows the effectiveness of the abduction mechanism.

How will this change as we address additional games and types of games? The set of recognition rules will certainly need to grow, to support new phrasings and constructions. However, we do not expect the set of instructional events to grow significantly. We only see two

Table 2. Instructional events for teaching spatial games.

Instructional Event	Description	Utterance	Ambiguity Reduction
Topic Introduction	Set up context for future interpretation	1	20
Game Classification	Extract type of game, enable partial	2, 4	1
Role Definition	Associate names with players	5, 6	2
Entity Introduction	Associate names with marks or pieces	–	–
Action Elaboration	Describe an action type	7	432
Configuration – Spatial	Define or reference a spatial configuration	3	1584
Configuration – Temporal	Define or reference a temporal configuration	9	3
State Description	Define or reference a named state	11	1
Win Conditions	Define condition for winning the game	8	384
Directive	Request an action	10	7

additions, based on preliminary analyses: (1) describing strategies, which requires combining multiple existing event types, and (2) describing tradeoffs.

In addition to instructional events, a small set of *communication events* integrate the multiple modalities. Our set of communication events includes UserStatement, UserRequest, UserQuery, EntityAdded, EntityDeleted, EntityMoved, and EntitySelected. Although others are possible (e.g., EntityRotated), we do not expect to need them for board game domains.

7.2 Multiple Representations

In order to reconcile the conflicting requirements of general language understanding with the production of task-specific operational rules, the system incrementally translates through multiple representations. Each translation benefits from simpler and more focused contextual expectations than would otherwise be possible. Comparing the relative sizes of these representations can provide some insight into the contribution of background knowledge.

One way to measure the effectiveness of an instructional system is to measure the compression ratio. How concise can the instruction be and how much information can be conveyed by a single sentence? When the instructor says "The first player to mark three squares in a row wins", this is a very concise way of conveying a lot of information, as illustrated in Table 1.

In this case, the intermediate representation is quite verbose. We have translated from a single sentence to a conjunction of 13 expressions, and on to a Horn clause with six subexpressions. On the other hand, a sentence like "TicTacToe is a marking game" produces an interpretation with one expression, (isa TicTacToe MarkingGame), but that imports 13 expressions into the game definition (in the form of two assertions and five rules). Figure 5 presents the compression profile for the sentences in the example transcript.

As a qualitative measure, this tells us something about the instruction process. In some cases, background knowledge is applied in the translation and operationalization of the interpretation; in other cases, it is the composition of general, but operational, knowledge that is applied. We take this as support for the incremental, progressive interpretation and translation process.

Another way to look at Figure 5 is that is shows the relative contributions of prior knowledge from ResearchCyc to learned knowledge. The intermediate representation consists of Cyc concepts and predicates, while the target representations are the expressions constituting GDL

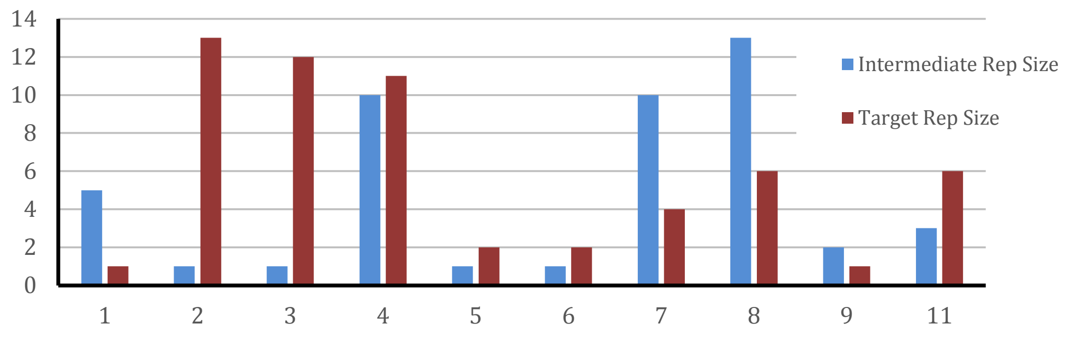


Figure 5. A comparison of representation sizes (number of conjuncts) by utterance number.

rules and properties. The final learned definition of Tic Tac Toe contains 12 rules, ten initialization statements, two numeric parameters and three entity names.

7.3 Multimodal Synergy

The example illustrates two ways in which verbal and graphical interaction are complementary. First, the introduction of a mark through both language and sketching creates a correspondence mapping between the name and the visual entity. That mapping also implies that the concrete visual entity represents a prototype shape rather than a particular instance in a specific location. Second, in demonstrating a winning configuration, the sketch *does* represent a concrete situation that is lifted to a general rule. The verbal and visual modalities constrain each other by influencing what is to be interpreted as a class or an instance.

7.4 Complementarity of Instruction and Demonstration

The example shows one way that learning by instruction and demonstration can work together: creating new goal rules by explaining and lifting demonstrated win configurations. This reflects our intuition that it is often easier to show than it is to tell. Yet, in its current form, this is a fairly rudimentary capability. The instructor signals a change in mode by starting a game or declaring a win or loss. Natural dialogs often switch spontaneously and seamlessly between instruction and demonstration and rely on the learner to recognize the transition.

7.5 Learning Hexapawn

In order to better understand the generality of our approach, we applied our system to learning the game of Hexapawn. This is a game that is similar in size and complexity to Tic Tac Toe, but involves moving pawns on a 3×3 board.

Rather than require the instructor to draw pawns, we instead created a sketch with an initial catalog of glyphs by importing vector graphics, as shown in Figure 6. This resulted in fewer deictic references and virtually no gestures during the instructional phase. This would need to change if we were to learn spatial relations or strategic concepts by example, or to explain how a knight moves or castling in Chess.

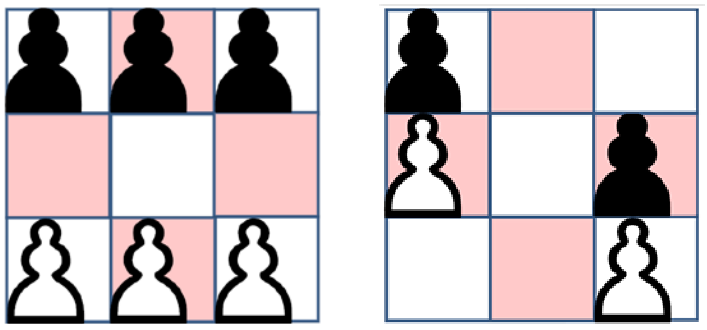


Figure 6. Hexapawn initial state and win configuration.

The main difference between Tic Tac Toe and Hexapawn is that that latter is a piece-moving game. This implies some background knowledge about the persistence of pieces, such as that movement removes a piece from its original location and places it at the destination. We can assume by default that a location only holds a single piece, although this can be easily overridden through instruction. Another difference is that while Tic Tac Toe conflates the player with his mark, there is a definite distinction in a piece moving game between a player and his or her pieces. The representation must support that level of indirection, and allow for the casual use of a piece to stand in for a player and vice versa. A third difference is that a piece moving game does not necessarily start out with an empty board. It was necessary to recognize references to the initial state and translate that into rules for populating the board.

In terms of the process of instruction, the two games are very similar. We are able to teach the rules of Hexapawn in 16 sentences. We introduce players, pieces, and game configurations in much the same way, but assume prior knowledge of a few more spatial relations than were necessary for Tic Tac Toe. For example, one instruction is "White may move wp diagonally up to capture bp." *Up, down, diagonal,* and *across* are all adjectives (and sometimes adverbs) that we operationalize directly in terms of grid coordinates. Clearly, these are concepts that could be induced through examples, but our purpose is not to start with a tabula rasa, but to find an appropriate level of discourse that makes instruction feasible and effective.

The ambiguity reduction in the Hexapawn transcript is similar to that of Tic Tac Toe. The number of possible interpretations of the input sentences typically ranges from one to 20, with outliers at 198 and 2160 for the sentence "The first player to move a piece across the board wins." As before, knowledge eliminates this ambiguity and lets the system proceed.

8. Related Work

The idea of an instructable machine has a long history in Artificial Intelligence. One early expression of this was the Teachable Language Comprehender (Quillian, 1969), a learning-by-reading system designed to assimilate factual statements from text into a persistent semantic memory. As such, it explored the role of background knowledge as an expectation context for language understanding. It did not have contextual expectations from a concrete task or from multimodal interaction.

The Understand program was another early instructable system (Hayes & Simon, 1974). That program took a natural language description of a task, in this case a Japanese tea ceremony that is isomorphic to the Towers of Hanoi, and produced a formal representation of the problem and problem space that could, in principle, be supplied to the General Problem Solver. This idea of interpreting language by looking for expected elements of a task representation is a key aspect of our current work. Today, however, with the availability of very large ontologies and lexicons, issues of ambiguity, subtle semantic differences, and scaling become more acute. As a consequence, our interpretation mechanism uses backward chaining abduction to fill in missing assumptions, and splits the translation process into multiple phases. These phases more clearly distinguish between assimilating declarative knowledge about the game and *operationalizing* that knowledge into executable form. We also close the feedback loop in order to generalize the learned knowledge through demonstration.

The Foo program introduced the concept of operationalizing advice (Mostow, 1983). Its task was to translate formally represented advice in the domain of playing Hearts into more concrete search preferences or constraints for playing the game. For example, advice to avoid taking points during the trick was reformulated and mapped onto a general heuristic search method. Operationalization is also an important component of our approach, although the input is not advice and the output is not a search procedure. When assimilating statements and sketch gestures, the learner converts the predicate calculus input into game rules that define the game as a state machine.

Broadly stated, we are taking a collaborative problem solving approach to instruction (Allen et al., 2002). Our task and approach have a different emphasis than PLOW (Allen et al., 2007), which also uses this approach. Both do one-shot learning, but PLOW uses language more as a running commentary on a demonstration than as instruction. In our case, demonstration is treated as a repair to instruction. Our architecture is less stratified than Allen et al.'s because there is no grammar of higher-level states through which the learner passes. Our reified communication events that assimilate statements and glyphs are similar to the communicative acts in his account.

Several prior systems have been built that learn Tic Tac Toe from purely visual inputs, either generated by a person manipulating a physical game board (Kaiser, 2012) or from a robot manipulating a specially engineered game board (Barbu, Narayanaswamy, & Siskind, 2010). Both of these systems require substantially more input data than ours to start playing the game. This is the advantage of using natural language to concisely and quickly communicate basic concepts about the game. On the other hand, communicating the full range of winning conditions is something that can conveniently be done via demonstration.

Prior work on multimodal interfaces has mostly focused on providing more natural interfaces to legacy computer systems. For example, QuickSet (Cohen et al., 1997) combined ink recognition and speech recognition to achieve higher accuracy by selecting semantically compatible recognition hypotheses from the best-N lists produced by each modality. We use typed input to factor out problems with speech recognition, and CogSketch's open-domain approach means that ink recognition is unnecessary. Much like human to human sketching, we are using language here to provide conceptual interpretations for what is sketched.

The approach that is most similar and complementary to ours is being pursued by the Soar group at the University of Michigan (Kirk & Laird, 2014). They are also teaching games, including Tic Tac Toe among others, through language, demonstration, and spatial interaction, in this case using a physical robot. Our sketch interface lets us avoid many of the difficult perceptual problems and focuses our efforts in different areas. The Soar project has emphasized spatial preposition learning early on, while we have deferred that in favor of starting with some basic prepositions and seeing what can be expressed with that vocabulary. (In prior work by Lockwood et al. (2008), sketched input was used with analogical generalization to learn spatial prepositions of contact in English and in Dutch from sketches, so we believe this combination of modalities could be used to model language learning as well.) We have also not pursued natural language generation, so there is less mixed-initiative dialog.

9. Limitations and Future Work

Our current implementation has a number of limitations that can be broadly classified into language, learning, and scaling issues. Although it sounds natural, the language interaction is not especially robust. Seemingly small changes in phrasing can cause parsing or interpretation to fail. While this is an active area of research for us, it is not the focus of this project in particular. One positive effect of the abductive interpretation process is that it is mostly additive. That is, additional knowledge does not break existing interpretation capabilities.

Our learner does not currently induce new intermediate predicates. This has not been an issue with the simple games we have addressed so far, but we will probably need to borrow some techniques from inductive logic programming (Muggleton & De Raedt, 1994) before attempting to learn more complex games.

We do not yet have a facility for repairing incorrectly learned rules through language, although there does not seem to be a deep theoretical impediment to this. We address the problem of over-fitting by letting the instructor select entities on the board to designate the relevant part of a winning configuration. This is only necessary when it is ambiguous, and the technique might need to be extended for more sophisticated games.

Scaling is one reason we have not yet attempted to learn the rules of Chess. The sheer number of legal moves makes it unpleasantly slow to play through the backchaining reasoner, and the sketch understanding system automatically computes qualitative spatial relations that slows significantly when there are 32 pieces on the board. Neither of these are intrinsic impediments. More control over spatial inference and compiling rules to a more efficient form for execution would make Chess a reasonable domain to learn. In the meantime, we intend to pursue subproblems, such as learning how different pieces move and learning endgame strategies.

In addition to teaching other games and game strategies, we are also looking at teaching physical reasoning tasks through multimodal interaction. Our main research goal is to identify instructional techniques that are amenable to such instruction. A prime candidate is analogy; this is one of the core capabilities of the Companions architecture, but, so far, this project has not used it at all. Nevertheless, we see a large role for it in bringing across concepts and action models from one game to another. While our current implementation uses inheritance to import fragments from abstract game types such as MarkingGame, transfer from one concrete game to another could greatly simplify instruction.

Abduction has proven to be an extremely powerful tool, so much so that we are currently re-implementing the facility to embed it more broadly in our reasoning engine and to support more general weighted abduction. We expect this to improve the flexibility of the language interpretation system and to be useful in other kinds of problem solving, such as plan recognition.

10. Conclusions

We have demonstrated one way to produce an executable program through informal language and sketching. We have found this combination to be a good medium for teaching board games. Our main purpose is to identify kinds of knowledge and expectations that help enable instruction, while being general enough to not presuppose too much knowledge of the domain.

The implemented system supports our earlier claims in a number of ways. We presented ten instructional events that constrain interpretation in the domain of learning simple board games. Translation from English into a conceptual representation and then on to more procedural rules allowed contextual expectations to be at the level of concepts and examples, rather than concrete operational rules. Propositional sketch representations grounded verbal abstractions, especially during the process of lifting winning configurations to goal rules. This might be termed *synergy of modality*. Finally, instruction is complemented by demonstration when we provided feedback to the learner during gameplay. This might be termed *synergy of instruction*. The result is a dialog that we believe to be a natural kind of instruction, in which we were able to convey twelve operational rules in only ten natural language utterances.

One reason for focusing on learning the game specification rather than strategies for playing well is that, once you have the basic rules for playing, everything else is optimization. It may well be the case that optimization (construed broadly) is an ability that itself is amenable to instruction. This leads to the obvious open question: Can the effectiveness of instruction reach the break-even point such that more and more background knowledge can be learned with less and less effort? We expect it will, but that is an empirical question to be explored in future research.

Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under contract D12AP00026. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the DARPA or the U.S. Government.

References

Allen, J., Blaylock, N., & Ferguson, G. (2002). A problem solving model for collaborative agents. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*. Bologna, Italy: ACM Press.

Allen, J., Chambers, N., Ferguson, G., Galescu, L., Jung, H., Swift, M., & Taysom, W. (2007). PLOW: A collaborative task learning agent. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence* (pp. 1514–1519). Vancouver, BC: AAAI Press.

Barbu, A., Narayanaswamy, S., & Siskind, J. (2010). Learning physically-instantiated game play through visual observation. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation* (pp. 1879–1886). Anchorage, AK: IEEE.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., & Clow, J. (1997). QuickSet: Multimodal interaction for distributed applications. *Proceedings of the Fifth ACM International Conference on Multimedia* (pp. 31–40). Seattle, WA: ACM Press.

Forbus, K., Hinrichs, T., de Kleer, J., & Usher, J. (2010). FIRE: Infrastructure for experience-based systems with common sense. *Proceedings of the AAAI Fall Symposium on Commonsense Knowledge*. Arlington, VA: AAAI Press.

Forbus, K., Klenk, M., & Hinrichs, T. (2009). Companion cognitive systems: Design goals and lessons learned so far. *IEEE Intelligent Systems*, *24,* 36–46.

Forbus, K., Usher, J., Lovett, A., & Wetzel, J. (2011). CogSketch: Sketch understanding for cognitive science research and for education*.* *Topics in Cognitive Science*, *3,* 648–666.

Genesereth, M., & Thielscher, M. (2014). *General game playing*. San Rafael, CA: Morgan & Claypool Publishers.

Hayes, J. R., & Simon, H. A. (1974). Understanding written problem instructions. In L. W. Gregg (Ed.), *Knowledge and cognition.* Hillsdale, NJ: Erlbaum Associates, 167-200.

Hobbs, J., Stickel, M., Appelt, D., & Martin, P. (1990). Interpretation as abduction. *Artificial Intelligence*, *63,* 69–142.

Kaiser, L. (2012). Learning games from videos guided by descriptive complexity. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence* (pp. 963–969). Toronto, CA: AAAI Press.

Kirk, J., & Laird, J. (2014). Interactive task learning for simple games. *Advances in Cognitive Systems, 3*, 11–28.

Lockwood, K., Lovett, A., & Forbus, K. (2008). Automatic Classification of Containment and Support Relations in English and Dutch. In C. Freksa, N. Newcombe, P. Gardenfors, & S. Wölfl, (Eds.), *Spatial Cognition VI: Learning, reasoning, and talking about space*. Berlin: Springer-Verlag, 283–294.

Mostow, D. J. (1983). Machine transformation of advice into a heuristic search procedure. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach.* Los Altos, CA: Morgan Kaufmann, 367-403.

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic and Algebraic Programming, 19/20*, 629–679.

Quillian, M. R. (1969). The teachable language comprehender: A simulation program and theory of language. *Communications of the ACM, 12,* 459–476.

Tomai, E. (2009). *A pragmatic approach to computational narrative understanding* (Technical Report Number NWU-EECS-09-17). Doctoral dissertation, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL.