

Protocols from perceptual observations

Chris J. Needham^{*}, Paulo E. Santos¹, Derek R. Magee,
Vincent Devin², David C. Hogg and Anthony G. Cohn

School of Computing, University of Leeds, Leeds, LS2 9JT, UK.

Abstract

This paper presents a cognitive vision system capable of autonomously learning protocols from perceptual observations of dynamic scenes. The work is motivated by the aim of creating a synthetic agent that can observe a scene containing interactions between unknown objects and agents, and learn models of these sufficient to act in accordance with the implicit protocols present in the scene. Discrete concepts (utterances and object properties), and temporal protocols involving these concepts, are learned in an unsupervised manner from continuous sensor input alone. Crucial to this learning process are methods for spatio-temporal attention applied to the audio and visual sensor data. These identify subsets of the sensor data relating to discrete concepts. Clustering within continuous feature spaces is used to learn object property and utterance models from processed sensor data, forming a symbolic description. The PROGOL Inductive Logic Programming system is subsequently used to learn symbolic models of the temporal protocols presented in the presence of noise and over-representation in the symbolic data input to it. The models learned are used to drive a synthetic agent that can interact with the world in a semi-natural way. The system has been evaluated in the domain of table-top game playing and has been shown to be successful at learning protocol behaviours in such real-world audio-visual environments.

Key words: cognitive vision, autonomous learning, unsupervised clustering, symbol grounding, inductive logic programming, spatio-temporal reasoning

^{*} Corresponding author.

Email address: chrisn@comp.leeds.ac.uk (Chris J. Needham).

¹ Paulo Santos is now at Centro Universitário da FEI, Sao Paulo, Brazil.

² Vincent Devin is now at France Telecom R&D, Meylan, France.

1 Introduction

This paper presents a cognitive vision system capable of autonomously learning protocols involving rudimentary language and visual objects. In this system, models of visual objects and utterances are obtained from unsupervised statistical learning algorithms. In order to form symbolic data for input into an inductive logic programming (ILP) system, the continuous perceptual observations are transformed using the models learned. Perceptual observations are taken to be any sensory input; here acoustic and visual inputs are used. The concept of *qualitative time* is introduced, since only key frames are deemed to be of importance. The direct link between perception and action is exploited; a change in what is perceived can only be brought about by an action. The ILP system is used to construct sets of definite clauses that express rules of behaviour for the perceived actions from the symbolic description of the scenes. In particular, the protocols learned encode the connection between utterances with visual objects that occur in the scene. This is intrinsically a solution to the anchoring problem [8] which is an instance of the symbol grounding problem [15]. The sets of definite clauses obtained are further used by a synthetic agent to perform actions in the world. Therefore, in this work we explore closing the loop between learning the connection between perception and action via bridging the gap between computer vision, pattern recognition and symbolic knowledge discovery.

The framework presented below is evaluated in the domain of simple table-top games. In this domain the system was able to learn complete protocols for the rules of the games, when different verbal utterances are made and also some aspects of the dynamics involved in playing the game (for instance, when objects should be placed on the table). The entire learning process is executed with minimal human intervention and assumes minimal domain specific knowledge of the scenes or of game concepts. In earlier work [22], we demonstrated that the same approach is capable of learning simple mathematical concepts such as numerical ordering and equality.

1.1 *The domain of table-top games*

We have chosen to work in the domain of simple table-top games involving interaction of one or two players with a small number of visual objects and incorporating spoken utterances. The reason for choosing such scenarios is that games contain rich protocols and their ‘complexity’ can be controlled by adding, excluding or modifying rules, actions and/or objects in the domain. Moreover, it may be argued that many real-world social-interaction scenarios may be modelled as games [14], which suggests that our framework may be

relevant to the development of a fully autonomous system that could learn how to behave in the real world.

Experimental data is collected using two standard PCs, two webcams, and a microphone in the arrangement shown in Figure 1. The games used in this work are described in greater detail in Section 4. Briefly, the following three games are played:

- (1) SNAP1. A generalised game of snap where two cards are played simultaneously, followed by an utterance dependent upon the figures on the cards. The utterances are either “colour” (when only the colours in the figures are the same), “shape” (when only the shapes in the figures are the same), “same” (when shape and colour match) or “nothing” (if no feature in the figures match to each other). The cards are removed, and “play” is then uttered indicating to play the next two cards.
- (2) SNAP2. A variation of the above game where cards are placed one on top of another and the resulting utterance is dependent upon the card which is visible, and the card at the previous time step.
- (3) Paper-scissors-stone (PSS). Played with two sets of three cards each depicting one of paper, scissors or stone. Two players simultaneously select one of the object cards and place them on the table. When the two figures in the cards are perceived, utterances “win”, “lose” and “draw” are spoken by one of the players (the one to be simulated by the synthetic agent). This player says “win” when its card *beats* the one shown by the other player - *paper beats (wraps) stone, scissors beats (cuts) paper, and stone beats (blunts) scissors*. A “play” is uttered when there are no cards on the table.

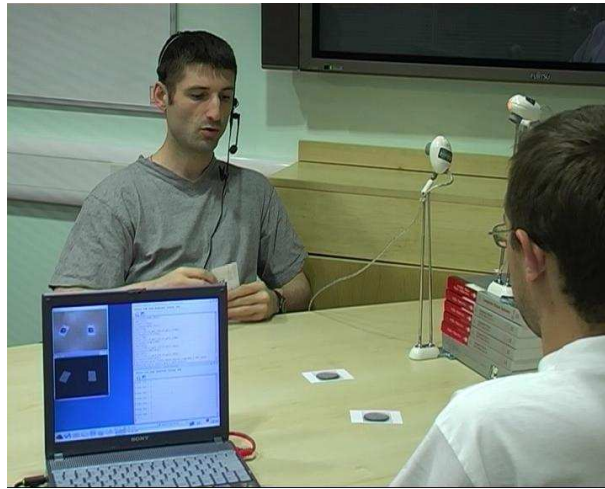


Fig. 1. Example of the data collection phase. A game is played on the table between two players. One camera points down at the table, and another captures the face of the participant to be replaced by the synthetic agent. The audio is captured by a microphone worn by this participant.

An overview of the framework is illustrated in Figure 2. Firstly, *attention mechanisms* are necessary to pick out salient (interesting) sounds, objects and features from the audio and visual input streams obtained in a setup similar to that shown in Figure 1.

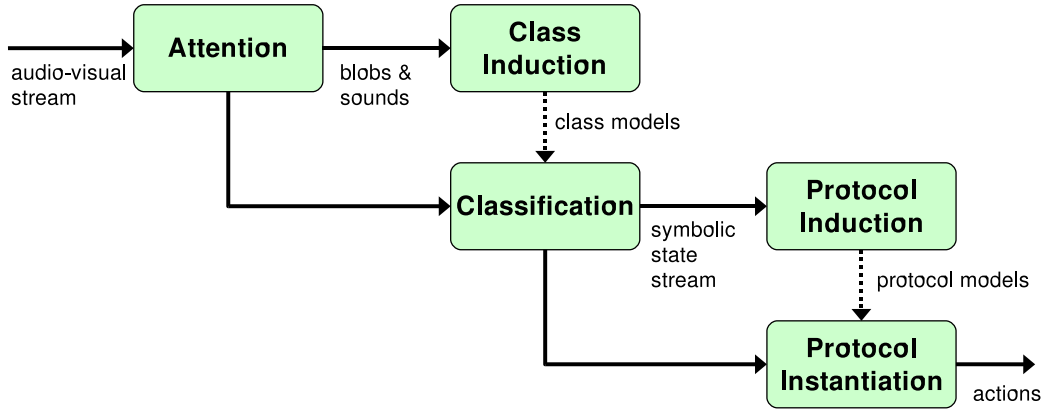


Fig. 2. Overview of the learning framework for the synthetic agent.

There are two phases of operation of our system: training and execution.

- In the *training* (or learning) phase, the synthetic agent observes the world, without participating. In this phase, *class induction* is performed on the blobs and sounds that are perceived, and *class models* are formed, which are used for classification of all the perceptual objects that have been seen in training. These are used to form a *symbolic data* description of the input signals. *Protocol induction* is then performed on this symbolic data stream, from which a set of *protocol models* (or rules) is formed.
- In the *execution* (or play) phase, the synthetic agent participates in games using the learned protocol models. An inference engine is used for *protocol instantiation* to infer the synthetic agent’s (vocal) response to the current symbolic description of the world.

We assume no knowledge about the type of objects or sounds presented to the system. Therefore, unsupervised clustering of both the audio and visual feature vectors is performed. Models are learned, based on this clustering, which can classify the perceptual inputs into classes. This provides a *symbolic* description of the input signals.

Protocol models are represented as ordered sets of definite clauses expressed in Prolog syntax. This provides the necessary flexibility to represent the relations between objects and actions that we require. Others have previously demonstrated the utility of (subsets of) first-order predicate logic in high-level scene interpretation (e.g. Neumann and Weiss [30]).

Inductive Logic Programming (ILP) in the form of PROGOL [28] is used for protocol induction. The reason for choosing PROGOL resides in its capability to construct theories from only (possibly noisy) positive examples. This coincides with our aim to learn protocol behaviour from observation in an unsupervised way. Moreover, ILP enables *concept generalisation* (e.g. to extend rules learned from observation of certain objects to unseen entities) and for the knowledge learned to be presented in a format that allows us to assess its ‘complexity’ and accuracy, besides serving as tools for further reasoning - the symbolic theories obtained are used in this work to construct equivalence classes between utterances in order to cope with over-clustering of the utterances in the audio signal.

Once learning is complete, the synthetic agent can process the perceptual inputs and infer the appropriate action response to the state of the world in real-time. The interactive agent grounds learned perceptual models and learned protocols to descriptions of objects in the real world, and plays back a suitable utterance.

The experiments reported in this paper are evaluated using new definitions of *soundness* and *completeness* that are inspired by their homonyms used to assess the semantics of inference systems. Soundness accounts for the extent to which the intended protocol (intended semantics) is represented by the rules obtained. In contrast, completeness measures how many of the rules found do not agree with the intended semantics.

The remainder of this paper is structured as follows: Section 2 reviews previous work on learning from audio-visual input. Section 3 details the system, discussing the application of the ideas presented above. Section 4 describes the methods used for evaluation, before presenting the experiments and results in detail. Discussion of our approach is presented in Section 5 alongside future research directions, before conclusions are drawn in Section 6.

2 Background

Most previous work on modelling the connection of language to the world through vision has not included the capacity for learning. Typically the aim has been to design a sufficiently rich conceptual framework with which to characterise the activities in a target domain [4,20,41]. Nagel [29] for example uses motion verbs as a conceptual framework to mediate between video sequences depicting traffic scenes and natural language descriptions of those scenes. More recently, Siskind has proposed the use of an event logic to map from visual input to single motion verbs such as *pick up*, *stack* and *move* [38].

A key issue has been the scalability of conceptual frameworks designed by hand to go beyond a prototype domain. This has motivated the use of learning procedures to populate a conceptual framework automatically through generalisation from extended observation of a target audio-visual (or language+visual) domain. Cangelosi and Parisi [6] adopt this approach within a perception-action setting in which visual input and a language command together determine an action such as the movement of a robotic arm. By representing this mapping as a feed-forward neural network, a standard learning procedure is used to set the parameters of the network from examples of visual and language inputs paired with the resulting actions.

For a simpler domain involving only visual and language input, Barnard *et al.* [5] model the relationship between features of image regions and single words naming those regions (e.g. *sky*, *water*) as a joint probability distribution. This distribution is learnt using an EM algorithm and can subsequently be used for various purposes through probabilistic inference, for example generating the conditional distribution for the name of a given region.

Roy and Pentland [35] demonstrate the use of mutual information between audio and video streams to segment words from a corpus containing phoneme sequences associated with simultaneous visual situations. The corpus is a video and sound recording of care-givers interacting with infants playing with a variety of objects. The system works from raw speech utterances and produces phoneme sequences using a trained recurrent neural network and hidden Markov model.

Our work addresses the general problem of unsupervised learning of game protocols from a corpus of raw visual and acoustic data depicting target scenarios. A problem not tackled in earlier work is the likely generation of too many or too few acoustic or visual object classes during unsupervised learning. With few exceptions (e.g. Roy and Pentland [35]), either an ideal categorisation is imposed by supervised learning or by prior design, or early processing is by-passed altogether and replaced by symbols denoting the acoustic or visual objects.

If during unsupervised learning too few classes are generated, it will be impossible to learn structural concepts that depend on a more refined categorisation. If too many classes are generated and this is uncorrected, redundant concepts will be generated which although identical in structure involve distinguished but semantically equivalent objects - this will impede learning as training examples will be required for each separate instance. A related problem is the misclassification of objects.

Within a learning context, earlier work integrating language with vision has addressed a number of audio-visual tasks involving different levels of language

complexity. At its simplest, there is the task of naming visual objects, involving single words in isolation [5]. At the other end of the spectrum, Roy [34] addresses the task of describing objects using noun phrases which optionally contain a shorter constituent noun phrase (e.g. the pink square; the green rectangle below the peach rectangle). From word sequences segmented from the acoustic signal, the joint probability distribution of consecutive words (bigrams) is estimated from relative frequencies and used in turn to generate legal utterances describing novel visual situations. This bigram model is equivalent to a stochastic regular grammar and could in principle be extended to a higher order language model using instead a stochastic context free grammar, although learning the grammar rules would then be more challenging.

Interestingly, stochastic grammars have also been used to model visual activities [2,17,25] when these can be characterised as a temporal sequence of visual primitives. However, they are not ideally suited to many other visual modelling tasks which involve essentially non-sequential data, for example the spatial relationships in a single visual image.

One way forward for the learning paradigm is to adopt a representational formalism that is sufficiently powerful to model the concepts of a target domain, such as the event logic used by Fern *et al.* [11], and then attempt to learn the necessary background knowledge connecting language and vision within this formalism. In our work, we have adopted logic programs expressed in Prolog syntax as our high-level formalism, and inverse entailment embodied in the PROGOL system as the learning procedure. A limitation of Prolog in this context is the absence of a mechanism for representing uncertainty, for example in the spatial relationship between two objects or the action taken in a given situation. Recent proposals for combining predicate logic with probabilistic inference could overcome this limitation, assuming equivalent inductive inference procedures can be developed [33]. For example, Markov networks have been used successfully for learning and reasoning about spatial and temporal uncertainties in physical situations that can be represented by a fixed conjunction of binary predicates [16].

PROGOL infers sets of Horn clauses using a statistical optimisation procedure and thereby deals with occasional misclassified objects. Like Roy [34], we work directly from raw acoustic and visual data, although our language domain contains only single word utterances. A key challenge for the future will be to automate learning of richer conceptual descriptions, such as those that are currently hand-crafted (e.g. Nagel [29]).

3 Learning perceptual categories and symbolic protocols

In this section we present the details of our implemented system, whereby models of visual objects and utterances, as well as a symbolic description of the protocol, are learned in a purely unsupervised manner. Figure 3 illustrates the overall learning scheme.

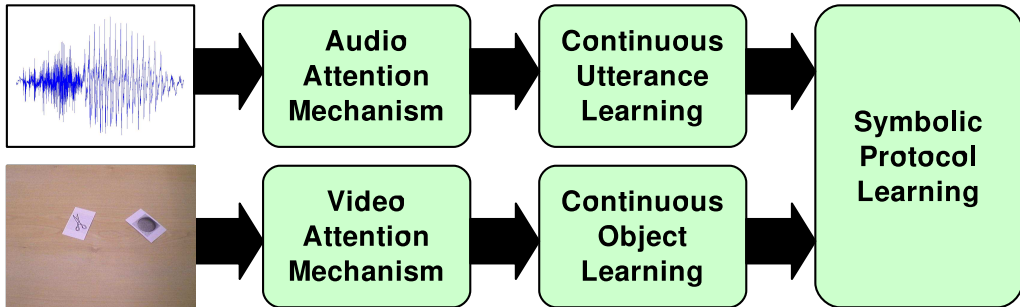


Fig. 3. Overview of the learning framework.

3.1 From continuous to symbolic data

Video streams of dynamic scenes contain huge quantities of data, much of which may be irrelevant to scene learning and interpretation. An attention mechanism is required to identify ‘interesting’ parts of the stream, in terms of spatial location (‘where’) and temporal location (‘when’). For autonomous learning, models or heuristics are required to determine what is of interest, and what is not. Such models could be based on motion, novelty, high (or low) degree of spatial variation, or a number of other factors. It is highly likely that no single factor could provide a generic attention mechanism for learning and interpretation in all scenarios. It is our view that attention from multiple cues is required for fully generic learning. This section details the attention mechanisms used to segment the continuous audio and video streams, and describes the algorithms used to classify the features extracted from the input signals in order to create symbolic data for protocol learning. Our aim has been to develop methods that are sufficiently robust to handle environments outside the laboratory.

3.1.1 Attention, learning and classification for visual objects

For our implementation in the game-playing domain, an attention mechanism which can identify salient areas of space and time is necessary. We have chosen motion as it is straight-forward to work with. The spatial aspect of our attention mechanism is based around a generic blob tracker [23] that works on

the principle of multi-modal (Gaussian mixture) background modelling, and foreground pixel grouping. This identifies the centroid location, bounding box and pixel segmentation of any separable moving object in the scene in each frame of the video sequence. The temporal aspect of our attention mechanism identifies key-frames where there is qualitatively zero motion for a number of frames (typically three), which are preceded by a number of frames (typically three) containing significant motion. Motion is defined as a change in any object’s centroid or bounding box above a threshold value (three standard deviations of tracker positional noise, typically five pixels in our experiments). This method for temporal attention is based on the domain limiting assumption that all objects remain motionless following a change in state (and that the process of change itself is not important). Figure 4 shows an example sequence of key-frames identified by the attention mechanism.

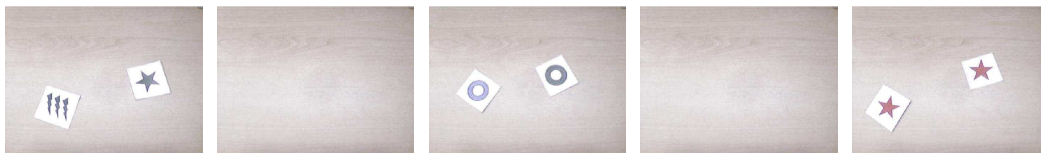


Fig. 4. Sequence of key-frames identified by the spatio-temporal attention mechanism. Intermediate frames where the objects are in motion, or subsequently remain stationary, are filtered out by the attention mechanism.

Features of the objects identified by the blob tracker (at the key-frames identified by the attention mechanism) are extracted and clustered. Currently three groups of features are extracted, corresponding to the attributes of texture, colour and position. The texture feature is used to classify global shape (e.g. a star) in addition to conventional textures. The remainder of this section discusses the methods used for unsupervised clustering and formation of classifiers for high dimensional feature vectors, describing colour and texture, which could indeed be applied to many other features that we may wish to use, for example global shape, local shape, or local textures. For our experiments, position is divided into two classes: `pos0` for objects on the left of the image, and `pos1` for objects on the right of the image. This has been done for simplicity (as it suffices for the domains in question and demonstrates that spatial position may be incorporated in the subsequent symbolic learning process). Position could be learned in a similar way to the other groups of features.

Rotationally invariant texture features

The texture feature set used in this paper was based on the Gabor wavelet [9] and other convolution based textural descriptions (Figure 5). These were applied at the object centroid provided by the object tracker, to form a 94 dimensional feature vector. To make these features rotationally invariant (where they were not already) sets of features at multiple orientations were applied to the

images and the statistics (mean, minimum, maximum and maximum/mean) of these feature sub-sets used as the feature description, rather than the raw values. The features used were: Gabor wavelets (various scales), equal covariance Gaussians (various scales), non-equal covariance Gaussians (various scales) and polar wavelet-like features defined by convolution masks of the form:

$$K(r, \theta) = G(r, \sigma)\sin(K_1\theta + K_2), \quad (1)$$

where r is the distance of a pixel in the mask from the object centroid, θ the angle that a line from the object centroid to a pixel in the mask subtends with the horizontal axis, $G(r, \sigma)$ is a Gaussian distribution with standard deviation σ (centred on the object centroid), K_1 relates to angular frequency ($K_1 = 1, 2$ or 4 in our experiments) and K_2 relates to the filter orientation (typically four values of K_2 were chosen in order to form a bank of filters). Convolution kernels of the form in Equation 1 were applied at various scales (by altering σ). The scales chosen for all feature descriptor types were selected as (approximately) evenly distributed between sensible limits, so as to be as general as possible.

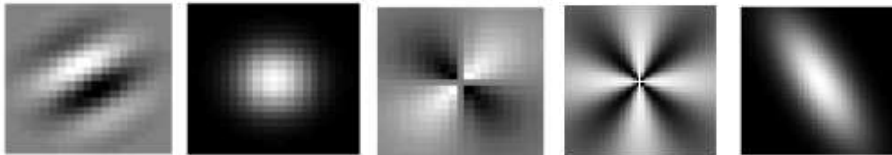


Fig. 5. Example wavelet and convolution based textural feature descriptors used.

Colour features

The colour feature set used in this paper consists of the bins of a 5×5 Hue-Saturation (HS) colour histogram. The blob tracker used as part of the attention mechanism provides a binary pixel mask defining the pixels belonging to each object identified. The (RGB) colour values of each of these pixel is projected into Hue-Saturation-Intensity (HSI) colour-space [39] and the Hue and Saturation (Intensity independent) dimensions of this space used to build a 2D histogram of the object pixel colour distribution. Each dimension is quantised into 5 levels (a compromise between the generality and specificity of the representation) to give a histogram with 25 bins (used as a 25 dimensional feature description).

Clustering and feature selection

For each attribute (texture/colour) we wish to form clusters of features vectors which define the different classes denoted by attribute labels. Since our system has no knowledge of which class each feature description belongs to, an unsupervised clustering approach must be taken. There are a number of methods

in the literature for unsupervised clustering such as K-means [13], agglomerative methods [1,37] and graph partitioning based methods [18]. Such methods generally work by associating similar data items in an n -dimensional feature space. All features are used in this clustering process and feature selection is usually seen as a supervised pre-processing step (if performed at all). An alternative approach is to combine the results of multiple clusterings (either multiple methods on a single feature set or a single method on multiple data sets) [40].

In the high dimensional feature vectors (texture responses from a variety of convolution masks and colour histogram bins), only a subset of the features will be important in the task of classifying the visual objects. More importantly, different descriptors will be important in different scenarios, depending upon the visual objects used. It is for this reason that a variety of textural descriptors are used to form the 94 dimensional texture feature vector. We use a feature selection method based on agreement between features to select the most important. Firstly, clustering is performed independently in each dimension of the feature space (1D sub-spaces) in order to induce N_c clusterings of the feature vectors. These clusterings are combined to form weights for edges in a graph with data items as nodes. This graph is then partitioned to form clusters of data items. Finally, supervised learning of a classifier is performed, using the cluster membership of data items from the graph partitioning.

We use agglomerative clustering on each sub-space, based loosely on that presented in [1]. To begin with, the data in each sub-space is normalised so that it has zero mean and unit standard deviation. This is performed so the same stopping criterion (below) for the sub-space clustering can be applied across all sub-spaces. One cluster per data item is initialised. For all cluster pairs C_i, C_j (containing N_i and N_j members respectively) we calculate $D(C_i, C_j)$: the mean distance between each member of C_i and each member of C_j .

$$D(C_i, C_j) = \frac{1}{N_i N_j} \sum_{s \in C_i} \sum_{t \in C_j} |s - t| \quad (2)$$

Iteratively the closest two clusters C_i and C_j are merged whilst $\min(D(C_i, C_j)) < T$, where T is a fixed threshold. The value of $T = K\sqrt{d}$, where d is the dimensionality of the sub-space (i.e. 1) and K is a constant (1 in all our experiments). This agglomerative clustering is performed on all 1D sub-spaces and the results of these clusterings are combined using a novel weighted version of the cluster-based similarity partitioning algorithm of Strehl and Ghosh [40]. In the original algorithm, a (non-directional, fully connected) weighted graph is formed where the vertices (unweighted) represent data items and the (weighted) edges represent the similarity between pairs of data items. Similarity is measured as the number of times the pair of data items occur in the same cluster over all sub-space clusterings. This graph is then partitioned

using a graph partitioning algorithm such as [18]³. Such algorithms attempt to form a set of sub-graphs, such that the edge cut required is minimised. This is an NP complete problem; however there are a number of suitable approximate methods at our disposal. We extend the method in [40] to form a graph in which the similarity is re-defined as the sum of the weights relating to each sub-space in which data items occur in the same cluster (Equation 3). In this way the relative importance of the individual features in our feature description can be weighted:

$$EdgeWeight(V_a, V_b) = \frac{\sum_{c=1}^{N_c} W_c S_c(V_a, V_b)}{\sum_{c=1}^{N_c} W_c} \quad (3)$$

where W_c is the weight associated with clustering c and $S_c(V_a, V_b)$ is 1 if the data items relating to vertices V_a and V_b are contained within the same cluster for clustering c , and 0 otherwise. N_c is the number of clusterings used. It should be noted that the denominator is simply a constant and, as such, has no effect on the clustering produced. However, the inclusion of this normalisation enables edge weights (and thus edge cuts) to be compared across different feature sets and weightings.

Once the initial clustering has been performed (with equal weights) the discriminative power of the individual sub-spaces used to build the clustering can be evaluated with respect to this clustering (i.e. assuming this clustering is “correct”). If a sub-space can discriminate well between any two classes then its discrimination power is good, as it is unusual for a low dimensional sub-space (1D in our experiments) to be able to discriminate between all classes. We re-weight the edges of the graph used to partition the data into clusters, to reflect how “correct” the current (in this case initial) clustering is. This involves evaluating how well each pair of clusters is discriminated between in each sub-space.

To do this, we consider the accuracy of a two class classifier based on a particular sub-space clustering (C_1, C_2 , etc.). If a class labelled dataset is available, such a classifier may be formed from the statistics of the associations of classes to clusters, as in Equation 4.

$$P(class|C_i) = \frac{|\{X : X \in class \wedge X \in C_i\}|}{|\{Y : Y \in C_i\}|} \quad (4)$$

However, class membership is not known in our case. If the clustering obtained from the graph partitioning method is assumed to be the “correct” class labelling, such a classifier may be formed. Figure 6 illustrates the theoretical accuracy of such a classifier as a Bayesian network.

³ Implementations of several graph partitioning algorithms are freely available to download from <http://www-users.cs.umn.edu/~karypis>, as part of the METIS and CLUTO packages.

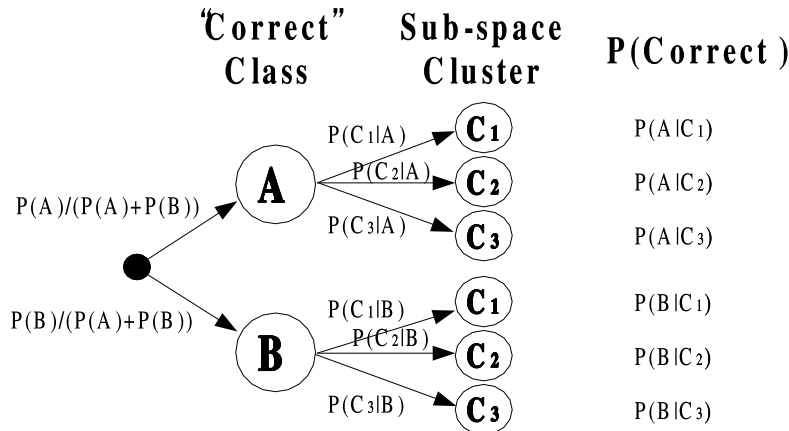


Fig. 6. Accuracy of a stochastic classifier for two class discrimination.

In Figure 6, the diagram on the left illustrates the probability distribution of the data items for a two-class discrimination problem. From this diagram it is easy to see the joint probability $P(L, C_i)$, where L is a label for the true class (A or B) and C_i is one of the sub-space clusters, is given by:

$$P(L, C_i) = P(C_i|L)P(L)/(P(A) + P(B)) \quad (5)$$

On the right under the heading $P(\text{Correct})$, alongside each node, is the probability $P(L|C_i)$ that classification is correct for a stochastic classifier (i.e. $P(\text{Correct}|L, C_i)$). The sum of the products of $P(\text{Correct}|L, C_i)$ and $P(L, C_i)$ over all pairs of clusters and labels gives the overall probability of a correct classification for this classifier based on a particular low dimensional cluster set $\mathcal{C} = \{C_1, C_2, \dots, C_n\}$, and a particular pair of labellings A, B :

$$P(\text{Correct}|A, B, \mathcal{C}) = \sum_{L=A,B} \left(\frac{P(L)}{P(A) + P(B)} \sum_{C_i \in \mathcal{C}} P(C_i|L)P(L|C_i) \right) \quad (6)$$

where $P(A)$, $P(B)$, $P(C_i|L)$ and $P(L|C_i)$ can be calculated from a co-occurrence frequency matrix of clusters formed from the graph partitioning vs. sub-space clusters for the training data. Thus we define discriminative power, $D(\mathcal{C})$, as the maximum value of $P(\text{Correct}|A, B, \mathcal{C})$ over all pairs of classes, as in Equation 7.

$$D(\mathcal{C}) = \max_{a,b=A,B,C_1,\dots,a \neq b} P(\text{Correct}|a, b, \mathcal{C}) \quad (7)$$

$D(\mathcal{C})$ lies between 0.5 (no discriminative power) and 1.0 (perfect discrimination for at least one pair of classes). This value can be used to weight the graph partitioning based clustering described in Equation 3; however a better approach is to threshold this value and use binary weights (Equation 8). We use a fairly low threshold (T_w) of 0.6 to exclude only those sub-spaces that are

very poorly discriminative. Essentially this is an unsupervised feature selection approach.

$$W_c = \begin{cases} 0 & \text{if } D(\mathcal{C}) < T_w \\ 1 & \text{if } D(\mathcal{C}) \geq T_w \end{cases} \quad (8)$$

In principle, our method could be applied iteratively by repeatedly using the clustering output to calculate new weights and re-clustering. Our experiments suggest that improved performance and convergence occurs in some circumstances; however in other circumstances over-fitting and other stability problems have been observed. We use only a single application of the re-weighting for this reason. It should be noted that our approach relies on the initial clustering having a (more or less) one-to-one mapping with the true class labelling (i.e at least 50% of items in a cluster belong to the same true class). If this is not the case, it is unlikely an improvement will be obtained.

Once a set of examples is partitioned, the partitions may be used as supervision for a conventional supervised statistical learning algorithm such as a Multi-Layer Perceptron, Radial Basis Function or Vector Quantisation based nearest neighbour classifier (the latter is used in our implementation). This allows for the construction of models that encapsulate the information from the clustering in such a way that they can be easily and efficiently applied to novel data. These models are used to generate training data suitable for symbolic learning. For each object identified by the attention mechanism, a (symbolic) property is associated with it for each semantic group using these learned classifiers.

3.1.2 Attention, learning and classification for spoken utterances

A symbolic description of audio events must be obtained from the input audio signal. This task is relatively straight-forward since we are dealing with a small number of isolated sounds (generally single words). Speech recognition and production software could be used to perform this task (e.g. HTK [42]), normally requiring supervised learning [7,12,32]; however an unsupervised approach to this task is favoured to fit in with the philosophy of learning models and rules for an autonomous synthetic agent. Such an approach also has the advantage that participants can make non-word utterances (e.g. animal noises) or make sounds using objects or instruments. The methods presented in this section are crude, yet adequate for our purposes. For more complex applications, more sophisticated techniques are likely to be required.

The participant of the game who we wish to replace with a synthetic agent speaks into a microphone. The audio is sampled at $8172Hz$ and an attention mechanism based on the energy of the signal is used to segment sounds. Non-

overlapping windows are formed each containing 512 samples, which is the power of 2 (needed for the Fourier transform) giving windows with the most similar duration to a frame of video. The energy for each window is calculated as the sum of absolute values of the samples. Utterances are contiguous windows with energy above the threshold and are of maximal duration (Figure 7).

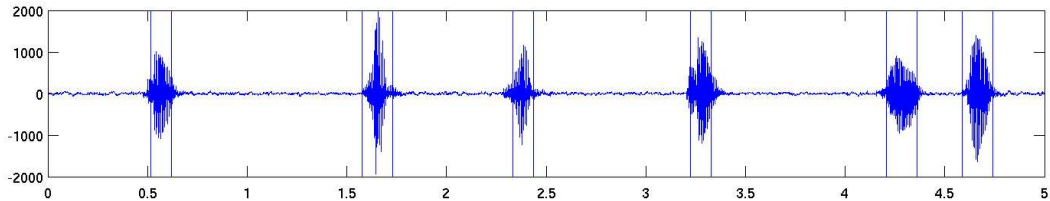


Fig. 7. The attention mechanism segments the audio signal.

Spectral analysis is performed on each detected window. The dimensionality of each spectrum is reduced from 512 to 17^4 by histogramming the absolute values. Reducing the spectrum to this dimensionality makes the clustering more robust to small variations in the pitch of the voice [19]. Each utterance detected is then represented by a temporal sequence of L reduced-dimensionality windows. L is chosen such that it is equal to the length of the shortest utterance (in windows) in the training set. This is achieved by linearly resampling the temporal sequence of spectral histograms for the windows that span an utterance. The utterances are now of identical length, and K-means clustering is performed on the set of utterances several times with different numbers of clusters.

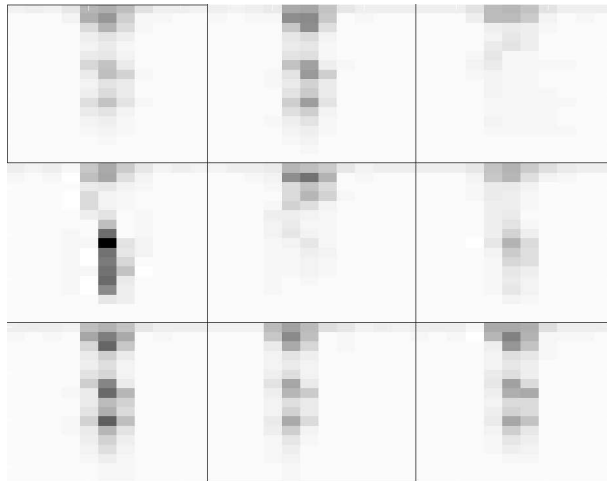


Fig. 8. Spectral analysis.

⁴ This depends upon the rate that the audio is sampled ($8172Hz$), the fundamental pitch frequency of the human voice (average around $275Hz$) and the size of the window used (512 samples equivalent to windows at $16Hz$). $275/16$ is approximately 17.

Figure 8 shows a set of nine clusters formed during one run of the process, visualised as spectrograms (plots of frequency vs normalised time) relating to the cluster centres. The number of clusters is automatically chosen such that the ratio between the mean distance of each utterance to the centre of the closest cluster and the mean distance between all the cluster centres is minimised. Each utterance of the training set is classified (nearest cluster centre) to create a symbolic data stream as shown in Figure 9; thus an utterance may be represented symbolically as one of a set of labels `utt1, utt2, . . .`. This method for automatically choosing the number of clusters tends to over-cluster the data (too many clusters are created); our approach to this problem is to build equivalence classes between the utterances, as discussed in Section 3.3.2.

3.2 Symbolic data representation

To drive a synthetic agent capable of interacting with its environment we wish to learn the protocols for the agent’s actions. In this work, actions are the vocal utterances; however, the principle should be applicable to other possible action types, such as movement of a robotic arm. Our aim is to encapsulate the protocol of the activity in a way in which actions can be fired following observation of perceptual inputs. Thus utterances of a participant are described in the form `action(utterance, time)`. The playing area is described by a list of objects, each described by their attributes of texture, colour and position in the form: `state([[texture1, colour1, position1], [texture2, colour2, position2]], time)`, which represents a state containing two objects. In addition, each time (`time(time)`.) and temporal successor (`successor(previous time, current time)`.) is denoted. Example audio-visual input for SNAP1 and the corresponding symbolic data stream is shown in Figure 9. When the table is empty, the description of perceptual inputs is an empty list, as no objects have been tracked, e.g. `state([], t518)`. Further details are discussed in Section 3.3.1.

Our synthetic agent interacts with its environment through actions (utterances) and perceives the environment through visual input of objects in the playing area. Thus an action by the agent will be fired immediately after the interpretation of perceptual input states. Therefore in the symbolic data presented for protocol learning, we wish for the actions to occur at the same qualitative time as the states (although quantitatively they occur successively). *This creates a direct link between perception and action.*

Our prototype system works in real time on live input data. As such, the frame rate of the tracker is variable, depending upon the number of objects it is tracking, the amount of motion in the scene, and what features it is extracting from the images for clustering. Thus, the use of frame numbers

is not sufficient to link times in the input video stream with the utterances obtained through processing the separate audio stream.

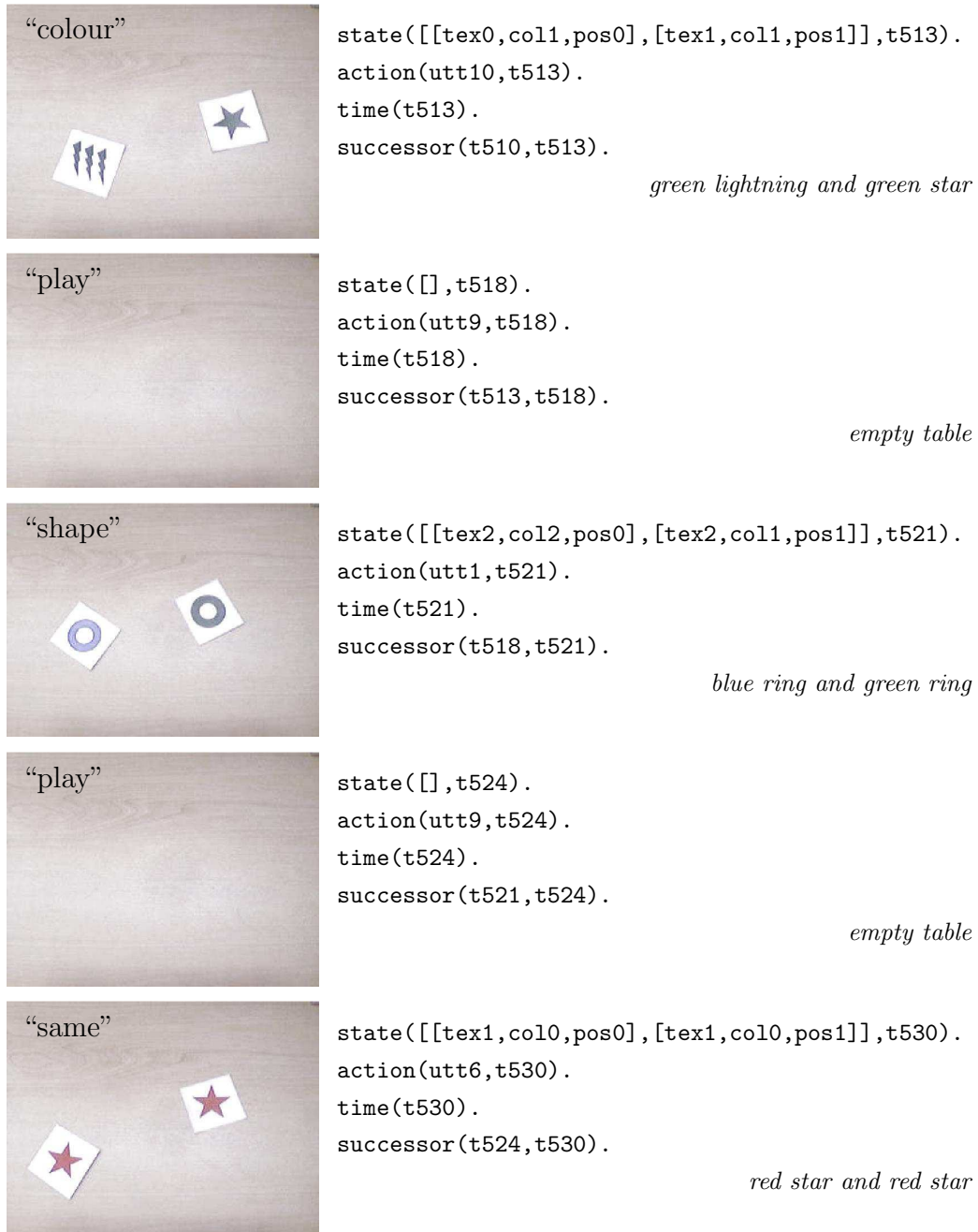


Fig. 9. Example audio-visual inputs and corresponding symbolic data representation for the game SNAP1. The symbols correspond to the perceptual categories learned by unsupervised clustering. In this case, the symbols `utt10`, `utt9`, `utt1` and `utt6` are symbols for utterances representing, respectively, the words “colour”, “play”, “shape” and “same”, and the `texX` relate to the texture/shape of objects, `colX` relate to the colour of the objects, and `posX` to the object’s position. The textual description in italics is provided for readers viewing a greyscale copy.

To form a direct link between states and actions, at the beginning of a training phase, a clock is started, and the number of seconds since the start is recorded for each (**state**, **time**, **successor**) triple from the object tracker. The number of seconds since the start is also recorded for each action utterance from the audio stream. The timings for this symbolic data stream from the audio is then synchronised with the timings from the video data stream. Each utterance is back-dated to the timing of the previous state description if the timing is not coincident already. This simple method works for our application. A more complex approach could be undertaken to provide a richer description where not only temporal successor relationships between events are used, but descriptions such as ‘before’, or ‘during’ may be useful for extended temporal events (in the future, the tracking of the movement of objects may be of interest, particularly when applied to the learning of robotic actions) for which a temporal representation such as Allen’s interval calculus [3,31] may be useful.

3.3 Learning symbolic protocols from perceptual data

3.3.1 Inductive Logic Programming

Inductive logic programming (ILP) [21,26] is the name given to the field of AI that studies inference methods by which given background knowledge B and examples E , the simplest consistent hypothesis H is found such that: $B \wedge H \models E$ where B, H, E are logic programs (informally: when B and H are true, E is also true). In our terms, we seek to induce a simple logic program H which when combined with the ‘state’ and ‘successor’ atoms (from the training data) B , entails the ‘action’ atoms E . No generic domain knowledge is included in the background B ⁵.

Our work uses CProgol4.4 [28] which is an implementation of the *definite modes language* [27]. PROGOL searches for the most statistically plausible hypothesis H using inverse entailment - a depth-first search through constructed hypotheses. PROGOL works by generalising a set of examples, given a list of *types*, accounting for the categories of objects in the domain under consideration, and a set of *syntactic biases*. These syntactic biases are user defined *mode declarations* that restrict the possible form of the proposed generalisations, restricting the possible search space. Mode declarations describe the predicates (with their argument types) that can be used either in the head or in the body of hypothesised generalisations. These declarations also state how variables in the argument of predicates in the head and body of formulae should be used in the formulae sought. In other words, variables can be declared in the modes as either input, output or grounded (constant). An example of mode declarations for the input data described in Section 3.2 is shown in Figure 10,

⁵ Implicit domain knowledge is provided by the mode declarations.

where `modeh` and `modeb` represent, respectively, predicates that should be in the head and in the body of the generalising formulae. The symbols `+`, `-` and `#`, represent respectively input, output and grounded variables and the terms `shape`, `colour`, `position` and `time` are term types. The number in the first argument of `modeh` and `modeb` bounds the number of alternative solutions for instantiating the predicate stated in the second argument.

```
:- modeh(1,action(#utterance,+time))?
:- modeb(100,state([[ -shape,-colour,-position],
                   [-shape,-colour,-position]],+time))?
:- modeb(100,state([],+time))?
:- modeb(100,successor(+time,-time))?
```

Fig. 10. Mode declarations for the game SNAP1.

The mode declarations in Figure 10 state that the generalising formula should have the predicate `action/2` in the head (with a grounded utterance and an input variable for time as arguments) and predicates `state/2` and `successor/2` in the body. In principle, there is no limit on the number of occurrences of each predicate in the mode declarations used with different combinations of input, output and grounded variables. Examples of PROGOL type declarations for the game SNAP1 are shown in Figure 11. The constant symbol names are chosen to make the example clear to read; in practice the semantic meaning of the object property or utterance labels is unknown, as the labelling come from the lower level systems and is devoid of semantics (except as provided by the lower level grounding). Although PROGOL is capable of generalising from positive and negative examples, we only have positive examples available.

```
utterance(play).    shape(ring).    colour(red).    position(pos0).
utterance(colour). shape(star).    colour(green).  position(pos1).
utterance(shape).  shape	flash).  colour(blue).
utterance(both).
utterance(nothing).
```

Fig. 11. Type declarations for the game SNAP1.

Briefly, PROGOL's search can be described as follows. For each positive example, PROGOL initially generates a most specific Horn clause, according to the mode declarations. This clause is further contrasted with the remaining examples in the search for a more general formula capable of subsuming the examples in the data set. An example of PROGOL's output for the SNAP1 game (given mode and type declarations as described above) is presented in Figure 12.

Formula (1) in Figure 12 shows that PROGOL built the connection between the utterance `play` and the empty state. The variable `A` in both the head (e.g. `action(play,A)`) and the body (e.g. `state([],A)`) indicates that a generalisation over time has been made; `A` represents an ungrounded time state, i.e.

```

(1) action(play,A)      :- state([],A).
(2) action(colour,A)   :- state([[B,C,D],[E,C,F]],A).
(3) action(shape,A)    :- state([[B,C,D],[B,E,F]],A).
(4) action(nothing,A)  :- state([[B,C,D],[E,F,G]],A).
(5) action(both,A)     :- state([[B,C,D],[B,C,E]],A).

```

Fig. 12. Unordered PROGOL output for the game SNAP1.

any time. Thus Formula (1) is interpreted as “if the table is empty at time A , then respond at time A with the action of uttering play”. Formulae (2-5) show the rules learned for non-empty states, expressed using co-occurring variables as arguments to `state/2`. E.g., in Formula (2), the repeated variable C in the second position in each object’s feature list encodes the condition that two objects have the same colour and that the appropriate action is to utter “colour”. Note that the rules are unordered and do not constitute a logic program that correctly specifies the intended protocol. In Section 3.4, we discuss how an inference engine orders these rules.

The key motivation for our use of inductive logic programming is that the generated rules can be applied to situations never seen before. For example, the complete rules for SNAP1 may be learned without an example in the training data of the utterance “colour” after two red objects are in the scene; it can generalise from the other examples in the training data where “colour” is uttered after two green or two blue objects are in the scene. Moreover, if the learning of object classifiers (for texture and colour) was allowed at run-time (i.e. during ‘play mode’), then the rules learned could also be applicable to objects and features never having been seen at all during training.

3.3.2 Building equivalence classes of utterances

As mentioned in Section 3.1.2, the method for utterance clustering is prone to cluster into more than the true number of clusters. However, we are able to use the generalisation rules found by PROGOL to construct equivalence classes among utterances. The procedure for generating equivalence classes is based on the hypothesis that rules with *similar* bodies are related to equivalent utterances in the rule heads. There are many possible ways of defining similarity in logic programs [10], and an investigation of those is outside the scope of this paper. As we shall see in Section 4.2, in this work we use unification and identity of bodies as our similarity measures between clause bodies.

To construct equivalence classes, firstly, every pair of input rules whose heads are of the form `action(utterance,time)` are checked for whether their bodies are similar. Clauses with empty bodies are discarded as they do not provide any evidence for equivalence. Assume a transitive, reflexive and symmet-

ric binary relation `equiv/2` defined over utterances. Thus, for every pair of clause heads whose bodies are similar, a statement `equiv/2` is created (and asserted), stating the hypothesis of equivalence between the utterances in their arguments. For instance, let `action(utt_i,t_x)` and `action(utt_j,t_y)` be two clause heads with similar bodies, then the hypothesis of equivalence between the utterances `utt_i` and `utt_j` is created and asserted as the statement `equiv(utt_i,utt_j)`. For notational convenience, we call atomic formula, such as `equiv(utt_i,utt_j)`, *equivalence pairs*. Two utterances `utt_i` and `utt_j` are hypothesised as being equivalent if the equivalence pair `equiv(utt_i,utt_j)` appears once (or more) in the training examples⁶. Finally, equivalence classes are created by taking the transitive closure of the relation `equiv/2`.

The process of building equivalence classes can be exemplified in the game SNAP1 if we assume that the audio clustering algorithm found many distinct representations for the same utterances. For instance, suppose this algorithm clustered the word “play” into 4 distinct classes, namely `p_1`, `p_2`, `p_3` and `p_4` and that PROGOL found the following four rules involving these symbols:

```
action(p_1,A) :- state([],A).
action(p_2,A) :- state([],A).
action(p_3,A) :- state([],A).
action(p_4,A) :- state([],A).
```

Therefore, the method for constructing equivalence classes suggests that [`p_1`, `p_2`, `p_3`, `p_4`] forms one class that is pairwise disjoint with respect to the classes combining the remainder of the utterances. In this case the bodies are all trivially similar, as they are identical, but this need not always be the case.

In this paper we are only obtaining equivalence classes of utterances, leaving aside the problem of over clustering on learning the models of visual objects. In fact, the problem of over clustering models of perceptual objects, in our setting, is more evident on the audio than on the visual domains. This is because the visual representation is invariant to variation in orientation and lighting, whereas there is little invariance in the audio representation to variation in pitch and dynamics (e.g., the participant may get excited when winning or if something unexpected happens).

⁶ The number of times that an equivalence pair is hypothesised as equivalent in the training examples could be used as a measure of confidence on the equivalence hypotheses, thus providing a thresholding value for the inclusion of objects in equivalence classes. This issue, however, is a matter for future research.

3.4 Inference engine for agent behaviour generation

The symbolic protocols learned by PROGOL are used by a Prolog program as a set of rules used to drive an interactive synthetic agent that can participate in its environment. This set of rules is, prior to input to the agent, automatically filtered and ordered according to the following criteria. Ground atomic formulae are kept at the beginning of the rule set (as initially given by PROGOL). These formulae will never be selected by the agent as they have their temporal variable instantiated to a very specific time point. Non-ground atomic formulae are moved to the end of the data set, since these are the most general rules (i.e. the action expressed in their heads is not constrained to any particular perceptual input). Non-atomic rules whose bodies unify are ordered from the one with the most specific to the one with most general body.

```
action(both,A)      :- state([[B,C,D],[B,C,E]],A).
action(shape,A)     :- state([[B,C,D],[B,E,F]],A).
action(colour,A)    :- state([[B,C,D],[E,C,F]],A).
action(nothing,A)  :- state([[B,C,D],[E,F,G]],A).
action(play,A)     :- state([],A).
```

Fig. 13. Sorted PROGOL output for the game SNAP1 corresponding to the unordered output in Fig 12.

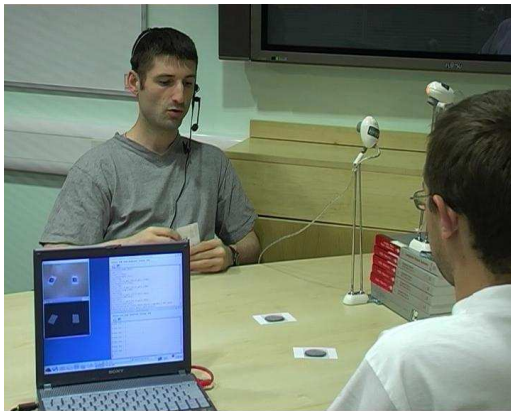
This ensures that more specific rules are fired first (otherwise they would be subsumed by more general rules). This ensures that rule 4 in Figure 12 (which is satisfied for every perceptual input of two objects) is ordered after rules 2 and 3. It also ensures that rule 5 (the most specific) is the first rule. Figure 13 shows the result of ordering the ruleset of Figure 12. Further examples are given in Section 4. The remaining rules are kept in the order output from PROGOL. It is worth pointing out that PROGOL ranks its output formulae according to their predictive power and compression with respect to the data set; therefore, rules at the bottom of the rule set are those that predict fewer examples than those handled by the rules at the top. The former usually represent idiosyncrasies of the data and, due to their positions in the rule set, may never be selected by Prolog. In addition to this, a cut is added as the last conjunct of each non-ground formula's body. This guarantees that only a single action will fire at any time step.

The synthetic agent, loaded with the learned rules (ordered as above), receives input from the perceptual system, and outputs the inferred utterance through a sound synthesis module. This module replays an audio clip of the appropriate response (the one closest to the cluster centre), automatically extracted from the training session.

For greater effect, a visual *waiting head* (Figure 14) is displayed on a screen, and in time with the replay of the audio clip, a corresponding facial movement

is made by the talking head. The video for this is captured with the audio of the participant to be replaced by a synthetic agent. The details of the video-realistic waiting head is not the focus of our work, but it adds realism to the demonstration system⁷.

As there is no robotic arm in the system, a human participant is required to follow the instructions uttered by the synthetic agent. Currently the agent executes a single action generation per time step. This restriction, however, might be relaxed without the need to change the central ideas of the system presented above.



(a) training



(b) playing



(c) waiting



(d) talking

Fig. 14. Synthetic agent. (a) During training, the audio and video stream of a participant are captured from microphone and webcam, objects are tracked by a separate webcam pointing at the table. (b) In the execution phase, the participant no longer says anything, he follows the instructions of the interactive talking head displayed on the screen and broadcast from the speakers. (c) Closeup of the waiting head displayed on the screen in (b). (d) The head talks by replaying the audio-video stream associated with the inferred action.

⁷ Movies demonstrating the work of this paper, both the learning and execution phases are online at <http://www.comp.leeds.ac.uk/vision/cogvis/>

4 Experiments and results

In this section we present some experimental results of applying the framework for learning protocol behaviour on three game playing scenarios (Section 4.2). The results are analysed using an evaluation method (Section 4.1) based on verifying soundness and completeness of the learned rule sets of Section 3.3. Note that the ordering constraints of Section 3.4 are not taken account of in the learned rule sets - they are just taken to be sets of definite clauses. As noted below, the experimental results for completeness would in some cases be better if the ordering constraints were to be taken into account. Informally, the method provides the extent to which the given model is represented in the learned rule set – *soundness* – and the extent to which the predictions of the learned rule set agree with the predictions of the model (for the same perceptual inputs) – *completeness*. Therefore, this method utilises logic-based concepts to provide a principled way of evaluating our system, which is one of the reasons for choosing a symbolic learning tool within this framework.

4.1 Evaluation method

In order to evaluate how well the protocol rules were learned from noisy vision data, we contrast the rules learned from each dataset with a hand-coded set of formulae; these hand coded rules can be understood as the *underlying semantics* of the game, i.e. they perfectly capture the protocol rules to be learned. If perfect learning takes place, then a logically equivalent set of rules should be learned from vision data. In practice, we are not worried about *exact* logical equivalence, but rather whether the same actions would be performed in identical circumstances (the learned formulae might possibly entail statements involving the performance of actions, and thus might not necessarily be in the hand coded set). Therefore we want to evaluate both:

- (1) on how many actions the learned set agrees with hand coded set; and
- (2) on how many actions the hand coded set agrees with learned set.

Making an analogy with model theory for inference systems, we call the first condition *soundness*, as it checks whether learned actions agree with the semantics, i.e. whether the learned rule set correctly predicts actions (given the gold standard of the hand coded set). The second condition might be called *completeness*, as it checks whether all the actions predicted by the semantics (under particular conditions) are similarly indicated in the learned rule set. Formally, we can characterise these ideas as follows.

Definition: Agreement between rule sets.

If $\mathcal{X}, \mathcal{Y}_1 \models \mathcal{Z}$ implies $\mathcal{X}, \mathcal{Y}_2 \models \mathcal{Z}$, then \mathcal{Y}_1 agrees with \mathcal{Y}_2 on \mathcal{Z} in situation \mathcal{X} .

Definition: Total soundness and completeness of a learned rule set with respect to a hand crafted rule set.

Let \mathcal{A} be a set of formulae representing possible actions (typically characterised syntactically as atomic formulae formed by a distinguished predicate). Let \mathcal{H} and \mathcal{L} be two sets of formulae representing the handcrafted and learned rule sets respectively. Let \mathcal{X} represent a symbolic description of the world at a given time (i.e. a set of ground atomic formulae – *state, time, successor, action* – as illustrated in Figures 9, 16 and 18).

If \mathcal{L} agrees with \mathcal{H} on every action $\theta \in \mathcal{A}$ for all \mathcal{X} , then \mathcal{L} is sound with respect to \mathcal{H} .

If \mathcal{H} agrees with \mathcal{L} on every action $\theta \in \mathcal{A}$ for all \mathcal{X} , then \mathcal{L} is complete with respect to \mathcal{H} .

However, in practice, the data may be noisy and incomplete causing rules to be missed, over-generalised or mis-represented. Therefore, we define *partial* soundness and completeness as follows.

Definition: Partial soundness and completeness of a learned rule set with respect to a hand crafted rule set.

Let \mathcal{A} , \mathcal{H} and \mathcal{L} and \mathcal{X} be as above.

If \mathcal{L} agrees with \mathcal{H} on a maximal subset of actions $\Theta \subset \mathcal{A}$ for some \mathcal{X} , then \mathcal{L} is $\frac{|\Theta|}{|\mathcal{A}|}$ sound with respect to \mathcal{H} .

If \mathcal{H} agrees with \mathcal{L} on a maximal subset of actions $\Theta \subset \mathcal{A}$ for some \mathcal{X} , then \mathcal{L} is $\frac{|\Theta|}{|\mathcal{A}|}$ complete with respect to \mathcal{H} .

A set of three games are used to show the performance of our cognitive vision system. The next section describes each game in detail, presenting typical outputs of the system and discussing experimental results based on the evaluation method above.

4.2 Experimental Evaluation

Experiment 1: SNAP1

This game demonstrates the system’s ability to generalise rules across perceptual categories, rather than grounding each instance. SNAP1 is a generalised snap game where two cards are played, followed by an utterance dependent upon the state of the two cards which are either the “same”, the same “colour”,

the same “shape” or “nothing”. The cards are then removed, and “play” is uttered indicating to play the next two cards. Figure 9 illustrates an example sequence of the game. For SNAP1, the learning process has been completed three times, each time with a dataset containing 100 objects and 100 utterances, equivalent to 50 rounds of the game.

Figure 15 presents an example of a typical rule set output by the system. This set shows that four distinct versions of the utterance “play” (represented by the symbolic labels `utt2`, `utt3`, `utt8` and `utt9`) were correctly learned; rules for “same”, “colour”, “shape” and “nothing” (`utt6`, `utt10`, `utt1` and `utt4` respectively) were properly built, along with one sub-optimal rule (in terms of compactness) required to explain `utt7` which was placed as the last rule in the data set, according to the criteria discussed in Section 3.4.

```

action(utt4,t65).
action(utt5,t198).
action(utt1,t237).
action(utt6,A) :- state([[B,C,D],[B,C,E]],A).
action(utt10,A) :- state([[B,C,D],[E,C,F]],A).
action(utt1,A) :- state([[B,C,D],[B,E,F]],A).
action(utt4,A) :- state([[B,C,D],[E,F,G]],A).
action(utt9,A) :- state([],A).
action(utt8,A) :- state([],A).
action(utt2,A) :- state([],A).
action(utt3,A) :- state([],A).
action(utt7,A) :- successor(B,A), state([[C,D,E],[F,G,H]],B).

```

Fig. 15. Sorted learned rule set for SNAP1 run 1. Key to symbolic utterance labels: “same” = `utt6`, “colour” = `utt10`, “shape” = `utt1`, “nothing” = `utt4`, “play” = `utt2`, `utt3`, `utt5`, `utt7`, `utt8`, `utt9`.

Game	Soundness	Completeness
SNAP1 run 1	100.0%	71.4%
SNAP1 run 2	100.0%	88.8%
SNAP1 run 3	60.0%	80.0%

Table 1

Results of the evaluation of the protocol rules learned from perceptual observation of three runs of the game SNAP1.

Table 1 presents the evaluation of three runs of SNAP1 according to the evaluation method introduced in Section 4.1. Runs 1 and 2 are totally sound (scored 100% for soundness) meaning that every prediction made by the intended semantic was also obtained by the learned set for the same perceptual input. Both runs scored less than 100% for completeness (i.e., 71.4% for run 1 and 88.8% for run 2) as idiosyncratic rules for “play” were found. These rules

represented that an action “play” should be valid at a time point t if, in the previous time point, there were any two objects on the table. Although this is a true fact in the domain in question, this rule does not represent the correct protocol of producing the utterance “play” as it allows it to be pronounced when the table is not empty.

Run 1 could not find rules for “colour” or “same”, thus scoring 60% with respect to soundness. However, 80% of the rules found could be mapped into the intended semantics. The misbehaviour of this experiment with respect to soundness may be explained by erroneous clustering of the colour feature, which in our current implementation is unreliable.

Analysis of run 3 reveals the sensitivity of PROGOL to misclassified data. We are learning from small amounts of data which is locally sparse, thus a classification error may make up a significant amount of the data used to form a generalisation. This may seem very fragile; however it is the behaviour that we should expect. SNAP1 run 3 contains six examples of two objects having the same colour and same texture, and one of the same shape but different colours, followed each time by the utterance “same”. PROGOL generalises the rule `action(same,A) :- state([[B,C,D],[B,E,F]],A)`. If the single action (following a state in which colour is misclassified) is removed from the training set, then SNAP1 run 3 would contain six examples of two objects having the same colour and same texture, followed each time by the utterance “same”, and PROGOL generalises the correct rule `action(same,A) :- state([[B,C,D],[B,C,E]],A)`.

The sound clustering of run 1 produced three classes for “play”, and one class for each of the other utterances. The method for constructing equivalence classes (proposed in Section 3.3.2) used identity of bodies as similarity measure in this experiment. With this it was able to cluster two of the “play” utterances into one single class leaving the remainder as classes containing one symbol (*unary classes*). In run 2 the system found the correct number of equivalence classes, namely, one class for “play” containing five different instances of this utterance, and unary classes for the other words. In run 3 only unary equivalence classes were obtained, while the expected result would be one class for the two utterances “nothing” and unary ones for “shape”, “same”, “colour” and “play”.

Experiment 2: SNAP2

This game demonstrates the system’s ability to generalise rules across perceptual categories, and introduces a temporal dependency on the previous state. SNAP2 is a variation of SNAP1 where cards are placed one on top of another

and the resulting utterance is dependent upon the card which is visible, and the card at the previous time step. An example sequence of the game is shown in Figure 16. For SNAP2, the learning process has been completed three times, each time with a dataset containing 50 objects and 50 utterances, equivalent to 50 rounds of the game.



Fig. 16. Example audio-visual inputs and corresponding symbolic data representation for the game SNAP2.

```

action(utt11,t48).
action(utt12,t101).
action(utt8,t131).
action(utt8,t160).
action(utt8,t184).
action(utt11,t218).
action(utt6,t256).
action(utt3,A) :- state([[B,C,D]],A), successor(E,A),
                  state([[B,C,D]],E).
action(utt1,A) :- state([[B,C,D]],A), successor(E,A),
                  state([[F,C,D]],E).
action(utt5,A) :- state([[B,C,D]],A), successor(E,A),
                  state([[F,C,G]],E).
action(utt7,A) :- state([[B,C,D]],A), successor(E,A),
                  state([[B,F,D]],E).
action(utt9,A) :- state([[B,C,D]],A), successor(E,A),
                  state([[B,F,D]],E).
action(utt10,A):- successor(B,A), state([[C,D,E]],B),
                  successor(F,B), state([[G,D,E]],F).
action(utt4,A) :- successor(B,A), state([[C,D,E]],B),
                  successor(F,B), state([[G,H,E]],F).
action(utt2,A).

```

Fig. 17. Sorted learned rule set for SNAP2 run 1. Key to symbolic utterance labels: “same” = utt3, utt6, utt9, “colour” = utt1, utt5, “shape” = utt2, utt7, utt11, “nothing” = utt4, utt8, utt10, utt12.

Game	Soundness	Completeness
SNAP2 run 1	100.0%	100.0%
SNAP2 run 2	100.0%	71.4%
SNAP2 run 3	75.0%	100.0%

Table 2

Results of the evaluation of the protocol rules learned from perceptual observation of three runs of the game SNAP2.

Table 2 shows the evaluation of the three runs of this game according to the method described in Section 4.1. The results in run 1 in SNAP2 show that our system could learn a totally sound and complete data set for this game (see Figure 17 for the rule set). Run 2 (not shown) was 100% sound but only 71.4% complete, because one rule for “nothing” and one for “same” were misleading. These rules were probably the result of systematic misclassification of the colour feature during the training period. However, the sorting method (described in Section 3.4) placed both of these misleading rules at the end of the rule set. Authentic rules for these words were thus higher ranked. Therefore, the erroneous rules, in this case, do not jeopardise the behaviour of the agent.

Run 3 scored 75% for soundness since its learned set did not find any rule representing the word “nothing”. This was due to a lack of examples about this utterance in the data set considered.

The same equivalence criteria used in SNAP1 (identity of bodies) was assumed in SNAP2. In run 1 of SNAP2, twelve utterances were clustered by the audio clustering method (four representing “nothing”, three “same”, two “shape” and two for “colour”); the equivalence method constructed the correct equivalence class for “colour”, and found a class with two (out of three) symbols for same, the remainder were not found. The reason for missing elements in equivalence classes, even though the result was sound and complete, was due to the fact that there were no useful rules in the answer set for some of the symbols representing the utterances. For example, `action(utt6,t256)` in Figure 17 is the only rule containing utterance `utt6` and it has no body, so will never fire. This usually occurs when the data set contains a restricted number of examples about these utterances. Therefore, the equivalence method did not have enough information to build the appropriate classes. This method suffered from the same problem, in runs 2 and 3 of SNAP2.

Experiment 3: PSS - Paper, Scissors, Stone

This game demonstrates the system’s ability to ground utterance actions with sets of specific perceptual inputs. PSS is played by two players, each simultaneously selecting one of the object cards that are placed on the table. The game protocol is as follows. Utterances “play”, “win”, “lose” and “draw” are spoken by the player to be replaced by a synthetic agent. The action “win” occurs when the object paper beats (wraps) the object stone, scissors beats (cuts) paper, and stone beats (blunts) scissors. In this game, the position of object cards as well as their texture is crucially important, since if the position of the cards was swapped then the resulting action would be altered from “win” to “lose” or vice-versa. Figure 18 illustrates an example sequence of the game, where the utterance is said by the player who places his card on the right of the playing area; a typical output of the learning process is shown in Figure 19.

For PSS, the learning process has been completed three times, each time with a dataset containing 200 objects and 200 utterances, equivalent to 100 rounds of the game. This is larger than the other games, since a greater number of rules are being learned in this case. It is worth noting that if fewer examples are available, the whole process does not fail completely, but a partial description of the protocol is learned. Table 3 presents the results of three runs of PSS.

Runs 1 and 2 were both totally sound and (approximately) 88% complete. The

reason why they are not totally complete is due to occasional misclassification's of visual objects, causing misleading formulae to be generalised from the data sets. Poor results were obtained for soundness in run 3 since none of the rules representing "lose" were constructed. This was due to the fact that only one class for the utterance "lose" was obtained by the audio clustering algorithm and this happened to occur in the head of a idiosyncratic rule in the learned set.



Fig. 18. Example audio-visual inputs and corresponding symbolic data representation for the game PSS.

```

action(utt13,t294).
action(utt13,t458).
action(utt11,t462).
action(utt9,A) :- state([[tex2,B,pos0],[tex0,B,pos1]],A).
action(utt13,A) :- state([[tex1,B,pos0],[tex0,B,pos1]],A).
action(utt9,A) :- state([[tex1,B,pos0],[tex2,C,pos1]],A).
action(utt1,A) :- state([[tex1,B,pos0],[tex1,C,pos1]],A).
action(utt3,A) :- state([[tex0,B,pos0],[tex1,C,pos1]],A).
action(utt11,A) :- state([[tex2,B,pos0],[tex1,C,pos1]],A).
action(utt1,A) :- state([[tex2,B,pos0],[tex2,C,pos1]],A).
action(utt1,A) :- state([[tex0,B,pos0],[tex0,C,pos1]],A).
action(utt11,A) :- state([[tex0,B,pos0],[tex2,C,pos1]],A).
action(utt3,A) :- state([[tex2,B,pos0],[tex0,C,pos1]],A).
action(utt7,A) :- state([[tex2,B,pos0],[tex0,C,pos1]],A).
action(utt7,A) :- state([[tex1,B,pos0],[tex2,C,pos1]],A).
action(utt11,A) :- state([[tex1,B,pos0],[tex0,C,pos1]],A).
action(utt9,A) :- state([[tex0,B,pos0],[tex1,C,pos1]],A).
action(utt7,A) :- state([[tex0,B,pos0],[tex1,C,pos1]],A).
action(utt13,A) :- state([[tex0,B,pos0],[tex2,C,pos1]],A).
action(utt13,A) :- state([[tex2,B,pos0],[tex1,C,pos1]],A).
action(utt8,A) :- state([],A).
action(utt2,A) :- state([],A).
action(utt14,A) :- state([],A).
action(utt5,A) :- state([],A).
action(utt6,A) :- state([],A).
action(utt12,A) :- state([],A).
action(utt10,A) :- successor(B,A), state([[tex2,C,pos0],[tex0,C,pos1]],B).
action(utt5,A) :- successor(B,A), state([[tex0,C,pos0],[tex2,D,pos1]],B).
action(utt12,A) :- successor(B,A), state([[tex1,C,pos0],[tex2,D,pos1]],B).
action(utt4,A).

```

Fig. 19. Sorted learned rule set for PSS run 1. Key to symbolic labels: “win” = utt3, utt7, utt9, “lose” = utt11, utt13, “draw” = utt1, “play” = utt2, utt4, utt5, utt6, utt8, utt10, utt12, utt14, paper = tex0, scissors = tex1, stone = tex2.

Game	Soundness	Completeness
PSS run 1	100.0%	88.5%
PSS run 2	100.0%	88.8%
PSS run 3	75.0%	76.9%

Table 3

Results of the evaluation of the protocol rules learned from perceptual observation of three runs of the game PSS.

In PSS we use term unification as our equivalence criteria. In run 1, from the 14 clusters representing the four utterances in this game, our method for hypothesising equivalences found five distinct classes. The three different representations of the utterance “win” and the two for “lose” were accurately

combined into two distinct classes. Six out of the eight symbols representing the word “play” were clustered into one class. The single symbol representing “draw” occupied an unary class. Two symbols for “play” are missing from their relative equivalence class as they occurred in the head of idiosyncratic rules. The exact number of equivalence classes were built for run 2, the four distinct versions of “play” and the two for “win” were joined into two distinct classes. The single “lose” and “draw” were assigned two unary classes. This method presented the same accuracy when applied to run 3.

In summary, from the experiments reported in this section we can conclude that our system could learn correct protocols from noisy visual data, as expressed by the many 100% soundness results shown in Tables 1, 2 and 3. Even in situations where rules for some items in the protocol were not found, our system was able to construct a partial description of behaviour that allows the agent to behave suitably given appropriate perceptual input. Most of the partial results for completeness do not jeopardise the actuation of the synthetic agent as the idiosyncratic rules obtained were, in general, occupying the end of the rule set as dictated by our sorting procedure (described in Section 3.4). The ordering procedure, however, has a palliative effect, as the cause of idiosyncratic rules resides in the occasional erroneous clustering of the colour feature and the generalisation of irrelevant facts from the data sets. A *de facto* solution for these issues is related to the integration of a more robust camera apparatus in our setting and the development of methods, within ILP, for choosing the most relevant formulae from a set of generalising rules.

The proposed method for building equivalence classes produced the exact number of classes for the game PSS, however it did not behave with the same accuracy in the SNAP1 and SNAP2 experiments. The reason for this difference in accuracy of the results for PSS and the SNAP games resides in the fact that the protocols obtained for the SNAP games involve only free variables in the body of the rules representing them, whereas rule bodies for PSS are composed of ground atoms. Therefore, similarity between bodies in the latter was trivially defined as equality. However, we had to use unification as similarity measure for the SNAP games, which resulted in a poorer set of equivalence classes than those obtained for PSS. Further research should concentrate on the development of appropriate similarity measures in order to overcome this problem.

5 Discussion

In this section we discuss the main pitfalls faced during the development of the research reported in the present paper. This discussion follows the order in which information is processed through the system, i.e., from attention mechanisms to symbolic learning of protocol behaviours.

The attention mechanisms used have been robust for our scenario. Very occasionally an utterance may be missed if spoken very softly, and it works in environments in which there is noise in the background. On the visual side, the attention mechanism based on motion followed by a number of static frames works 100% of the time unless objects are placed in the scene *very* quickly (if motion lasts for less than 3 frames (< 0.5 seconds)).

Once features have been extracted from interesting objects and sounds, continuous unsupervised methods are used to cluster the data items. It may appear that we are using simple objects and few words; however, the main purpose of this work is to create a synthetic agent which integrates learning at both perceptual and task levels, with as few (general) assumptions as possible. It would be trivial to classify more complex objects than those used in this paper in a perfect way, using a supervised classifier and specific feature descriptors or model based approach. However, classifying a number of examples into an unknown number of classes in an unsupervised manner is an open problem.

The ideas presented treat video and audio in largely the same way; i.e., as an input signal from which interesting perceptual objects are extracted and clustered. Presently, only one clustering is used. We could form several clusterings and choose the one that works best in the context, i.e. use symbolic reasoning to decide which clustering is best, based on the protocols it can learn. We would also like to explore feeding back information from the symbolic learning phase, in order to re-label the classes to which each perceptual object belongs based on the context of the learned protocol. This would allow classifiers to be re-learned using the new labels as supervision.

The main limitation of the vision system presented is reliable colour classification. We are using inexpensive U.S.B. webcams, which struggle to capture colour well enough to easily distinguish different coloured objects. This has led to a large number of misclassifications of the colours of objects. However, it has allowed us to investigate how the system behaves in the presence of noise. In situations when examples are locally sparse (e.g. when there are few examples of a particular utterance), then PROGOL tends to over-generalise the data, particularly if noisy examples are present. This is understandable, and generally PROGOL *does* learn rules of the right specificity from *very few examples* (in SNAP2 run 1 sound and complete rules are learned for the game where, on average, each utterance class has only four examples!).

The use of PROGOL for inductive logic programming has allowed the formulation of protocol models from *positive only examples*. In summary, it has learned protocols that:

- generalise temporal sequences of state observations;
- generalise equality of perceptual classes of objects;
- ground symbols to perceptual objects autonomously;
- can be used by a synthetic agent to interact with its environment.

Learning complex temporal dependency in the protocols is, perhaps, the next step of development of this research. The present work assumes the `successor`⁸ predicate as the only temporal relation of interest. This should be extended to include a range of temporal relations, which would allow a greater range of scenarios to be captured. McCallum’s work on Nearest Sequence Memory [24] in instance-based state identification may be a useful way to proceed in order to capture variable length temporal dependencies between states.

The complexity of the symbolic learning task would be reduced if negative examples were supplied, constraining the search space. We have not wanted to incorporate supervision into our framework, which would be necessary if we chose to learn from negative as well as positive examples.

The symbolic protocols learned were used to construct accurate sets of equivalence classes among over-clustered sets of utterance symbols. The criteria for deciding which elements should be included in each class was based on user-defined similarity criteria between formulae bodies. The development of automatic procedures for choosing such criteria is subject of current research.

5.1 *Future work*

There are many ways that this work may be extended. Outside of the game playing scenarios presented, it would be of more interest to generalise the approach to human behaviour in traffic scenes and sporting activities, where the learning task is likely to be more challenging. It may also be possible to learn from observation in industrial inspection tasks to accept or reject industrial parts based on unsupervised learning of perceptual categories and symbolic models learned using ILP. In all of these examples, particularly interesting points include dealing with:

- higher (and variable) order temporal models;
- when the actions that we wish to learn are not only grounded to both the present and previous `states`, but also to the `action` at the previous time-step;
- scenarios involving a large number of rules;
- stochastic actions, when a non-deterministic choice can be made;
- incremental learning:
 - learning from all the information seen so far, and adapting the models as more examples are seen;
 - multi-stage learning, where concepts such as orderings are incorporated

⁸ This does not specifically preclude higher 1st order temporal models, but it makes their formulation unlikely.

- in the form of background knowledge gained in a previous learning phase from perceptual observation;
- a richer language structure, within a similar framework.

We would like to propose a game of greater ‘complexity’ than those presented in this paper (but feasible to be solved by our approach), and discuss the adaptations that our framework should incur in order to learn the protocols of this and other scenarios where similar behaviours are present.

The game of ‘play your cards right’, played on some TV shows, involves a pack of playing cards, a sequence of which are laid in a row face down. The first card is turned over, and based on the value of this card the contestant has to say “higher” or “lower”. Then the next card is turned over, the contestant “loses” if they were wrong, or continues in the game saying “higher” or “lower” until the final card, when if they correctly guess “higher” or “lower”, then they “win”.

With the aim of learning the protocol of the game for an autonomous agent, this game encompasses a number of interesting challenges:

- the actions (utterances) that we wish to learn are not only grounded to both the present and previous **state**, but also to the **action** at the previous time-step;
- the number of rules is large; even if only five values of cards are used then many rules are necessary;
- stochastic actions exist; if the middle value card is turned over, then there is no single correct protocol to follow - a stochastic choice of either “higher” or “lower” must be made (in fact the action is strictly non-deterministic always).
- incorporating concepts such as the ordering of the cards, in the form of background knowledge from a previous learning phase from perceptual observation.

This game may also seem to be simple, yet a number of background concepts are needed to tackle it. Sensible approaches that we should consider, include the development of an incremental learning system allowing the agent to learn the protocol while experimenting with it. This learning system should also search for abstract formulae that could encompass basic mathematical concepts about the domain to be fed back into the learning process as background knowledge. Preliminary results in this direction were reported in [36]. An engine for interpreting stochastic actions should also be developed to handle cases where multiple, distinct, actions are possible from the same perceptual input. Research into the development of such an engine is well under way.

6 Conclusion

Completely unsupervised learning of rules of behaviour from observation is a crucial issue in the development of fully autonomous agents that are able to act appropriately in the real world, including interacting with other individuals.

We have presented a novel approach for learning protocol behaviours from perceptual observation that incorporates computer vision, audio processing and symbolic learning methods. The results obtained on learning the protocols of simple table-top games show that this system is able to successfully learn perceptual classes and rules of behaviours from real world audio-visual data in an autonomous manner. Through selecting simple components (e.g. for acoustic processing), we have been able to focus on the potential for constructive interaction between these components. For example, the ability to resolve failings in sound categorisation through examining induced rule-sets was an unexpected finding that may not have surfaced had we used a more sophisticated acoustic processor from the start. Progressively scaling this approach to more complex games in order to reach the level of interactions between any number of agents in any environment is a challenge for future investigations.

7 Acknowledgements

This work was funded by the European Union, as part of the CogVis project (Contract IST-2000-29375). We would like to thank Brandon Bennett and Aphrodite Galata for many useful discussions throughout the course of this research and also the anonymous reviewers for the comments that helped improve this paper.

References

- [1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. European Conference on Computer Vision*, volume 4, pages 113–127, 2002.
- [2] S. Aksoy, C. Tusk, K. Koperski, and G. Marchisio. Scene modeling and image mining with a visual grammar. In *Frontiers of Remote Sensing Information Processing*, pages 35–62. World Scientific, 2003.
- [3] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

- [4] N. Badler. *Temporal Scene Analysis: Conceptual Descriptions of Object Movements*. PhD thesis, Computer Science, University of Toronto, 1975.
- [5] K. Barnard, P. Duygulu, N. de Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *Journal of Machine Learning Research*, 3:1107–1135, 2003.
- [6] A. Cangelosi and D. Parisi. The processing of verbs and nouns in neural networks: Insights from synthetic brain imaging. *Brain and Language*, 89(2):401–408, 2004.
- [7] R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue. *Survey of the State of the Art in Human Language Technology*. Cambridge University Press, Cambridge; New York, 1996.
- [8] S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.
- [9] J. Daugman. Uncertainty relation for resolution in space, spatioil frequency, and orientation optimised by two-dimensional visual cortical filters. *Journal of the Optical Society of America*, 2(7):1160–1169, 1985.
- [10] M. I. Sessa F. Formato, G. Gerla. Similarity-based unification. *Fundamenta Informaticae*, 40:393 – 414, 2000.
- [11] A. Fern, R. Givan, and J. Siskind. Specific-to-general learning for temporal events with application to learning event definitions from video. *Journal of Artificial Intelligence Research*, 17:379–449, 2002.
- [12] J. L. Flanagan and L. R. Rabiner, editors. *Speech Synthesis*. Dowden Hutchinson and Ross Inc., Stroudburg, 1973.
- [13] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768, 1965.
- [14] S. Hargreaves-Heap and Y. Varoufakis. *Game Theory, A Critical Introduction*. Routledge, London, 1995.
- [15] S. Harnard. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
- [16] S. Hongeng. Unsupervised learning of multi-object event classes. In *15th British Machine Vision Conference (BMVC'04)*, pages 487–496, London, UK, 2004.
- [17] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.
- [18] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *Proc. International Conference on Parallel Processing*, volume 3, pages 113–122, 1995.
- [19] E. Keller. *Fundamentals of speech synthesis and speech recognition: basic concepts, state-of-the-art and future challenges*. John Wiley and Sons Ltd., Chichester, 1994.

- [20] H. Kollnig, H-H. Nagel, and M. Otte. Association of motion verbs with vehicle movements extracted from dense optical flow fields. In *Proc. of European Conference on Computer Vision*, volume 2, pages 338–347, 1994.
- [21] N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.
- [22] D. Magee, C. J. Needham, P. Santos, A. G. Cohn, and D. C. Hogg. Autonomous learning for a cognitive agent using continuous models and inductive logic programming from audio-visual input. In *Proceedings of the AAAI Workshop on Anchoring Symbols to Sensor Data*, 2004.
- [23] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 20(8):581–594, 2004.
- [24] A. R. McCallum. Hidden state and reinforcement learning with instance-based state identification. *IEEE Transactions on Systems, Man and Cybernetics (Special issue on Robot Learning)*, 26(3):464–473, 1996.
- [25] D. Moore and I. Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *Proc. AAAI National Conf. on Artificial Intelligence*, 2002.
- [26] S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, San Diego, 1992.
- [27] S. Muggleton. Learning from positive data. In S. Muggleton, editor, *ILP96*, volume 1314 of *Lecture Notes of Artificial Intelligence*, pages 358–376. SV, 1996.
- [28] S. Muggleton and J. Firth. CProlog4.4: a tutorial introduction. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, pages 160–188. Springer-Verlag, 2001.
- [29] H-H. Nagel. From image sequences towards conceptual descriptions. *Image and Vision Computing*, 6(2):59–74, 1988.
- [30] B. Neumann and T. Weiss. Navigating through logic-based scene models for high-level scene interpretations. In *3rd International Conference on Computer Vision Systems - ICVS 2003*, pages 212–222. Springer, 2003.
- [31] C. Pinhanez and A. Bobick. Interval scripts: a programming paradigm for interactive environments and agents. *Pervasive and Ubiquitous Computing*, 7(1):1–21, 2003.
- [32] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [33] L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *Algorithmic Learning Theory, 15th International Conference, ALT 2004*, pages 19–36, Padova, Italy, 2004.
- [34] D. Roy. Learning words and syntax for a visual description task. *Computer Speech and Language*, 16(3):353–385, 2002.

- [35] D. Roy and A. Pentland. Learning words from sights and sounds: A computational model. *Cognitive Science*, 26(1):113–146, 2002.
- [36] P. Santos, D. Magee, A. Cohn, and D. Hogg. Combining multiple answers for learning mathematical structures from visual observation. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI-04)*, pages 17–24, Valencia, Spain, 2004.
- [37] H. Shin and C. Kim. A simple yet effective technique for partitioning. *IEEE Transactions on VLSI Systems*, 1(3):380–386, 1993.
- [38] J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90, 2001.
- [39] A. R. Smith. Color gamut transform pairs. *Computer Graphics*, 12(3):12–19, 1978.
- [40] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.
- [41] J. Tsotsos, J. Mylopoulos, H. D. Covvey, and S. W. Zucker. A framework for visual motion understanding. *IEEE Pattern Analysis and Machine Intelligence*, pages 563–573, 1980. Special Issue on Computer Analysis of Time-Varying Imagery.
- [42] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book*. Microsoft Corporation, 2000.