Learning Temporal, Relational, Force-Dynamic Event Definitions from Video*

Alan Fern and **Jeffrey Mark Siskind** and **Robert Givan**
School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907–1285 USA
{afern,qobi,givan}@purdue.edu

Abstract

We present and evaluate a novel implemented approach for learning to recognize events in video. First, we introduce a sublanguage of event logic, called k-AMA, that is sufficiently expressive to represent visual events yet sufficiently restrictive to support learning. Second, we develop a specific-to-general learning algorithm for learning event definitions in k-AMA. Finally, we apply this algorithm to the task of learning event definitions from video and show that it yields definitions that are competitive with hand-coded ones.

Introduction

Humans conceptualize the world in terms of objects and events. This is reflected in the fact that we talk about the world using nouns and verbs. We perceive events taking place between objects, we interact with the world by performing events on objects, and we reason about the effects that actual and hypothetical events performed by us and others have on objects. We also *learn* new object and event types from novel experience. In this paper, we present and evaluate novel implemented techniques that allow a computer to learn to recognize new event types from video input.

We wish the acquired knowledge of event types to support multiple modalities. Humans can observe someone *fax*ing a letter for the first time and quickly be able to recognize future occurrences of faxing, perform faxing, and reason about faxing. It thus appears likely that humans use and learn event representations that are sufficiently general to support fast and efficient use in multiple modalities. A long-term goal of our research is to allow similar cross-modal learning and use of event representations. We intend the same learned representations to be used for vision (as described in this paper), planning (something that we are beginning to investigate), and robotics (something left to the future).

A crucial requirement for event representations is that they capture the *invariants* of an event type. Humans classify both picking up a cup off a table and picking up a dumbbell off the floor as *picking up*. This suggests that human event representations are *relational*. We have an abstract relational notion of *picking up* that is parameterized by the participant objects rather than distinct propositional notions instantiated for specific objects. Humans also classify an event as *picking up* no matter whether the hand is moving slowly or quickly, horizontally or vertically, leftward or rightward, or along a straight path or circuitous one. It appears that it is not the characteristics of participant-object motion that distinguish *picking up* from other event types. Rather, it is the fact that the object being picked up changes from being supported by resting on its initial location to be supported by being grasped by the agent. This suggests that the primitive relations used to build event representations are *force dynamic* (Talmy 1988). Finally, humans distinguish between picking up and putting down a cup despite the fact that both before putting down the cup and after picking up the cup the same state of affairs holds, namely the human is holding the cup. This suggests that event representations are *temporal*. The order of world states matters when defining event types.

Another desirable property of event representations is that they be *perspicuous*. Humans can introspect and describe the defining characteristics of event types. Such introspection is what allows us to create dictionaries. To support such introspection, the representation language should allow such characteristics to be explicitly manifest in event definitions and not be emergent consequences of distributed parameters as in neural networks or hidden Markov models.

We present a novel system that learns to recognize events from video input using temporal, relational, force-dynamic representations. This is not the first system to perform visual event recognition. We review prior work and compare it to the current work later in the paper. In fact, one of us has built two such prior systems. HOWARD (Siskind & Morris 1996) learns to classify events from video using temporal, relational representations. But these representations are not force dynamic. LEONARD (Siskind 2001) classifies events from video using temporal, relational, force-dynamic representations but does not learn these representations. It uses a library of hand-code representations. This work adds a learning component to LEONARD. We describe the representation language used to support learning and the algorithms used to learn representations in that language. We

*This work was supported in part by NSF grants 9977981-IIS and 0093100-IIS, an NSF Graduate Fellowship for Fern, and the Center for Education and Research in Information Assurance and Security at Purdue University. Part of this work was performed while Siskind was at NEC Research Institute, Inc.

also evaluate the performance of the approach and compare the performance of learned representations to the prior hand-coded representations. A companion paper (Fern, Givan, & Siskind 2002) addresses theoretical issues of the approach, elaborating on the algorithms, proving their correctness, and analyzing their complexity.

Representing Event Types

LEONARD uses *event logic* (Siskind 2001) to represent event types. An event-logic expression Φ is either a primitive event type or one of $\neg\Phi$, $\Diamond_R\Phi$, $\Phi_1 \vee \Phi_2$, $\Phi_1 \wedge_R \Phi_2$, Φ^+, $\forall x\Phi$, or $\exists x\Phi$, where x is a variable and R is a subset of interval relations (Allen 1983). In this paper, we use only the connective $\wedge_R$ with R being either $\{\mathsf{m}\}$ or $\{=\}$. $\Phi \wedge_R \Psi$ denotes occurrence of Φ and Ψ during intervals related by a relation in R. $\Phi \wedge_{\{=\}} \Psi$, denoting simultaneous occurrence of Φ and Ψ, and $\Phi \wedge_{\{\mathsf{m}\}} \Psi$, denoting occurrence of Φ followed immediately by occurrence of Ψ, are sufficiently common that we abbreviate them as $\Phi \wedge \Psi$ (without a subscript) and $\Phi;\Psi$ respectively.

LEONARD represents event types with definitions in event logic like the following:

$$\textsc{PickUp}(x,y,z) \triangleq \left[\begin{array}{l}(\textsc{Supports}(z,y) \wedge \textsc{Contacts}(z,y));\\ (\textsc{Supports}(x,y) \wedge \textsc{Attached}(x,y))\end{array}\right]$$

This means that an event of x picking up y off of z is defined as a sequence of two states where z supports y by way of contact in the first and x supports y by way of attachment in the second. SUPPORTS, CONTACTS, and ATTACHED are primitive force-dynamic relations. These are recovered from video input by a force-dynamic model-reconstruction process (Siskind 2000). Events occurrences are inferred from these primitive force-dynamic relations using event definitions like the above (the actual definitions are more complex) by an event-classification process. Previously, these definitions were hand coded. In the work reported here, we learn these definitions by applying the same LEONARD force-dynamic model-reconstruction process to training videos and inducing definitions, using the novel techniques presented in this paper, from the primitive force-dynamic relations produced by model reconstruction. We then evaluate these definitions by using them to classify test videos processed with the standard LEONARD model-reconstruction and event-classification processes.

We determined that only a subset of the full event-logic language was used in the hand-coded definitions. This subset proved both to support a learning algorithm with certain properties (see the companion paper) and to be an effective restrictive learning bias (as demonstrated by the experimental results described later in this paper). We call this sublanguage AMA. An *A formula* (intuitively 'and') is a conjunction $(L_1 \wedge \cdots \wedge L_l)$ of literals L_i. For now, these literals are primitive force-dynamic relations. The theoretical results in the companion paper are limited to positive literals. In the experimental results section, we discuss extensions to support negative literals. An *MA formula* (intuitively 'meets-and') is a sequence $(A_1;\ldots;A_m)$ of A formulas A_i. An *AMA formula* (intuitively 'and-meets-and') is a conjunction $(M_1 \wedge \cdots \wedge M_n)$ of MA formulas M_i. The above definition for PICKUP is an MA formula (and hence an AMA formula). Intuitively, A formulas describe states of the world, MA formulas describe sequences of states, what we call *timelines*, and AMA formulas describe an event in terms of multiple timelines that it must satisfy.

Learning Event Definitions

We adopt a specific-to-general ILP approach to learning event definitions from positive-only data. The AMA language is suitable for this approach because of three key properties. First, the least general MA formula (and hence AMA formula) that covers any force-dynamic model produced by model reconstruction exists and is unique up to semantic equivalence. We call these *model covers*. Second, the least general AMA generalization (LGG) of two AMA formulas exists and is unique up to semantic equivalence. Third, we have a tractable syntactic notion of generalization for AMA, refining semantic generalization, for which the corresponding LGG of two formulas exists and is unique up to mutual syntactic generalization. (See the companion paper for discussion and proof of these properties.) We refer to this "syntactic LGG" as the "LGG" from here on[1]. Our learning algorithm operates by producing model covers for the training examples and folding the binary LGG operator over these to yield the least general AMA formula that covers the training set. This works because the binary LGG operator is commutative and associative.

A key characteristic of our domain that allows the above properties to hold is that our primitive event types are *liquid* (Shoham 1987), i.e. if a primitive holds of an interval, it holds of every subinterval of that interval. This means that for any A formula A, the MA formulas A, $A;A$, $A;A;A$, ... are equivalent. We use this in finding model covers and computing the LGG of two AMA formulas.

Before presenting the algorithms to do so, we define some notation. An MA formula $(A_1;\ldots;A_m)$ has a set of $m+1$ *transition points* t_i, one at the beginning, one between each pair of adjacent A formulas, and one at the end. This set is finite and totally ordered, yielding an adjacency relation. A *segment* $A_{t_it_{i+1}}$ denotes the A formula between a pair $\langle t_i, t_{i+1}\rangle$ of adjacent transition points. An MA formula corresponds to a transition-point set and a set of segments between the pairs of adjacent transition points. And vice versa.

First, we show how to compute a model cover. The output of model reconstruction is a set of primitive event occurrences $L@[s_1,s_2]$ where L is a primitive event type and $[s_1,s_2]$ is a maximal interval during which it occurred. Because such primitive event types are liquid, occurrence during all subintervals of $[s_1,s_2]$ is implicit. Take the set of s_i that appear in such primitive event occurrences as a set of transition points, ordered by the $<$ relation on time instants. For every pair $\langle t_i, t_{i+1}\rangle$ of adjacent transition points, take $A_{t_it_{i+1}}$ to be the conjunction of all primitive event types L where $L@[s_1,s_2]$ is a primitive event occurrence, $s_1 \leq t_i$,

[1] Our algorithm for syntactic LGG is exponentially more efficient than our best algorithm for the true semantic LGG.

and $t_{i+1} \leq s_2$. This yields a model cover. Note that constructing $A_{t_i t_{i+1}}$ requires primitive-event liquidity.

Next, we show how to compute the LGG of two AMA formulas. First, consider two MA formulas M_1 and M_2. Let T_1 and T_2 be the transition-point sets of M_1 and M_2 respectively. A *proto-interdigitation* T' of T_1 and T_2 is $T_1 \cup T_2$ along with a total, reflexive, transitive 'ordering' relation that satisfies the following: (a) the order on T' is consistent with the orders on T_1 and T_2, (b) no two elements of T_1 (T_2) are equated, and (c) the minimal (maximal) elements of T_1 and T_2 are equated. An *interdigitation* T of T_1 and T_2 is the partition of a proto-interdigitation under equality.

Given this, the LGG of two MA formulas M_1 and M_2 can be computed as follows. Let T_1 and T_2 be the transition point sets of M_1 and M_2 respectively. Let $A^1_{t^1_i t^1_{i+1}}$ be the segment in M_1 between transition points t^1_i and t^1_{i+1} in T_1. Similarly, let $A^2_{t^2_i t^2_{i+1}}$ be the segment in M_2 between transition points t^2_i and t^2_{i+1} in T_2. Let T be an interdigitation of T_1 and T_2. T can be interpreted as a transition-point set where the components are interpreted as transition points ordered by the interdigitation relation. Compute the segments of T as follows. For each transition point t_i in T, let $f_1(t_i)$ be the latest member of T_1 occurring in T no later than t_i. Define $f_2(t_i)$ similarly for T_2. Dually, define $g_1(t_i)$ to be the earliest member of T_1 occurring in T no earlier than t_i. Define $g_2(t_i)$ similarly for T_2. Now, for every pair $\langle t_i, t_{i+1} \rangle$ of adjacent transition points in T take $A_{t_i t_{i+1}}$ to be the intersection of $A^1_{f_1(t_i)\, g_1(t_{i+1})}$ and $A^2_{f_2(t_i)\, g_2(t_{i+1})}$. This yields an MA formula that generalizes M_1 and M_2. The conjunction of all such formulas derived from all interdigitations is the LGG of M_1 and M_2. The conjunction of the LGGs of all pairs of MA formulas M_1 and M_2 in the AMA formulas Φ_1 and Φ_2 respectively is the LGG of Φ_1 and Φ_2. Note that constructing $A_{t_i t_j}$, again, requires primitive-event liquidity. Timelines in the LGG that subsume other such timelines can be pruned in polynomial time using an algorithm presented in our companion paper.

This LGG computation is clearly exponential in the input size. The companion paper shows that the size of the *smallest* LGG of two AMA formulas, which may be smaller than the LGG computed here, can be exponential in the size of the input formulas, so no better worst-case bound can be achieved. This motivates restricting the AMA language to k-AMA: AMA formulas that contain only k-MA formulas, those with no more than k A formulas.

There is also a learning motivation for the k-AMA restriction: the hypothesis space grows exponentially as k is relaxed and yet the AMA conjunction of k-MA timelines can capture much of the structure that formulas with longer MA timelines capture. Thus, we study k-AMA rather than unbounded single MA timelines as hypotheses.

The *k-cover* of an AMA formula Φ is the least general k-AMA formula Φ' that covers Φ. We can compute the k-cover of Φ as follows. First consider MA formulas M with more than k A formulas. A *k-digitation* of M is a subset T' of the transition-point set T of M of size no greater than $k+1$ that contains the minimal and maximal elements of T and is ordered by the same ordering relation as T. Form a new MA formula M' whose transition points are a k-digitation of M. Each segment $A'_{t'_i t'_{i+1}}$ of M' is the intersection of all segments $A_{t_j t_{j+1}}$ of M where $t'_{i'} \leq t_j$ and $t'_{j'} \leq t_j$ in T. M' generalizes M and is k-MA. The k-cover of an MA formula M with no more than k A formulas is M itself. Otherwise, it is the conjunction of all such M' for all k-digitations of M. The k-cover of an AMA formula Φ is the conjunction of the k-covers of the MA formulas in Φ. We note that the k-cover of an AMA formula Φ may be exponentially larger than Φ. However, in practice, after pruning redundant k-MA formulas, we have found that k-covers do not exhibit undue size growth.

Given this, we restrict our learner to k-AMA by computing the k-cover of the output of both the model cover computation and the LGG computation each time it is performed.

Experiments Results

Prior to the work reported in this paper, LEONARD needed hand-coded event definitions. We have augmented LEONARD with the ability to learn definitions using our k-AMA learning algorithm and evaluated its performance.[2] LEONARD is a three-stage pipeline. The raw input consists of a video-frame sequence depicting events. First, a segmentation-and-tracking component transforms this input into a polygon movie: a sequence of frames, each frame being a set of convex polygons placed around the tracked objects in the video. Next, a model-reconstruction component transforms the polygon movie into a force-dynamic model. This model describes the changing support, contact, and attachment relations between the tracked objects over time. Finally, an event-recognition component determines which events, from a library of event definitions, occurred in the model and, accordingly, in the video. The learning process uses the early stages of the LEONARD pipeline to produce force-dynamic models from the training movies and applies the k-AMA learning algorithm to these models.

Relational Data

LEONARD produces relational models that involve objects and (force dynamic) relations between those objects. Thus event definitions include variables to allow generalization over objects. For example, a definition for PICKUP(x, y, z) recognizes both PICKUP(**hand**, **block**, **table**) as well as PICKUP(**man**, **dumbbell**, **floor**). Despite the fact that our k-AMA learning algorithm is propositional, we are still able to use it to learn relational definitions.

We take a straightforward object-correspondence approach to relational learning. We view the models output by LEONARD as containing relations applied to constants. Since we (currently) support only supervised learning, we have a set of distinct training examples for each event type. There is an implicit correspondence between the objects filling the same role across

[2] The code and data set reported here is available from `ftp://dynamo.ecn.purdue.edu/pub/qobi/ama.tar.Z`.

the different training models for a given type. For example, models showing PICKUP(**hand**, **block**, **table**) and PICKUP(**man**, **dumbbell**, **floor**) have implicit correspondences ⟨**hand**, **man**⟩, ⟨**block**, **dumbbell**⟩, and ⟨**table**, **floor**⟩. We outline two relational learning methods that differ in how much object-correspondence information they require.

Complete Object Correspondence This first approach assumes that a complete object correspondence is given, as input, along with the training examples. Given such information, we can propositionalize the training models by replacing corresponding objects with unique constants. The propositionalized models are then given to our propositional k-AMA learning algorithm which returns a propositional k-AMA formula. We then lift this propositional formula by replacing each constant with a distinct variable. Lavrac, Dzeroski, & Grobelnik (1991) took a similar approach.

Partial Object Correspondence The above approach assumes complete object-correspondence information. While it is sometimes possible to provide all correspondences (for example, by color-coding objects that fill identical roles when recording training movies), such information is not always available. When only a partial (or even no) object correspondence is available, we can automatically complete the correspondence and apply the above technique.

For the moment, assume that we have an evaluation function that takes two relational models and a candidate object correspondence, as input, and yields an evaluation of correspondence quality. Given a set of training examples with missing object correspondences, we perform a greedy search for the best set of object-correspondence completions over the models. Our method works by storing a set P of propositionalized training examples (initially empty) and a set U of unpropositionalized training examples (initially the entire training set). For the first step, when P is empty, we evaluate all pairs of examples from U, under all possible correspondences, select the two that yield the highest score, remove them from U, propositionalize them according to the best correspondence, and add them to P. For each subsequent step, we use the previously computed values of all pairs of examples, one from U and one from P, under all possible correspondences. We then select the example from U along with the correspondence that yields the highest average score relative to all models in P. This example is removed from U, propositionalized according to the winning correspondence, and added to P. For a fixed number of objects, the effort expended is polynomial in the size of the training set. However, if the number of objects b in a training example grows, the number of correspondences grows as b^b. Thus, it is important that the events involve only a modest number of objects.

Our evaluation function is based on the intuition that object roles for visual events (as well as events from other domains) can often be inferred by considering the changes between the initial and final moments of an event. Specifically, given two models and an object correspondence, we first propositionalize the models according to the correspondence. Next, we compute ADD and DELETE lists for each model. The ADD list is the set of propositions that are true at the final moment but not the initial moment. The DELETE list is the set of propositions that are true at the initial moment but not the final moment. (These add and delete lists are motivated by STRIPS action representations.) Given such ADD_i and DELETE_i lists for models 1 and 2, the evaluation function returns the sum of the cardinalities of $\text{ADD}_1 \cap \text{ADD}_2$ and $\text{DELETE}_1 \cap \text{DELETE}_2$. This measures the similarity between the ADD and DELETE lists of the two models. We have found that this evaluation function works well in the visual-event domain.

Negative Information

The AMA language does not allow negated propositions. Negation, however, is sometimes necessary to adequately define an event class. It turns out that we can easily get the practical advantages of negation without incorporating negation into the AMA language. We do this by appending a new set of propositions to our models that intuitively represents the negation of each proposition. Assume the training examples contain the propositions $\{p_1, \ldots, p_n\}$. We introduce a new set $\{\bar{p}_1, \ldots, \bar{p}_n\}$ of propositions and add these into the training models. It is a design choice as to how we assign truth values to these new propositions.

In our experiments, we compare two methods for assigning a truth value to $\bar{p}_i$. The first method, called *full negation*, assigns true to $\bar{p}_i$ in a model iff p_i is false in the model. The second method, called *boundary negation*, differs from full negation in that it only allows $\bar{p}_i$ to be true in the initial and final moments of a model. $\bar{p}_i$ must be false at all other times. We have found that boundary negation provides a good trade-off between no negation, which often produces overly general results, and full negation, which often produces overly specific and much more complicated results. Both methods share the property that they produce models where p_i and $\bar{p}_i$ are never simultaneously true. Thus our learning methods will never produce formulas with states that contain both p_i and $\bar{p}_i$.

Data Set

Our data set contains examples of 7 different event classes: *pick up*, *put down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble*. Each of these involve a hand and two to three blocks. For a detailed description of these event classes, see Siskind (2001). Key frames from sample video sequences of these event classes are shown in figure 1. The results of segmentation, tracking, and model reconstruction are overlayed on the video frames. We recorded 30 movies for each of the 7 event classes resulting in a total of 210 movies comprising 11946 frames. We replaced one movie, `assemble-left-qobi04`, with a duplicate copy of `assemble-left-qobi11`, because of segmentation and tracking errors.

Some of the events classes are hierarchical in that occurrences of events in one class contain occurrences of events in one or more simpler classes. For example, a movie depicting a MOVE(a, b, c, d) event (i.e. a moves b from c to d) contains subintervals where PICKUP(a, b, c) and PUTDOWN(a, b, d) events occur. In evaluating the learned definitions, we wish to detect both the events

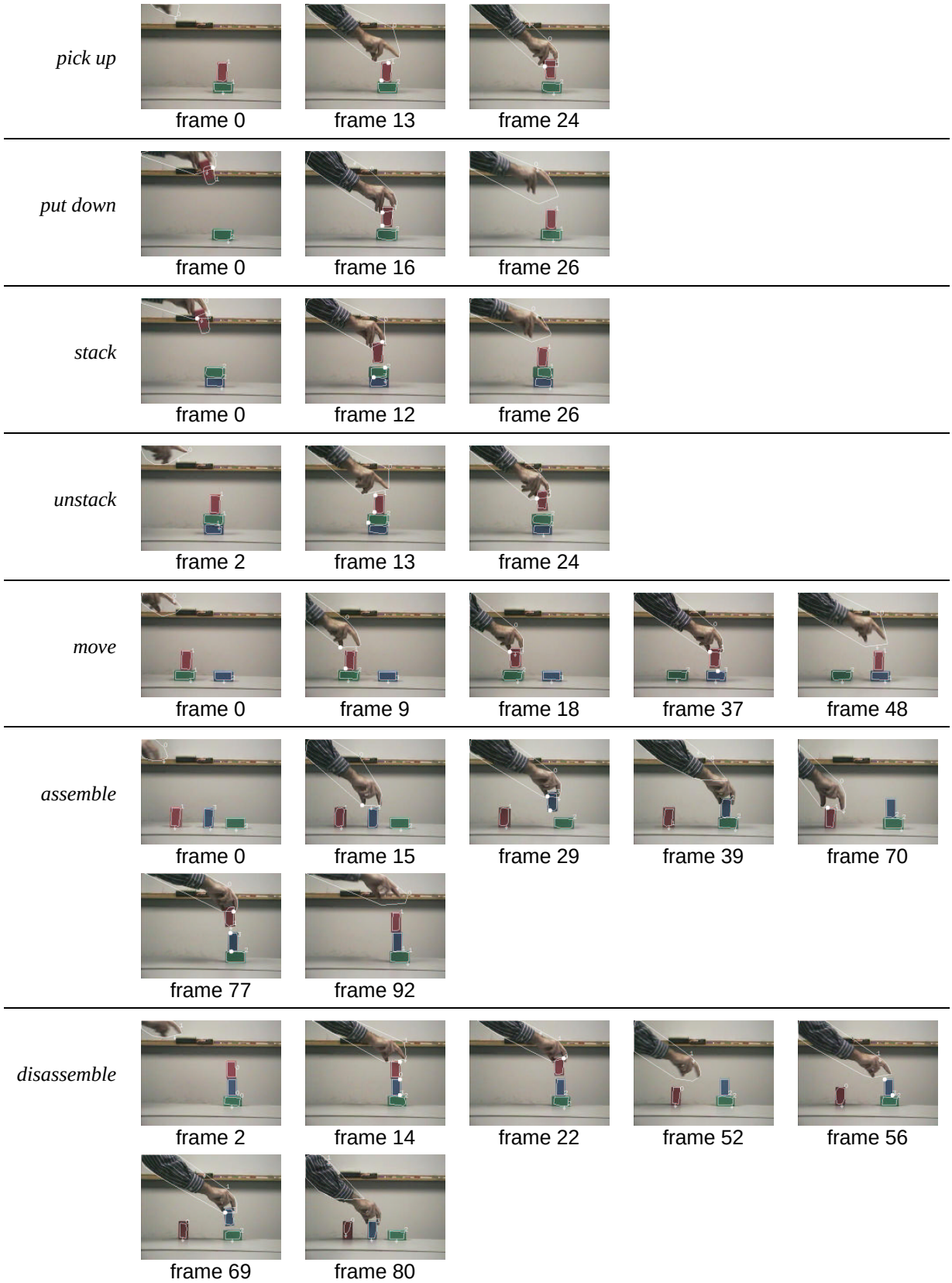

Figure 1: Key frames from sample videos of the seven event types.

that correspond to an entire movie as well as subevents that correspond to portions of that movie. For example, given a movie depicting MOVE(a, b, c, d), we wish to detect not only the MOVE(a, b, c, d) event but also the PICKUP(a, b, c) and PUTDOWN(a, b, c) subevents as well. For each movie type in our data set, we have a set of *intended* events and subevents that should be detected. If a definition does not detect an intended event, we deem the error a false negative. If a definition detects an untended event, we deem the error a false positive. For example, if a movie depicts a MOVE(a, b, c, d) event, the intended events are MOVE(a, b, c, d), PICKUP(a, b, c), and PUTDOWN(a, b, c). If the definition for *pick up* detects the occurrence of PICKUP(c, b, a) and PICKUP(b, a, c), but not PICKUP(a, b, c), it will be charged two false positives as well as one false negative. We evaluate our definitions in terms of false positive and negative rates as describe below.

Experimental Procedure

For each event class, we evaluate the k-AMA learning algorithm using a leave-one-movie-out cross-validation technique with training-set sampling. The parameters to our learning algorithm are k and the degree D of negative information used: either positive propositions only (P), boundary negation (NPN), or full-negation (N). The parameters to our evaluation procedure include the target event class E and the training-set size N. Given this information, the evaluation proceeds as follows: For each movie M (the held-out movie) from the 210 movies, apply the k-AMA learning algorithm to a randomly drawn training sample of N movies from the 30 movies of event class E (or 29 movies if M is one of the 30). Use LEONARD to detect all occurrences of the learned event definition in M. Based on E and the event class of M, record the number of false positives and false negatives in M, as detected by LEONARD. Let FP and FN be the total number of false positives and false negatives observed over all 210 held-out movies respectively. Repeat the entire process of calculating FP and FN 10 times and record the averages as $\overline{\text{FP}}$ and $\overline{\text{FN}}$.

Since some event classes occur more frequently in our data than others (because simpler events occur as subevents of more complex events but not vice versa), we do not report $\overline{\text{FP}}$ and $\overline{\text{FN}}$ directly. Instead, we normalize $\overline{\text{FP}}$ relative to the number of times LEONARD detected the target event within all 210 movies and $\overline{\text{FN}}$ relative to human assessment of the total number of occurrences of the target event within all 210 movies. The normalized value of $\overline{\text{FP}}$ estimates the probability that the target event did not occur, given that it was predicted to occur, while the normalized value of $\overline{\text{FN}}$ estimates the probability that the event was not predicted to occur, given that it did occur.

Results

To evaluate our k-AMA learning approach, we ran leave-one-movie-out experiments, as described above, for varying k, D, and N. The 210 example movies were recorded with color-coded objects to provide complete object-correspondence information. We compared our learned event definitions to the performance of two sets of hand-coded definitions. The first set HD_1 of hand-coded definitions appeared in Siskind (2001). We manually revised these to yield another set HD_2 of hand-coded definitions that gives a significantly better $\overline{\text{FN}}$ at a cost in $\overline{\text{FP}}$ performance. Figure illustrates sample definitions for PICKUP(x, y, z) from HD_1 and HD_2 along with a sample machine-generated definition produced by our method.

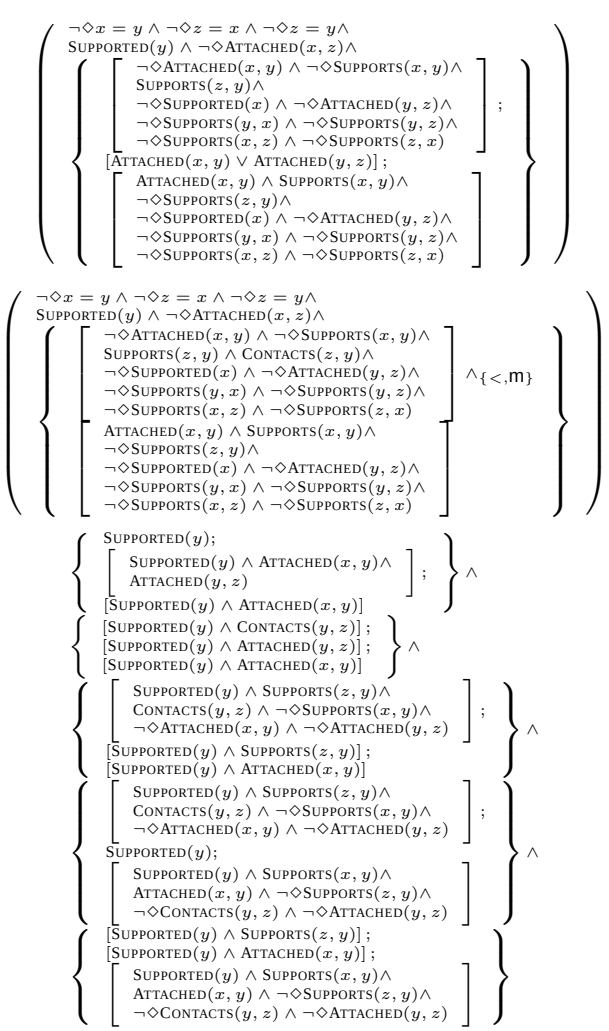

Figure 2: The definitions for PICKUP(x, y, z) in HD_1 and HD_2 along with the machine-generated definition for $k = 3$ and $D =$ NPN produced by training on all 30 *pick up* movies.

Object Correspondence To evaluate our algorithm for finding object correspondences, we ignored the correspondence information provided by color coding and applied the algorithm to all training models for each event class. The algorithm selected the correct correspondence for all 210 training models. Thus, for this data set, the learning results will be identical regardless of whether or not correspondence information is provided manually. In light of this, the rest of our experiments use the manual correspondence informa-

k	D		*pu*	*pd*	*st*	*un*	*mo*	*as*	*di*
2	NPN	$\overline{\text{FP}}$	0	0.14	0	0	0	0.75	0
		$\overline{\text{FN}}$	0	0.19	0.12	0.03	0	0	0
3	NPN	$\overline{\text{FP}}$	0	0	0	0	0	0	0
		$\overline{\text{FN}}$	0	0.2	0.45	0.10	0.03	0.07	0.10
4	NPN	$\overline{\text{FP}}$	0	0	0	0	0	0	0
		$\overline{\text{FN}}$	0	0.2	0.47	0.12	0.03	0.07	0.17
3	P	$\overline{\text{FP}}$	0.42	0.5	0	0.02	0	0	0
		$\overline{\text{FN}}$	0	0.19	0.42	0.11	0.03	0.03	0.10
3	N	$\overline{\text{FP}}$	0	0	0	0	0	0	0
		$\overline{\text{FN}}$	0.04	0.39	0.58	0.16	0.13	0.2	0.2
3	NPN	$\overline{\text{FP}}$	0	0	0	0	0	0	0
		$\overline{\text{FN}}$	0	0.2	0.45	0.10	0.03	0.07	0.10
HD_1		$\overline{\text{FP}}$	0.01	0.01	0	0	0	0	0
		$\overline{\text{FN}}$	0.02	0.22	0.82	0.62	0.03	1.0	0.5
HD_2		$\overline{\text{FP}}$	0.13	0.11	0	0	0	0	0
		$\overline{\text{FN}}$	0.0	0.19	0.42	0.02	0.0	0.77	0.0

Table 1: $\overline{\text{FP}}$ and $\overline{\text{FN}}$ for both learned definitions, varying both k and D, and hand-coded definitions.

tion, provided by color-coding, rather than recomputing it.

Varying k The first three rows of table 1 show the $\overline{\text{FP}}$ and $\overline{\text{FN}}$ values for all 7 event classes for $k \in \{2, 3, 4\}$, $N = 29$ (the maximum), and $D = \text{NPN}$. Similar trends were found for $D = \text{P}$ and $D = \text{N}$. The general trend is that, as k increases, $\overline{\text{FP}}$ decreases and $\overline{\text{FN}}$ increases. Such a trend is a consequence of our k-cover approach. This is because, as k increases, the k-AMA language contains strictly more formulas. Thus for $k_1 > k_2$, the k_1-cover will never be more general than the k_2-cover. This strongly suggests (but does not prove) that $\overline{\text{FP}}$ will be non-increasing with k and $\overline{\text{FN}}$ will be non-decreasing with k.

Our results show that 2-AMA is overly general for *put down* and *assemble*, i.e. it gives high $\overline{\text{FP}}$. In contrast, 4-AMA is overly specific, as it achieves $\overline{\text{FP}} = 0$ for each event class but with a significant penalty in $\overline{\text{FN}}$. 3-AMA appears to provide with a good trade-off between the two, achieving $\overline{\text{FP}} = 0$ for each event class, while yielding reasonable $\overline{\text{FN}}$.

Varying D Rows four through sixth of table 1 show $\overline{\text{FP}}$ and $\overline{\text{FN}}$ for all 7 event classes for $D \in \{\text{P}, \text{NPN}, \text{N}\}$, $N = 29$, and $k = 3$. Similar trends were observed for other values of k. The general trend is that, as the degree of negative information increases, the learned event definitions become more specific. In other words, $\overline{\text{FP}}$ decreases and $\overline{\text{FN}}$ increases. This makes sense since, as more negative information is added to the training models, more specific structure can be found in the data and exploited by the k-AMA formulas. We can see that, with $D = \text{P}$, the definitions for *pick up* and *put down* are overly general, as they produce high $\overline{\text{FP}}$. Alternatively, with $D = \text{N}$, the learned definitions are overly specific, giving $\overline{\text{FP}} = 0$, at the cost of high $\overline{\text{FN}}$. In these experiments, as well as others, we have found that $D = \text{NPN}$ yields the best of both worlds: $\overline{\text{FP}} = 0$ for all event classes and lower $\overline{\text{FN}}$ than achieved with $D = \text{N}$.

Experiments not shown here have demonstrated that, without negation for *pick up* and *put down*, we can increase k arbitrarily, in an attempt to specialize the learned definitions, and never significantly reduce $\overline{\text{FP}}$. This indicates that negative information plays a crucial role in constructing definitions for these event classes.

Comparison to Hand-Coded Definitions The bottom two rows of table 1 show the results for HD_1 and HD_2. We have not yet attempted to automatically select the parameters for learning (i.e. k and D). Rather, here we focus on comparing the hand-coded definitions to the parameter set that we judged to be best performing across all event classes. We believe, however, that these parameters could be selected reliably using cross-validation techniques on a larger data set. In that case, the parameters would be selected on a per–event-class basis and would likely result in an even more favorable comparison to the hand-coded definitions.

The results show that the learned definitions significantly outperform HD_1 on the current data set. The HD_1 definitions were found to produce a large number of false negatives on the current data set. Manual revision of HD_1 yielded HD_2. Notice that, although HD_2 produces significantly fewer false negatives for all event classes, it produces more false positives for *pick up* and *put down*. This is because the hand definitions utilize *pick up* and *put down* as macros for defining the other events.

The performance of the learned definitions is competitive with the performance of HD_2. The main differences in performance are: (a) for *pick up* and *put down*, the learned and HD_2 definitions achieve nearly the same $\overline{\text{FN}}$ but the learned definitions achieve $\overline{\text{FP}} = 0$ whereas HD_2 has significant $\overline{\text{FP}}$, (b) for *unstack* and *disassemble*, the learned definitions perform moderately worse than HD_2 with respect to $\overline{\text{FN}}$, and (c) the learned definitions perform significantly better than HD_2 on *assemble* events.

We conjecture that further manual revision could improve HD_2 to perform as well as (and perhaps better than) the learned definitions for every event class. Nonetheless, we view this experiment as promising, as it demonstrates that our learning technique is able to compete with, and sometimes outperform, hand-coded definitions.

Varying N It is of practical interest to know how training set size affects our algorithm's performance. For this application, it is important that our method work well with fairly small data sets, as it can be tedious to collect event data.

Table 2 shows the $\overline{\text{FN}}$ of our learning algorithm for each event class, as N is reduced from 29 to 5. For these experiments, we used $k = 3$ and $D = \text{NPN}$. Note that $\overline{\text{FP}} = 0$ for all event classes and all N and hence is not shown. We expect $\overline{\text{FN}}$ to increase as N is decreased, since, with specific-to-general learning, more data yields more-general definitions. Generally, $\overline{\text{FN}}$ is flat for $N > 20$, increases slowly for $10 < N < 20$, and increases abruptly for $5 < N < 10$. We also see that, for several event classes, $\overline{\text{FN}}$ decreases slowly, as N is increased from 20 to 29. This indicates that a larger data set might yield improved results.

Related Work

Prior work has investigated various subsets of the pieces of learning and using temporal, relational, and force-dynamic

N	*pu*	*pd*	*st*	*un*	*mo*	*as*	*di*
29	0.0	0.20	0.45	0.10	0.03	0.07	0.10
25	0.0	0.20	0.47	0.16	0.05	0.09	0.10
20	0.01	0.21	0.50	0.17	0.08	0.12	0.12
15	0.01	0.22	0.53	0.26	0.14	0.20	0.16
10	0.07	0.27	0.60	0.36	0.23	0.32	0.26
5	0.22	0.43	0.77	0.54	0.35	0.57	0.43

Table 2: $\overline{\text{FN}}$ for $k = 3$, $D =$ NPN, and various values of N.

representations for recognizing events in video. But none, to date, combine all the pieces together. The following is a representative list and not meant to be comprehensive. Borchardt (1985) presents temporal, relational, force-dynamic event definitions but these definitions are neither learned nor applied to video. Regier (1992) presents techniques for learning temporal event definitions but the learned definitions are neither relational, force dynamic, nor applied to video. Yamoto, Ohya, & Ishii (1992), Starner (1995), Brand & Essa (1995), Brand, Oliver, & Pentland (1997), and Bobick & Ivanov (1998) present techniques for learning temporal event definitions from video but the learned definitions are neither relational nor force dynamic. Pinhanez & Bobick (1995) and Brand (1997a) present temporal, relational event definitions that recognize events in video but these definitions are neither learned nor force dynamic. Brand (1997b) and Mann & Jepson (1998) present techniques for analyzing force dynamics in video but neither formulate event definitions nor apply these techniques to recognizing events or learning event definitions.

Conclusion

We have presented k-AMA, a novel restrictive bias on event logic, along with a novel learning algorithm for that hypothesis space. This language is sufficiently expressive to support learning temporal, relational, force-dynamic event definitions from video. To date, however, the definitions are neither cross-modal nor perspicuous. And while the performance of learned definitions matches that of hand-coded ones, we wish to surpass hand coding. In the future, we intend to address cross-modality by applying k-AMA learning to the planning domain. And we believe that addressing perspicuity will lead to improved performance.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Bobick, A. F., and Ivanov, Y. A. 1998. Action recognition using probabilistic parsing. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 196–202.

Borchardt, G. C. 1985. Event calculus. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 524–527.

Brand, M., and Essa, I. 1995. Causal analysis for visual gesture understanding. In *Proceedings of AAAI Fall Symposium on Computational Models for Integrating Language and Vision*.

Brand, M.; Oliver, N.; and Pentland, A. 1997. Coupled hidden Markov models for complex action recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Brand, M. 1997a. The inverse hollywood problem: From video to scripts and storyboards via causal analysis. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 132–137.

Brand, M. 1997b. Physics-based visual understanding. *Computer Vision and Image Understanding* 65(2):192–205.

Fern, A.; Givan, R.; and Siskind, J. M. 2002. Specific-to-general learning for temporal events. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*.

Lavrac, N.; Dzeroski, S.; and Grobelnik, M. 1991. Learning nonrecursive definitions of relations with LINUS. In *Proceedings of the Fifth European Working Session on Learning*, 265–288.

Mann, R., and Jepson, A. D. 1998. Toward the computational perception of action. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 794–799.

Pinhanez, C., and Bobick, A. 1995. Scripts in machine understanding of image sequences. In *AAAI Fall Symposium Series on Computational Models for Integrating Language and Vision*.

Regier, T. P. 1992. *The Acquisition of Lexical Semantics for Spatial Terms: A Connectionist Model of Perceptual Categorization*. Ph.D. Dissertation, University of California at Berkeley.

Shoham, Y. 1987. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence* 33(1):89–104.

Siskind, J. M., and Morris, Q. 1996. A maximum-likelihood approach to visual event classification. In *Proceedings of the Fourth European Conference on Computer Vision*, 347–360. Cambridge, UK: Springer-Verlag.

Siskind, J. M. 2000. Visual event classification via force dynamics. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 149–155.

Siskind, J. M. 2001. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research* 15:31–90.

Starner, T. E. 1995. Visual recognition of American Sign Language using hidden Markov models. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Talmy, L. 1988. Force dynamics in language and cognition. *Cognitive Science* 12:49–100.

Yamoto, J.; Ohya, J.; and Ishii, K. 1992. Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings of the 1992 IEEE Conference on Computer Vision and Pattern Recognition*, 379–385. IEEE Press.