Symbolic Probabilistic Reasoning for Narratives

Hannaneh Hajishirzi and Erik T. Mueller

hajishir@illinois.edu,etm@us.ibm.com
University of Illinois at Urbana-Champaign, IBM TJWatson

Abstract

We present a framework to represent and reason about narratives that combines logical and probabilistic representations of commonsense knowledge. Unlike most natural language understanding systems which merely extract facts or semantic roles, our system builds probabilistic representations of the temporal sequence of world states and events implied by a narrative. We use probabilistic transitions to represent ambiguities and uncertainties in the narrative sentences. We present exact and approximate reasoning algorithms that take a representation of a narrative, derive all possible or the most likely interpretations of the narrative, and answer probabilistic queries by marginalizing over these interpretations. In our experiments, we show that our representation together with reasoning algorithms enables semantic understanding of narratives and answering probabilistic questions whose responses are not contained in the narrative. We report our results for two domains of Robocup soccer commentaries and a children story with focus on spatial contexts. To this end, we apply natural language processing (NLP) tools together with statistical approaches over common sense knowledge bases to represent a narrative in our framework.

1 Introduction

Understanding narratives and answering questions about them is an important problem in both natural language processing and linguistics (Hobbs *et al.* 1993). It is fundamental to question answering systems, help desk systems, dialog generators, and robot command interfaces. In many applications, questionnaires ask semantic questions whose responses are not contained in the written text. For example, it is very hard to answer a question like "Who has the possession of the ball at every step?" about soccer commentaries which do not explicitly mention the possession of the ball at every sentence. Answering this question requires understanding real events that happen in the game and following those events to track the possession of the ball.

There are recent approaches that map narratives to a sequence of meaningful events (Branavan *et al.* 2009; Vogel & Jurafsky 2010; Chen, Kim, & Mooney 2010). This mapping allows deeper understanding of the narratives, but deos not succeed in answering semantic questions without using advanced reasoning techniques. Answering questions whose answers are not included in the text requires a powerful representation integrating with reasoning algorithms and commonsense and domain-specific prior knowledge.

In this paper we use a powerful framework (extension of probabilistic situation calculus (Reiter 2001)) to represent narratives that incorporates events, the effects of events, and probability. We introduce reasoning algorithms in this framework to answer questions about the narrative. Our experiments demonstrate the benefit of our representation and reasoning algorithms to understand the semantics of narratives and answering semantic questions in two domains of children stories and soccer commentaries.

In this paper, we model a narrative as a sequence of probabilistic transitions which express uncertain changes imposed by the narrative sentences. For example, the verb *went* in *John went to the store* shows uncertainty between *walk*, *run*, *drive*, and so on. Probabilistic transitions give rise to deterministic events according to a probability distribution that depends on the current state. For instance, when the agent is tired, the agent is more likely to *drive* than to *run*. Deterministic events in turn give rise to changes in the world according to a transition function that also depends on the current state. For example, the deterministic event *run* makes the runner tired and *drive* uses up gasoline.

We then present exact and approximate inference algorithms for answering probabilistic queries about a narrative represented as above. An advantage of our exact algorithm is that it maintains and grows all feasible interpretations of the narrative in a forest of trees in an online fashion. This property allows to answer queries in real-time about dynamic narratives in which the text is iteratively growing. When a new sentence appears, our algorithm updates the current state forward and then backward to keep the forest consistent. For example, the sentence *John put his wallet on the table* implies that John was holding his wallet at the previous time step and our algorithm propagates this information backward for consistency. Our approximate algorithm approximates the most likely interpretation of the narrative.

We further extend our algorithms to answer queries about narratives with sentences universally quantified over variables. An example of such sentence is *John moves the briefcase*. This implies that all the objects inside the briefcase have also been moved.

Finally, we use our framework for answering semantic questions about Robocup soccer commentaries and a children story with a focus on spatial contexts. We show that our system is able to find responses not contained in the text.

Related Work: There are several symbolic narrative understanding systems (e.g., (Hobbs *et al.* 1993)) that do not model uncertainty, an essential part for narrative understanding. (Narayanan & Harabagiu 2004) uses reasoning about actions for question answering, however, their transition rep-

resentation and reasoning algorithms are different than ours. In addition, they do not show an approach to compute uncertainties of sentences. Still, their approach cannot answer semantic questions whose responses not contained in the text.

Hidden Markov Models (HMMs) (Rabiner 1989) and logical HMMs (Kersting, Raedt, & Raiko 2006) use sequences of variables to represent dynamic systems. In contrast, our model applies a transition model as a distribution over deterministic events. Both formalisms are universal and can represent each other, however, problem solving methods can take advantage of each formalism. In particular, relational elements in our representation allow more compact representation and more efficient algorithms in our narrative understanding problem.

There have been various reasoning formalisms that tackle probabilistic actions in propositional domains, however, they are not focused on the application of narrative understanding. (Baral & Tuan 2002; Iocchi *et al.* 2004; Reiter 2001) introduce exact reasoning, but their methods do not maintain the possible interpretations online and are restricted to answer queries about the last time step. Furthermore, their methods are required to know at the initial time step which propositions would appear in the later steps.

2 Narrative Representation

In this section we define our framework Probabilistic Action Model (PAM) to represent narratives. PAM is a general probabilistic logical framework for representing dynamic systems and consists of elements to model prior knowledge and dynamic transitions. The transitions are modeled using a probability distribution over several deterministic events.

Language: The language of PAM consists of a set of constants, variables, predicates (called *fluents*), and deterministic events. In PAM every constant and variable has a *type*, and every fluent and deterministic event has a schema. Specifically, $f(\tau_1, \ldots, \tau_k)$ (or $(f(\vec{\tau}))$ is a fluent schema where f is a fluent symbol and $\tau_1, \ldots, \tau_k$ are types.

Definition 1. The language $\mathcal{L}$ of a RPAM is a tuple $\mathcal{L} = (T, I, Itype, V, Vtype, F, DA)$ consisting of
- a set of types T
- a set of constants I and a function *Itype* which maps every constant to a type $Itype : I \rightarrow T$
- a set of variables V and a function $Vtype : V \rightarrow T$
- a set of fluent schemas F
- a set of deterministic event schemas DA

Grounding a fluent or action schema is defined as replacing every variable in the schema with a constant. In PAM, a world state s is defined as a complete truth assignment to all the groundings of fluent schemas F. It is generally the case that the truth value of all the fluents are not known at a specific time step, therefore, we use notion of *partial* states. A partial state σ is a function $\sigma : GF \rightarrow \{true, false, unknown\}$ where GF is the set of ground fluents. We can interchangeably represent a partial state σ as a conjunction of fluent literals where a fluent literal is in the form of either f (for $\sigma(f) = true$) or $\neg f$ (for $\sigma(f) = false$).

For example, the language $\mathcal{L}$ consists of types $\{o, l\}$, constants $\{Glass : o, Room : l\}$, and the fluent schemas $\{Holding(o), At(o, l), Atloc(l), Tired()\}$. Then, in this domain $Holding(Glass) \wedge \neg At(Glass, Room)$ is a partial state, while $Holding(Glass) \wedge At(Glass, Room) \wedge Atloc(Room) \wedge \neg Tired()$ is a complete state.

Deterministic Events: We define effect axioms to represent deterministic event schemas.

Definition 2. Let $da(\vec{x})$ be a deterministic event schema, and $Effect(\vec{x}, \vec{y})$ and $Precond(\vec{x}, \vec{y})$ be partial states whose variables appear in $\vec{x}, \vec{y}$. Then, effect axioms for *da* is:
- $da(\vec{x})$ **initiates** $Effect(\vec{x}, \vec{y})$ **when** $Precond(\vec{x}, \vec{y})$.

Note that every fluent schema $f(\vec{x}, \vec{y})$ appearing in *Effect* and *Precond* can contain free variables denoted by $\vec{y}$. The semantics of the above definition is as follows: For every time t and every free variable y, if *Precond* holds at time t, then the deterministic event *da* initiates *Effect* at time $t + 1$.

For example, $Run(l_2)$ **initiates** $Atloc(l_2) \wedge Tired()$ **when** $Empty(l_2)$ describes that the agent runs to the empty location l_2. $Walkobj(l_2)$ **initiates** $Atloc(l_2) \wedge At(o, l_2)$ **when** $Holding(o), Empty(l_2)$ describes that the location of all the objects o held by the agent is changed to l_2.

Transition Probability is associated with every sentence of the narrative and is modeled as a probability distribution over different deterministic events.

Definition 3. Let $\psi_1(\vec{x}), \ldots, \psi_N(\vec{x})$ be partial states partitioning the world (called *Partitions*(*pa*)) . The following is the *transition model pa*:
$pa(\vec{x})$ **produces**
- $da_1^1, \ldots, da_1^M$ **with** $p_1^1, \ldots, p_1^m$ **when** $\psi_1(\vec{x})$
- $\ldots$
- $da_N^1, \ldots, da_N^M$ **with** $p_N^1, \ldots, p_N^m$ **when** $\psi_N(\vec{x})$

where p_J^i is the probability of choosing event da_J^i in the partition ψ_J and $p_J^1 + \ldots + p_J^m = 1$.

For example, the probabilistic transitions $Movobj$ and Put are defined as: $Moveobj(l_2)$ **produces**
- $Runobj(l_2), Walkobj(l_2)$ **with** .8, .2 **when** $Empty(l_2)$
- $Nothing$ **with** 1.0 **when** $\neg Empty(l_2)$

$Put(o)$: **produces**
- $PutDown(o)$ **with** 1.0 **when** $Holding(o)$
- $Nothing$ **with** 1.0 **when** $\neg Holding(o)$

Finally, PAM is defined using the above definitions.

Definition 4. A RAM $(\mathcal{L}, EA, PA, P_0)$ consists of
- language $\mathcal{L} = (T, I, Itype, V, Vtype, F, DA)$ (Def. 1)
- effect axioms EA (Def. 2)
- probability transitions PA (Def. 3)
- graphical model prior probability P_0

A narrative for a PAM is modeled as $Tx = (T, OA)$ where OA is the sequence of T sentences as grounded probabilistic transitions (a transition $a(s_1, \ldots, s_k)$ is grounded if every argument s_i of transition a is a constant).

3 Query Answering Algorithm

Our query answering algorithm takes as input a narrative model $Tx = (T, OA = \langle a_1, \ldots, a_T \rangle)$ and builds the PAM representation corresponding to the narrative where each sentence a_t is mapped to a transition probability of PAM. A query about the narrative inquires about the probability of a formula (conjunction of literals) in the *PAM*. Our query answering algorithm applies a exact, approximate, and lifted reasoning algorithms to answer queries. For our reasoning algorithms we assume that the transition probability corresponding to every sentence is available. Next in section 4 we show how we compute transition distribution.

Algorithm 1. *ExactInf*(*Tx*, *PAM*, q)
- Input: Narrative $Tx = (T, OA)$ for *PAM* , formula q
- Output: $p(q) \in [0, 1]$

1. *Forest* ← *MakeForest*(*Tx*, *T*)
2. *PATHS* ← *DetPathsFromForest*(*Forest*)
3. **for** $\langle ev_1, \ldots, ev_T \rangle \in$ (*PATHS*)
4. $path^i \leftarrow \langle ev_1, \ldots, ev_T \rangle$
5. $q_0 \leftarrow BackupToRoot(q, path^i)$
6. $Qr^i \leftarrow P_0(q_0 | path^i[0].\sigma)$
7. **return** $\sum_i PPath(path^i, T)(Qr^i)$

Algorithm 2. *AppxInf*(*Tx*,*PAM*,*q*)
- Input: Narrative $Tx = (T, OA)$ for *PAM*, Query q
- Output: *true*,*false*

1. $\langle ev_1, \ldots, ev_T \rangle \leftarrow MostLikelySeq(Tx, PAM)$
2. $s_t \leftarrow ProgressSeq(\langle ev_1, \ldots, ev_T \rangle, s_0)$
3. **return** $s_t \wedge q$

Figure 1: Algorithm *ExactInf* to find all the interpretations of the narrative and compute the probability of query. Algorithm *AppxInf* to approximate the most likely interpretation of the narrative and evaluate the truth of the query.

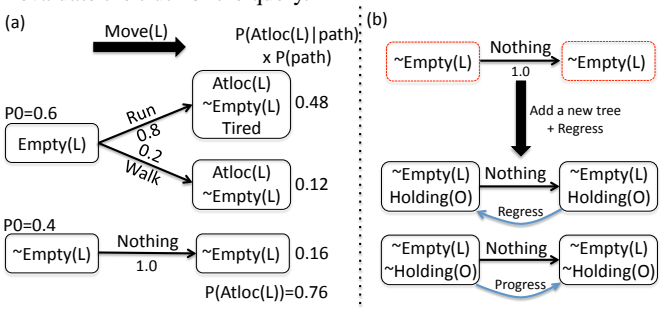

Figure 2: (a) Interpretation forest for *Move(L)* and calculation of P(*Atloc(L)*). (b) Adding a new tree in the forest with observation of $Holding(O)$ at one branch at step 1.

3.1 Exact Reasoning

Algorithm *ExactInf* (Algorithm 1) answers queries about a narrative model $Tx = (T, OA)$ for a PAM. It first builds an interpretation forest for the narrative. Then, for every interpretation in the forest it updates the query backward to time 0 and computes the probability of the query using the prior distribution P_0. Finally, it integrates all the answers.

Propositional Reasoning We first assume that the events do not include free variables in their effect axioms. In this setting we define an interpretation forest as follows:

Definition 5. Forest *F* is an interpretation forest of the narrative *Tx* if the nodes represent the current state of the narrative and the edges represent deterministic events corresponding to every sentence and has following properties:

1. The labels of roots partition the set of states *S*.
2. Sum of probabilities of the edges $e(da, p) = \langle n, - \rangle$ leading from any node n is 1 i.e., $\sum_{e=\langle n,- \rangle}(e.p) = 1$.
3. For edge $e(da, p) =$ between nodes n_t and n_{t+1}: $n_{t+1} = Progress(n_t, da)$, $n_t = Regress(n_{t+1}, da, n_t)$.
4. If a fluent f appears in a node *n*, then it should appear at all descendants of the node unless there is an edge $e(da, p)$ leading to *n* and f is in *Effects*(da).

Our exact reasoning algorithm uses *MakeForest* subroutine to build the interpretation forest of the narrative to maintain all, and only, feasible interpretations of the narrative in an online fashion. *MakeForest* initializes interpretations by setting the current state of the interpretation into *Partitions*(a_0). It then grows each interpretation by adding new branches associated with every deterministic event da_t of the sentence a_t. Then, the algorithm updates the current state of the narrative by *progressing* with da_t. To maintain the properties of the forest, it replicates the current node into several nodes and *regresses* the information back to the root.

Figure 2(a) shows the forest construction with transition $Move(L)$. If $Holding(O)$ is observed at step 1 at one branch of the tree then its truth value should be known in the roots to satisfy property (4). Then to satisfy property (1) new trees should be generated with roots $(\neg Empty(L), \neg Holding(O))$, $(Empty(L), Holding(O))$, $(Empty(L), \neg Holding(O))$, $(\neg Empty(L), Holding(O))$. Figure 2(b) shows a part of this update step.

Progress subroutine $Progress(s_{t-1}, da)$ takes as input an event da and the current state s_{t-1} and returns the updated state s_t if the preconditions of the event da is consistent with s_{t-1}. The current state s_t is then updated by applying the effect axioms of the event da. *Regress* subroutine $Regress(s_t, da, s_{t-1})$ takes as input an event da and current and previous states of the narrative and updates s_{t-1} using the preconditions of da.

An interpretation of the narrative is a path $path = \langle (-, \sigma_0), (da_1, \sigma_1), \ldots, (da_T, \sigma_T) \rangle$ in the forest where σ_t is a narrative state at time t and da_t is a deterministic event. The following indicates the likelihood of the interpretation: $PPath(path, 0) = P_0(\sigma_0)$ and $PPath(path, t) = PPath(path, t-1)PA(da_t | a_t, \sigma_{t-1})$.

To compute the probability of query formula, *ExactInf* builds q_0^i by regresses the query q backward to the root given the i^{th} interpretation. Note that regressing a formula backward with a deterministic sequence does not change its probability. The algorithm computes the probability of the query conditioned on the root label of each interpretation because the root formula encodes the state of the narrative in that interpretation. Finally, the algorithm marginalizes over all derived probabilities and answers the query using equation $P(q|OA) = \sum_i PPath(path^i, T) P_0(q_0^i | path^i \cdot \sigma_0)$. Figure 2 shows the calculation for $P(Atloc(L))$.

Lemma 1. *PPath* returns a probability distribution over all the interpretations in the forest from time step 0 to time step T (i.e., $\sum_i PPath(path^i, T) = 1$).

Theorem 1. Let $Tx = (T, OA)$ be a narrative model for a *PAM*. If q is a conjunction of literals at an arbitrary time step, then *ExactInf* returns the correct answer to the probability of query formula q given *Tx*.

Lifted Reasoning We introduce a lifted reasoning approach to handle free variables that appear in effect axioms. One naive approach is to ground all the free variables (similar to (Helmert 2008)) and use the propositional reasoning algorithm. The problem with grounding is that we need to deal with a huge state space. Instead, we use a lifted algorithm that considers a variable x as a potential set of constants. The algorithm does not ground variable x unless a constant c included in x appears in the narrative. In this case, the constant c is no longer in the span of variable x which still includes other constants.

More specifically, our algorithm replaces every free vari-

able x with a new auxiliary constant *AuxC* plus a table *possTable* that restricts its possible values. For example, if $possTable(AuxC) = \{C_1, \ldots, C_k\}$ and Q is an arbitrary fluent then: $Q(AuxC) \equiv Q(C_1) \vee \ldots \vee Q(C_k)$. After replacing variables with auxiliary constants, our algorithm treats every auxiliary constant as a regular constant in the operations. In this case, the interpretation forest is no longer well-defined if the algorithm directly applies auxiliary constants in progression and regression. To satisfy the properties of the forest the algorithm *Normalizes* the node formulas by grounding some other fluent schemas. A normal formula is defined as follows:

Definition 6 (normal formula)**.** A formula $fm = f_1 \wedge \ldots \wedge f_k$ is in *normal* form means that: if a constant C is a grounding of a variable x in the arguments of one fluent schema f_i, then C is the grounding of variable x in all other fluent schemas that has x as an argument.

Procedure *Normalize*(*fm*) is applied after each progression or regression operation, It first checks whether a formula *fm* is in normal form or not. If it is not then *Normalize* returns a set of formulas $\{fm_1, \ldots, fm_k\}$ whose disjunctions have the same truth value as *fm* and every fm_i is in normal form. Then, *Normalize* replicates the current node (labeled by *fm*) in the forest into k different nodes whose formulas are fm_i. For example, if the combination $Q(AuxC), P(AuxC), Q(C)$ exists in *fm* then it extracts *C* from the $possTable(AuxC)$ and builds two disjuncts by including two new literals $P(C)$ and $\neg P(C)$ as $(P(AuxC), Q(AuxC), Q(C), P(C))$ and $(P(AuxC), Q(AuxC), Q(C), \neg P(C))$.

Lemma 2. If formula $fm' = fm_1 \vee \ldots \vee fm_k$ is the result of *Normalize*(*fm*), then every fm_i is (a) normal and (b) fm' has the same truth value as *fm*.

The following corollary shows that the lifted reasoning improves the running time over grounding.

Corollary 1 (Complexity)**.** Let T be the length of the narrative, *B* be the maximum branching factor, *GF* be the ground fluents, *PathF* be the fluents appearing in an interpretation path. If *I* represents the constants and *PathO* represents fluent schemas affected by the normalization step then the running time of lifted reasoning is $O(2^{|PathF|+|PathO|}|F|B^T)$ while the running time with grounding is $O(2^{|GF||I|}B^T)$.

ExactInf algorithm returns exact probability of the query formula and allows to answer queries about every time step. Moreover, it uses the notion of partial states instead of complete states. In other exact algorithms (e.g., (Baral & Tuan 2002)) every world state should be represented completely if they relax the hidden assumption that the initial state of the narrative is known. Even with partial states, the number of trees in the forest generated by our algorithm is exponential in terms of number of fluents and deterministic events that appear in the narrative. It is just feasible for narratives that are short or have few ambiguities. Therefore, approximate reasoning is of much interest.

3.2 Approximate Reasoning

The *AppxInf* algorithm returns the truth value of a query formula given sentences $\langle a_1, \ldots, a_T \rangle$ and PAM representation. The algorithm uses a Viterbi-like (Rabiner 1989) dynamic programming subroutine that approximates the most likely event sequence corresponding to a narrative given the PAM representation instead of storing all the possible interpretations of the narrative. This subroutine returns path $\langle ev_1, \ldots, ev_T \rangle$ as an approximation of the most likely event sequence corresponding to the narrative using the following recurrence relations.

$$
\begin{aligned}
&V_{1,da} = Pa(da|a_1, s_0),\ S_{1,da} = Prog(s_0, da),\ Path_{1,da} = [da] \\
&V_{t,da} = Pa(da|a_t, s_{t-1}) + V_{t-1,ev} + l_{s_{t-1},da} \\
&S_{t,da} = Prog(s_{t-1}, da), Path_{t,da} = Path_{t-1,ev} + [da] \qquad (1)
\end{aligned}
$$

where $ev = \arg\max_{da \in DA}(V_{t-1,da})$, $s_{t-1} = S_{t-1,ev}$, and $l_{s,da}$ is a loss function.

Here $V_{t,da}$ (not a probability function) shows the value of the best sequence $Path_{t,da}$ for the first t sentences, and $S_{t-1,da}$ shows the current state of the narrative at time $t-1$. The algorithm initializes the value of the path with the probabilities of events for the first sentence. Then, at each time step the algorithm integrates the probability of event da and the maximum value derived for the step $t-1$. If the preconditions of da is not consistent with the current state s_{t-1} we penalize the value of choosing this event using a loss function $l_{s_{t-1},da}$ which is a real number between 0 and -1. Current state $S_{t,da}$ is derived by *Progressing* state s_{t-1} with event da. The path $Path_{t,ev}$ is updated by keeping a pointer to the previous selected event in the recursive step. Finally, the best path is derived as: $ev_T = \arg\max_{da \in DA}(V_{T,da})$ and $ev_{1..T-1} = Path_{ev_T,T}$

Finally, *AppxInf* checks if the query holds in the current state of the narrative. The current state of the narrative at the query time is computed by progressing the initial state and regressing the final state. Notice that the Viterbi-like approach does not return the exact most likely event sequence as it depends on the loss function at every step that the event is inconsistent. To deal with free variables we use the approach in (Nance, Vogel, & Amir 2006) for parametrized filtering with a sequence of deterministic events.

4 Experiments: Reading Comprehension

In this section we show how we represent a narrative using PAM for two domains: children story in spatial contexts and Robocup soccer commentaries. Moreover, we show the performance of our system for answering semantic questions whose answers are not contained in the text.

4.1 Robocup Soccer Commentaries

We use a dataset from (Chen, Kim, & Mooney 2010) including four commentaries of Robocup soccer games. We first extract fluents and manually build effect axioms for about 20 events in the domain. We then apply an iterative learning approach (Hajishirzi & Amir 2011) to compute the transition probability corresponding to every sentence. Next, we apply *AppxInf* algorithm and answer queries about the commentaries.

We compare the precision of our algorithm with a baseline algorithm that returns a part of the text that has similar features to the query. The baseline algorithm is a rough implementation of a traditional reading comprehension system (Rilo & Thelen 2000). We evaluate the precision of the returned responses as the percentage of the steps that the algorithm answers the query correctly.

Approach	1	2	3	4	Avg.
Query I: Who has the possession of the ball?					
AppxInf	**.76**	**.70**	**.80**	**.74**	**.75**
Baseline	.27	.27	.26	.24	.26
Query II: Did a correct pass happen at this step?					
AppxInf	**.95**	**.93**	**.99**	**.98**	**.96**
Baseline	.82	.50	.59	.57	.64

Table 1: Results of answering questions about four Robocup commentaries using our approach *AppxInf* and a baseline.

For example, to answer the query "Who has the possession of the ball?" the baseline reports sentences that term "ball" has appeared. Another example query is "Did a correct pass happen at the current time step?". The baseline algorithm looks for sentences that "pass" appears in them. This way, they really do not distinguish between *pass* and *bad-pass* events. The accuracies of both approaches are reported in Table 1. The results suggests that mapping sentences to meaningful events using our representation provides deeper understanding of the narrative as it can answer questions whose responses are not contained in the text.

4.2 Children Story

Here we apply our framework to a less structured domain, children stories with focus on spatial contexts. We use statistical approaches over commonsense knowledge bases together with NLP tools and build corresponding PAM representation. We manually describe effect axioms for about 15 verbs extracted from 1000 most common nouns. In addition, we build a set of predicates which may be extended given the narrative. Our example narrative (Figure 3)(a) is a children story[1] . We first apply C&C tools (Curran, Clark, & Bos 2007) that maps texts to logical representations by generating a logical symbol for every element of the text. We apply postprocessing and derive the verbs and their arguments from the narrative. Therefore, the narrative is represented as a sequence of relationships of the form $verb(args)$ (Figure 3(b)).

Transition Probability: The verbs in the sentences show uncertainty between different deterministic events which are manually described by effect axioms. We map every sentence to a probability distribution over different deterministic events and build the transition model of PAM.

We use *hyponyms* of a verb as possible deterministic events corresponding to that verb. In linguistics, a hyponym is a word that shares a *type-of* relationship with its *hypernym*. For example, *run*, *walk*, and *drive* are all hyponyms of *go* (their *hypernym*). We use WordNet (Fellbaum 1998) to extract the list of hyponyms (deterministic events) associated with the verb. We then compute transition distribution $P(ev|s)$ of event ev given the sentence $s = verb(args)$. In fact, we identify how frequently event ev has been used together with the arguments $args$ in a corpus (Lee 1999). The corpus consists of tuples in the form of (noun, verb, frequency) extracted for 1000 most common nouns.

To compute the transition probability we divide the frequency of the tuple $(ev, args)$ over the frequency of $args$ in the corpus: $P(ev|s) = |(ev, args)|/|args|$. If either of the terms in the $args$ do not appear in the corpus, we generalize that term by replacing it with its hypernym and compute the frequency of new arguments. In addition, if the tuple $(ev, args)$ do not appear in the corpus, we find similar nouns $Sims(arg)$ for each term arg in the $args$. We then compute the weighted sum over frequencies of similar nouns and the event ev. The weights are derived by computing the semantic distance (Lee 1999) between similar nouns and arg. Therefore, $P(ev|s) \propto \prod_{arg \in args} \sum_{sim \in Sims(arg)} |ev, sim| Dist(sim, arg)$.

To evaluate the precision of our method we test on another corpus, SemCor[2]. We transform every sentence $verb(args)$ in SemCor to the form of $hypernym(verb)(args)$ using C&C tools and WordNet. For every new relation we apply the above frequency counting and compute the probability of all deterministic events associated with $hypernym(verb)$. For evaluation, we rank the events according to their probabilities and check the rank of the original verb of the sentence. Table 2 shows the result of our frequency checking and a baseline to see if the original verb is among the first k candidates. The baseline algorithm returns the most frequent event independent of the $args$ in the sentence. Table 3 shows transition probabilities derived using our approach for some verbs including the verbs in the children story.

Prior Distribution We build a part of prior distribution

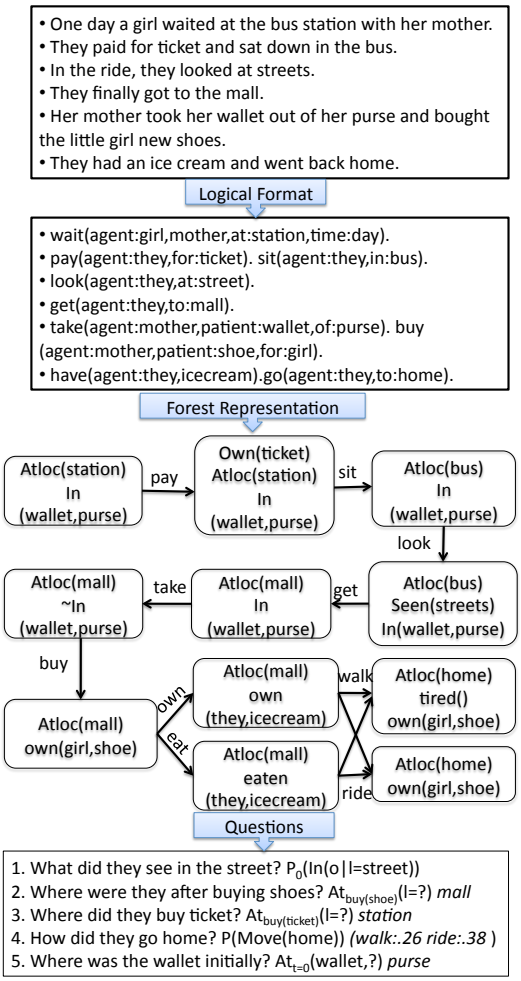

Figure 3: (a) Children story (b) logical representation corresponding to the story (c) interpretation forest (d) semantic questions about the story and responses.

[1] http://www.goodnightstories.com/

[2] http://www.seas.smu.edu/~sanda/research/

top k	top 1	top 2	top 3	top 4	top 5
our method	**.24**	**.52**	**.66**	**.75**	**.92**
baseline	.14	.43	.59	.68	.89

Table 2: Comparing the rank of the correct event in top k returned deterministic events by our approach vs. Baseline.

	verb,args					
	memorize,pattern		make,tea		go,home	
event,p	study	.35	cook	.42	ride	.38
	review	.22	make	.37	walk	.28
	absorb	.18	throw	.16	run	.36
	memorize	.17	dip	.05		

Table 3: Some examples for the transition distributions over the events (hyponyms of the verb)

P_0 with a focus on spatial contexts by computing the probabilities $P(l, o)$, the probability that object o is in location l. We extract object-location correlations from Open Mind Common Sense (OMCS) knowledge base (Singh 2002), `findin` file, which contains a set of sentences of the form, "you often find (object o) (in location l)". For each object o and location l in OMCS we calculate $P(l, o)$ as the frequency of the appearance of object o close to location l in corpus (American fiction downloaded from Project Gutenberg)[3]. To compute $P(l|o)$ and $P(o|l)$ we divide this number by the frequency of the object o or the location l in the corpus, respectively. Table 4 shows some results of object-location probabilities including terms of our story.

Answering Questions: We build the PAM representation and the forest corresponding to the story (Figure 3(c)) where we use the transition probabilities from Table 3. Notice that we infer the missing arguments of the events when we build the forest as we know the effect axioms and the number of arguments for each event.

Using our framework we can answer different categories of questions (Figure 3(d)) whose responses are not contained in the text. First category (Questions 2, 3, and 5) inquire about the object locations or properties that can be explicitly inferred from the interpretation forest. Second category (question 1) is about object locations which can be inferred from the prior probability P_0, a probabilistic version of OpenMind. Third category(Question 4) inquires about event probabilities derived from the transition distribution.

Answering these types of questions using current question answering techniques is very hard. For example our rough implementation of (Rilo & Thelen 2000) is not able to answer these question as it looks for parts of texts that have similar features to the question. The reason is that the responses are not mentioned in the narrative, and reasoning is required to infer responses.

5 Discussion and Future Work

In this paper we introduced a framework together with exact and approximate reasoning algorithms for representing narratives and answering queries about them. We suggest that using action theories allows understanding semantics of the narrative as well as answering questions whose responses are not contained in the text using reasoning in the framework. We demonstrate our results in a Robocup commentary domain and a less structured domain of children stories.

Our current system is a flexible structure that can be augmented with richer NLP tools such as dismbiguation, co-reference resolution, etc. For example, *have ice cream* in our story should be disambiguated to *eating* and therefore a branch of the tree should be removed. Our immediate future work is to augment our system with such NLP tools and apply it in Remedia corpus for reading comprehension. Our idea is to manually construct effect axioms for some events and a noise event *Nothing*. This way we can model all the verbs of the stories. Our current approach only works if the sentences are consecutive; In future, we would like to apply partial planning ideas and deal with overlapping events that happen in the narrative. Also, we would like to use this approach to find missing events of narratives.

[3]See `http://www.gutenberg.org/`.

Objects						Locations	
Animal		Ball		Bed		street	
wild	.64	game	.43	bedroom	.43	vehicle	.40
water	.16	hand	.33	hotel	.29	pavement	.28
forest	.05	park	.09	hospital	.16	curb	.25
cage	.05	basket	.08	store	.12	bus stop	.7

Table 4: Prior distribution $P_0(l|o)$ for locations given an object, and $P_0(o|l)$ for objects given a location.

References

Baral, C., and Tuan, L. 2002. Reasoning about actions in a probabilistic setting. In *AAAI*.

Branavan, S.; Chen, H.; Zettlemoyer, L.; and Barzilay, R. 2009. Reinforcement learning for mapping instructions to actions. In *ACL-IJCNLP*, 82–90.

Chen, D.; Kim, J.; and Mooney, R. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *JAIR* 37:397–435.

Curran, J.; Clark, S.; and Bos, J. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *ACL Demo*.

Fellbaum, C., ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Hajishirzi, H., and Amir, E. 2011. Reasoning about robocup soccer narratives. In *Common Sense*.

Helmert, M. 2008. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence*.

Hobbs, J. R.; Stickel, M. E.; Appelt, D. E.; and Martin, P. 1993. Interpretation as abduction. *Artificial Intelligence* 63:69–142.

Iocchi, L.; Lukasiewicz, T.; Nardi, D.; and Rosati, R. 2004. Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *ECAI*.

Kersting, K.; Raedt, L.; and Raiko, T. 2006. Logical hidden markov models. *JAIR*.

Lee, L. 1999. Measures of distributional similarity. In *ACL*.

Nance, M.; Vogel, A.; and Amir, E. 2006. Reasoning about partially observed actions. In *AAAI'06*.

Narayanan, S., and Harabagiu, S. 2004. Question answering based on semantic structures. In *Coling*, 693–701.

Rabiner, L. R. 1989. A tutorial on hmm and selected applications in speech recognition. *IEEE* 77(2).

Reiter, R. 2001. *Knowledge In Action*. MIT Press.

Rilo, E., and Thelen, M. 2000. A rule-based question answering system for reading comprehension tests. In *In ANLP/NAACL Workshop*.

Singh, P. 2002. The public acquisition of commonsense knowledge. In *AAAI Spring Symposimum*.

Vogel, A., and Jurafsky, D. 2010. Learning to follow navigational directions. In *ACL*.