Using SUMO to Represent Storytelling Knowledge

Jeffrey Cua
Center for Language Technologies
De La Salle University
Manila, Philippines
jeffrey.cua@delasalle.ph

Ethel Ong
Center for Language Technologies
De La Salle University
Manila, Philippines
ethel.ong@delasalle.ph

Adam Pease
Articulate Software
Angwin, California
USA
apease@articulatesoftware.com

ABSTRACT
An ontology is a conceptual model of what "exists" in a domain, brought into machine-interpretable form by means of expressions in formal logic. Since ontologies are content theories about the kinds, properties and relations of objects, story generators can use them as concept dictionaries for actions and events in a story world. This paper presents SUMO Stories, a story planner system that uses the Suggested Upper Merged Ontology (SUMO) to represent the storytelling knowledge of Picture Books. Picture Books is an existing story generation system that utilizes a commonsense knowledge base to generate fables for children age 4-6 years old. The story planner of SUMO Stories utilizes Sigma, the inference engine of SUMO, in generating story plans. The story plans are stated in first order logic and represent the elements of a children's story. Stories are constrained to follow a basic plot, where a child violates a rule and learns the moral lesson at the end of the story.

Keywords
Ontology, First-Order Logic, Story Generation

1. INTRODUCTION
A story is not just simply text; a good story contains creativity, originality and even imagination, which would keep the target readers interested. Before addressing these challenging goals, we first have to ensure that we can generate a simple story that makes sense, which is one that does not conflict with commonsense information about the world. A story is made up of words, these words represent concepts about the things and events that may or may not exist in the real world. A person cannot create a story from nothing. Without background conceptual knowledge in which he/she can represent with words there could be no story. Only through daily experiences and learning can someone obtain the knowledge about the world necessary to construct a good story. Thus, for systems to have the potential to achieve the same level of expressiveness as people, they must also be provided with the same collection of knowledge about the basic relations between things and events.

Swartjes [15] developed a story world ontology containing two layers, an upper story world ontology and a domain-specific world ontology. The upper domain-independent story world ontology models a vast amount of possible actions and events. It is limited to high-level concepts that are meta, generic or abstract to address a broad range of domain areas. A domain-specific story world ontology applies the upper ontology to a certain domain.

Kooijman [7] suggests the use of the Suggested Upper Merged Ontology (SUMO) as an upper ontology to capture the semantics of world knowledge. SUMO [10] is an open source formal and public ontology. It is a collection of well-defined and well-documented concepts, interconnected into a logical theory. Although originally just an upper ontology, it now contains some 20,000 terms and 70,000 axioms, having incorporated many formal domain ontologies. Axioms are in first-order logic form (with some higher order extensions) and reflect commonsense notions that are generally recognized among the concepts. They place a constraint on the interpretation of concepts and provide guidelines for automated reasoning systems such as Sigma [12]. Sigma incorporates a version of the Vampire [13] theorem prover, and the SInE axiom selection method [5] as well as the full TPTP suite of theorem provers [14]. SUMO is stated in SUO-KIF, a variant of KIF [4], which is first-order logic with equality and some higher-order extensions.

This paper presents SUMO Stories, an automatic story generator that uses first-order logic to declaratively describe models of the world, specifically those aspects of the world that represent storytelling knowledge for children's stories of the fable form. The story planner then utilizes Sigma to infer this knowledge in order to generate a story plan also in first-order logic. Although there exist a number of choices for knowledge representation in formal logic, we chose first-order logic because it enables a less restricted semantics compared to description logic. In particular, the axiomatic nature of actions and their consequences, essential for reasoning about narrative structures, is not supported by description logics, which focus on category and instance membership reasoning.

In this paper, we focus on discussing the representation of Picture Books' [6] storytelling knowledge [11] in SUMO, and the interaction between the story planner and Sigma to derive the story plan. The paper ends with a summary of what we have learned including the issues discovered, and presents further work that can be done.

2. REPRESENTING STORYTELLING KNOWLEDGE IN SUMO
Story generators require two types of knowledge, the narratological or operational knowledge representing concepts about narrative structures, and the storytelling domain knowledge comprising of the knowledge about the world, character representations, and a causal network of actions and events that can take place in the story world. Narratological knowledge includes the plot and the theme that drive the flow of the story. Stories in Picture Books follow the classic story pattern of Machado [9] to drive its plot, which has five phases comprising of the introduction, problem occurrence, rising action, resolution of the problem, and climax.

2.1 Narratological Knowledge

Story phases are modeled in SUMO Stories (SUMOs) as story attributes to depict the time progression of a story, for example:

```
(instance ?STORY Story)
(attribute ?STORY ProblemPhase)
```

These could be considered a replacement for the temporal qualifications of SUMO. Because temporal qualifications are higher-order logic, they currently pose problems for inferencing.

Story phases help restrict the occurrence of events or the performance of actions in certain parts of the story. This is especially true for initializing attributes, locations and other elements in the story during specific phases, e.g., in the *introduction phase*. Every time a phase transition occurs, the system checks if there are any attributes that conflict with each other; for instance, a character having the attribute of *tranquility* during the introduction phase may experience *anxiety* during the problem phase. This scenario arises due to the lack of temporal predicates to allow attribute changes in a character to be asserted.

There is no built-in method to remove an assertion in SIGMA, however, which leads to the solution of creating a new snapshot of the current world state by removing all conflicting assertions and re-asserting only those that still hold true based on the current phase of the story. Conflicting attributes are those with the same parent class, e.g., *honest* and *dishonest* belong to the *TraitAttribute* class, where one is an instance of a *PositiveAttribute* and the other an instance of a *NegativeAttribute*, as shown in the following axioms:

```
(subclass NegativeAttribute TraitAttribute)
(subclass PositiveAttribute TraitAttribute)
(instance Dishonest NegativeAttribute)
(instance Honest PositiveAttribute)
```

Other conflicting *trait* attributes include *cowardly* and *brave*, while conflicting *visibility* attributes include *hidden* and *visible*. Using this approach, the system would only allow one of the *TraitAttribute* and prevents a situation wherein a character has multiple positive and multiple negative attributes. To resolve this, the system manually allows certain attributes to have many instances and checks if a certain attribute is a *contrary attribute* of an existing attribute using the axiom below.

```
(contraryAttribute Honest Dishonest)
```

Children's stories also have themes to teach a particular value, e.g., learning to be brave, or learning to sleep early. Themes give a different set of possible actions and events to generate varying stories from the same input story elements. Currently, the story planner determines the theme based from the input of the user on what he/she wants to learn. Adding themes to axioms is also vital to the story plan progression to set initial attributes or to force an event to happen since not all events have to actually be based from previous events and current world state. In the example axiom below for the *learning to sleep early* theme, the child has to disobey the rule of the parent to sleep early.

```
(=>
  (and
    (instance ?STORY Story)
    (attribute ?STORY IntroductionPhase)
    (attribute ?STORY SleepEarly)
    (instance ?CHAR StoryCharacter)
    (attribute ?CHAR Child)
    (attribute ?CHAR MainRole))
  (attribute ?CHAR Disobedient))
```

2.2 Axioms for Domain Knowledge

SUMO Stories based its storytelling domain knowledge (the semantic description about concepts, objects and their relations) from the domain of Picture Books [6]. Whereas Picture Books uses binary relations patterned after ConceptNet [8] to represent commonsense relations between two concepts [11], SUMOs utilizes a more formal ontology.

The axioms that comprise the ontology of SUMOs are divided into three main categories: factual axioms, action axioms, and event axioms. Factual axioms represent the characters, locations and objects that may comprise the story. These axioms define the story elements by specifying their attributes which could be their traits, age, role, visibility, physical state, name and parent class. A sample factual axiom describing story characters is shown below. This axiom states that "*if Adult1 is a female adult elephant character and Child1 is a child elephant character, then Adult1 has the name Edna and she is the mother of Child1*". Picture Books generates fables that convey morals or lessons and utilize animals as characters.

```
(=>
  (and
    (instance ?Adult1 ElephantCharacter)
    (attribute ?Adult1 Female)
    (attribute ?Adult1 Adult)
    (instance ?Child1 ElephantCharacter)
    (attribute ?Child1 Child))
  (and
    (name ?Adult1 "Edna")
    (mother ?Child1 ?Adult1)))
```

Action axioms define the actions that story characters may do given the world state. In SUMOs, the story revolves around and progresses through actions of characters. By specifying the choices that a character can make, more options are available to generate varying stories from the same input picture. Currently, only characters are able to have action axioms, however it does not necessarily mean that all axioms for characters are actions. The example action axiom below states that "*if a child is near some toys, then the child may play with the toys*". The *capability* predicate specifies the possible action, while the *agent* and the *patient* arguments reflect thematic roles for specifying the doer and the receiver of the actions, respectively.

```
(=>
  (and
    (instance ?STORY Story)
    (attribute ?STORY ProblemPhase)
    (instance ?CHILD StoryCharacter)
    (attribute ?CHILD Child)
    (instance ?TOYS Toys)
    (orientation ?CHILD ?TOYS Near))
  (and
    (capability RecreationOrExercise
         agent ?CHILD)
    (capability RecreationOrExercise
         patient ?TOYS)))
```

Event axioms include explicit events, (e.g., *a fragile lamp breaking if a child is not careful during playtime* [3]), attribute changes, thoughts and even actions of characters (e.g., *if a child is hurt he will immediately cry*). The example event axiom below states that "*if it is night time and a child continues to play (instead of going to bed early), he or she will have a headache*".

```
(=>
  (and
    (instance ?STORY Story)
    (attribute ?STORY RisingActionPhase)
    (instance ?CHILD StoryCharacter)
    (attribute ?CHILD Child)
    (instance ?NIGHT NightTime)
    (instance ?PLAYING RecreationOrExercise)
    (agent ?PLAYING ?CHILD))
  (and
    (attribute ?CHILD HeadAche)
    (exists (?R)
      (instance ?R RisingActionEvent))))
```

When a child plays during the night it becomes a rising action event. There are other events as well, such as problem events and solution events, corresponding to those that occur during a problem phase and a solution phase, respectively, of a story. Rising action events are typically actions that will cause consequences because of the problem (i.e., *the child violated his/her parent's rule to go to bed early*).

3. GENERATING STORY PLANS

Figure 1 shows the architecture of SUMO Stories, including its interface with SUMO and Sigma. SUMOs has two main modules – Story Editor and Story Planner, both of which assert and query the ontology using Sigma, but the purpose for each is different.

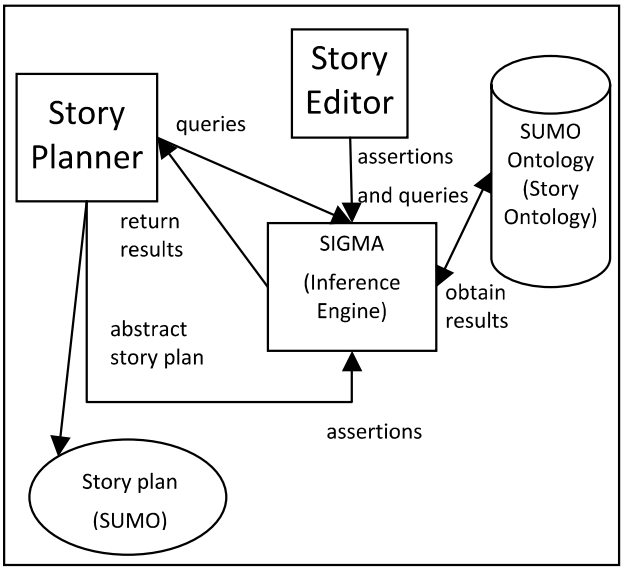

Figure 1. Architectural Design

The Story Editor processes the input story elements – theme, background, characters and objects, chosen by the user and generates assertions corresponding to these. It then queries Sigma for additional information such as attributes and initial locations of the characters and objects in the background.

The following axioms comprise the first part of the story plan. Lines 1 to 5 contain axioms to instantiate a story with its theme (*sleep early*) and current phase (*introduction phase*). The time is associated with the theme (*night time*) as well but the day is set to a default value. A group, *children*, is instantiated in line 6 to track the number of child characters in the story. By keeping track of the number of child characters in the story we have a simple metric to change the flow of the story by giving the child more possible actions to react to events. For example, when the child breaks the lamp he/she will not only probably hide but will also blame another child in the process.

```
1>  (instance Story1 Story)
2>  (attribute Story1 SleepEarly)
3>  (attribute Story1 IntroductionPhase)
4>  (instance NightTime2 NightTime)
5>  (instance Friday3 Friday)
6>  (instance Children Group)
```

Lines 7 to 17 assert the different characters and their attributes; *RabAdult* being the parent of *RabChild* (the main character) is automatically inferred from the domain knowledge. Notice that line 11 initializes the current state of *RabChild* to *tranquility* as the story has not reached the problem phase yet.

```
7>  (instance RabChild RabbitCharacter)
8>  (attribute RabChild Child)
9>  (attribute RabChild Female)
10> (member RabChild Children)
11> (attribute RabChild Tranquility)
12> (instance RabAdult RabbitCharacter)
13> (attribute RabAdult Adult)
14> (attribute RabAdult Female)
15> (attribute RabAdult Tranquility)
16> (attribute RabChild MainRole)
17> (memberCount Children 1)
```

Line 18 asserts the object (*toys*) and line 19 asserts the background (*living room*) which is the setting of the story. Lines 20 to 22 place the main character and the object in the location, as well as initialize their orientation (*near*). These are dictated by the theme as well.

```
18> (instance Toys6 Toys)
19> (instance LR7 LivingRoom)
20> (located RabChild LR7)
21> (located Toys6 LR7)
22> (orientation RabChild Toys6 Near)
```

The Story Planner handles the progression of the story and creates assertions from results of queries to Sigma until the story plan is completed. The generated axioms are asserted back to Sigma for inclusion in the SUMO Stories' domain ontology to be used again for further inferencing.

Queries sent to Sigma can be classified into three types. Concept-based queries concern classes and instances, and are used to determine direct and indirect subclass and class-instance relationships. Relation-based queries infer knowledge by considering transitivity, symmetry and inversion of relations [2]. Action-based queries identify a set of actions based on the current

world state to drive the story. A fourth category, time-event queries, containing quantification over formula, is currently not supported by Sigma's embedded first-order theorem provers. These sorts of queries involve reasoning about temporal and event-based specifications and will be supported in the future by Sigma's currently experimental higher-order theorem proving component [1].

From the initial set of assertions, the planner issues its first concept-based query, "`(name RabChild ?X)`", to determine a name for the main character, *RabChild*, and receives "*Rizzy*" as a result. This is asserted to the story plan as shown in line 23.

```
23>   (name RabChild "Rizzy")
```

The next query then checks if the parent has any desires for the child's state or behavior, which is actually the theme of the story. If there is, this would be the rule of the parent. The query "`(desires RabAdult ?X)`" would return and add assertion 24 to the story plan for the *learning to sleep early* theme. Note that this is in higher-order logic which Sigma cannot interpret, and the quote is used to treat it as a list that can simply be matched and unified with other lists.

```
24> (desires RabAdult
       '(exists (?X0)
          (and
            (instance ?X0 Sleeping)
            (manner ?X0 Early)
            (agent ?X0 RabChild)))
```

The parent must communicate this desire to the child, so an instance of the `Stating` process is created, with the parent as the agent of the action and the child as the patient, as shown in assertions 25 to 28.

```
25> (instance Stating8 Stating)
26> (agent Stating8 RabAdult)
27> (patient Stating8 RabChild)
28> (patient Stating8
       '(exists (?X0)
          (and
            (instance ?X0 Sleeping)
            (manner ?X0  Early)
            (agent ?X0 RabChild))))
```

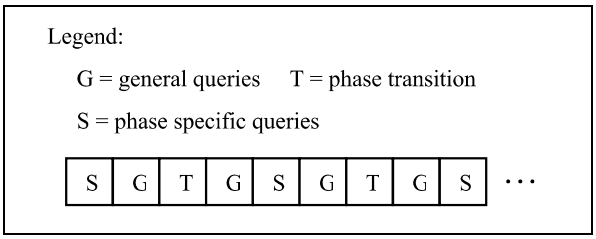

Figure 2. Sequence of Queries during Phase Transition

At this point, the introduction phase of the story plan has been completed. The next phase of the story would be the problem phase. However, during the transition to the next phase, a set of processes is performed. This involves general queries that check if there are any new events and new attributes, and remove old or conflicting information. The current *StoryPhase* attribute is also removed and replaced with the next one. Before going to phase-specific queries, the system again queries if there are any new desires, attributes, and events at the start of each new phase. These general queries are performed before and after the transition to the next phase. The sequence of queries is depicted in Figure 2.

During the transition from the introduction phase to the problem phase, the following queries are performed to retrieve various conditions or states of the story world:

- Query to check for new attributes of a character "`(attribute <character> ?X)`"
- Query to check for new attributes of an object "`(attribute <object> ?X)`"
- Query to check a character's desires, beliefs, states or other predicates "`(<predicate> <character> ?X)`"
- Query to check for new events "`(instance ?X <StoryEvent>)`"

Based from the results of these queries, new assertions shown in lines 29 to 30 are created. It is necessary to assert that the child character is *disobedient* (line 29) in this theme so he/she would disobey the rule (line 30) given by the parent character.

```
29> (attribute RabChild Disobedient)
30> (disapproves RabChild
      (exists (?X0)
         (and
           (instance ?X0 Sleeping)
           (manner ?X0 Early)
           (agent ?X0 RabChild))))
```

The problem phase begins by identifying any possible actions the child character can do given the current state of the story world with the query "`(capability RabChild ?X)`". This is the first action-based query to start the story flow. The ontology contains an axiom that states that if a child is near a toy he/she can play ("*RecreationOrExercise*") with it, resulting in the addition of assertions 31 to 33.

```
31> (instance ROE9 RecreationOrExercise)
32> (agent ROE9 RabChild)
33> (patient ROE9 Toys6)
```

The more story knowledge that is added to the ontology, the larger the set of possible actions (i.e., capabilities) a character can do (e.g. a child can look, touch, or clean the toy). However, the current effort only chooses the first item from the list (if Sigma does return a list) of actions as they are the only ones populated with sufficient knowledge to lead to the progress of the story.

If the parent gave a command at the start of the story, the system checks if the child actually did it. It would query if there is an instance of *sleeping* with its corresponding semantic roles.

```
(and
  (instance ?X Sleeping)
  (manner ?X Early)
  (agent ?X RabChild))
```

Sigma returns "*no*" which then leads for the system to query for the occurrence of any event in which the child participated that contradicts an adult's command. Since the only action that the child was involved in was *RecreationOrExercise*, the system

asserts line 34 that "*the child did not sleep early*" to the story plan as the problem of the story (violation of stated rule).

```
34> (not
      (exists (?X0)
        (and
          (instance ?X0 Sleeping)
          (manner ?X0 Early)
          (agent ?X0 RabChild))))
```

The rising action phase contains queries to check for the possible actions of the child and the parent. Line 35 is the assertion depicting the effect of *not sleeping early* on the main character.

```
35> (attribute RabChild HeadAche)
```

A forced event is set to happen where the parent will go to the room to see where the child is, to see if anything transpired, resulting in the assertion in line 36. This is to drive the interaction between the adult and child in order to lead to the solution.

```
36> (located RabAdult LR7)
```

After the adult enters the same room where the child is, the system will continually query what the adult and the child characters can do until a rising action event has occurred. However, in the *sleep early* theme, when the child character experiences a headache (due to playing during the night which indirectly means he did not sleep early) it is already a rising action event and no further actions are done in this phase. The system was able to determine that *the child has a headache* using the general queries during the phase transition asking Sigma if there are any new attributes for the story characters.

In the solution phase, the story planner continues to query Sigma for possible actions of the child until a solution event has been asserted. Since the parent and the child are near to each other (on the account that they are in the same room as asserted in line 36), when the system queries what the child can do, it returns that he/she can complain (*ExpressingDisapproval*) to his/her parent about his/her headache, which are asserted as lines 37 to 40.

```
37> (instance ED11 ExpressingDisapproval)
38> (agent ED11 RabChild)
39> (patient ED11 HeadAche)
40> (patient ED11 RabAdult)
```

Actions that show remorse or positive change due to previous events are the ones that are labeled as solution events. These include a child confessing, apologizing, complaining, or becoming brave. In the example above, after the child experiences the side effects of *not going to sleep early*, he/she complained regarding his/her headache to his/her parent. The adult would then repeat the same rule he/she had stated at the start of the story, if he/she made one, as shown in assertions 41 to 44.

```
41> (instance S12 Stating)
42> (agent S12 RabAdult)
43> (patient S12 RabChild)
44> (patient S12
      '(exists (?X0)
         (and
           (instance ?X0 Sleeping)
           (manner ?X0 Early)
           (agent ?X0 RabChild))))
```

The last phase, climax, queries the new trait attribute of the child which is assumed to be a positive one already.

However, in the *sleep early* theme, there is a unique event associated in the climax phase, a *change day* event. For a child character to actually learn the benefits of sleeping early he/she should experience it himself/herself. If the system queries and learns that there is a change day event, it will assert lines 45 to 46.

```
45> (instance Saturday13 Saturday)
46> (instance NightTime14 NightTime)
```

The system would also remove previous assertions 4 "`(instance NightTime2 NightTime)`" and 5 "`(instance Friday3 Friday)`" to give an impression that a whole day has passed setting an assumption that all future assertions would be done in this new period. The system would also reset some attributes of the child character as the story transitions to a new day (e.g. *headache will be gone*).

Note that this is a workaround for representing events in sequence without the use of higher-order logic. Issues related to this workaround are discussed in the next section of this paper.

In the climax phase, it has been assumed that the child has learned his/her lesson and would follow the rule of his/her parent, as shown in assertions 48 to 52. The attributes of the child would also be changed to *obedient* (trait) and *happiness* (state).

```
48> (attribute RabChild Obedient)
49> (instance Sleeping15 Sleeping)
50> (agent Sleeping15 RabChild)
51> (manner Sleeping15 Early)
52> (attribute RabChild Happiness)
```

4. ISSUES AND ANALYSIS

Sigma has a suite of theorem provers that handle only first-order logic, which presented some serious limitations. There are a few partial workarounds. The first is that Sigma converts higher-order statements into statements with a first-order interpretation by quoting. For example:

```
(desires RabChild
     (exists (S1) (instance S1 Sleeping)))
```

becomes

```
(desires RabChild
    '(exists (S1) (instance S1 Sleeping)))
```

The quoted sentence becomes an uninterpreted list rather than a logical sentence. Quantifying over a list term is allowed in first-order logic. The following query can now be performed:

```
(desires Rab5
   (exists (S1) (instance S1 ?X)))
```

The correct answer of "*Sleeping*" will be returned through simple list unification. However, most of the semantics of the statement is lost with this approach. For example, given:

```
(desires Rab5
   '(exists (S1)
      (and
        (instance S1 Sleeping)
        (manner S1 Early)
        (agent S1 Rab4))))
```

The same query will return no result because the "*and*" ceases to have its logical meaning. It becomes just another opaque term, and the two lists above do not unify.

The second partial workaround is that it is possible to create "*summary*" predicates that have a first-order use, while having a higher-order definition. For example, the predicate "*capability*" in SUMO states that a particular entity is capable of playing a particular kind of role in a particular kind of process. The definition requires the use of higher-order logic (in particular, a logic of possibility and necessity). However, once that predicate is defined, we can simply state that a child is capable of playing:

```
(capability Playing agent Child1)
```

There are serious limitations to this as well. The kinds of possible statements one can make in higher-order logic are infinite. It is typically not possible, even in a restricted knowledge representation project, to cover all the things that one may need to say through the creation of these sorts of summary predicates, and even then, first-order inference will not be able to take advantage of the definitions of these statements.

5. CONCLUSION

The paper presented our work in using an upper ontology in first-order logic to represent storytelling domain knowledge and narratological concepts. The story knowledge created is simplified, most of which is in first-order logic. Briefly, in higher-order logic one can quantify over statements, whereas in first-order logic one can only quantify over terms.

Our story planner then interacts with the inference engine Sigma to retrieve relevant knowledge from the ontology and to make new assertions in order to generate story plans, specifically the events, actions, emotions, objects, characters and location comprising the target story. As the formal logic notation is readable by only a select group of people, another engine, specifically a language generator, is needed to transform the story plan to surface text in English (or any desired language).

Though the system is designed to be able to create branching stories, the current version selects only the first item in the list of possible actions, as other options do not have sufficient knowledge available in the ontology. If enough knowledge is added, the static selection could then be replaced with a smart action selector algorithm that could also detect if the action has been done already and choose another one.

The use of probabilistic logic can be explored to address the issue of a smart action selector in order to find possible future actions in a story and how likely they are. Although a full higher-order logic can be used, it is not necessary and makes inferencing very hard. A logic of possibility and necessity approach would miss the notions of different events being more likely than others, and would mean that choosing story outcomes would be part of an external control program, rather than being explicit in the knowledge representation of the story rules. The current approach of using the first-order *capability* relation does not allow us to give full logical statements, but just roles and action types. Wrapping full logical statements in probabilities would provide full flexibility.

Axioms that are created to represent storytelling knowledge are usually events and actions caused by characters that continuously change the world state. The changing attributes of the instances in the world should be associated at a specified part of the event (e.g. during, after, before, start, end) to be able to correctly represent it.

Other concepts such as conversations (statements, questions) and thoughts (desires and disapprovals) also require higher-order logic. For example, we may want to state that "*Mother believes the child is sad.*" This sentence states a relation between "*Mother*" and the proposition that "*the child is sad*". The belief relation quantifies over statements, and is therefore higher-order.

Another promising approach is to integrate full higher-order logical reasoning with Sigma. An initial attempt has been made in this direction with LEO-II [1]. However, this work was begun only in 2010, is still experimental, and is not yet suitable for application projects such as the SUMO Stories effort. Hopefully, future projects will be able to take advantage of this new work.

6. ACKNOWLEDGMENTS

This work would not have been possible without our strong collaboration with Mr. Adam Pease from Articulate Software (USA). Mr. Pease is one of the invited speakers in the 6th National Natural Language Processing Research Symposium organized by the College of Computer Studies (De La Salle University) in September 2009. His visit, talks (in the symposium), and subsequent lectures for and consultations with members of the university were made possible through funding from DOST-PCASTRD and the Visiting Scholars Program of the university.

7. REFERENCES

[1] Benzmüller, C., and Pease, A. 2010. Reasoning with Embedded Formulas and Modalities in SUMO. In *Proceedings of European Conference on Artificial Intelligence 2010 Workshop on Automated Reasoning about Context and Ontology Evolution*, August 16-17 2010, Portugal.

[2] Corda, I., Bennett, B., and Dimitrova, V. 2008. Interacting with an Ontology to Explore Historical Domains. In *Proceedings of the 2008 First International Workshop on Ontologies in Interactive Systems*, 65-74, IEEE Computer Society.

[3] Cua, J., Manurung, R., Ong, E., and Pease, A. 2010. Representing Story Plans in SUMO. In *Proceedings of the NAACL Human Language Technology 2010 Workshop on Computational Approaches to Linguistic Creativity* (Los Angeles, USA, June 5, 2010). NAACL-HLT CALC '10. ACL, NJ, USA, 40-48.

[4] Genesereth, M. 1991. Knowledge Interchange Format. In *Proceedings of the Second International Conference on the Principles of KnowledgeRepresentation and Reasoning*, Allen, J., Fikes, R., Sandewall, E. (eds), 238-249, Morgan Kaufman Publishers.

[5] Hoder, K. 2008. *Automated Reasoning in Large Knowledge Bases*. PhD Thesis, Charles University, Prague, Czech Republic.

[6] Hong, A.J., Solis, C., Siy, J.T., Tabirao, E. and Ong, E. 2008. Picture Books: An Automated Story Generator. In *Proceedings of the 5th National Natural Language Processing Research Symposium* (Manila, Philippines, November 2008). 5NNLPRS, DLSU, Manila, Philippines.

[7] Kooijman, R. 2004. *De virtueleverhalenverteller: voorstelvoor het gebruik van een upper-ontology en eennieuwearchitectuur*. Technical Report, University of Twente, Department of Electrical Engineering, Mathematics and Computer Science.

[8] Liu, H. and Singh, P. 2004. Commonsense Reasoning in and over Natural Language. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, 293-306, Wellington, New Zealand, Springer Berlin.

[9] Machado, J. 2003. Storytelling. In *Early Childhood Experiences in Language Arts: Emerging Literacy*, 304-319. Clifton Park, N.Y., Thomson/Delmar Learning.

[10] Niles, I. and Pease, A. 2001. Towards A Standard Upper Ontology. In *Proceedings of Formal Ontology in Information Systems* (Maine, USA, October 17-19, 2001). FOIS 2001, USA, 2-9. See also www.ontologyportal.org

[11] Ong, E. 2010. A Commonsense Knowledge Base for Generating Children's Stories. In *Proceedings of the 2010 AAAI Fall Symposium Series on Common Sense Knowledge* (Virginia, USA, November 11-13, 2010). CSK '10, AAAI, USA, 82-87.

[12] Pease, A. and Benzmüller C. 2010. Sigma: An Integrated Development Environment for Logical Theories. In *Proceedings of the ECAI 2010 Workshop on Intelligent Engineering Techniques for Knowledge Bases* (Lisbon, Portugal, August 16-17, 2010). I-KBET-2010, Portugal.

[13] Riazanov, A. and Voronkov, A. 2002. The Design and Implementation of Vampire. *AI Communications*, 15(2-3), 91-110.

[14] Sutcliffe, G. 2007. TPTP, TSTP, CASC, etc. In *Lecture Notes in Computer Science*, V. Diekert, M. Volkov, and A. Voronkov (eds), Vol. 4649/2007, 6-22, Springer Berlin / Heidelberg, ISBN 978-3-540-74509-9.

[15] Swartjes, I. 2006. *The Plot Thickens: Bringing Structure and Meaning into Automated Story Generation*. Master's Thesis, University of Twente, The Netherlands.