# Story understanding through multi-representation model construction

**Erik T. Mueller**
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA
etm@us.ibm.com

## Abstract

We present an implemented model of story understanding and apply it to the understanding of a children's story. We argue that understanding a story consists of building multi-representation models of the story and that story models are efficiently constructed using a satisfiability solver. We present a computer program that contains multiple representations of commonsense knowledge, takes a narrative as input, transforms the narrative and representations of commonsense knowledge into a satisfiability problem, runs a satisfiability solver, and produces models of the story as output. The narrative, models, and representations are expressed in the language of Shanahan's event calculus.

## 1 Introduction

Story understanding is a fundamental unsolved problem in artificial intelligence and computational linguistics. In order for a computer program to understand a story text, it must be able to make inferences about states and events not explicitly described in the text. To do this it must have knowledge about the world and an ability to reason using this knowledge—in short it must be able to perform commonsense reasoning, itself a fundamental unsolved problem.

Story understanding has largely been ignored of late. We seek to remedy this situation by applying current research on commonsense reasoning to the story understanding problem. In this paper[1] we present an implemented model of commonsense reasoning for story understanding that has been applied to the understanding of a children's story.

[1]This is a condensed version of a paper that is under preparation.

### 1.1 Model-based multi-representation story understanding

We propose that understanding a story consists of building multi-representation models of the states and events described in the story. The representations are concerned with multiple realms such as space, time, needs, and feelings. There may be several representations for a single realm. Space, for example, may be represented at different levels of the spatial semantic hierarchy (Kuipers, 2000) such as topological space and metric space as well as at different levels of granularity such as room-scale and object-scale space. We further propose that models are efficiently constructed using a powerful engine, in particular a satisfiability solver, that operates in conjunction with multiple, rich representations of the commonsense world.

### 1.2 Scope and methodology

We are concerned with in-depth understanding in contrast to information extraction. Since research on commonsense reasoning to date has focused on small benchmark problems, it would be difficult to launch into the problem of in-depth understanding of adult-level stories right away. Instead, we and others have proposed to start by handling children's stories (Hirschman et al., 1999; McCarthy et al., 2002). We have formed a corpus of 15 early reader stories for pre-school and kindergarten students, drawn from the Random House Step into Reading® series. In this paper, we treat one of the stories in this corpus. The representations we develop for this story will, we hope, be applicable to the understanding of the remaining 14 stories as well as other early reader stories—though the representations will certainly require elaboration.

Since our primary research focus is on in-depth understanding, we make the simplifying assumption that the narrative text has already been parsed into event calculus formulas (Shanahan, 1997). We manually annotate

the narrative text with event calculus formulas, which are similar to the predicate-argument structures produced by semantic parsers (Alshawi, 1992; Beale et al., 1995; Gildea and Jurafsky, 2002). In a complete story understanding program, a semantic parser would feed its surface-level understanding of a story to our program, which would in turn produce a more detailed understanding.

### 1.3 Brief history of story understanding

Starting in the 1960s, a number of programs have been written that are able to read and understand a handful of stories.[2] Several programs built in the 1970s were based on the knowledge structures of scripts, plans, and goals (Schank and Abelson, 1977). The BORIS in-depth story understanding program (Dyer, 1983) integrated scripts, plans, and goals with other knowledge structures including emotions, interpersonal relations, spatiotemporal maps, and story themes.

Starting in the late 1980's, many story understanding researchers, frustrated by the lack of robustness of story understanding programs, shifted their focus from narrow coverage deep understanding to broad coverage shallow understanding or information extraction. It is currently unknown how to produce a deep understanding program with broad coverage. Two routes are apparent: (1) start with a broad coverage shallow understanding program and make it progressively deeper (Riloff (Riloff, 1999) argues for this approach), or (2) start with a narrow coverage deep understanding program and make its coverage progressively broader. In this paper we take the second route.

### 1.4 Model-based story understanding

Cognitive psychologists have argued that the reader of a narrative creates a situation or mental model of the narrative including the goals and personalities of the characters and the physical setting (Bower, 1989). Our earlier story understanding program, ThoughtTreasure (Mueller, 1998), built models of a story consisting of a sequence of time slices, where each time slice is a snapshot of (a) the physical world and (b) the mental world of each story character. The physical world was represented using spatial occupancy arrays and mental states were represented using finite automata.

In this paper we use the term model in the sense of Tarskian semantics. A model or interpretation of a language maps constant symbols of the language to elements of a domain $D$, $n$-ary function symbols to functions from $D^n$ to $D$, and $n$-ary predicate symbols to a subset of $D^n$. We confine our attention to finite domains. Time is represented by the integers 0 through a maximum time.

[2] Mueller (Mueller, 2002) provides a more detailed history of story understanding programs.

A debate over model-based versus proof-based reasoning rages in the fields of artificial intelligence (Levesque, 1986; Davis, 1991) and psychology (Johnson-Laird, 1993; Rips, 1994). The degree to which readers generate inferences and construct mental models during reading is also debated (McKoon and Ratcliff, 1992; Graesser et al., 1994). For the purposes of building and debugging a working story understanding program, the model-based approach has several advantages. First, with a model-based program the consequences of a set of formulas are immediately apparent by inspecting the models. This makes debugging faster than with a proof-based program in which facts are individually considered and proved. Second, model construction may be performed automatically, whereas proof construction often requires human guidance. Third, the process of answering a question about a story is simplified since the program may read the answer directly off the model without having to perform complex reasoning.

### 1.5 Multi-representation story understanding

The view that understanding stories involves multiple representations has been argued by Minsky (Minsky, 1986), who points out that understanding requires knowledge and skills from many realms such as the physical, social, conversational, procedural, sensory, motor, tactile, temporal, economic, and reflective realms. Several previous story understanding programs have used multiple representations. BORIS used 17 types of representation and ThoughtTreasure used five.

### 1.6 Reasoning through satisfiability

Satisfiability solvers take as input a set of boolean variables and a propositional formula over those variables and produce as output zero or more models of the formula—truth assignments for the variables such that the formula is satisfied. Satisfiability solvers may be used to perform a variety of forms of reasoning useful in understanding and answering questions about a story.

Deduction may be performed in the satisfiability framework by checking that one formula is true in every model of another formula.

Story understanding has been viewed as an abductive task (Charniak and McDermott, 1985; Hobbs et al., 1993). A satisfiability solver may be used to perform abduction for story understanding by providing the stated information as input to the solver and allowing the solver to find models that include the stated information as well as the unstated information.

Story understanding tasks such as predicting next events (McKoon and Ratcliff, 1986) require projection. A satisfiability solver may be used to perform projection by asserting the initial states and events and allowing the solver to find models of the ensuing states and events.

Planning consists of taking an initial state and a goal state, and producing a sequence of events such that the goal state results from those events. Kautz and Selman (Kautz and Selman, 1996) have demonstrated the efficiency of planning using satisfiability.

### 1.7 Satisfiability versus multi-agent systems for model construction

Several previous story understanding programs have used multi-agent systems to build representations. Charniak's early story understanding program (Charniak, 1972) used agents called demons to generate inferences. BORIS used demons to build representations as it parsed a story from left to right.

Our previous story understanding program ThoughtTreasure used a multi-agent system in which different understanding agents were responsible for maintaining different components of the model while processing a story. The understanding agents interacted with each other in order to decide on a suitable update to the model. Because of the many potential interactions, the understanding agents proved difficult for the programmer to write, maintain, and extend.

In the present work, instead of attempting to hand code a collection of agents to build models, we use a powerful solution engine to build models automatically given representations of commonsense knowledge.

### 1.8 The event calculus

We have chosen to express our representations for story understanding in the version of Shanahan's circumscriptive event calculus that uses forced separation (Shanahan, 1997). This language is an extension of many-sorted first-order predicate calculus with explicit time and can be used to express diverse representations. The event calculus predicates important for this paper are as follows:

- *Happens*$(e, t)$ represents that an event $e$ happens at time $t$.
- *HoldsAt*$(f, t)$ represents that a fluent $f$ holds at time $t$.
- *Initiates*$(e, f, t)$ represents that if event $e$ occurs at $t$ then fluent $f$ starts holding after $t$.
- *Terminates*$(e, f, t)$ represents that if event $e$ occurs at $t$ then fluent $f$ stops holding after $t$.

Reasoning using the event calculus is carried out as follows: If $\Delta_1$ and $\Delta_2$ are conjunctions of *Happens* and temporal ordering formulas, $\Sigma$ is a conjunction of *Initiates*, *Terminates*, and *Releases* axioms, $\Phi$ is the conjunction of the event calculus axioms ECF1 to ECF5 (Shanahan, 1997), $\Psi$ is a conjunction of state constraints, $\Pi$ is a conjunction of trajectory axioms, $\Omega$ is a conjunction of uniqueness-of-names axioms, and $\Gamma$ is a conjunction of *HoldsAt* formulas, then we are interested in the following:

$$CIRC[\Delta_1 \wedge \Delta_2; Happens] \wedge$$
$$CIRC[\Sigma; Initiates, Terminates, Releases] \wedge$$
$$\Phi \wedge \Psi \wedge \Pi \wedge \Omega \models \Gamma$$

Deduction and projection are performed by taking $\Delta_1$, $\Delta_2$, $\Sigma$, $\Phi$, $\Psi$, $\Pi$, and $\Omega$ as input, and producing $\Gamma$ as output. Abduction and planning are performed by taking $\Delta_1$, $\Sigma$, $\Phi$, $\Psi$, $\Pi$, $\Omega$, and $\Gamma$ as input, and producing $\Delta_2$ as output.

### 1.9 The story understanding program

Our story understanding program operates as follows: The main program takes the event calculus narrative and axiomatization as input, formulates deductive or abductive reasoning problems, and sends them to the satisfiability encoder. The satisfiability encoder sends encoded problems back to the main program. The main program sends encoded problems to the satisfiability solver. The satisfiability solver sends solutions to problems back to the main program, which produces models as output. The main program consists of 6332 lines of Python and Java code. The satisfiability encoder consists of 3658 lines of C code. The program uses off-the-shelf satisfiability solvers.

More specifically, the event calculus narrative provided as input consists of:

- annotation of the story sentences as *Happens* and *HoldsAt* formulas,
- the structure of room-scale topological space, and
- (optionally, to reduce the number of models) initial and intermediate events and fluents, represented by *Happens* and *HoldsAt* formulas.

Coreference annotation must be performed on the story sentences, so that unique story entities such as actors and physical objects are represented by unique constants in the above formulas.

### 1.10 The Snowman

The story handled by our program is an adaptation for early readers of the children's story "The Snowman" by Raymond Briggs.

It is not yet possible to process the entire Snowman story as a single satisfiability problem—the problem does not fit in memory. We therefore break the story into several segments, where each segment contains one or more time points and each segment follows the previous segment in story time. The following shows how we have divided the Snowman story into segments SNOWMAN1 through SNOWMAN8, along with the manual event calculus annotation of the sentences:

SNOWMAN1:
**This segment models the falling of individual snowflakes.**
SNOWMAN2:
Hooray!
*Happens*(*CryForJoy*(*James*), 3)
It is snowing!
*HoldsAt*(*Snowing*(*JamesOutside*), 3)
James gets dressed.
*Happens*(*GetDressed*(*James*), 5)
He runs outside.

*Happens*(*WalkThroughDoor21*(*James, JamesFrontDoor1Fl*), 10)
He makes a pile of snow.
*Happens*(*HoldSome*(*James, Snowball1, Snow1*), 12)
He makes it bigger and bigger.
*Happens*(*RollAlong*(*James, Snowball1, Snow1*), 13)
He puts a big snowball on top.
*Happens*(*PlaceOn*(*James, Snowball2, Snowball1*), 17)
SNOWMAN3:
**This segment models James going into the house to get a scarf, hat, and orange.**
SNOWMAN4:
He adds a scarf and a hat.
*Happens*(*PlaceOn*(*James, JamesScarf, Snowball2*), 0)
*Happens*(*PlaceOn*(*James, JamesHat, Snowball2*), 1)
He adds an orange for a nose.
*Happens*(*PutInside*(*James, JamesOrange, Snowball2*), 2)
He adds coal for eyes and buttons.
*Happens*(*PutInside*(*James, JamesCoal, Snowball2*), 4)
There!
What a fine snowman!
SNOWMAN5:
It is nighttime.
*Nighttime*(0)
James sneaks downstairs.
*Happens*(*WalkDownStaircase*(*James, JamesStaircase1To2*), 1)
He looks out the door.
*Happens*(*LookAt*(*James, Snowman*), 4)
What does he see?
The snowman is moving!
*Happens*(*Move*(*Snowman*), 5)
James invites him in.
*Happens*(*InviteIn*(*James, Snowman, JamesFoyer1Fl*), 6)
The snowman has never been inside a house.
SNOWMAN6:
Hello, cat!
*Happens*(*Greet*(*Snowman, JamesCat*), 0)
Hello, lamp!
*Happens*(*Greet*(*Snowman, JamesLamp*), 1)
Hello, paper towels!
*Happens*(*Greet*(*Snowman, JamesPaperTowels*), 2)
The snowman takes James's hand.
*Happens*(*Hold*(*Snowman, JamesHand*), 7)
SNOWMAN7:
They go up, up, up into the air!
*Happens*(*StartFlyingFromTo*(*Snowman, JamesOutsideGround, JamesOutsideSky*), 0)
They are flying!
*HoldsAt*(*FlyingFromTo*(*Snowman, JamesOutsideGround, JamesOutsideSky*), 1)
What a wonderful night!
SNOWMAN8:
It is morning.
*Morning*(0)
James jumps out of bed.
*Happens*(*RiseFrom*(*James, JamesBed*), 1)
He runs downstairs.
*Happens*(*WalkDownStaircase*(*James, JamesStaircase1To2*), 4)
He runs into the kitchen.
He runs outside.
*Happens*(*WalkThroughDoor21*(*James, JamesFrontDoor1Fl*), 7)
But the snowman has gone.

### 1.11 Remainder of the paper

In Section 2, we discuss our method for transforming event calculus reasoning problems into satisfiability problems. In Section 3, we discuss our multi-representation axiomatization of the commonsense knowledge needed to understand the Snowman story. In Section 4, we discuss the processing of the Snowman story by our program using the axiomatization. We conclude with future work.

## 2 A satisfiability encoding of the event calculus

We have implemented a method for encoding event calculus problems in propositional conjunctive normal form, which enables them to be solved using an off-the-shelf satisfiability solver.

Solving event calculus problems using satisfiability solvers has several advantages over solving those problems using other methods. First, satisfiability solvers are faster at solving event calculus planning problems than planners based on abductive logic programming (Shanahan, 2000; Shanahan and Witkowski, 2002). Second, solving event calculus problems using theorem proving requires computation of circumscription. The rules for computing circumscription are complicated in general (Lifschitz, 1994). One rule is given by Proposition 2 of Lifschitz, which reduces circumscription to predicate completion:

> If $F(x)$ does not contain $P$, then the circumscription CIRC[$\forall x F(x) \Rightarrow P(x); P$] is equivalent to $\forall x F(x) \Leftrightarrow P(x)$

Many cases of circumscription in the event calculus reduce directly to simple predicate completion using Proposition 2, but some do not. Notably the circumscription of *Happens* (= $P$) in a disjunctive event axiom or compound event axiom (= $\forall x F(x) \Rightarrow P(x)$) cannot be achieved using Proposition 2 because $F(x)$ does contain *Happens* in those axioms.

Our encoding method handles a larger subset of the event calculus than the method previously proposed (Shanahan and Witkowski, 2002). The method of Shanahan and Witkowski separately maps into conjunctive normal form each type of event calculus axiom such as effect axioms and precondition axioms. Our encoding method maps arbitrary axioms to conjunctive normal form by applying syntactic transformations. The generality of our method enables it to handle a larger subset of the event calculus. Table 1 provides a comparison of the coverage of the two encodings. Both methods use explanation closure frame axioms (Haas, 1987) to cope with the frame problem instead of circumscription. In our method the frame axioms are extended to allow fluents to be released from the commonsense law of inertia. Neither

| | Shanahan/ Witkowski | Our encoding |
|---|---|---|
| compound events | | |
| concurrent events | | |
| determining fluents | | ✓ |
| disjunctive event axioms | | ✓ |
| effect axioms without conditions | ✓ | ✓ |
| effect axioms with conditions | | ✓ |
| precondition axioms | ✓ | ✓ |
| releases axioms | | ✓ |
| state constraints | | ✓ |
| three-argument *Happens* | | |
| trajectory axioms | | ✓ |
| trigger axioms | | ✓ |

Table 1: Coverage of event calculus satisfiability encoding methods

method handles continuous time—both support discrete time. Due to space limitations, the complete encoding method cannot be presented here.

# 3 A multi-representation axiomatization for the Snowman story

We have created a multi-representation axiomatization of the commonsense knowledge necessary to understand the Snowman story using Shanahan's event calculus. Table 2 shows how our axiomatization compares with other event calculus axiomatizations.

| Axiomatization | Axioms | Implemented |
|---|---|---|
| our Snowman axiomatization | 181 | ✓ |
| egg cracking (Morgenstern, 2001) | 79 | |
| egg cracking (Shanahan, 1998) | 49 | |
| robot sensors and motors (Shanahan, 1996) | 33 | ✓ |
| beliefs, car crash reports (Lévy and Quantz, 1998) | 30 | |
| robot mail delivery (Shanahan, 2000) | 25 | ✓ |
| chemical plant safety (Shanahan, 2000) | 7 | ✓ |
| shopping trip (Shanahan, 2000) | 6 | ✓ |

Table 2: Axiomatizations using Shanahan's event calculus

The axiomatization is broken down into the following representations:[3]

[3]The axiomatization does not cover all aspects of the Snowman story: It does not deal with the snowman disappearing and melting and the resulting thoughts of James. It does not deal with the creation and destruction of objects such as the snowballs. These are assumed to exist over all time points.

- CTime: clock time.
- ECTime: the event calculus model of time.
- Feeling: simple positive, neutral, and negative emotions, and positive, neutral, and negative attitudes toward objects.
- OMSpace: object-scale metric space, with falling and collisions.
- OTSpace: object-scale topological space.
- PlayNeed: the need to play, with a simple model of needs and intentions.
- RTSpace: room-scale topological space.
- Sleep: sleeping and body posture.
- Snow: snow and snow falling from the sky.
- SpeechAct: some simple speech acts.
- Vision: some simple aspects of vision.

Due to space limitations, in this paper we present only one of the eleven representations: RTSpace.

## 3.1 Room-scale topological space

The predicates, functions, fluents, and events of RTSpace are shown in Table 3.

| Formula | English |
|---|---|
| *Adjacent*(*location1*, *location2*) | *location1* is adjacent to *location2*. |
| *At*(*object*, *location*) | *object* is at *location*. |
| *BuildingOf*(*room*) = *building* | The building of *room* is *building*. |
| *DoorClose*(*actor*, *door*) | *actor* closes *door*. |
| *DoorIsOpen*(*door*) | *door* is open. |
| *DoorLock*(*actor*, *door*) | *actor* locks *door*. |
| *DoorOpen*(*actor*, *door*) | *actor* opens *door*. |
| *DoorUnlocked*(*door*) | *door* is unlocked. |
| *DoorUnlock*(*actor*, *door*) | *actor* unlocks *door*. |
| *Floor*(*room*) = *integer* | The floor of *room* is *integer*. |
| *GroundOf*(*outside*) = *ground* | The ground of *outside* is *ground*. |
| *LookOutOnto*(*room*) = *outside* | *room* looks out onto *outside*. |
| *NearPortal*(*object*, *portal*) | *object* is at a location that has *portal*. |
| *Side1*(*portal*) = *location* | Side one of *portal* is *location*. |
| *Side2*(*portal*) = *location* | Side two of *portal* is *location*. |
| *SkyOf*(*outside*) = *sky* | The sky of *outside* is *sky*. |
| *WalkDownStaircase*(*actor*, *staircase*) | *actor* walks down *staircase*. |
| *WalkThroughDoor12*(*actor*, *door*) | *actor* walks through side one of *door*. |
| *WalkThroughDoor21*(*actor*, *door*) | *actor* walks through side two of *door*. |
| *WalkUpStaircase*(*actor*, *staircase*) | *actor* walks up *staircase*. |

Table 3: RTSpace

This representation of space consists of locations (rooms and outside areas), which are connected by portals (doors and staircases).

A state constraint says that an object is at one location at a time:

**Axiom 1.**
$HoldsAt(At(object, location1), time) \wedge$
$HoldsAt(At(object, location2), time) \Rightarrow$
$location1 = location2$

A state constraint says that an object is near a portal if and only if there is a location such that the object is at the location and one of the sides of the portal is the location:

**Axiom 2.**
$HoldsAt(NearPortal(object, portal), time) \Leftrightarrow$
$\exists\ location\ (Side1(portal) = location \vee$
$Side2(portal) = location) \wedge$
$HoldsAt(At(object, location), time)$

A precondition axiom states that for an actor to unlock a door, the actor must be awake, the door must not already be unlocked, and the actor must be near the door:

**Axiom 3.**
*Happens*(*DoorUnlock*(*actor*, *door*), *time*) ⇒
*HoldsAt*(*Awake*(*actor*), *time*) ∧
¬*HoldsAt*(*DoorUnlocked*(*door*), *time*) ∧
*HoldsAt*(*NearPortal*(*actor*, *door*), *time*)

An effect axiom states that if an actor unlocks a door, the door will be unlocked:

**Axiom 4.**
*Initiates*(*DoorUnlock*(*actor*, *door*),
*DoorUnlocked*(*door*), *time*)

We have similar precondition and effect axioms for locking a door.

A state constraint says that if a door is open, it is unlocked:

**Axiom 5.**
*HoldsAt*(*DoorIsOpen*(*door*), *time*) ⇒
*HoldsAt*(*DoorUnlocked*(*door*), *time*)

A precondition axiom states that for an actor to open a door, the actor must be awake, the door must not already be open, the door must be unlocked, and the actor must be near the door:

**Axiom 6.**
*Happens*(*DoorOpen*(*actor*, *door*), *time*) ⇒
*HoldsAt*(*Awake*(*actor*), *time*) ∧
¬*HoldsAt*(*DoorIsOpen*(*door*), *time*) ∧
*HoldsAt*(*DoorUnlocked*(*door*), *time*) ∧
*HoldsAt*(*NearPortal*(*actor*, *door*), *time*)

An effect axiom states that if an actor opens a door, the door will be open:

**Axiom 7.**
*Initiates*(*DoorOpen*(*actor*, *door*), *DoorIsOpen*(*door*), *time*)

We have similar precondition and effect axioms for closing a door.

Precondition axioms state that for an actor to walk through a side of a door, the actor must be awake and standing, the door must be open, and the actor must be at the side of the door that the actor walks through:

**Axiom 8.**
*Happens*(*WalkThroughDoor12*(*actor*, *door*), *time*) ⇒
*HoldsAt*(*Awake*(*actor*), *time*) ∧
*HoldsAt*(*Standing*(*actor*), *time*) ∧
*HoldsAt*(*DoorIsOpen*(*door*), *time*) ∧
*HoldsAt*(*At*(*actor*, *Side1*(*door*)), *time*)

**Axiom 9.**
*Happens*(*WalkThroughDoor21*(*actor*, *door*), *time*) ⇒
*HoldsAt*(*Awake*(*actor*), *time*) ∧
*HoldsAt*(*Standing*(*actor*), *time*) ∧
*HoldsAt*(*DoorIsOpen*(*door*), *time*) ∧
*HoldsAt*(*At*(*actor*, *Side2*(*door*)), *time*)

Effect axioms state that if an actor walks through one side of a door, the actor will be at the other side of the door:

**Axiom 10.**
*Side2*(*door*) = *location* ⇒
*Initiates*(*WalkThroughDoor12*(*actor*, *door*),
*At*(*actor*, *location*), *time*)

**Axiom 11.**
*Side1*(*door*) = *location* ⇒
*Initiates*(*WalkThroughDoor21*(*actor*, *door*),
*At*(*actor*, *location*), *time*)

**Axiom 12.**
*Side1*(*door*) = *location* ⇒
*Terminates*(*WalkThroughDoor12*(*actor*, *door*),
*At*(*actor*, *location*), *time*)

**Axiom 13.**
*Side2*(*door*) = *location* ⇒
*Terminates*(*WalkThroughDoor21*(*actor*, *door*),
*At*(*actor*, *location*), *time*)

We have similar precondition and effect axioms for walking up and down a staircase.

A state constraint says that if an actor is outside, the actor is dressed:

**Axiom 14.**
*HoldsAt*(*At*(*actor*, *outside*), *time*) ⇒
*HoldsAt*(*Dressed*(*actor*), *time*)

Two locations are adjacent if and only if they have a portal in common:

**Axiom 15.**
*Adjacent*(*location1*, *location2*) ⇔
∃ *portal* (*Side1*(*portal*) = *location1* ∧
*Side2*(*portal*) = *location2*) ∨
(*Side2*(*portal*) = *location1* ∧
*Side1*(*portal*) = *location2*)

State constraints fix the location of ground and sky:

**Axiom 16.**
*GroundOf*(*outside*) = *ground* ⇒
*HoldsAt*(*At*(*ground*, *outside*), *time*)

**Axiom 17.**
*SkyOf*(*outside*) = *sky* ⇒
*HoldsAt*(*At*(*sky*, *outside*), *time*)

## 4 Processing the Snowman story

The complete run of the Snowman story takes 45 minutes on a machine with a 700 MHz Pentium III processor and 512 megabytes of RAM. Statistics on processing the segments are shown in Table 4.

Due to space limitations, we cannot show the model of all the story segments. We present here the model of the SNOWMAN2 segment:

0
*Asleep*(*James*)
*At*(*JamesBed*, *JamesBedroom2Fl*)
*At*(*JamesCoal*, *JamesOutside*)
*At*(*JamesHat*, *JamesBedroom2Fl*)
*At*(*JamesOrange*, *JamesKitchen1Fl*)
*At*(*JamesScarf*, *JamesBedroom2Fl*)
*At*(*James*, *JamesBedroom2Fl*)
*At*(*Snow1*, *JamesOutside*)
*At*(*Snowball1*, *JamesOutside*)

| Name | Vars | Clauses | Encode | Solve |
|---|---|---|---|---|
| SNOWMAN1 | 5,489 | 48,952 | 59.45 | 73.29 |
| SNOWMAN2 | 12,415 | 227,781 | 768.59 | 57.67 |
| SNOWMAN3 | 8,503 | 152,382 | 365.53 | 32.60 |
| SNOWMAN4 | 3,963 | 69,262 | 88.53 | 14.68 |
| SNOWMAN5 | 7,227 | 149,997 | 355.71 | 14.72 |
| SNOWMAN6 | 6,628 | 117,614 | 262.39 | 7.40 |
| SNOWMAN7 | 2,551 | 7,451 | 6.83 | 0.41 |
| SNOWMAN8 | 5,470 | 90,091 | 151.11 | 20.43 |

Table 4: Snowman story runtime statistics (times in seconds)

*At*(*Snowball2, JamesOutside*)
*At*(*Snowman, JamesOutside*)
*Awake*(*Snowman*)
*Calm*(*James*)
*Calm*(*Snowman*)
*Diameter*(*Snowball1*, 1)
*Diameter*(*Snowball2*, 1)
*DoorIsOpen*(*JamesDoor2Fl*)
*DoorUnlocked*(*JamesDoor2Fl*)
*DoorUnlocked*(*JamesKitchenDoor1Fl*)
*Dressed*(*Snowman*)
*HungryToPlay*(*James*)
*LikeSnow*(*James*)
*LikeSnow*(*Snowman*)
*Like*(*Snowman, James*)
*LyingOn*(*James, JamesBed*)
*Lying*(*James*)
*NearPortal*(*JamesBed, JamesDoor2Fl*)
*NearPortal*(*JamesCoal, JamesFrontDoor1Fl*)
*NearPortal*(*JamesHat, JamesDoor2Fl*)
*NearPortal*(*JamesOrange, JamesKitchenDoor1Fl*)
*NearPortal*(*JamesScarf, JamesDoor2Fl*)
*NearPortal*(*James, JamesDoor2Fl*)
*NearPortal*(*Snow1, JamesFrontDoor1Fl*)
*NearPortal*(*Snowball1, JamesFrontDoor1Fl*)
*NearPortal*(*Snowball2, JamesFrontDoor1Fl*)
*NearPortal*(*Snowman, JamesFrontDoor1Fl*)
*SatiatedFromPlay*(*Snowman*)
*Sleep0*(*James*)
*Sleep3*(*Snowman*)
*Standing*(*Snowman*)
*Happens*(*StartSnowing*(*JamesOutside*), 0)
1
+*Snowing*(*JamesOutside*)
*Happens*(*WakeUp*(*James*), 1)
2
**Event occurrences are shown at the end of each time point. Only changes in what fluents hold from one time point to the next are shown. Thus after the WakeUp event occurs above, James is no longer asleep and he is awake:**
-*Asleep*(*James*)
-*Sleep0*(*James*)
+*Awake*(*James*)
+*Sleep1*(*James*)
**An axiom in the Feeling representation triggers this event in response to the snow:**
*Happens*(*BecomeHappy*(*James*), 2)
**An axiom in the PlayNeed representation triggers this event in response to the snow:**
*Happens*(*IntendToPlay*(*James, JamesOutside*), 2)
3
-*Calm*(*James*)
-*HungryToPlay*(*James*)
+*Happy*(*James*)
+*IntentionToPlay*(*James, JamesOutside*)
*Happens*(*CryForJoy*(*James*), 3)
4
*Happens*(*RiseFrom*(*James, JamesBed*), 4)
5
-*LyingOn*(*James, JamesBed*)
-*Lying*(*James*)
-*Sleep1*(*James*)
+*Sleep2*(*James*)
+*Standing*(*James*)
*Happens*(*GetDressed*(*James*), 5)
6
-*Sleep2*(*James*)
+*Dressed*(*James*)
+*Sleep3*(*James*)
*Happens*(*WalkThroughDoor12*(*James, JamesDoor2Fl*), 6)
7
**James was in his bedroom from time points 0 to 6 inclusive. After he walks through the bedroom door above, he is no longer in his bedroom:**
-*At*(*James, JamesBedroom2Fl*)
+*At*(*James, JamesHallway2Fl*)
+*NearPortal*(*James, JamesStaircase1To2*)
*Happens*(*WalkDownStaircase*(*James, JamesStaircase1To2*), 7)
8
-*At*(*James, JamesHallway2Fl*)
-*NearPortal*(*James, JamesDoor2Fl*)
+*At*(*James, JamesFoyer1Fl*)
+*NearPortal*(*James, JamesFrontDoor1Fl*)
+*NearPortal*(*James, JamesKitchenDoor1Fl*)
*Happens*(*DoorUnlock*(*James, JamesFrontDoor1Fl*), 8)
9
+*DoorUnlocked*(*JamesFrontDoor1Fl*)
*Happens*(*DoorOpen*(*James, JamesFrontDoor1Fl*), 9)
10
+*DoorIsOpen*(*JamesFrontDoor1Fl*)
*Happens*(*WalkThroughDoor21*(*James, JamesFrontDoor1Fl*), 10)
11
-*At*(*James, JamesFoyer1Fl*)
-*NearPortal*(*James, JamesKitchenDoor1Fl*)
-*NearPortal*(*James, JamesStaircase1To2*)
**Optional intermediate fluents fix the time point at which James acts on his intention to play, thereby reducing the number of models:**
+*ActOnIntentionToPlay*(*James, JamesOutside*)
+*At*(*James, JamesOutside*)
*Happens*(*Play*(*James, JamesOutside*), 11)
12
-*ActOnIntentionToPlay*(*James, JamesOutside*)
-*IntentionToPlay*(*James, JamesOutside*)
+*SatiatedFromPlay*(*James*)
*Happens*(*HoldSome*(*James, Snowball1, Snow1*), 12)
13
+*Holding*(*James, Snowball1*)
*Happens*(*RollAlong*(*James, Snowball1, Snow1*), 13)
14
-*Diameter*(*Snowball1*, 1)
+*Diameter*(*Snowball1*, 2)
*Happens*(*LetGoOf*(*James, Snowball1*), 14)

15
-*Holding*(*James*, *Snowball1*)
*Happens*(*HoldSome*(*James*, *Snowball2*, *Snow1*), 15)
16
+*Holding*(*James*, *Snowball2*)
*Happens*(*RollAlong*(*James*, *Snowball2*, *Snow1*), 16)
17
-*Diameter*(*Snowball2*, 1)
+*Diameter*(*Snowball2*, 2)
*Happens*(*PlaceOn*(*James*, *Snowball2*, *Snowball1*), 17)
18
-*Holding*(*James*, *Snowball2*)
+*On*(*Snowball2*, *Snowball1*)

## 5 Conclusions and future work

We have described a model-based multi-representation approach to story understanding that can be used to produce a detailed understanding of a children's story.

Future work includes the following. First, the program should be parallelized and run on a collection of networked machines so that it can solve much larger problems and solve them quickly to facilitate debugging. Second, the multi-representation axiomatization should be elaborated for a second story, and eventually for the entire early reader corpus. Third, algorithms for minimizing event occurrences in abduction should be added to the program. Fourth, a meta-level reasoning module should be added to formulate event calculus reasoning problems, including setting up the story and segment initial states. Fifth, the story understanding system should be hooked up to a semantic parser for input and natural language generator for output. Finally, a natural language question answering module should be added.

## References


Alshawi, H., editor (1992). *The Core Language Engine*. MIT Press, Cambridge, MA.

Beale, S., Nirenburg, S., and Mahesh, K. (1995). Semantic analysis in the Mikrokosmos machine translation project. In *Proceedings of the Second Symposium on Natural Language Processing*.

Bower, G. H. (1989). Mental models in text understanding. In Bennett, A. F. and McConkey, K. M., editors, *Cognition in individual and social contexts*, pages 129–144. Elsevier, Amsterdam.

Charniak, E. (1972). Toward a model of children's story comprehension. Technical Report AITR-266, Cambridge, MA: Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Charniak, E. and McDermott, D. (1985). *Introduction to artificial intelligence*. Addison-Wesley, Reading, MA.

Davis, E. (1991). Lucid representations. Technical Report TR1991-565, New York: Computer Science Department, New York University.

Dyer, M. G. (1983). *In-depth understanding: A computer model of integrated processing for narrative comprehension*. MIT Press, Cambridge, MA.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Graesser, A. C., Singer, M., and Trabasso, T. (1994). Constructing inferences during narrative text comprehension. *Psychological Review*, 101(3):371–395.

Haas, A. R. (1987). The case for domain-specific frame axioms. In Brown, F. M., editor, *The frame problem in artificial intelligence: Proceedings of the 1987 workshop*, pages 343–348, Los Altos, CA. Morgan Kaufmann.

Hirschman, L., Light, M., Breck, E., and Burger, J. D. (1999). Deep Read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 325–332.

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

Johnson-Laird, P. N. (1993). *Human and machine thinking*. Lawrence Erlbaum, Hillsdale, NJ.

Kautz, H. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Annual Conference on Innovative Applications of Artificial Intelligence*, pages 1194–1201, Menlo Park, CA. AAAI Press.

Kuipers, B. (2000). The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233.

Levesque, H. J. (1986). Making believers out of computers. *Artificial Intelligence*, 30(1):81–108.

Lévy, F. and Quantz, J. J. (1998). Representing beliefs in a situated event calculus. In Prade, H., editor, *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, pages 547–551, Chichester, UK. John Wiley.

Lifschitz, V. (1994). Circumscription. In Gabbay, D. M., Hogger, C. J., and Robinson, J. A., editors, *Handbook of logic in artificial intelligence and logic programming*, volume 3: Nonmonotonic reasoning and uncertain reasoning, pages 298–352. Oxford University Press, Oxford.

McCarthy, J., Minsky, M., Sloman, A., Gong, L., Lau, T., Morgenstern, L., Mueller, E. T., Riecken, D., Singh, M., and Singh, P. (2002). An architecture of diversity for commonsense reasoning. *IBM Systems Journal*, 41(3):530–539.

McKoon, G. and Ratcliff, R. (1986). Inferences about predictable events. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 12(1):82–91.

McKoon, G. and Ratcliff, R. (1992). Inference during reading. *Psychological Review*, 99(3):440–466.

Minsky, M. (1986). *The society of mind*. Simon & Schuster, New York.

Morgenstern, L. (2001). Mid-sized axiomatizations of commonsense problems: A case study in egg cracking. *Studia Logica*, 67:333–384.

Mueller, E. T. (1998). *Natural language processing with ThoughtTreasure*. Signiform, New York. `http://www.signiform.com/tt/book/`.

Mueller, E. T. (2002). Story understanding. In Nadel, L., editor, *Encyclopedia of Cognitive Science*, volume 4, pages 238–246. Nature Publishing Group, London.

Riloff, E. (1999). Information extraction as a stepping stone toward story understanding. In Ram, A. and Moorman, K., editors, *Understanding language understanding: Computational models of reading*, pages 435–460. MIT Press, Cambridge, MA.

Rips, L. J. (1994). *The psychology of proof*. MIT Press, Cambridge, MA.

Schank, R. C. and Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum, Hillsdale, NJ.

Shanahan, M. (1996). Robotics and the common sense informatic situation. In Wahlster, W., editor, *Proceedings of the Twelfth European Conference on Artificial Intelligence*, pages 684–688, Chichester, UK. John Wiley.

Shanahan, M. (1997). *Solving the frame problem*. MIT Press, Cambridge, MA.

Shanahan, M. (1998). A logical formalisation of Ernie Davis's egg cracking problem. In *Fourth Symposium on Logical Formalizations of Commonsense Reasoning*. `http://www.dcs.qmw.ac.uk/~mps/egg_murray.ps.Z`.

Shanahan, M. (2000). An abductive event calculus planner. *Journal of Logic Programming*, 44(1–3):207–240.

Shanahan, M. and Witkowski, M. (2002). Event calculus planning through satisfiability. Unpublished paper.