Understanding Goal-Based Stories through Model Finding and Planning

Erik T. Mueller
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA

Abstract

We present an approach for understanding goal-based stories that combines model finding and planning. We present an algorithm that takes narrated actions, narrated properties, and a domain description as input and produces weighted models as output. We demonstrate the use of the approach on two examples.

Introduction

The story understanding task consists of taking a story as input and producing an understanding of the story as output. Story understanding is an important problem because it has many possible applications in computer systems such as advisory, alert, question answering, search, and summarization systems. Story understanding can be broken down into the tasks of parsing and inferencing. In *parsing*, story text or speech is converted into predicate-argument structure representations (Alshawi 1992; Gildea & Jurafsky 2002; Palmer, Gildea, & Kingsbury 2005). In *inferencing*, those representations are elaborated by filling in missing details (Graesser, Singer, & Trabasso 1994; Kintsch 1998).

Propositional satisfiability (SAT) solving (Du, Gu, & Pardalos 1997) can be used to perform inferencing in story understanding (Mueller 2003; 2004a; 2004b; 2006; 2007). SAT solving is an appealing technique because of the widespread availability of SAT solvers and the fact that their efficiency is continually being improved. A story and commonsense knowledge are represented as formulas in first-order logic. Inferencing is then performed in two steps:

1. The first-order formulas are translated into a formula of propositional logic in conjunctive normal form using the technique of renaming subformulas (Plaisted & Greenbaum 1986).
2. A SAT solver is run on the propositional formula to produce models of the story.

Each model produced by the SAT solver is one possible understanding of the story.

Step 1 is the computational bottleneck, typically taking about 15 times as long as the second step (Mueller 2003). It becomes an even greater bottleneck when mental states are represented. For example, consider the following story:

[Willa1] Willa was hungry. She grabbed the Michelin guide. She got in her car. (Schank & Abelson 1977, p. 71; Schank & Riesbeck 1981, p. 182)

Understanding this story requires inferring that Willa has a goal to satisfy hunger and that her actions constitute part of a plan to achieve this goal. To reason about Willa's mental plan, we must represent mental trees of hypothetical world states nested inside actual world states. Grounding these states gives rise to a large number of propositions making the SAT approach computationally infeasible.

In this paper, we present an efficient approach for inferencing in story understanding that supports reasoning about mental plans. The approach combines model finding and planning. A model finder is used to find models consistent with the story. A planner is used to prune models in which actions are not goal-based. The model finder and planner operate in an interleaved fashion in order to avoid an explosion in the number of models. We make several simplifying assumptions:

- The story has a single character.
- Exactly one action is performed at each timepoint.
- The timepoints of narrated actions and properties and the maximum timepoint are given.

Definitions

We use a formalism for reasoning about action similar to that of Ghallab, Nau, and Traverso (2004). We start with a finite set of *actions* A, a finite set of *fluents* (or properties of the world) F, and a finite set of *timepoints* $T = \{0, 1, \ldots, n\}$ for some *maximum timepoint* n. Let $V = \{\mathbf{T}, \mathbf{F}\}$ be the set of *truth values*.

Definition 1. A *state* is a function $\sigma : F \to V$ specifying the truth value of every fluent.

Definition 2. A *story domain* is a tuple $D = (\Sigma, \gamma, G)$ consisting of

- a set Σ of states not containing $\square$,
- a *state transition function* $\gamma : \Sigma \times A \to \Sigma \cup \{\square\}$, and
- a function $G : \Sigma \to 2^F$ specifying the *active goals* in a state.

If $\gamma(\sigma, a) = \square$, then action a cannot be performed in state σ.

Definition 3. A *story* for a story domain $D = (\Sigma, \gamma, G)$ is a tuple $S = (\alpha, \phi)$ consisting of

- a function $\alpha : T \rightarrow A \cup \{\square\}$ specifying the *narrated actions* and
- a function $\phi : F \times T \rightarrow V \cup \{\mathbf{U}\}$ specifying the *narrated fluents*.

If $\alpha(t) = \square$, then the action performed at timepoint t is unknown (not specified in the narrative). If $\phi(f, t) = \mathbf{U}$, then the truth value of fluent f at timepoint t is unknown.

Definition 4. A *story understanding problem* is a tuple $U = (D, S)$ consisting of

- a story domain D and
- a story S for D.

Definition 5. A *story interpretation* for a story domain $D = (\Sigma, \gamma, G)$ is a tuple $I = (\pi, \delta)$ consisting of

- a sequence of actions (elements of A) $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ and
- a sequence of states (elements of Σ) $\delta = \langle \sigma_0, \sigma_1, \ldots, \sigma_n \rangle$.

Definition 6. A story interpretation $I = (\pi, \delta)$ for a story domain $D = (\Sigma, \gamma, G)$ is a *model* of a story understanding problem $U = (D, S)$ where $S = (\alpha, \phi)$ if and only if

- for every $a \in A$ and $t \in T - \{n\}$, if $\alpha(t) \neq \square$, then $\alpha(t) = \pi_t$,
- for every $f \in F$ and $t \in T$, if $\phi(f, t) \neq \mathbf{U}$, then $\phi(f, t) = \delta_t(f)$, and
- for every $t \in T - \{n\}$, $\delta_{t+1} = \gamma(\delta_t, \pi_t)$.

If s is a sequence and i is a nonnegative integer, then s_i denotes the ith element of s (0-origin).

Definition 7. A *planning problem* is a tuple $P = (D, \sigma, g)$ consisting of

- a story domain $D = (\Sigma, \gamma, G)$,
- an initial state $\sigma \in \Sigma$, and
- a goal fluent $g \in F$.

Definition 8. Let $\pi = \langle a_0, \ldots, a_{k-1} \rangle$ be a sequence of actions and $P = (D, \sigma, g)$ be a planning problem where $D = (\Sigma, \gamma, G)$. Then π is a *solution* for P if and only if there exists a sequence of states (elements of Σ) $\langle \sigma_0, \ldots, \sigma_k \rangle$ such that

- $\sigma_0 = \sigma$,
- $\sigma_k(g) = \mathbf{T}$, and
- for every $i \in \{0, 1, \ldots, k-1\}$, $\sigma_{i+1} = \gamma(\sigma_i, a_i)$.

So far, the notion of a model does not take character goals into account. A model is any interpretation consistent with the narrated actions, narrated fluents, and state transition function. To take goals into account, we place additional requirements on models along the lines of $\mathcal{M}$odular-$\mathcal{E}$ (Kakas, Michael, & Miller 2005). We define a goal-based model as a model in which each action of a character is performed as part of a plan to achieve some active goal of the character. Specifically, an action performed at a timepoint must be the first action of some plan starting from that timepoint to achieve some active goal at that timepoint.

The notion of a goal-based model is flexible and able to account for many instances of goal-based stories. The sequence of story actions is not required to be a single plan for a single active goal. There may be several active goals at a timepoint, active goals may differ from timepoint to timepoint, and actions may be performed on behalf of different active goals from timepoint to timepoint.

Definition 9. A sequence of actions π is a *goal-based behavior* in a story domain $D = (\Sigma, \gamma, G)$ at a state $\sigma \in \Sigma$ if and only if there exists an active goal $g \in G(\sigma)$ such that π is a solution to the planning problem (D, σ, g).

Definition 10. A story interpretation $I = (\pi, \delta)$ for a story domain $D = (\Sigma, \gamma, G)$ is a *goal-based model* of a story understanding problem $U = (D, S)$ if and only if

- I is a model of U and
- for every $t \in T - \{n\}$, there exists a sequence of actions π' such that $\pi_t = \pi'_0$ and π' is a goal-based behavior in D at δ_t.

Now, many models qualify as goal-based models. Given a solution to a planning problem, there are many actions we can prepend to create a solution one step longer. For example, we can prepend the action of blinking to any plan for eating at a restaurant. Therefore, we compute weights for actions and goal-based models that favor shorter plans. Weights are real numbers ranging from 0 (worst) to 1 (best).

Definition 11. If a is an action, $D = (\Sigma, \gamma, G)$ is a story domain, and $\sigma \in \Sigma$, then $\mathit{weight}(a, \sigma, D) = 2^{k-l}$ where

- k is the length of the shortest sequence of actions π such that π is a goal-based behavior in D at σ and
- l is the length of the shortest sequence of actions π such that $a = \pi_0$ and π is a goal-based behavior in D at σ.

Definition 12. If story interpretation $I = (\pi, \delta)$ for a story domain D is a goal-based model of a story understanding problem $U = (D, S)$, then $\mathit{weight}(I) = \prod_{i \in \{0,1,\ldots,n-1\}} \mathit{weight}(\pi_i, \delta_i, D)$.

The function G specifies the set of fluents that the story character wishes to be true in a state σ. For example, we may define G so that the character wishes to be satiated in any state in which the character is hungry. Notice that our formalism allows us to infer both the goals of characters as well as the fluents that give rise to those goals. For example, if the story states that the character ate a sandwich but does not state that the character was hungry, we can infer that the character was hungry and had the goal to be satiated. To formalize this process of inference, we define entailment and goal entailment.

Definition 13. Let D be a story domain, Θ be a set of story interpretations for D, f be a fluent, t be a timepoint, and v be a truth value. Then Θ *entails* $f = v$ at t if and only if for every $(\pi, \delta) \in \Theta$, $\delta_t(f) = v$.

Definition 14. Let D be a story domain, Θ be a set of story interpretations for D, g be a fluent, and t be a timepoint. Then Θ *goal-entails* g at t if and only if for every $(\pi, \delta) \in \Theta$, $g \in G(\delta_t)$.

Typically we take Θ to be a set of goal-based models whose weights are greater than some threshold.

Algorithm *understand*(A, F, n, D, S)
Inputs: set of actions A, set of fluents F, integer n, story domain $D = (\Sigma, \gamma, G)$, story $S = (\alpha, \phi)$
$r \leftarrow$ *get_initial*(F, D, S)
for t **in** $\{0, 1, \ldots, n-1\}$ **do**
  $r' \leftarrow \emptyset$
  for (π, δ, w) **in** r **do**
    $r' \leftarrow r' \cup$ *extend*$(A, F, D, S, t, \pi, \delta, w)$
  end for
  $r \leftarrow r'$
end for
return r

Figure 1: Understanding a story

Algorithm *get_initial*(F, D, S)
Inputs: set of fluents F, story domain $D = (\Sigma, \gamma, G)$, story $S = (\alpha, \phi)$
$r \leftarrow \emptyset$
for σ **in** Σ **do**
  if *satisfies*$(F, \sigma, \phi, 0)$ **then**
    $r \leftarrow r \cup \{(\langle\,\rangle, \langle\sigma\rangle, 1)\}$
  end if
end for
return r

Figure 2: Building an initial partial interpretation

Algorithms

The top-level algorithm for story understanding is shown in Figure 1. The algorithm for building an initial partial interpretation is shown in Figure 2. The algorithm for determining whether a state satisfies the narrated fluents is shown in Figure 3. The algorithm for extending a partial interpretation by one timepoint is shown in Figure 4. If $s = \langle s_0, s_1, \ldots, s_k\rangle$ is a sequence, then $s.e$ denotes the sequence $\langle s_0, s_1, \ldots, s_k, e\rangle$. The algorithm for computing a weight is shown in Figure 5. A sample planning algorithm is shown in Figure 6.

The top-level algorithm *understand* interleaves model finding and planning as follows. First *understand* invokes *get_initial* to generate a set of partial interpretations in which only the state at timepoint 0 is provided. Then *understand* enters a loop. On each iteration of the loop, *understand* invokes *extend* on each partial interpretation to extend it by one timepoint. The *extend* algorithm generates extensions of a partial interpretation that are consistent with the story (model finding). If the global parameter *ALLMODELS* is **false**, then *extend* invokes *get_weight*, which invokes *find_plans*, to generate plans that are used to prune the extensions (planning). Generated plans are discarded and the *understand* algorithm proceeds to the next iteration of the loop, which performs more model finding and planning.

Implementation

The algorithms have been implemented in Python. The implementation makes use of several optimizations:

Algorithm *satisfies*(F, σ, ϕ, t)
Inputs: set of fluents F, state σ, narrated fluents ϕ, integer t
for f **in** F **do**
  if $\phi(f, t) \neq$ **U and** $\phi(f, t) \neq \sigma(f)$ **then**
    return false
  end if
end for
return true

Figure 3: Determining whether a state satisfies the narrated fluents

Algorithm *extend*$(A, F, D, S, t, \pi, \delta, w)$
Inputs: set of actions A, set of fluents F, story domain $D = (\Sigma, \gamma, G)$, story $S = (\alpha, \phi)$, integer t, sequence of actions π, sequence of states δ, real number w
$\sigma = \delta_t$
if $\alpha(t) \neq \square$ **then**
  actions $\leftarrow \{\alpha(t)\}$
else
  actions $\leftarrow A$
end if
$r \leftarrow \emptyset$
for a **in** *actions* **do**
  $\sigma' \leftarrow \gamma(\sigma, a)$
  if $\sigma' \neq \square$ **and** *satisfies*$(F, \sigma', \phi, t+1)$ **then**
    weight $\leftarrow$ *get_weight*(A, D, σ, a)
    if *weight* > 0 **or** *ALLMODELS* **then**
      $r \leftarrow r \cup \{(\pi.a, \delta.\sigma', w \cdot$ *weight*$)\}$
    end if
  end if
end for
return r

Figure 4: Extending a partial interpretation

Algorithm *get_weight*(A, D, σ, a)
Inputs: set of actions A, story domain $D = (\Sigma, \gamma, G)$, state σ, action a
$k \leftarrow -1$
for l **in** $\{1, 2, \ldots,$ *MAXPLANLEN*$\}$ **do**
  for *goal* **in** $G(\sigma)$ **do**
    for π **in** *find_plans*$(A, D,$ *goal*$, \sigma, l, \langle\,\rangle, \{\sigma\})$ **do**
      if $k = -1$ **then**
        $k \leftarrow l$
      end if
      if $\pi_0 = a$ **then**
        return 2^{k-l}
      end if
    end for
  end for
end for
return 0

Figure 5: Computing a weight

Algorithm *find_plans*$(A, D, goal, \sigma, l, \pi, visited)$
Inputs: set of actions A, story domain $D = (\Sigma, \gamma, G)$, fluent *goal*, state σ, integer l, sequence of actions π, set of states *visited*
if $\sigma(goal) = \mathbf{T}$ **then**
  return $\{\pi\}$
end if
if $l = 0$ **then**
  return $\emptyset$
end if
$r \leftarrow \emptyset$
for a **in** A **do**
  $\sigma' \leftarrow \gamma(\sigma, a)$
  if $\sigma' \neq \square$ **and** $\sigma' \notin$ *visited* **then**
    $r \leftarrow r \cup$ *find_plans*$(A, D, goal, \sigma', l-1, \pi.a,$
      $visited \cup \{\sigma'\})$
  end if
end for
return r

Figure 6: Planning

- The set of states Σ includes only those states satisfying a collection of *state constraints*.
- The value of the state transition function $\gamma(\sigma, a)$ for a particular state σ and action a is only computed the first time it is needed. The value is then cached for future use.
- The narrated fluents are represented as a list of triples of fluent, timepoint, and truth value so that *satisfies* does not have to loop through all fluents.

For convenience in specifying the story domain and story, the implementation supports a first-order notation that is grounded to produce a propositional notation. For example, if the agents are *Agent1* and *Agent2* and the vehicles are *Vehicle1* and *Vehicle2*, then the grounding of the first-order expression *GetIn*$(agent, vehicle)$ is the four propositions *GetIn*$(Agent1, Vehicle1)$, *GetIn*$(Agent2, Vehicle1)$, *GetIn*$(Agent1, Vehicle2)$, and *GetIn*$(Agent2, Vehicle2)$.

Examples

We present a run of Willa1 as well as the following story:

> [Willa2] Willa was bored. She picked up the Michelin guide.

Each story was run twice. In the first run, all models were considered (*ALLMODELS* = **true**). In the second run, only goal-based models were considered (*ALLMODELS* = **false**). Summary statistics are shown in Table 1. Each row provides the story name (**story**), the maximum timepoint (n), the number of models (**all**), the number of goal-based models (**gb**), the number of goal-based models with weights greater than or equal to .5 (**gb** $\geq .5$), and the number of goal-based models with weights equal to 1 (**gb** $= 1$). The number of models of Willa2 is significantly smaller than the number of models of Willa1 because of the values of n.

The Willa story domain includes the following *operators*:

story	n	**all**	**gb**	**gb** $\geq .5$	**gb** $= 1$
Willa1	5	110,640	72	8	2
Willa2	2	1,424	54	38	16

Table 1: Summary statistics

Eat$(agent, food)$
  precond: *At*$(agent, location)$, *At*$(food, location)$
  effects: $\neg$*Hungry*$(agent)$, $\neg$*At*$(food, location)$,
    Satiated$(agent)$

PickUp$(agent, physobj)$
  precond: *At*$(agent, location)$, *At*$(physobj, location)$
  effects: *Holding*$(agent, physobj)$

SetDown$(agent, physobj)$
  precond: *At*$(agent, location)$, *At*$(physobj, location)$
  effects: $\neg$*Holding*$(agent, physobj)$

Read$(agent, guide)$
  precond: *Holding*$(agent, guide)$,
    At$(Restaurant1, location)$
  effects: *KnowLocation*$(agent, Restaurant1, location)$

Read$(agent, guide)$
  precond: *Holding*$(agent, guide)$
  effects: *Entertained*$(agent)$

GetIn$(agent, vehicle)$
  precond: *At*$(agent, location)$, *At*$(vehicle, location)$
  effects: *In*$(agent, vehicle)$

GetOut$(agent, vehicle)$
  precond: *In*$(agent, vehicle)$
  effects: $\neg$*In*$(agent, vehicle)$

Drive$(agent, vehicle, building)$
  precond: $location1 \neq location2$, *In*$(agent, vehicle)$,
    At$(agent, location1)$, *At*$(vehicle, location1)$,
    At$(building, location2)$,
    KnowLocation$(agent, building, location2)$
  effects: $\neg$*At*$(agent, location1)$,
    $\neg$*At*$(vehicle, location1)$,
    At$(agent, location2)$,
    At$(vehicle, location2)$

Drive$(agent, vehicle, building)$
  precond: $location1 \neq location2$,
    Holding$(agent, physobj)$,
    In$(agent, vehicle)$, *At*$(physobj, location1)$,
    At$(agent, location1)$, *At*$(vehicle, location1)$,
    At$(building, location2)$,
    KnowLocation$(agent, building, location2)$
  effects: $\neg$*At*$(physobj, location1)$,
    At$(physobj, location2)$

We allow multiple instances of an operator with a given predicate symbol, which allows us to represent conditional effects of actions. The first *Drive* operator represents that, if an agent drives a vehicle to a building, then the agent and vehicle change location. The second *Drive* operator represents

that, if an agent drives a vehicle to a building and the agent is holding an object, then that object also changes location.

The Willa story domain also includes the following state constraints:

$Hungry(agent) \leftrightarrow \neg Satiated(agent)$

$Bored(agent) \leftrightarrow \neg Entertained(agent)$

$At(object, location1) \wedge At(object, location2) \rightarrow location1 = location2$

$Holding(agent, physobj) \wedge At(agent, location) \rightarrow At(physobj, location)$

$In(agent, vehicle) \wedge At(agent, location) \rightarrow At(vehicle, location)$

The Willa story domain also includes the following active goals:

$Hungry(agent) \rightsquigarrow Satiated(agent)$

$Bored(agent) \rightsquigarrow Entertained(agent)$

Willa1

The Willa1 story is represented as follows:

$\neg HoldsAt(Bored(Willa), 0)$
$HoldsAt(At(Car1, Location1), 0)$
$HoldsAt(At(Restaurant1, Location2), 0)$
$HoldsAt(At(Salad1, Location2), 0)$
$HoldsAt(Hungry(Willa), 0)$
$Happens(PickUp(Willa, MichelinGuide), 0)$
$Happens(GetIn(Willa, Car1), 2)$

The output of the Willa1 run in which only goal-based models were considered is as follows.

```
9 actions
19 fluents
524288 states total
16512 states satisfying state constraints
0
160 models
5484 find_plans invocations
1
16 models
18240 find_plans invocations
2
44 models
4652 find_plans invocations
3
10 models
16496 find_plans invocations
4
26 models
38092 find_plans invocations
5
72 models
65 seconds
---
model 1 weight 1.0:
0
!At(Car1,Location2)
!At(MichelinGuide,Location2)
!At(Restaurant1,Location1)
!At(Salad1,Location1)
!At(Willa,Location2)
!Bored(Willa)
!Holding(Willa,MichelinGuide)
!Holding(Willa,Salad1)
!In(Willa,Car1)
!KnowLocation(Willa,Restaurant1,Location1)
!KnowLocation(Willa,Restaurant1,Location2)
!Satiated(Willa)
At(Car1,Location1)
At(MichelinGuide,Location1)
At(Restaurant1,Location2)
At(Salad1,Location2)
At(Willa,Location1)
Entertained(Willa)
Hungry(Willa)
Happens PickUp(Willa,MichelinGuide) 1.0
1
+Holding(Willa,MichelinGuide)
Happens Read(Willa,MichelinGuide) 1.0
2
+KnowLocation(Willa,Restaurant1,Location2)
Happens GetIn(Willa,Car1) 1.0
3
+In(Willa,Car1)
Happens Drive(Willa,Car1,Restaurant1) 1.0
4
-At(Car1,Location1)
-At(Willa,Location1)
-At(MichelinGuide,Location1)
+At(Car1,Location2)
+At(Willa,Location2)
+At(MichelinGuide,Location2)
Happens Eat(Willa,Salad1) 1.0
5
-At(Salad1,Location2)
-Hungry(Willa)
+Satiated(Willa)
```

At timepoint 0, the truth values of all fluents is shown. At later timepoints, fluents that become false are indicated with a minus sign (-) and fluents that become true are indicated with a plus sign (+).

The only difference between model 2, which is also of weight 1, and model 1 is that $KnowLocation(Willa, Restaurant1, Location1)$ is initially true in model 2 and initially false in model 1.

Model 3 gets a weight of .5 because Willa proceeds more slowly toward her goal by setting down the Michelin guide at timepoint 4:

```
---
model 3 weight 0.5:
0
!At(Car1,Location2)
!At(MichelinGuide,Location2)
!At(Restaurant1,Location1)
!At(Salad1,Location1)
!At(Willa,Location2)
!Bored(Willa)
!Holding(Willa,MichelinGuide)
!Holding(Willa,Salad1)
!In(Willa,Car1)
```

```
!KnowLocation(Willa,Restaurant1,Location1)
!KnowLocation(Willa,Restaurant1,Location2)
!Satiated(Willa)
At(Car1,Location1)
At(MichelinGuide,Location1)
At(Restaurant1,Location2)
At(Salad1,Location2)
At(Willa,Location1)
Entertained(Willa)
Hungry(Willa)
Happens PickUp(Willa,MichelinGuide) 1.0
1
+Holding(Willa,MichelinGuide)
Happens Read(Willa,MichelinGuide) 1.0
2
+KnowLocation(Willa,Restaurant1,Location2)
Happens GetIn(Willa,Car1) 1.0
3
+In(Willa,Car1)
Happens Drive(Willa,Car1,Restaurant1) 1.0
4
-At(Car1,Location1)
-At(Willa,Location1)
-At(MichelinGuide,Location1)
+At(Car1,Location2)
+At(Willa,Location2)
+At(MichelinGuide,Location2)
Happens SetDown(Willa,MichelinGuide) 0.5
5
-Holding(Willa,MichelinGuide)
```

Five other similar models with weight .5 follow. The action $Eat(Willa, Salad1)$ is not performed in any of these models. The models differ on the sequence of actions performed and on the initial truth value of $KnowLocation(Willa, Restaurant1, Location1)$.

Willa2

The Willa2 story is represented as follows:

$\neg HoldsAt(Hungry(Willa), 0)$
$HoldsAt(At(Car1, Location1), 0)$
$HoldsAt(At(Restaurant1, Location2), 0)$
$HoldsAt(At(Salad1, Location2), 0)$
$HoldsAt(Bored(Willa), 0)$
$Happens(PickUp(Willa, MichelinGuide), 0)$

The output of the Willa2 run in which only goal-based models were considered is as follows.

```
9 actions
19 fluents
524288 states total
16512 states satisfying state constraints
0
160 models
5244 find_plans invocations
1
16 models
13112 find_plans invocations
2
54 models
67 seconds
---
model 1 weight 1.0:
0
!At(Car1,Location2)
!At(MichelinGuide,Location2)
!At(Restaurant1,Location1)
!At(Salad1,Location1)
!At(Willa,Location2)
!Entertained(Willa)
!Holding(Willa,MichelinGuide)
!Holding(Willa,Salad1)
!Hungry(Willa)
At(Car1,Location1)
At(MichelinGuide,Location1)
At(Restaurant1,Location2)
At(Salad1,Location2)
At(Willa,Location1)
Bored(Willa)
In(Willa,Car1)
KnowLocation(Willa,Restaurant1,Location1)
KnowLocation(Willa,Restaurant1,Location2)
Satiated(Willa)
Happens PickUp(Willa,MichelinGuide) 1.0
1
+Holding(Willa,MichelinGuide)
Happens Read(Willa,MichelinGuide) 1.0
2
-Bored(Willa)
+Entertained(Willa)
```

Fifteen other similar models with weight 1 follow. Willa reads the Michelin Guide in all of them, but they differ on the truth values of various fluents not constrained by the story.

Related Work

Starting with Charniak's (1972) program, artificial intelligence researchers have written a number of story understanding programs. Ram and Moorman (1999) and Mueller (2002) provide overviews.[1] Schank and Abelson (1977) pointed out the importance of the knowledge structures of scripts, plans, goals, and themes in story understanding. Wilensky (1978) implemented the PAM program for understanding goal-based stories. PAM takes a story text as input, parses the story, makes inferences, answers questions about the story, and retells the story from different points of view. PAM uses a procedural representation of planning knowledge, whereas our approach uses a declarative representation. PAM handles stories involving goal relationships such as subsumption, conflict, competition, and concord, which our approach does not address.

Our work is similar in spirit to prior story understanding work (Schank & Riesbeck 1981; Dyer 1983). The primary difference is that we increase the flexibility of story understanding by incorporating modern action formalisms and reasoning algorithms (Ghallab, Nau, & Traverso 2004; Mueller 2006). In previous papers we have revisited scripts (Mueller 2004b; 2007) and in the present paper we revisit plans and goals.

The interpretation task discussed in this paper is similar to the plan recognition task (Kautz 1991; Carberry 2001),

[1] A list of over 50 story understanding programs is provided at `http://xenia.media.mit.edu/~mueller/storyund/storyres.html`.

which consists of taking observed actions as input and producing recognized plans and goals as output. The present work differs from plan recognition in two significant ways. First, most work on plan recognition assumes that a hierarchical plan library is available. In our approach, the story understanding process uses classical planning to generate plans from scratch given knowledge of the preconditions and effects of actions. This enables recognition of any plan for achieving a goal derivable from the planning knowledge and not just plans stored in the planning library. That is, our approach enables recognition of novel plans. Second, plan recognition is concerned with connecting actions via plans to goals. We are concerned with filling in missing details of all kinds, not just plans and goals. Our approach simultaneously infers goals, missing actions, and missing properties.

Conclusions

We have shown how model finding can be combined with planning to produce models of goal-based stories. By interleaving model finding and planning, we avoid an explosion of models.

Some areas for further work are the following:

1. The implementation could be tested and run on further stories.
2. Correctness proofs for the algorithms could be devised.
3. The approach could be extended to deal with stories in which temporal relations between narrated actions and fluents are specified, instead of stories in which the exact timepoints of narrated actions and fluents are specified.
4. The formalism could be extended to handle multiple characters. A goal-based behavior of a character would be defined as a sequence of actions that achieves an active goal of that character. A goal-based model would be defined as a model in which each action performed by a character is the first action of a goal-based behavior of that character.
5. The approach could be extended to deal with goal relationships such as competition (Wilensky 1983).
6. Even for small story domains and stories, the set of states Σ can be very large. Techniques for lazy evaluation could be devised to allow our approach to scale to larger story domains and stories.

References

Alshawi, H., ed. 1992. *The Core Language Engine*. Cambridge, MA: MIT Press.

Carberry, S. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction* 11(1–2):31–48.

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

Du, D.; Gu, J.; and Pardalos, P. M., eds. 1997. *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Boston, MA: American Mathematical Society.

Dyer, M. G. 1983. *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. Cambridge, MA: MIT Press.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. San Francisco: Morgan Kaufmann.

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.

Kakas, A. C.; Michael, L.; and Miller, R. 2005. Modular-E: an elaboration tolerant approach to the ramification and qualification problems. In McIlraith, S.; Peppas, P.; and Thielscher, M., eds., *Seventh International Symposium on Logical Formalizations of Commonsense Reasoning*.

Kautz, H. A. 1991. A formal theory of plan recognition and its implementation. In Allen, J. F.; Kautz, H. A.; Pelavin, R. N.; and Tenenberg, J. D., eds., *Reasoning about Plans*. San Mateo, CA: Morgan Kaufmann. 69–125.

Kintsch, W. 1998. *Comprehension: A paradigm for cognition*. Cambridge: Cambridge University Press.

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

Mueller, E. T. 2003. Story understanding through multi-representation model construction. In Hirst, G., and Nirenburg, S., eds., *Text Meaning: Proceedings of the HLT-NAACL 2003 Workshop*, 46–53. East Stroudsburg, PA: Association for Computational Linguistics.

Mueller, E. T. 2004a. Event calculus reasoning through satisfiability. *Journal of Logic and Computation* 14(5):703–730.

Mueller, E. T. 2004b. Understanding script-based stories using commonsense reasoning. *Cognitive Systems Research* 5(4):307–340.

Mueller, E. T. 2006. *Commonsense Reasoning*. San Francisco: Morgan Kaufmann.

Mueller, E. T. 2007. Modelling space and time in narratives about restaurants. *Literary and Linguistic Computing* 22(1):67–84.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–105.

Plaisted, D. A., and Greenbaum, S. 1986. A structure-preserving clause form translation. *Journal of Symbolic Computation* 2:293–304.

Ram, A., and Moorman, K., eds. 1999. *Understanding Language Understanding: Computational Models of Reading*. Cambridge, MA: MIT Press.

Schank, R. C., and Abelson, R. P. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum.

Schank, R. C., and Riesbeck, C. K., eds. 1981. *Inside Computer Understanding: Five Programs plus Miniatures*. Hillsdale, NJ: Lawrence Erlbaum.

Wilensky, R. 1978. Understanding goal-based stories. Technical Report YALE/DCS/tr140, New Haven, CT: Computer Science Department, Yale University.

Wilensky, R. 1983. *Planning and Understanding: A Computational Approach to Human Reasoning*. Reading, MA: Addison-Wesley.