Experiences with Planning for Natural Language Generation

Alexander Koller\(^1,2\) and Ron Petrick\(^1\)

\(^1\)School of Informatics, University of Edinburgh
\(^2\)Department of Computational Linguistics, Saarland University

Scheduling and Planning Applications woRKshop (SPARK-08), Sydney, Australia

15 September 2008
Motivation

- **Natural Language Generation** (NLG) is a major subfield of natural language processing, concerned with computing natural language sentences or texts that convey a piece of information to the user.

- Can be viewed as a problem involving actions, beliefs, and goals.
  
  A speaker tries to change the mental state of the hearer by applying actions that correspond to the utterance of words or sentences.

⇒ Problems in NLG result in computationally challenging domains (often more challenging than in other areas of natural language processing).

⇒ Obvious parallels to **planning**.
NLG and planning

• The link between NLG and planning has a long tradition
  – Appelt (1985)
  – Young & Moore (1994)
  – …

• More recently, there has been a resurgence of interest in applying modern planning techniques to NLG
  – Steedman & Petrick (2007)
  – Benotti (2008)
  – …

• Recent approaches use planning as a modelling tool, with the hope of improved efficiency over traditional NLG methods
This work

1. Describes two recent planning domains that arise from NLG
   - Sentence generation
   - Generating Instructions in Virtual Environments (GIVE)

2. Shares our experiences in these domains using off-the-shelf planners that perform well in traditional planning domains
   - FF (Hoffmann & Nebel, 2001)
   - SGPLAN (Hsu et al., 2006)
   - Java implementation of GraphPlan (Blum & Furst, 1997)

⇒ Empirical results using off-the-shelf planners are mixed
⇒ Domains are freely available; we believe these problems offer interesting challenges for the planning community
I. Sentence generation

• **Sentence generation** is the problem of computing, from a grammar and semantic representation, a sentence that expresses this meaning

  1. **Sentence planning**: the semantic representation is enriched with more information, e.g., referring expressions
  2. **Surface realization**: the enriched representation is translated into a natural language sentence using the grammar

• The process of determining referring expressions often interacts with the realization step; beneficial to perform both steps together

• E.g., SPUD system (*Stone et al.*, 2003)
Example: “The white rabbit sleeps.”

- Tree Adjoining Grammar (Joshi & Schabes, 1997)

Elementary trees contribute **semantic content**. Trees are instantiated by substituting individuals for **semantic roles**.

- Knowledge base
  - $a$ and $b$ are rabbits
  - $a$ is white
  - $b$ is brown
  - $e$ is an event in which $a$ sleeps
Example: “The white rabbit sleeps.” (2)

A derivation to express \{sleep(e, a)\}:

+ Knowledge base

A. Koller & R. Petrick / Scheduling and Planning Applications woRKshop (SPARK-08) / 2008-09-15
Sentence generation as planning

• The sentence generation problem can be translated into a planning problem (Koller & Stone, 2007)

• Actions add elementary trees to a derivation

```lisp
(:action add-sleeps
  :parameters (?u - node
               ?xself - individual
               ?xsubj - individual)
  :precondition (and (subst S ?u)
                     (referent ?u ?xself)
                     (sleep ?xself ?xsubj))
  :effect (and (not (subst S ?u))
             (expressed sleep ?xself ?xsubj)
             (subst NP (subj ?u))
             (referent (subj ?u) ?xsubj)
             (forall (?y - individual)
              (when (not (= ?y ?xself))
               (distractor (subj ?u) ?y))))
```

• A plan must satisfy the syntactic and semantic linguistic requirements

```
[add-sleeps(root, a), add-rabbit(subj(root), a), add-white(subj(root), a)]
```
II. GIVE Challenge

• “Generating Instructions in Virtual Environments” (Koller et al., 2007)

• A challenge for the NLG community

  Build an NLG system capable of producing natural language instructions to guide a human user in performing some task in a virtual environment

• A theory-neutral task that tests all components of an NLG system

• Virtual 3D client; evaluation possible over the Internet

• First GIVE evaluation will take place in late 2008
Virtual 3D GIVE client
Planning in instruction giving

• A GIVE problem is very similar to a Gridworld problem
  – Discretised tiles of equal size
  – User can turn in $90^\circ$ steps in either direction and move forward
  – Additional need to press buttons in the right order, reason about more objects, and navigate more complicated room shapes

• PDDL description of available actions

```pddl
(:action move
 :parameters (?from - position
                ?to - position
                ?ori - orientation)
 :precondition (and (player-pos ?from)
                (adjacent ?from ?to ?ori)
                (not-blocked ?from ?to)
                (not-alarmed ?to))
 :effect (and (not (player-pos ?from))
          (player-pos ?to)))
```
Example GIVE map
Challenges for NLG

• Plan summarisation
  “Turn left; turn left; walk forward”
  ⇒ “Turn around and walk through the door”

• Plan elaboration
  “Press the green button on the wall to your right”
  ⇒ “Walk to the centre of the room; turn right; now press the green button in front of you”

• ...
Challenges for planning

• Strict run-time requirements
  – Planning must happen in (near) real time
  – User can interact with the world while the system is deliberating

• System must monitor a user’s actions and compare them against the generated instruction set
  – Mental state of the user is not known
  – User’s actions may not match the generated instructions exactly but still meet the intended goal ⇒ identify “equivalent” plans
  – User can communicate intentions through action and inaction

• Plans are often non-trivial, e.g., 108 steps in sample domain

  [  turn-left(north,west),
    move(pos_5_2,pos_4_2,west),
    manipulate-b1-off-on(pos_5_2), . . . ]
Experiments and results
Experiment 1: Sentence generation

• Compute a plan representing a sentence of the form
  
  “Mary likes the Adj$_1$ . . . Adj$_n$ rabbit.”

• Knowledge base for each problem instance
  
  – $m$ rabbits
  – $n$ properties are required to distinguish the target reference from other rabbits, where $n \leq m$

• The $n$ properties are realised as $n$ different adjectives, in any order
Experiment 1: Results

![Graph](image)

- SGPLAN total
- FF total
- FF grounding
- SGPLAN parsing
- GraphPlan (Java)

Runtime (ms) vs. \((m,n)\)

A. Koller & R. Petrick / Scheduling and Planning Applications wORKshop (SPARK-08) / 2008-09-15
Experiment 2: Minimal GIVE worlds

- Simplified GIVE world consisting of a $2n$ by $h$ grid and buttons in alternating positions
- Player starts in position (1,1)
- All buttons must be pressed to successfully complete the game
Experiment 2: Results

A. Koller & R. Petrick / Scheduling and Planning Applications woRKshop (SPARK-08) / 2008-09-15
Experiment 3: GIVE + extra positions

We vary the structure of the GIVE world by adding another $w$ by $h$ empty “junk” positions to the right of the minimal world.
Experiment 3: Results

Runtime (sec) vs. $w$ ($h = 20, n = 5$)
Observations

• Modern planners are fairly good at the sentence generation task
  – FF generates non-trivial 14-word sentences in about 2 seconds
  – Special purpose NLG algorithms can generate referring expressions in milliseconds (Areces et al., 2008)
  – Larger, but realistically-sized, instances are still problematic for the planners we tested

• GIVE domain examples are more problematic
  – Focus is on generating relatively short plans in large universes
  – Planning times over a couple seconds negatively affect the overall response time of the NLG system
  – Restricting bottleneck seems to be the initial grounding stage
Conclusions

• NLG problems offer a real-world testbed for planning under uncertainty, resource management, execution monitoring, plan presentation, reasoning about knowledge and sensing, . . .

• Mixed results with off-the-shelf planners in two NLG planning domains

• FF and SGPLAN are good at controlling search in our domains but the total planning time is dominated by grounding time; large problem instances remain a challenge

• It’s not all bad news for planning! The GIVE challenge wouldn’t have been possible 10 years ago...

• Future work: experiments with other planners; other grammars . . .

• Domain descriptions and code available online

http://code.google.com/p/crisp-nlg/
http://www.give-challenge.org/
References


A. Koller & R. Petrick / Scheduling and Planning Applications woRKshop (SPARK-08) / 2008-09-15


