Enhancing a Dialogue System through Dynamic Planning

Luciana Benotti
Luciana.Benotti@stud-inf.unibz.it

European Masters Program in Computational Logic
Free University of Bozen - Bolzano, Italy
Universidad Politécnica de Madrid, Spain

EMCL Students Workshop - December 2006
Outline

1. Dialogue systems
   - What is a dialogue system?
   - FrOz: A state-of-the-art dialogue system

2. Dynamic planning in a dialogue system
   - Why does FrOz need planning?
   - Blackbox: An off-the-shelf planner
   - Dynamic planning
   - Evaluation

3. Final reflections and future work
Outline

1. Dialogue systems
   - What is a dialogue system?
   - FrOz: A state-of-the-art dialogue system

2. Dynamic planning in a dialogue system
   - Why does FrOz need planning?
   - Blackbox: An off-the-shelf planner
   - Dynamic planning
   - Evaluation

3. Final reflections and future work
What is a dialogue system?

A dialogue system is a piece of software that interprets natural language inputs, answers back in natural language, and mimics certain human conversational capacities such as:

- syntactic and semantic interpretation of the input,
- intention recognition,
- mixed-initiative management,
- answer construction,
- etc.
FrOz: A computer game

FrOz is a dialogue system that implements a text adventure:

1. FrOz starts by describing the game world
   
   You are in a brown couch.
   There is a red apple on the couch.

2. The player types, in natural language, instructions for the protagonist character to follow

   take the apple on the couch.

3. FrOz executes the action and responds to the player by describing the result

   You have the red apple.
FrOz architecture

FrOz offers a **generic framework** for text adventures which can be instantiated with different scenarios:
FrOz: Playing the game

- All the game output is generated on the fly, there is no canned text.
- The game resolves referring expressions using querying the player A-Box.
- The player executes actions in order to modify the game world.
- The game is able to handle conjunctive actions.
Outline

1. Dialogue systems
   - What is a dialogue system?
   - FrOz: A state-of-the-art dialogue system

2. Dynamic planning in a dialogue system
   - Why does FrOz need planning?
   - Blackbox: An off-the-shelf planner
   - Dynamic planning
   - Evaluation

3. Final reflections and future work
Our goal

1. The player tries to execute an action which she presupposes executable in the current world state: eat the apple.

2. The game fails with You can’t do this! You do not have the apple because not all the action preconditions hold in the game world.

3. The player needs to make explicit the implicit actions: take the apple and eat it.

We want the game to be able to find out on its own the sequence of implicit actions, and to execute it autonomously (when suitable).
Characteristics for a solution

The relevant information we have is:

- A description of the commands that can be executed over the world: the actions.
- The representation of the world state where the action input by the player fails: the initial state.
- A description of the preconditions of the action intended by the player: the goal.

We need a sequence of actions that achieves the goal when executed on the initial state.
Characteristics for a solution

The relevant information we have is:

- A description of the commands that can be executed over the world: the **actions**.
- The representation of the world state where the action input by the player fails: the **initial state**.
- A description of the preconditions of the action intended by the player: the **goal**.

We need a sequence of actions that achieves the goal when executed on the initial state.

Clearly, the solution will involve **planning**.
### The actions in the game

FrOz represents the actions that can be executed over the game world in a STRIPS-based format.

<table>
<thead>
<tr>
<th>take (patient: X)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>precond</strong></td>
</tr>
<tr>
<td>instance(X accessible),</td>
</tr>
<tr>
<td>instance(X takeable),</td>
</tr>
<tr>
<td>not(instance(X inventory-object))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>world effects add:</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance(X inventory-object)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>world effects delete:</th>
</tr>
</thead>
<tbody>
<tr>
<td>related(X</td>
</tr>
<tr>
<td>individual-filler(X has-location) has-location)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>player effects add:</th>
</tr>
</thead>
<tbody>
<tr>
<td>instance(X inventory-object)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>player effects delete:</th>
</tr>
</thead>
<tbody>
<tr>
<td>related(X</td>
</tr>
<tr>
<td>individual-filler(X has-location) has-location)</td>
</tr>
</tbody>
</table>
The world effects and the player effects can be different.

<table>
<thead>
<tr>
<th>unlock(p: X i: Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>precond</strong></td>
</tr>
<tr>
<td>instance(X locked)</td>
</tr>
<tr>
<td>instance(Y key)</td>
</tr>
<tr>
<td>instance(Y i-o)</td>
</tr>
<tr>
<td>related(Y X fits-in)</td>
</tr>
<tr>
<td><strong>world effects</strong></td>
</tr>
<tr>
<td>add:</td>
</tr>
<tr>
<td>instance(X unlocked)</td>
</tr>
<tr>
<td><strong>player effects</strong></td>
</tr>
<tr>
<td>add:</td>
</tr>
<tr>
<td>instance(X unlocked)</td>
</tr>
<tr>
<td>related(Y X fits-in)</td>
</tr>
<tr>
<td>delete:</td>
</tr>
<tr>
<td>instance(X locked)</td>
</tr>
</tbody>
</table>
Blackbox is a planner developed in the University of Washington that:

- Translates the planning problem into a propositional formula.
- Uses a satisfiability solver to find a model for the formula. The model is then translated into a plan.
- I.e., Blackbox implements the planning paradigm known as planning as satisfiability.

Crucially, Blackbox:

- finds optimal plans
- and does it extremely fast!
Blackbox: PDDL specification language

Blackbox input language:

- Is a subset of the Planning Domain Definition Language.
- Specifies actions in a basic STRIPS-style format.

A sample action is:

```plaintext
:action take
 :parameters (?x - takeable ?y - top)
 :precondition
   (accessible ?x)
   (no-inventory-object ?x)
   (has-location ?x ?y)
 :effect
   (inventory-object ?x)
   (not(has-location ?x ?y))
```
From FrOz actions into PDDL actions

Each action in the text adventure:

<table>
<thead>
<tr>
<th>take(patient:X)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>precond</strong> instance(X accessible), instance(X takeable), not(instance(X inventory-object))</td>
</tr>
<tr>
<td><strong>world</strong> add: instance(X inventory-object)</td>
</tr>
<tr>
<td><strong>effects</strong> delete: related(X individual-filler(X has-location) has-location)</td>
</tr>
</tbody>
</table>

⇓ must be specified in PDDL.

:action take
  :parameters (?x - takeable ?y - top)
  :precondition (accessible ?x) (has-location ?x ?y)
    (no-inventory-object ?x)
  :effect (inventory-object ?x)
    (not(has-location ?x ?y))

These actions constitute the **planning domain**.
The **Dynamic Planning** module main tasks are:

- **Codify** the planning problem (initial state and goal).
- **Invoke** Blackbox with a domain and a problem.
- **De-codify** the plan and insert it into the game cycle.
FrOzA: A simple example

1. The planning problem includes:
   - The world state: The facts believed by the player.
   - The goal: The failed action is executable.

2. The plan found is:
   1. take the apple

3. The sequence of actions executed is:
   1. take the apple
   2. eat the apple

You are in a brown couch.
There is a brown frog, a green apple and a green frog in the couch.
The couch is in a drawing-room.
There is a chest and a brown table in the drawing-room.
The drawing-room has a exit leading south.
> eat the apple
You have the apple.
You eat the apple.
The player can explore the world with the action `look`.

FrOzA will execute on its own only those actions that the player knows how to perform.

Once the player tries to open the chest with the key that is on the table, FrOzA will:
1. take the key
2. unlock the chest with it

When the action succeeds, the player learns that the key fits into the chest.
Later in the game, the player locks the wooden chest.

The player tries to open the chest (which is locked) a second time. **This time FrOzA will unlock the chest on its own:**

1. unlock the chest with the key
2. open it
Outline

1. Dialogue systems
   - What is a dialogue system?
   - FrOz: A state-of-the-art dialogue system

2. Dynamic planning in a dialogue system
   - Why does FrOz need planning?
   - Blackbox: An off-the-shelf planner
   - Dynamic planning
   - Evaluation

3. Final reflections and future work
FrOzA achieves our goal: it finds autonomously a suitable sequence of actions and executes it.

The interaction with FrOzA is more effective and robust.

Interestingly, planning capabilities become more powerful as the player’s knowledge increases.

We showed that a state-of-the-art reasoning tool can be used to handle a dialogue phenomenon.

To do:
- Codify more complex and realistic scenarios in order to perform user evaluation.
- Modify the generation component such that its output is more natural when planning is used. For example:
  
  You eat the apple [after taking it].