Designer codes dialogue as sequence of prompts and responses.

- Pretty tedious especially for adding in common functionality to each state such as "cancel" or "goback" and confirmations.

Overview

Form Filling Dialogue

- Confirmations
- Shortcuts

Form Filling Dialogue

Overview
Form-Filling Dialogues with CSLU Toolkit

- Single state with self-loop and complex Tcl code for form
- Can change recognizer grammar with Tcl code
- Use package command to bring in TTS functionality
- Omit RAD level, and just make Tcl program
- All domain information stored in data structures
- Drives dialogue based on forms
- Tcl program
- Recognizer, wave handling
- Very similar to a frame
- Collection of items called form
- Might not be needed
- Interpretation Grammar
- Recognition Grammar
- Prompt where do you want to go to
- Ask destination
- Ask time
- Ask origin
- Ask destination
- Have data structure that drives the dialogue flow
Controlling what slot to work on

Work on the first slot that does not have a value

Tcl Code

```tcl
set formslots {dest origin}
set prompt(dest) "Where do you want to fly to"
set grammar(dest) "portland | newyork | ..."
set prompt(origin) "Where do you want to fly from"
set grammar(origin) "portland | newyork | ..."

proc Converse {} {
    while {1} {
        DoPrompt
        GetResponse
    }
}
```
• Allow user to say 'repeat' so that the system will repeat what it just said.

proc GetResponse {} {
    set ::value($::doslot) $words
    # run Speech Recognizer
    # set Speech Recognizer to user grammar for slot $::doslot
}

Understanding the Response
Explicit Confirmations

- Form can indicate which items should have immediate
- So is that a slice of pizza with shopping
- Form could specify how to build a confirmation
- Could delay confirmation query till end of form
- If no repeat original prompt
- Load recognition grammar of yes and no
- Is “Did you say”...
- Could have Tel code ask for a confirmation after each query

•

Overview

Form Filling Dialogue

- Shortcuts

Explicit Confirmations

•

OVERVIEW
Overview

Form Filling Dialogue

• Form Filling Dialogue

Confirmations

• Confirmations

⇒ Shortcuts

Implicit Confirmations

• At each step, could say previous item and prompt for next
  
  - Or designer can specify implicit confirmation prompt

  Flying from San Francisco. Where do you want to fly to?

  • Add "no" to grammar of next prompt

  - Or designer can specify implicit confirmation prompt

  So San Francisco. Where do you want to fly to?

  • At each step, could say previous item and prompt for next

Will else could we do?

• Will else could we do?

- Will this work? Will users know to do this naturally?
Concatenating Grammars

Actually, need something more complicated that allows any order of extra information.

```tcl
set grammar ::explicitgrammar($::doslot) foreach slot $::form {
   if {![info exists ::value($slot)] && $slot != $::doslot} {
      append grammar " " $::implicitgrammar($slot)
   }
}
```

Allowing Multiple Items in Response

- When system asks question, user might give extra information
  Where do you want to go to? Denver from Chicago
  - Note that extra information had contextual clues around it
  - Where do you want to go? Denver leaving at 5 pm Denver from Chicago

- Have two forms of grammars for each item
  - ```tcl
   set grammar(origin) "portland | newyork | ..."
   ```
  - ```tcl
   set grammar(dest) "to (portland | newyork | ...)"
   ```

- Build current grammar by concatenating item grammars

- When item is not explicitly prompted for: e.g. Denver
  ```tcl
  set grammar(Denver) "origin | destination | ..."
  ```

- When item is explicitly prompted for: e.g. Denver from Chicago
  ```tcl
  set grammar(origin) "portland | newyork | ..."
  ```
  ```tcl
  set grammar(dest) "to (portland | newyork | ...)"
  ```
Then start prompting for missing items.

From first response, fill in as much information as possible.

- Make grammar by concatenating subgrammars (explicit versions)
- Designer specifies the initial prompt
- When flight would you like?

Rather than ask for a specific parameter, ask general question.

---

**Initial Prompt**

---

**Filling in the values**

- Other alternative is to use a separate interpretation grammar with parallel.
- Result from recognizer will then have appropriate slot values embedded.
- Explicit grammar for origin:
  - Ensure even numbered tokens have the slot value.
  - Ensure odd numbered tokens have the slot name.
- Denver from Chicago
- Explicitly given values often have contextual classes.
- How do we get values for each parameter?