Dialog Management

Ling575
Spoken Dialog Systems
April 19, 2017

Roadmap

- Dialog Management

  - Basic:
    - Finite State Models
    - Frame-based Models

  - Advanced:
    - Information State Models
    - Statistical Dialog Models

Dialogue Manager

- Holds system together: Governs interaction style

Dialogue Manager

- Holds system together: Governs interaction style
  - Takes input from ASR/NLU

Dialogue Manager

- Holds system together: Governs interaction style
  - Takes input from ASR/NLU

  - Maintains dialog state, history
    - Incremental frame construction
    - Reference, ellipsis resolution
    - Determines what system does next

Dialogue Manager

- Holds system together: Governs interaction style
  - Takes input from ASR/NLU
  - Maintains dialog state, history
    - Incremental frame construction
    - Reference, ellipsis resolution
    - Determines what system does next
  - Interfaces with task manager/backend app

Dialogue Manager

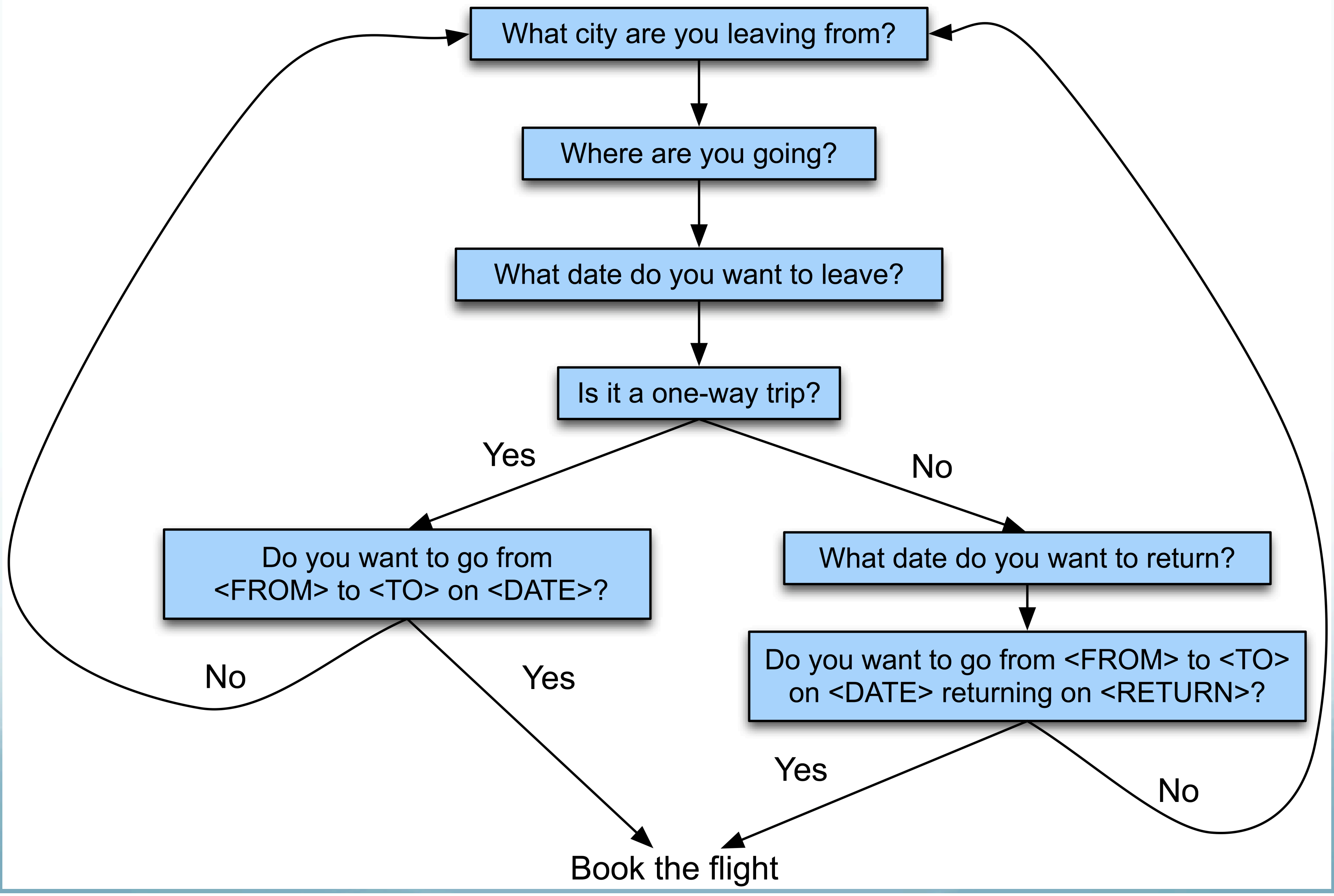
- Holds system together: Governs interaction style
  - Takes input from ASR/NLU

  - Maintains dialog state, history
    - Incremental frame construction
    - Reference, ellipsis resolution
    - Determines what system does next

  - Interfaces with task manager/backend app

  - Formulates basic response, passes to NLG,TTS

Finite-State Management


What city are you leaving from?
Where are you going?
What date do you want to leave?
Is it a one-way trip?
Yes
No
Do you want to go from <FROM> to <TO> on <DATE>?
What date do you want to return?
Do you want to go from <FROM> to <TO> on <DATE> returning on <RETURN>?
No
Yes
Yes
No
Book the flight

Finite-State Dialogue Management

- Simplest type of dialogue management
  - States:
    - Questions system asks user
  - Arcs:
    - User responses

Finite-State Dialogue Management

- Simplest type of dialogue management
  - States:
    - Questions system asks user
  - Arcs:
    - User responses
- System controls interactions:
  - Interprets all input based on current state
  - Assumes any user input is response to last question

Finite-State Dialogue Management

- Initiative:
  - Control of the interaction
- Who’s in control here?

Finite-State Dialogue Management

- Initiative:
  - Control of the interaction
- Who’s in control here?
  - System!
    - “system initiative”/”single initiative”
  - Natural?

Finite-State Dialogue Management

- Initiative:
  - Control of the interaction
- Who’s in control here?
  - System!
    - “system initiative”/”single initiative”
  - Natural? No!
    - Human conversation goes back and forth
- Deploy targeted vocabulary / grammar for state
  - Add ‘universals’ – accessible anywhere in dialog
    - ‘Help’, ‘Start over’

Pros and Cons

- Advantages

Pros and Cons

- Advantages
  - Straightforward to encode
  - Clear mapping of interaction to model
  - Well-suited to simple information access
  - System initiative
- Disadvantages

Pros and Cons

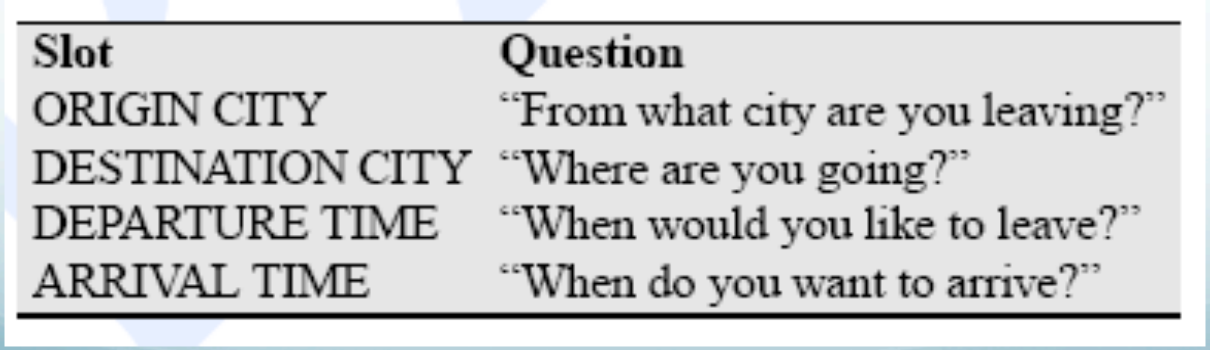
- Advantages
  - Straightforward to encode
  - Clear mapping of interaction to model
  - Well-suited to simple information access
  - System initiative
- Disadvantages
  - Limited flexibility of interaction
    - Constrained input – single item
    - Fully system controlled
    - Restrictive dialogue structure, order
  - Ill-suited to complex problem-solving

Frame-based Dialogue Management

- Essentially form-filling
  - User can include any/all of the pieces of form
  - System must determine which entered, remain
  - Rules determine next action, question, information presentation

Frame-based Dialogue Management

- Essentially form-filling
  - User can include any/all of the pieces of form
  - System must determine which entered, remain
  - Rules determine next action, question, information presentation



Slot	Question
ORIGIN CITY	"From what city are you leaving?"
DESTINATION CITY	"Where are you going?"
DEPARTURE TIME	"When would you like to leave?"
ARRIVAL TIME	"When do you want to arrive?"

Frames and Initiative

- Mixed initiative systems:
  - A) User/System can shift control arbitrarily, any time
    - Difficult to achieve
  - B) Mix of control based on prompt type

Frames and Initiative

- Mixed initiative systems:
  - A) User/System can shift control arbitrarily, any time
    - Difficult to achieve
  - B) Mix of control based on prompt type
- Prompts:
  - Open prompt: ‘How may I help you?’

Frames and Initiative

- Mixed initiative systems:
  - A) User/System can shift control arbitrarily, any time
    - Difficult to achieve
  - B) Mix of control based on prompt type
- Prompts:
  - Open prompt: ‘How may I help you?’
    - Open-ended, user can respond in any way
  - Directive prompt: ‘Say yes to accept call, or no o.w.’

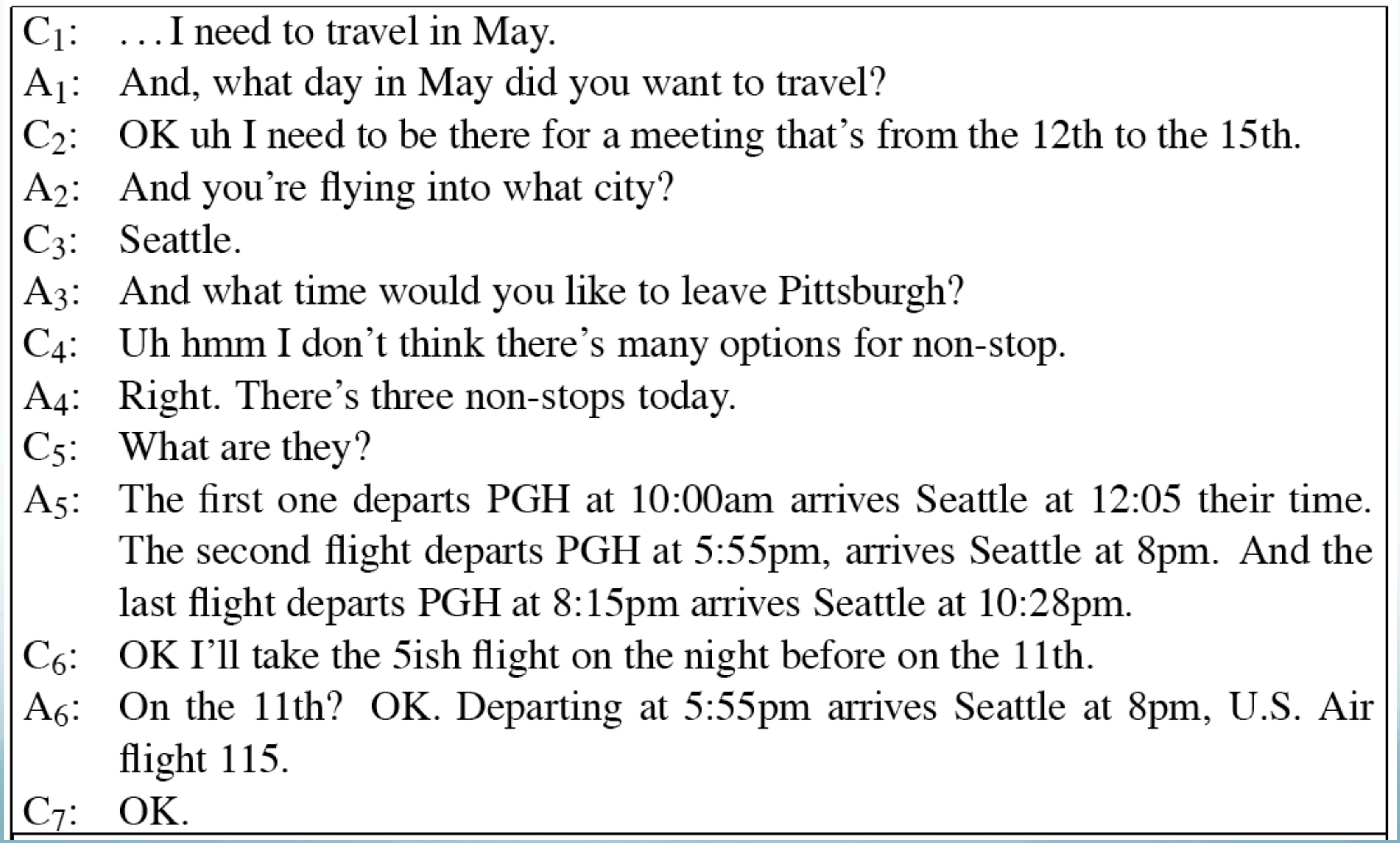
Frames and Initiative

- Mixed initiative systems:
  - A) User/System can shift control arbitrarily, any time
    - Difficult to achieve
  - B) Mix of control based on prompt type
- Prompts:
  - Open prompt: ‘How may I help you?’
    - Open-ended, user can respond in any way
  - Directive prompt: ‘Say yes to accept call, or no o.w.’
    - Stipulates user response type, form

Dialogue Management: Confirmation

- Miscommunication common in SDS
  - “Error spirals” of sequential errors
    - Highly problematic
  - Recognition, recovery crucial
- Confirmation strategies can detect, mitigate
  - Explicit confirmation:

Dialog Example



C_1: …I need to travel in May.

A_1: And, what day in May did you want to travel?

C_2: OK uh I need to be there for a meeting that's from the 12th to the 15th.

A_2: And you're flying into what city?

C_3: Seattle.

A_3: And what time would you like to leave Pittsburgh?

C_4: Uh hmm I don't think there's many options for non-stop.

A_4: Right. There's three non-stops today.

C_5: What are they?

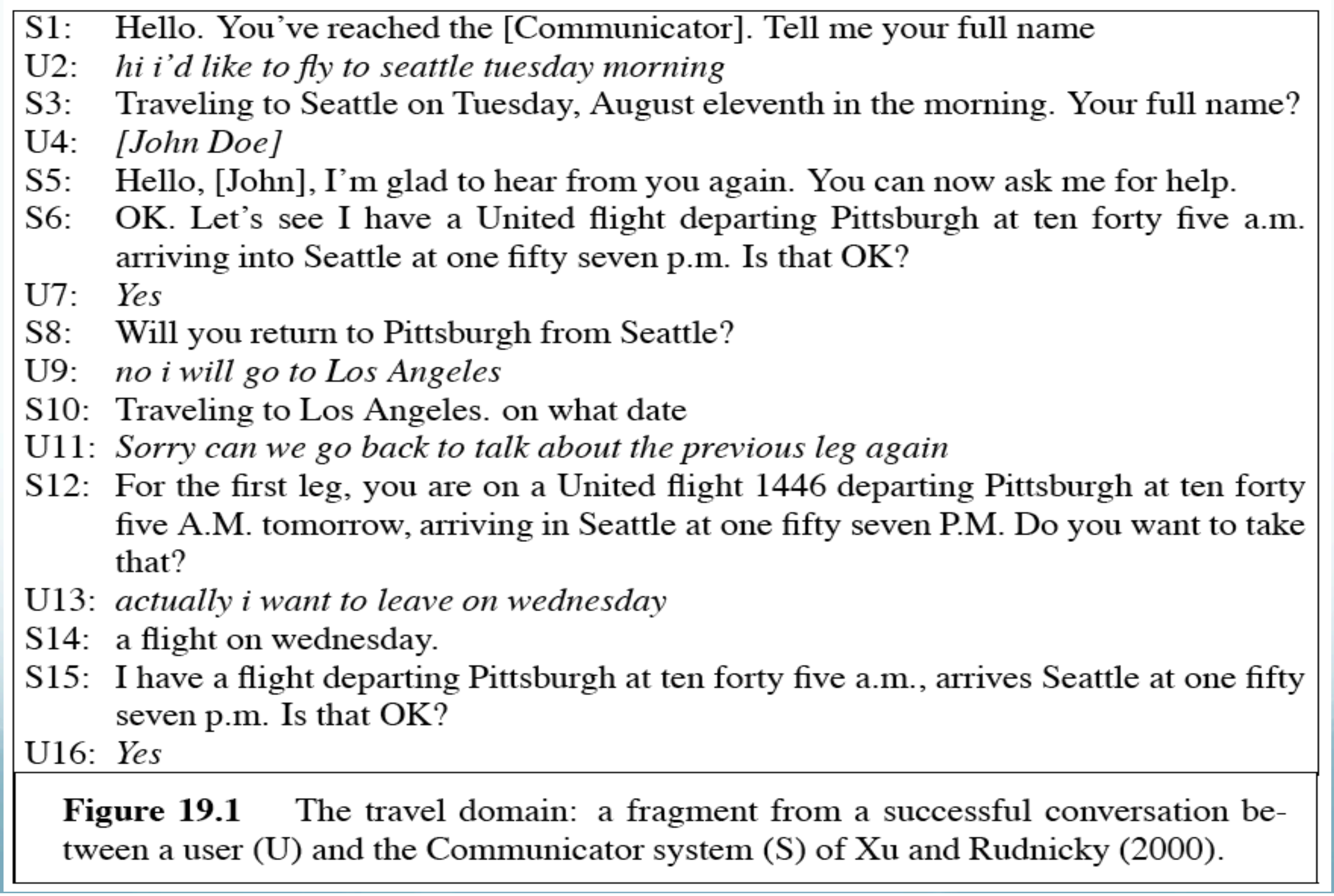
A_5: The first one departs PGH at 10:00am arrives Seattle at 12:05 their time. The second flight departs PGH at 5:55pm, arrives Seattle at 8pm. And the last flight departs PGH at 8:15pm arrives Seattle at 10:28pm.

C_6: OK I'll take the 5ish flight on the night before on the 11th.

A_6: On the 11th? OK. Departing at 5:55pm arrives Seattle at 8pm, U.S. Air flight 115.

C_7: OK.

Travel Planning



S1: Hello. You've reached the [Communicator]. Tell me your full name

U2: *hi i'd like to fly to seattle tuesday morning*

S3: Traveling to Seattle on Tuesday, August eleventh in the morning. Your full name?

U4: *[John Doe]*

S5: Hello, [John], I'm glad to hear from you again. You can now ask me for help.

S6: OK. Let's see I have a United flight departing Pittsburgh at ten forty five a.m. arriving into Seattle at one fifty seven p.m. Is that OK?

U7: *Yes*

S8: Will you return to Pittsburgh from Seattle?

U9: *no i will go to Los Angeles*

S10: Traveling to Los Angeles. on what date

U11: *Sorry can we go back to talk about the previous leg again*

S12: For the first leg, you are on a United flight 1446 departing Pittsburgh at ten forty five A.M. tomorrow, arriving in Seattle at one fifty seven P.M. Do you want to take that?

U13: *actually i want to leave on wednesday*

S14: a flight on wednesday.

S15: I have a flight departing Pittsburgh at ten forty five a.m., arrives Seattle at one fifty seven p.m. Is that OK?

U16: *Yes*

Figure 19.1 The travel domain: a fragment from a successful conversation between a user (U) and the Communicator system (S) of Xu and Rudnicky (2000).

Dialogue Management: Confirmation

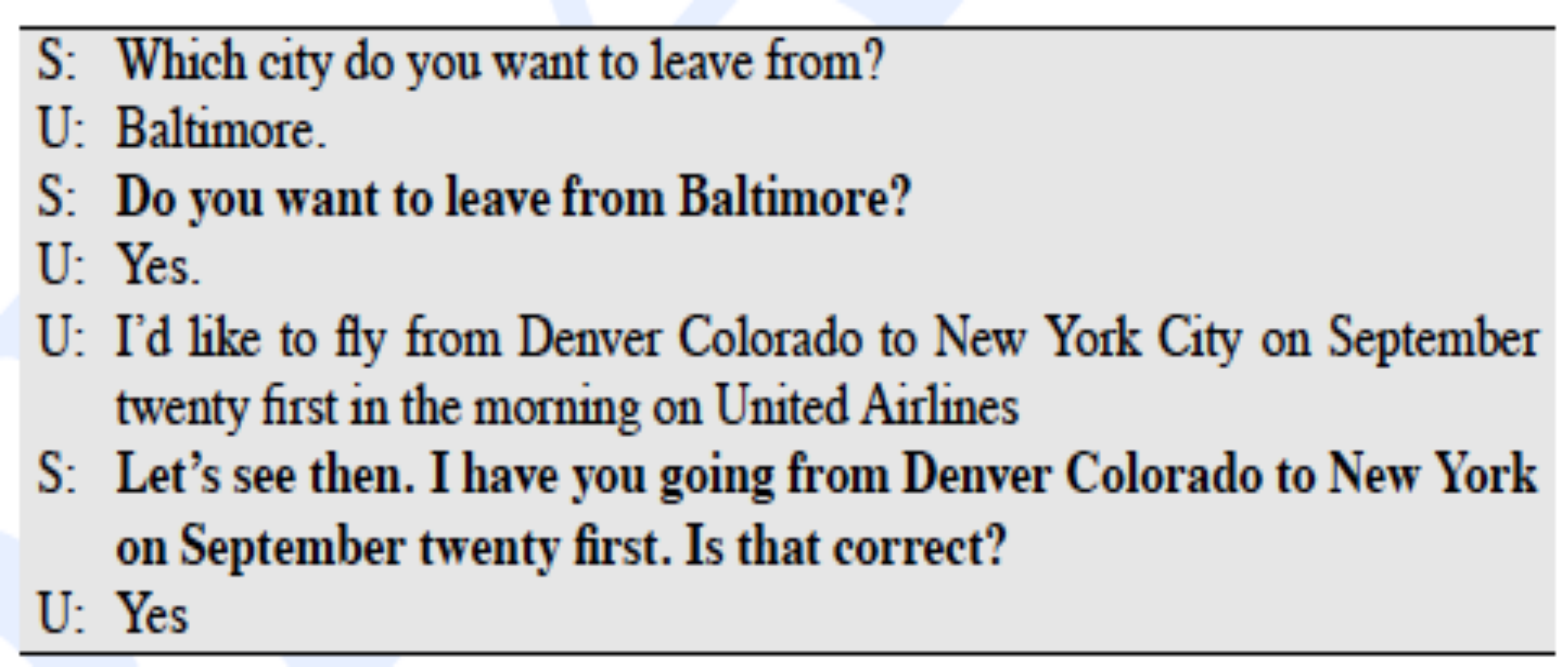
- Miscommunication common in SDS
  - "Error spirals" of sequential errors
    - Highly problematic
  - Recognition, recovery crucial
- Confirmation strategies can detect, mitigate
  - Explicit confirmation:
    - Ask for verification of each input
  - Implicit confirmation:

Dialogue Management: Confirmation

- Miscommunication common in SDS
  - "Error spirals" of sequential errors
    - Highly problematic
  - Recognition, recovery crucial
- Confirmation strategies can detect, mitigate
  - Explicit confirmation:
    - Ask for verification of each input
  - Implicit confirmation:
    - Include input information in subsequent prompt

Confirmation Strategies

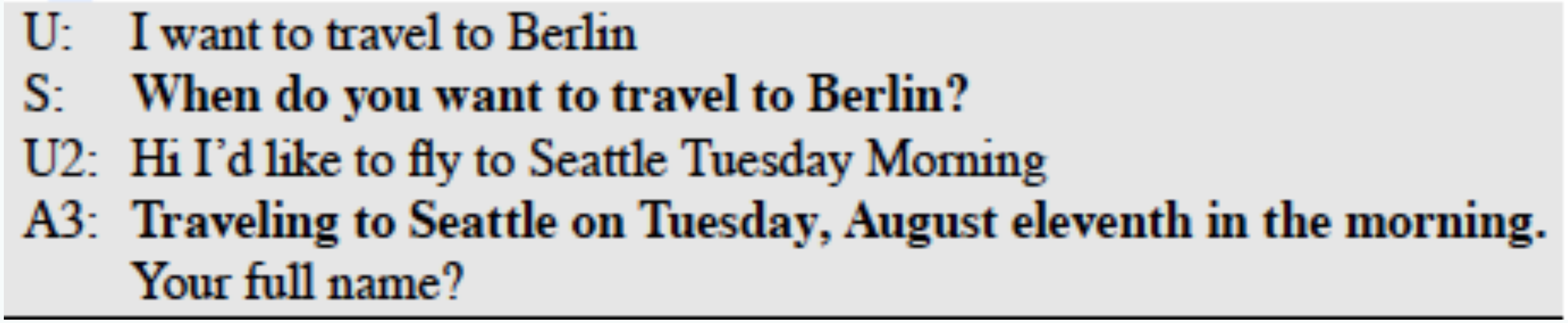
- Explicit:



S: Which city do you want to leave from?
U: Baltimore.
S: **Do you want to leave from Baltimore?**
U: Yes.
U: I'd like to fly from Denver Colorado to New York City on September twenty first in the morning on United Airlines
S: **Let's see then. I have you going from Denver Colorado to New York on September twenty first. Is that correct?**
U: Yes

Confirmation Strategy

- Implicit:



U: I want to travel to Berlin

S: **When do you want to travel to Berlin?**

U2: Hi I'd like to fly to Seattle Tuesday Morning

A3: **Traveling to Seattle on Tuesday, August eleventh in the morning. Your full name?**

Pros and Cons

- Grounding of user input
  - Weakest grounding
    - I.e. continued att'n, next relevant contibution

Pros and Cons

- Grounding of user input
  - Weakest grounding insufficient
    - I.e. continued att'n, next relevant contibution
  - Explicit:

Pros and Cons

- Grounding of user input
  - Weakest grounding insufficient
    - I.e. continued att’n, next relevant contibution
  - Explicit: highest: repetition
  - Implicit:

Pros and Cons

- Grounding of user input
  - Weakest grounding insufficient
    - I.e. continued att'n, next relevant contibution
  - Explicit: highest: repetition
  - Implicit: demonstration, display
- Explicit:

Pros and Cons

- Grounding of user input
  - Weakest grounding insufficient
    - I.e. continued att'n, next relevant contibution
  - Explicit: highest: repetition
  - Implicit: demonstration, display
- Explicit:
  - Pro: easier to correct; Con: verbose, awkward, non-human
- Implicit:

Pros and Cons

- Grounding of user input
  - Weakest grounding insufficient
    - I.e. continued att'n, next relevant contibution
  - Explicit: highest: repetition
  - Implicit: demonstration, display
- Explicit:
  - Pro: easier to correct; Con: verbose, awkward, non-human
- Implicit:
  - Pro: more natural, efficient; Con: less easy to correct

Frame-based Systems: Pros and Cons

- Advantages
  - Relatively flexible input – multiple inputs, orders
  - Well-suited to complex information access (air)
  - Supports different types of initiative

- Disadvantages
  - Ill-suited to more complex problem-solving
    - Form-filling applications

Richer Dialog Management

- Alternative Dialog Management approaches
  - More flexible interaction, motivated by human-human

Richer Dialog Management

- Alternative Dialog Management approaches
  - More flexible interaction, motivated by human-human

  - Information State
    - General interpretation of speech in terms of dialog acts
      - Similar to “speech acts”, e.g. statement, wh-q, yn-q, check,..
    - Model of knowledge, belief state of current dialog

Richer Dialog Management

- Alternative Dialog Management approaches
  - More flexible interaction, motivated by human-human

  - Information State
    - General interpretation of speech in terms of dialog acts
      - Similar to "speech acts", e.g. statement, wh-q, yn-q, check,..
    - Model of knowledge, belief state of current dialog

  - Statistical dialog management
    - Builds on reinforcement learning approaches (planning)
    - Aims to automatically learns best sequence of actions
    - Models uncertainty in system understanding of user

Designing Dialog

- Apply user-centered design

Designing Dialog

- Apply user-centered design
  - Study user and task: How?

Designing Dialog

- Apply user-centered design
  - Study user and task: How?
    - Interview potential users, record human-human tasks
  - Study how the user interacts with the system

Designing Dialog

- Apply user-centered design
  - Study user and task: How?
    - Interview potential users, record human-human tasks
  - Study how the user interacts with the system
    - But it's not built yet....

Designing Dialog

- Apply user-centered design
  - Study user and task: How?
    - Interview potential users, record human-human tasks
  - Study how the user interacts with the system
    - But it’s not built yet....
      - Wizard-of-Oz systems: Simulations
        - User thinks they’re interacting with a system, but it’s driven by a human
      - Prototypes

Designing Dialog

- Apply user-centered design
  - Study user and task: How?
    - Interview potential users, record human-human tasks
  - Study how the user interacts with the system
    - But it’s not built yet....
      - Wizard-of-Oz systems: Simulations
        - User thinks they’re interacting with a system, but it’s driven by a human
      - Prototypes
  - Iterative redesign:
    - Test system: see how users really react, what problems occur, correct, repeat

SDS Evaluation

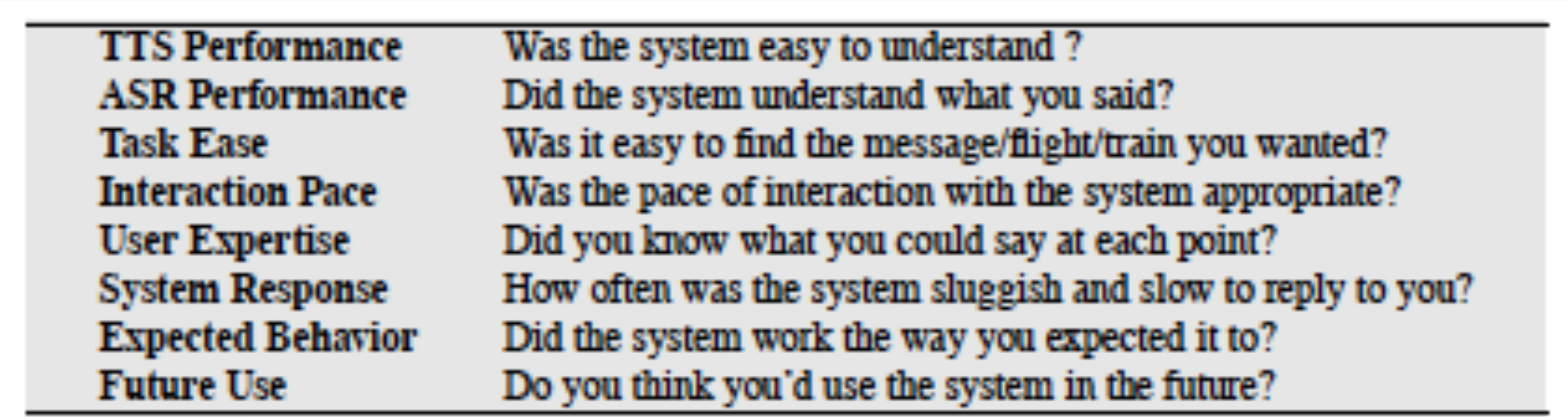
- Goal: Determine overall user satisfaction
  - Highlight systems problems; help tune

SDS Evaluation

- Goal: Determine overall user satisfaction
  - Highlight systems problems; help tune
- Classically: Conduct user surveys

SDS Evaluation

- Goal: Determine overall user satisfaction
  - Highlight systems problems; help tune
- Classically: Conduct user surveys



TTS Performance	Was the system easy to understand ?
ASR Performance	Did the system understand what you said?
Task Ease	Was it easy to find the message/flight/train you wanted?
Interaction Pace	Was the pace of interaction with the system appropriate?
User Expertise	Did you know what you could say at each point?
System Response	How often was the system sluggish and slow to reply to you?
Expected Behavior	Did the system work the way you expected it to?
Future Use	Do you think you'd use the system in the future?

Figure 24.14 User satisfaction survey, adapted from Walker et al. (2001).

SDS Evaluation

- User evaluation issues:

SDS Evaluation

- User evaluation issues:
  - Expensive; often unrealistic; hard to get real user to do
- Create model correlated with human satisfaction
- Criteria:

SDS Evaluation

- User evaluation issues:
  - Expensive; often unrealistic; hard to get real user to do
- Create model correlated with human satisfaction
- Criteria:
  - Maximize task success
    - Measure task completion: % subgoals; Kappa of frame values
  - Minimize task costs
    - Efficiency costs: time elapsed; # turns; # error correction turns
    - Quality costs: # rejections; # barge-in; concept error rate

PARADISE Model

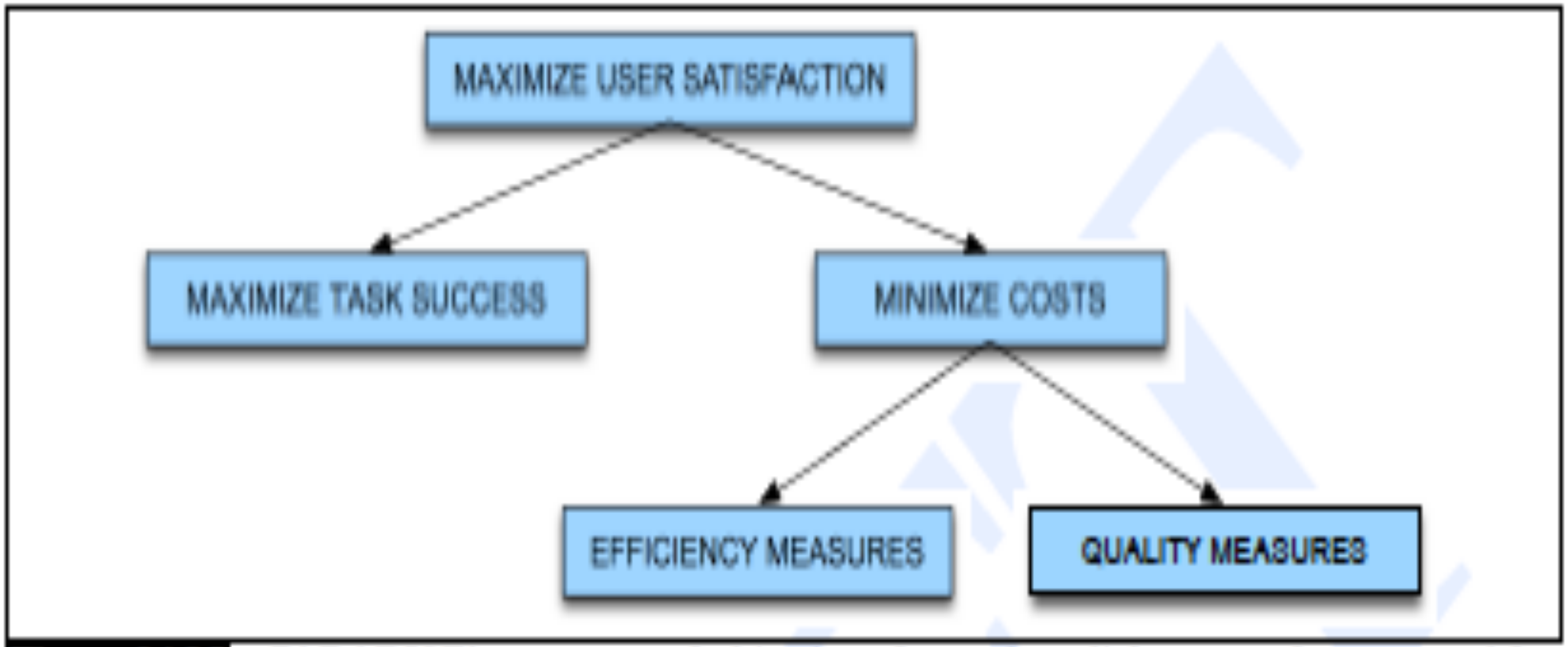


Figure 24.15 PARADISE's structure of objectives for spoken dialogue performance. After Walker et al. (1997).

PARADISE Model

- Compute user satisfaction with questionnaires
- Extract task success and costs measures from corresponding dialogs
  - Automatically or manually
- Perform multiple regression:
  - Assign weights to all factors of contribution to Usat
  - Task success, Concept accuracy key
- Allows prediction of accuracy on new dialog

Information State Models

- Challenges in dialog management
  - Difficult to evaluate
    - Hard to isolate from implementations
    - Integration inhibits portability
  - Wide gap between theoretical and practical models
    - Theoretical: logic-based, BDI, plan-based, attention/ intention
    - Practical: mostly finite-state or frame-based
    - Even if theory-consistent, many possible implementations
  - Implementation dominates

Why the Gap?

- Theories hard to implement
  - Underspecified
  - Overly complex, intractable
    - e.g. inferring all user intents
- Theories hard to compare
  - Employ diff't basic units
  - Disagree on basic structure

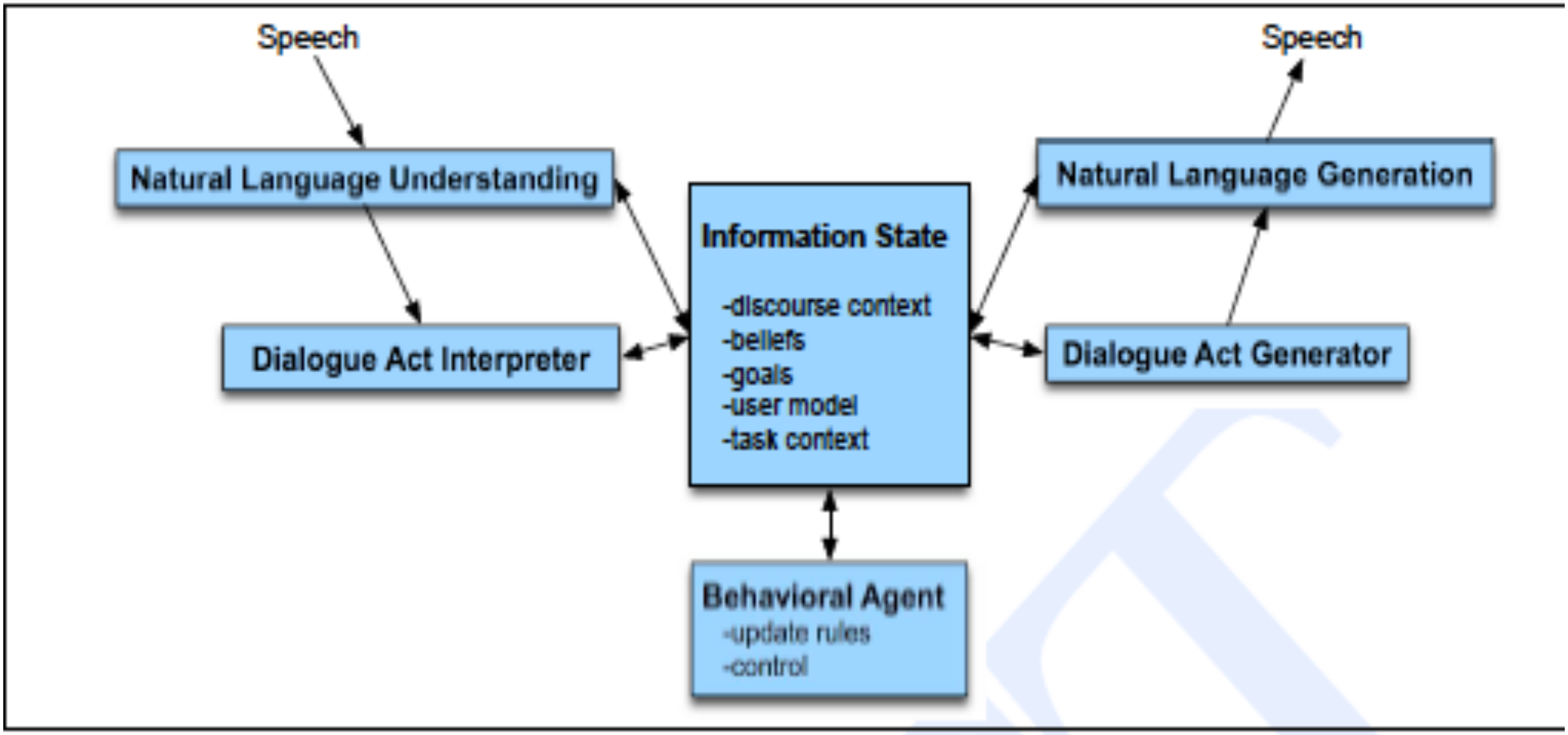
- Implementation is hard
  - Driven by technical limitations, optimizations
  - Driven by specific tasks
- Most approaches simplistic
  - Not focused on model details

Information State Approach

- Approach to formalizing dialog theories
- Toolkit to support implementation (Trindikit)
    - Designed to abstract out dialog theory components
- Example systems & related tools

Information State Architecture

- Simple ideas, complex execution

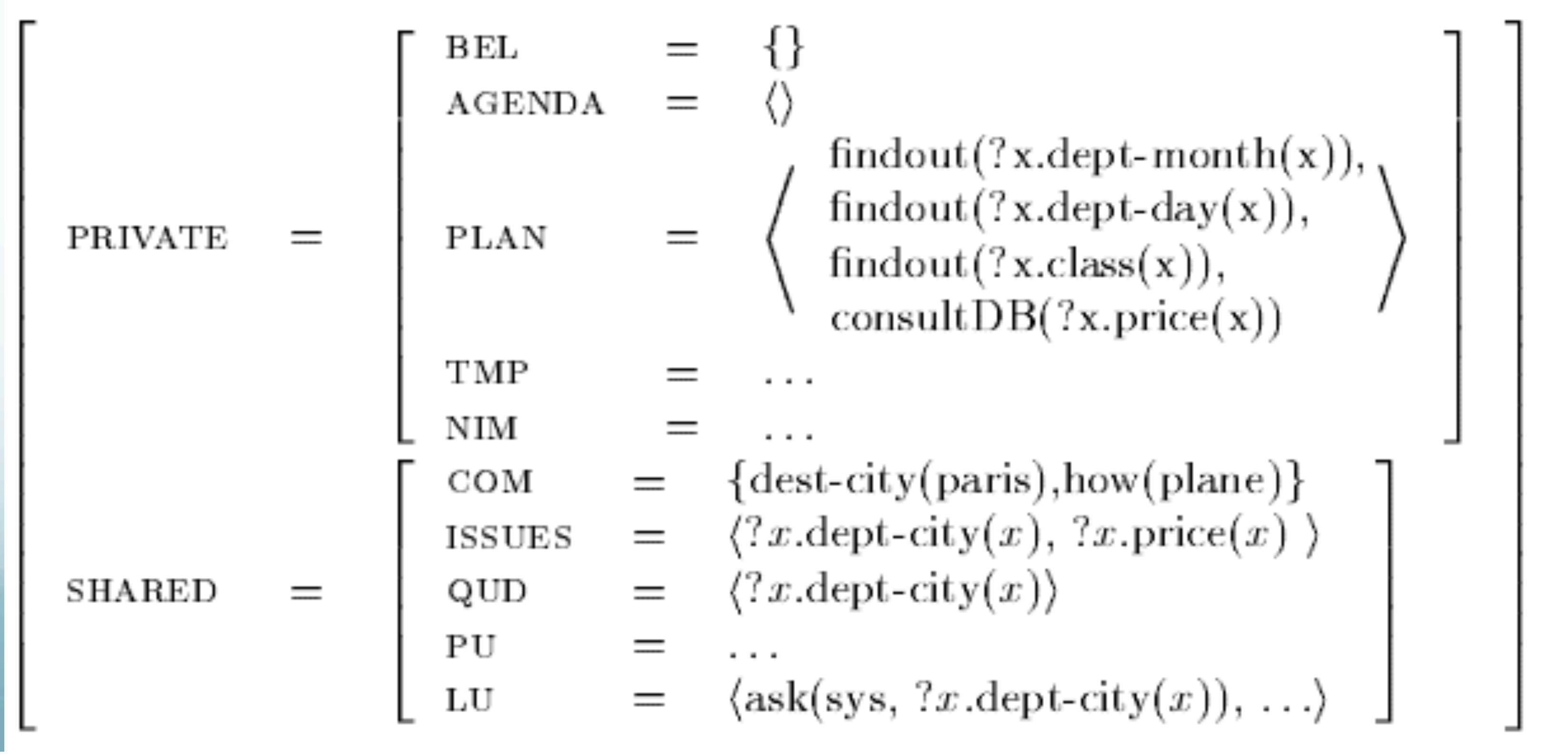


Information State Theory of Dialog

- Components:
  - Informational components:
    - Common context and internal models (belief, goals, etc)
  - Formal representations:
  - Dialog moves: recognition and generation
    - Trigger state updates
  - Update rules:
    - Describe update given current state, moves, etc
  - Update strategy:
    - Method for selecting rules if more than one applies
      - Simple or complex

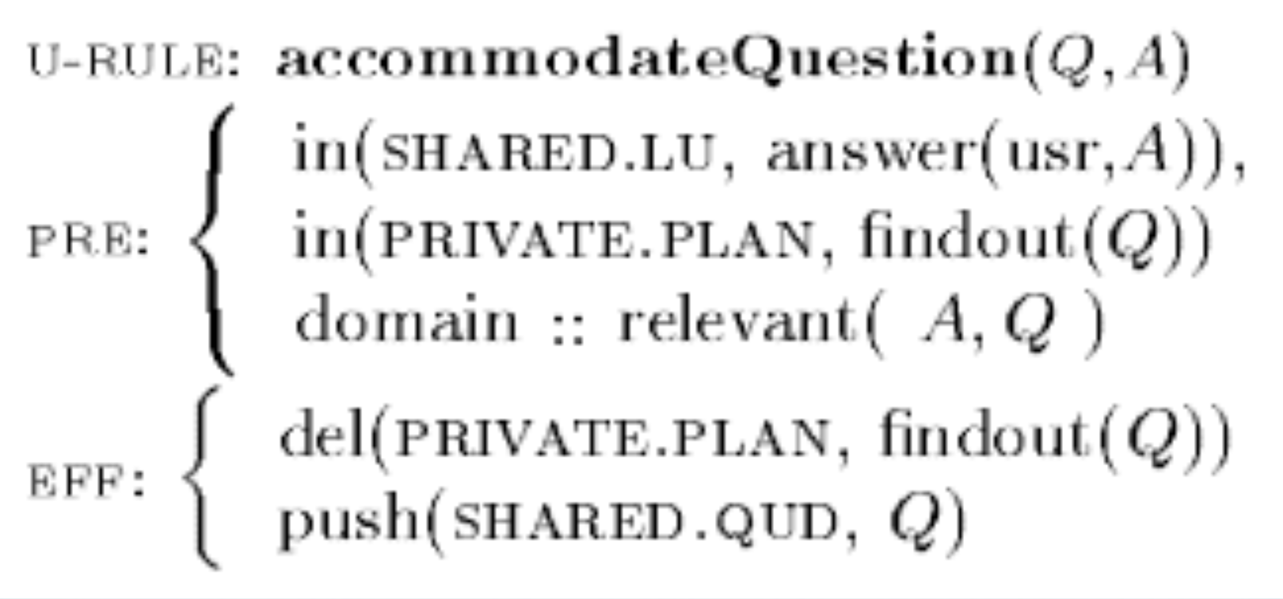
Example Dialog

- S: Welcome to the travel agency!
- U: flights to paris
- S: Okay, you want to know about price. A flight. To Paris. Let's see. What city do you want to go from?



$$\left[\begin{array}{lll}
\text{PRIVATE} & = & \left[\begin{array}{lll}
\text{BEL} & = & \{\} \\
\text{AGENDA} & = & \langle\rangle \\
\text{PLAN} & = & \left\langle\begin{array}{l}
\text{findout(?x.dept-month(x)),} \\
\text{findout(?x.dept-day(x)),} \\
\text{findout(?x.class(x)),} \\
\text{consultDB(?x.price(x))}
\end{array}\right\rangle \\
\text{TMP} & = & \ldots \\
\text{NIM} & = & \ldots
\end{array}\right] \\
\text{SHARED} & = & \left[\begin{array}{lll}
\text{COM} & = & \{\text{dest-city(paris),how(plane)}\} \\
\text{ISSUES} & = & \langle ?x.\text{dept-city}(x),\ ?x.\text{price}(x)\ \rangle \\
\text{QUD} & = & \langle ?x.\text{dept-city}(x)\rangle \\
\text{PU} & = & \ldots \\
\text{LU} & = & \langle \text{ask}(\text{sys},\ ?x.\text{dept-city}(x)),\ \ldots\rangle
\end{array}\right]
\end{array}\right]$$

Example Update Rule



U-RULE: **accommodateQuestion**(Q, A)

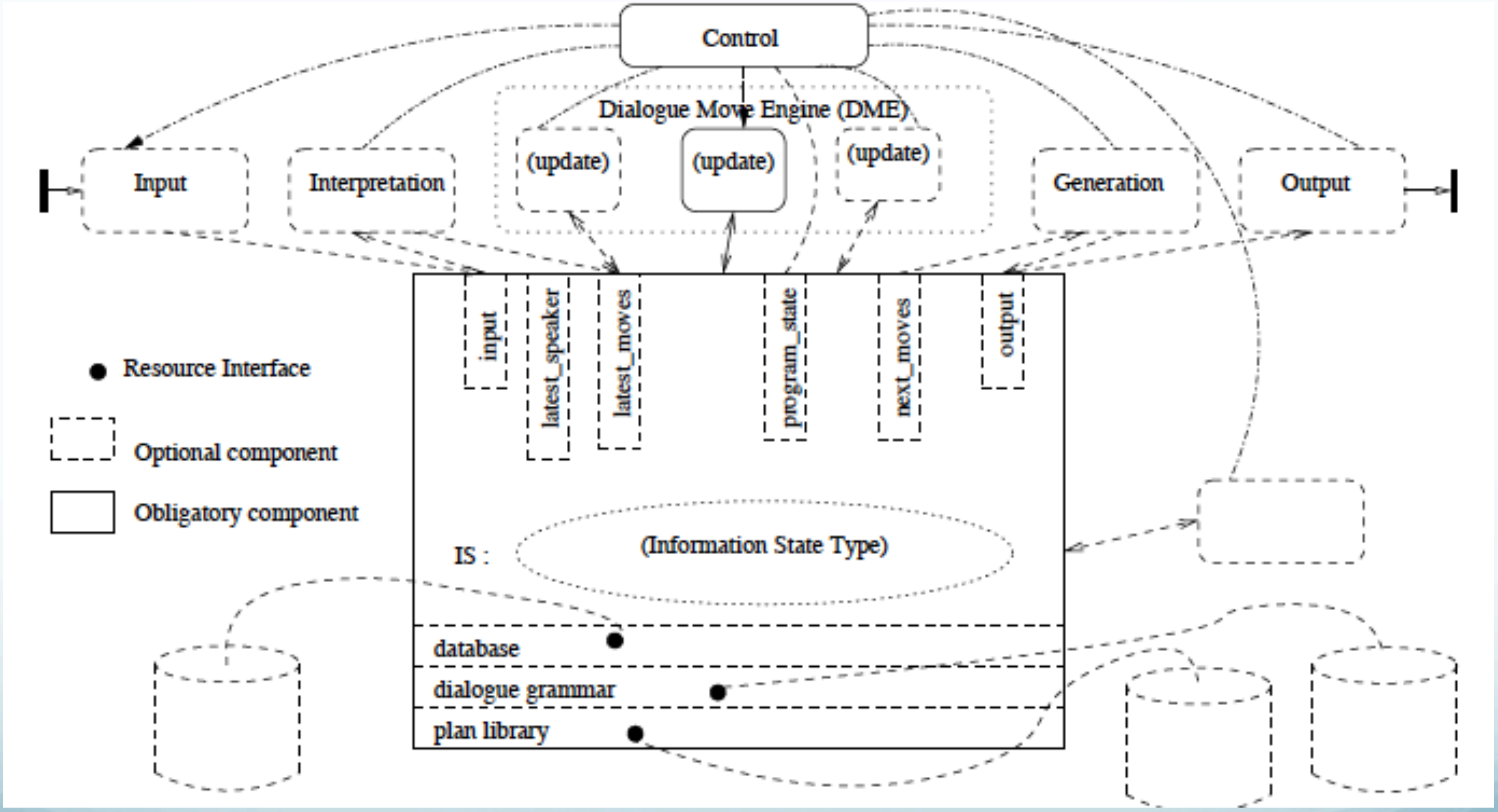
$$\text{PRE:} \left\{ \begin{array}{l} \text{in}(\text{SHARED.LU},\ \text{answer}(\text{usr}, A)), \\ \text{in}(\text{PRIVATE.PLAN},\ \text{findout}(Q)) \\ \text{domain} :: \text{relevant}(\ A, Q\ ) \end{array} \right.$$

$$\text{EFF:} \left\{ \begin{array}{l} \text{del}(\text{PRIVATE.PLAN},\ \text{findout}(Q)) \\ \text{push}(\text{SHARED.QUD},\ Q) \end{array} \right.$$

Implementation

- Dialog Move Engine (DME)
  - Implements an information state dialog model
  - Observes/interprets moves
  - Updates information state based on moves
  - Generates new moves consistent with state
- Full system requires: DME+
  - Input/output components
  - Interpretation: determine what move made
  - Generation: produce output for ‘next move’
  - Control system to manage components

Trindikit Architecture



Multi-level Architecture

- Separates types of design expertise, knowledge
- Domain & language resources → Domain system
- Dialog theory → Abstract DME
  - IS, update rules, etc
- Software Engineering → Trindikit
  - basic types, control

OpenDial

- Modern Java-based implementation
  - Significantly influenced by information structure model
  - Modeling uses declarative XML framework
  - Supports probabilistic models based on Bayes Nets
  - Hooks to Nuance ASR
- http://www.opendial-toolkit.net

Dialogue Acts

- Extension of speech acts
  - Adds structure related to conversational phenomena
    - Grounding, adjacency pairs, etc
- Many proposed tagsets
  - We'll see taxonomies soon

Dialogue Act Interpretation

- Automatically tag utterances in dialogue
- Some simple cases:
  - **YES-NO-Q:** Will breakfast be served on USAir 1557?
  - I don’t care about lunch.
  - Show me flights from L.A. to Orlando

Dialogue Act Interpretation

- Automatically tag utterances in dialogue
- Some simple cases:
  - **YES-NO-Q:** Will breakfast be served on USAir 1557?
  - **Statement:** I don’t care about lunch.
  - Show me flights from L.A. to Orlando

Dialogue Act Interpretation

- Automatically tag utterances in dialogue
- Some simple cases:
  - **YES-NO-Q:** Will breakfast be served on USAir 1557?
  - **Statement:** I don't care about lunch.
  - **Command:** Show me flights from L.A. to Orlando
- Is it always that easy?
  - Can you give me the flights from Atlanta to Boston?
  - Yeah.

Dialogue Act Interpretation

- Automatically tag utterances in dialogue
- Some simple cases:
  - **YES-NO-Q:** Will breakfast be served on USAir 1557?
  - **Statement:** I don't care about lunch.
  - **Command:** Show me flights from L.A. to Orlando
- Is it always that easy?
  - Can you give me the flights from Atlanta to Boston?
  - Yeah.
    - Depends on context: Y/N answer; agreement; back-channel

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:
  - Word information:
    - *Please, would you*: request; *are you*: yes-no question

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:
  - Word information:
    - *Please, would you*: request; *are you*: yes-no question
    - N-gram grammars
  - Prosody:

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:
  - Word information:
    - *Please, would you*: request; *are you*: yes-no question
    - N-gram grammars
  - Prosody:
    - Final rising pitch: question; final lowering: statement
    - Reduced intensity: *Yeah:* agreement vs backchannel

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:
  - Word information:
    - *Please, would you*: request; *are you*: yes-no question
    - N-gram grammars
  - Prosody:
    - Final rising pitch: question; final lowering: statement
    - Reduced intensity: *Yeah:* agreement vs backchannel
  - Adjacency pairs:

Dialogue Act Recognition

- How can we classify dialogue acts?
- Sources of information:
  - Word information:
    - *Please, would you*: request; *are you*: yes-no question
    - N-gram grammars
  - Prosody:
    - Final rising pitch: question; final lowering: statement
    - Reduced intensity: *Yeah:* agreement vs backchannel
  - Adjacency pairs:
    - Y/N question, agreement vs Y/N question, backchannel
    - DA bi-grams

HW #2

- Build a basic dialog system
  - Using a standard framework
    - Probably VoiceXML
- Work through system tutorial/”Hello world” example
- Implement System-initiative weather interface
- Implement revised Mixed-initiative system

VoiceXML

VoiceXML

- W3C standard for voice interfaces
    - XML-based ‘programming’ framework for speech systems
        - Provides recognition of:
            - Speech, DTMF (touch tone codes)

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input

VoiceXML

- W3C standard for voice interfaces
  - XML-based 'programming' framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps
    - Structures voice interaction

VoiceXML

- W3C standard for voice interfaces
  - XML-based ‘programming’ framework for speech systems
    - Provides recognition of:
      - Speech, DTMF (touch tone codes)
    - Provides output of synthesized speech, recorded audio
    - Supports recording of user input
    - Enables interchange between voice interface, web-based apps
    - Structures voice interaction
    - Can incorporate Javascript/PHP/etc for functionality

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative
  - Can implement frame-based mixed initiative

Capabilities

- Interactions:
  - Default behavior is FST-style, system initiative
  - Can implement frame-based mixed initiative
  - Support for sub-dialog call-outs

Speech I/O

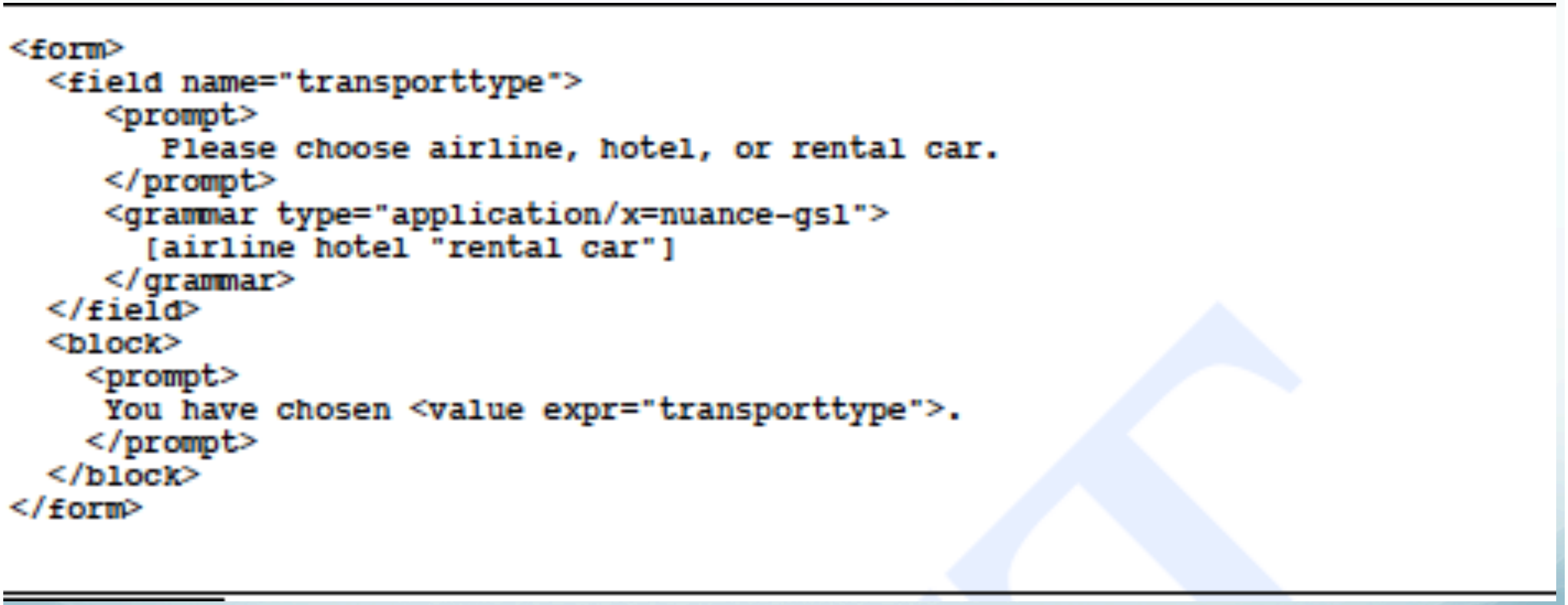
- ASR:
  - Supports speech recognition defined by
    - Grammars
    - Trigrams
    - Domain managers: credit card nos etc

Speech I/O

- ASR:
  - Supports speech recognition defined by
    - Grammars
    - Trigrams
    - Domain managers: credit card nos etc
- TTS:
  - <ssml> markup language
  - Allows choice of: language, voice, pronunciation
  - Allows tuning of: timing, breaks

Simple VoiceXML Example

- Minimal form:



```
<form>
  <field name="transporttype">
     <prompt>
        Please choose airline, hotel, or rental car.
     </prompt>
     <grammar type="application/x=nuance-gsl">
       [airline hotel "rental car"]
     </grammar>
  </field>
  <block>
    <prompt>
     You have chosen <value expr="transporttype">.
    </prompt>
  </block>
</form>
```

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name="transporttype">

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name=“transporttype”>

    - Prompt for user input
      - <prompt> Please choose airline, hotel, or rental car.</prompt>
        - Can include URL for recorded prompt, backs off

Basic VXML Document

- Main body: <form></form>
  - Sequence of fields: <field></field>
    - Correspond to variable storing user input
      - <field name="transporttype">
    - Prompt for user input
      - <prompt> Please choose airline, hotel, or rental car.</prompt>
        - Can include URL for recorded prompt, backs off
    - Specify grammar to recognize/interpret user input
      - <grammar>[airline hotel "rental car"]</grammar>

Other Field Elements

- Context-dependent help:
    - <help>Please select activity.</help>

Other Field Elements

- Context-dependent help:
    - <help>Please select activity.</help>

- Action to be performed on input:
  - <filled>
    - <prompt>You have chosen <value exp=“transporttype”>.
    - </prompt></filled>

Control Flow

- Default behavior:
  - Step through elements of form in document order

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next=“weather.xml”>
  - Other position in form: <goto next=“#departdate”>
-

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next="weather.xml">
  - Other position in form: <goto next="#departdate">
- Conditionals:
  - <if cond="varname=='air'">….</if>

Control Flow

- Default behavior:
  - Step through elements of form in document order
- Goto allows jump to:
  - Other form: <goto next=“weather.xml”>
  - Other position in form: <goto next=“#departdate”>
- Conditionals:
  - <if cond=“varname==‘air’”>....</if>
- Guards:
  - Default: Skip field if slot value already entered

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout

General Interaction

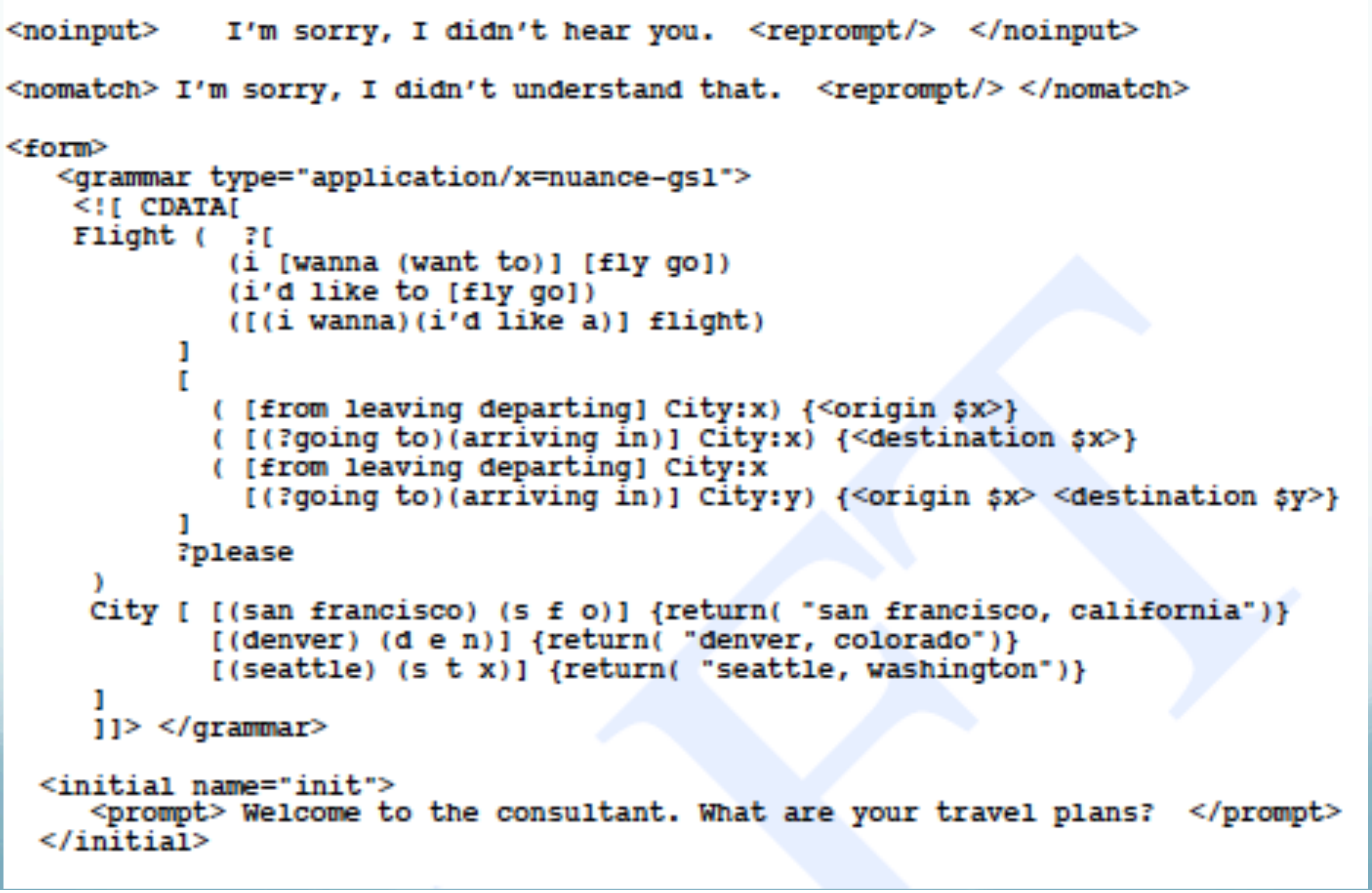
- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout
- <nomatch>:
  - Speech, but nothing valid recognized

General Interaction

- ‘Universals’:
  - Behaviors used by all apps, specify particulars
  - Pick prompts for conditions
- <noinput>:
  - No speech timeout
- <nomatch>:
  - Speech, but nothing valid recognized
- <help>:
  - General system help prompt

Complex Interaction

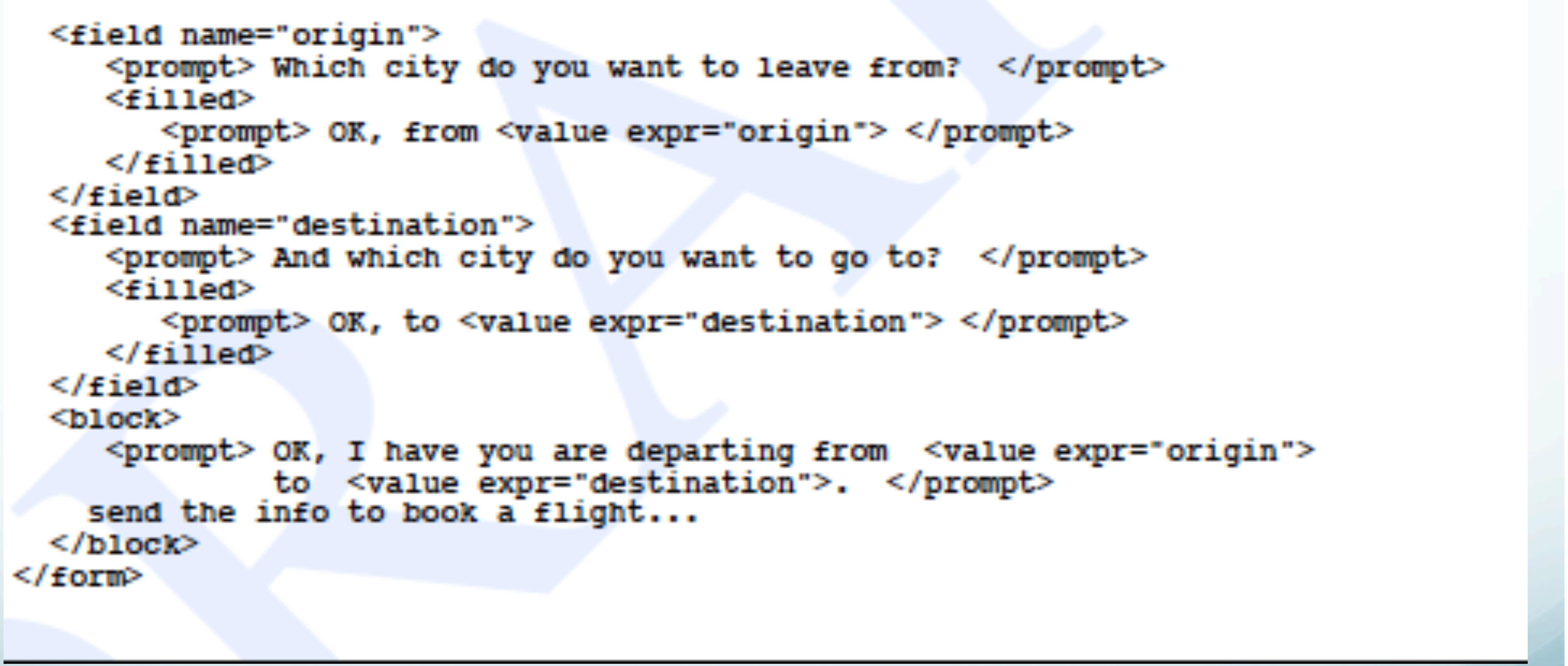
- Preamble, grammar:



```
<noinput>    I'm sorry, I didn't hear you.  <reprompt/>  </noinput>

<nomatch> I'm sorry, I didn't understand that.  <reprompt/> </nomatch>

<form>
   <grammar type="application/x=nuance-gsl">
    <![ CDATA[
    Flight (  ?[
             (i [wanna (want to)] [fly go])
             (i'd like to [fly go])
             ([(i wanna)(i'd like a)] flight)
          ]
          [
            ( [from leaving departing] City:x) {<origin $x>}
            ( [(?going to)(arriving in)] City:x) {<destination $x>}
            ( [from leaving departing] City:x
              [(?going to)(arriving in)] City:y) {<origin $x> <destination $y>}
          ]
          ?please
     )
     City [ [(san francisco) (s f o)] {return( "san francisco, california")}
            [(denver) (d e n)] {return( "denver, colorado")}
            [(seattle) (s t x)] {return( "seattle, washington")}
     ]
     ]]> </grammar>

  <initial name="init">
     <prompt> Welcome to the consultant. What are your travel plans?  </prompt>
  </initial>
```

Mixed Initiative

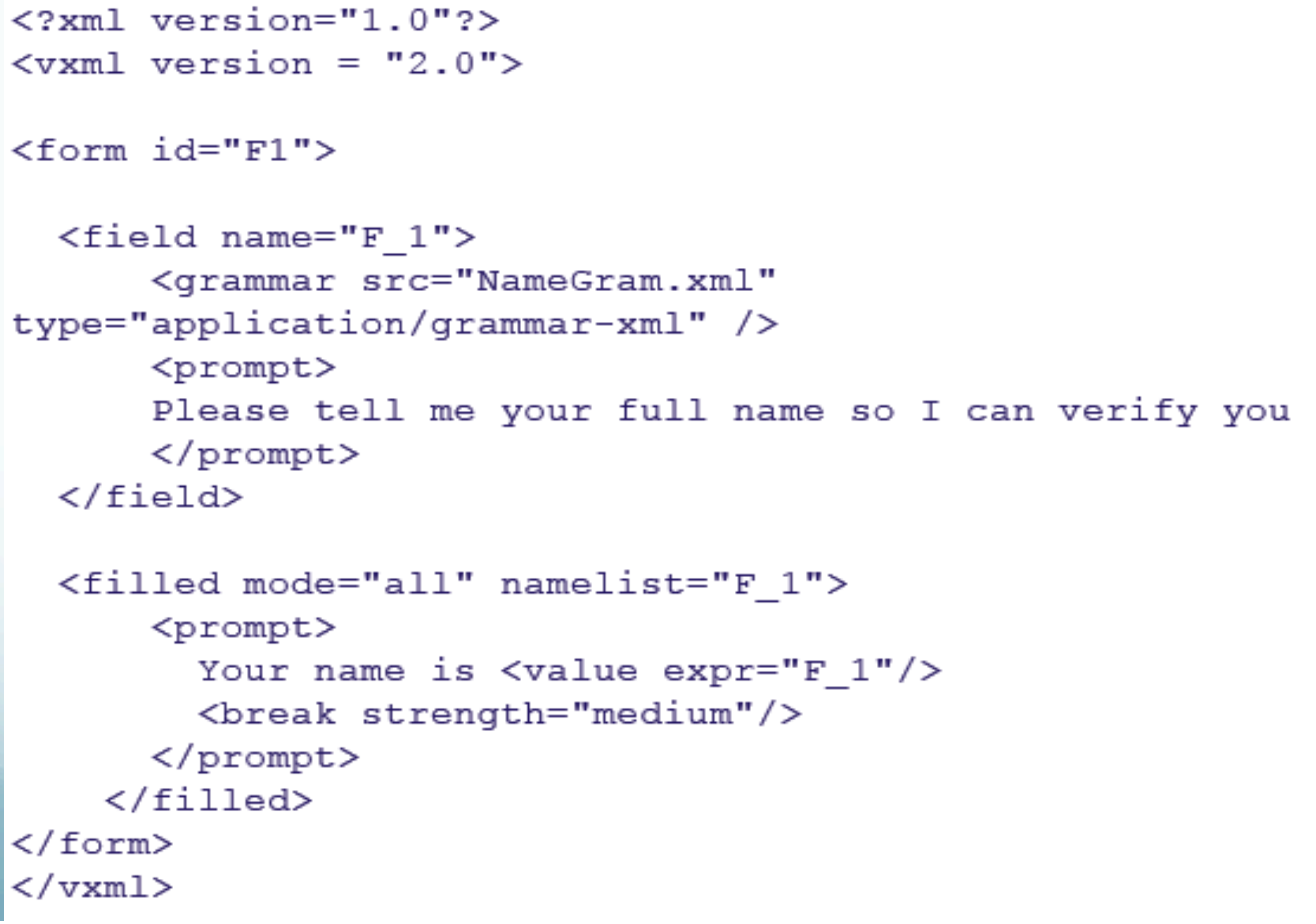
- With guard defaults



```
  <field name="origin">
     <prompt> Which city do you want to leave from?  </prompt>
     <filled>
        <prompt> OK, from <value expr="origin"> </prompt>
     </filled>
  </field>
  <field name="destination">
     <prompt> And which city do you want to go to?  </prompt>
     <filled>
        <prompt> OK, to <value expr="destination"> </prompt>
     </filled>
  </field>
  <block>
     <prompt> OK, I have you are departing from  <value expr="origin">
              to  <value expr="destination">.  </prompt>
    send the info to book a flight...
  </block>
</form>
```

Complex Interaction

- Preamble, external grammar:



```
<?xml version="1.0"?>
<vxml version = "2.0">

<form id="F1">

  <field name="F_1">
      <grammar src="NameGram.xml"
type="application/grammar-xml" />
      <prompt>
      Please tell me your full name so I can verify you
      </prompt>
  </field>

  <filled mode="all" namelist="F_1">
      <prompt>
        Your name is <value expr="F_1"/>
        <break strength="medium"/>
      </prompt>
    </filled>
</form>
</vxml>
```

Multi-slot Grammar

-
  ```
  <?xml version= "1.0"?>
     <grammar xml:lang="en-US" root = "TOPLEVEL">
      <rule id="TOPLEVEL" scope="public">
       <item>
  <!-- FIRST NAME RETURN -- >
     <item repeat="0-1">
      <ruleref  uri="#FIRSTNAME"/>
      <tag>out.firstNameSlot=rules.FIRSTNAME.firstNameSubslot;</tag>
     </item>
  <!-- MIDDLE NAME RETURN -->
     <item repeat="0-1">
      <ruleref  uri="#MIDDLENAME"/>
        <tag>out.middleNameSlot=rules.MIDDLENAME.middleNameSubslot;</tag>
     </item>
  <!-- LAST NAME RETURN -- >
    <ruleref  uri="#LASTNAME"/>
      <tag>out.lastNameSlot=rules.LASTNAME.lastNameSubslot;</tag>
     </item>
  <!-- TOP LEVEL RETURN-->
    <tag> out.F_1= out.firstNameSlot + out.middleNameSlot + out.lastNameSlot; </tag>
   </rule>
  ```

Multi-slot Grammar II

```
<rule id="FIRSTNAME" scope="public">
 <one-of>
  <item> matt<tag>out.firstNameSubslot="matthew";</tag></item>
  <item> dee <tag> out.firstNameSubslot="dee ";</tag></item>
  <item> jon <tag> out.firstNameSubslot="jon ";</tag></item>
  <item> george <tag>out.firstNameSubslot="george ";</tag></item>
  <item> billy <tag> out.firstNameSubslot="billy ";</tag></item>
 </one-of>
 </rule>
 <rule id="MIDDLENAME" scope="public">
 <one-of>
  <item> bon <tag>out.middleNameSubslot="bon ";</tag></item>
  <item> double ya <tag> out.middleNameSubslot="w ";</tag></item>
  <item> dee <tag> out.middleNameSubslot="dee ";</tag></item>
 </one-of>
 </rule>
 <rule id="LASTNAME" scope="public">
 <one-of>
  <item> henry <tag> out.lastNameSubslot="henry "; </tag></item>
  <item> ramone <tag> out.lastNameSubslot="dee ";  </tag></item>
  <item> jovi <tag> out.lastNameSubslot="jovi ";  </tag></item>
  <item> bush <tag> out.lastNameSubslot=""bush ";  </tag></item>
  <item> williams <tag> out.lastNameSubslot="williams "; </tag></item>
 </one-of>
 </rule>
```

Augmenting VoiceXML

- Don’t write XML directly
  - Use php or other system to generate VoiceXML
    - Used in ‘Let’s Go Dude’ bus info system

Augmenting VoiceXML

- Don’t write XML directly
  - Use php or other system to generate VoiceXML
    - Used in ‘Let’s Go Dude’ bus info system
- Pass input to other web services
  - i.e. to RESTful services

Augmenting VoiceXML

- Don’t write XML directly
  - Use php or other system to generate VoiceXML
    - Used in ‘Let’s Go Dude’ bus info system
- Pass input to other web services
  - i.e. to RESTful services
- Access web-based audio for prompts