



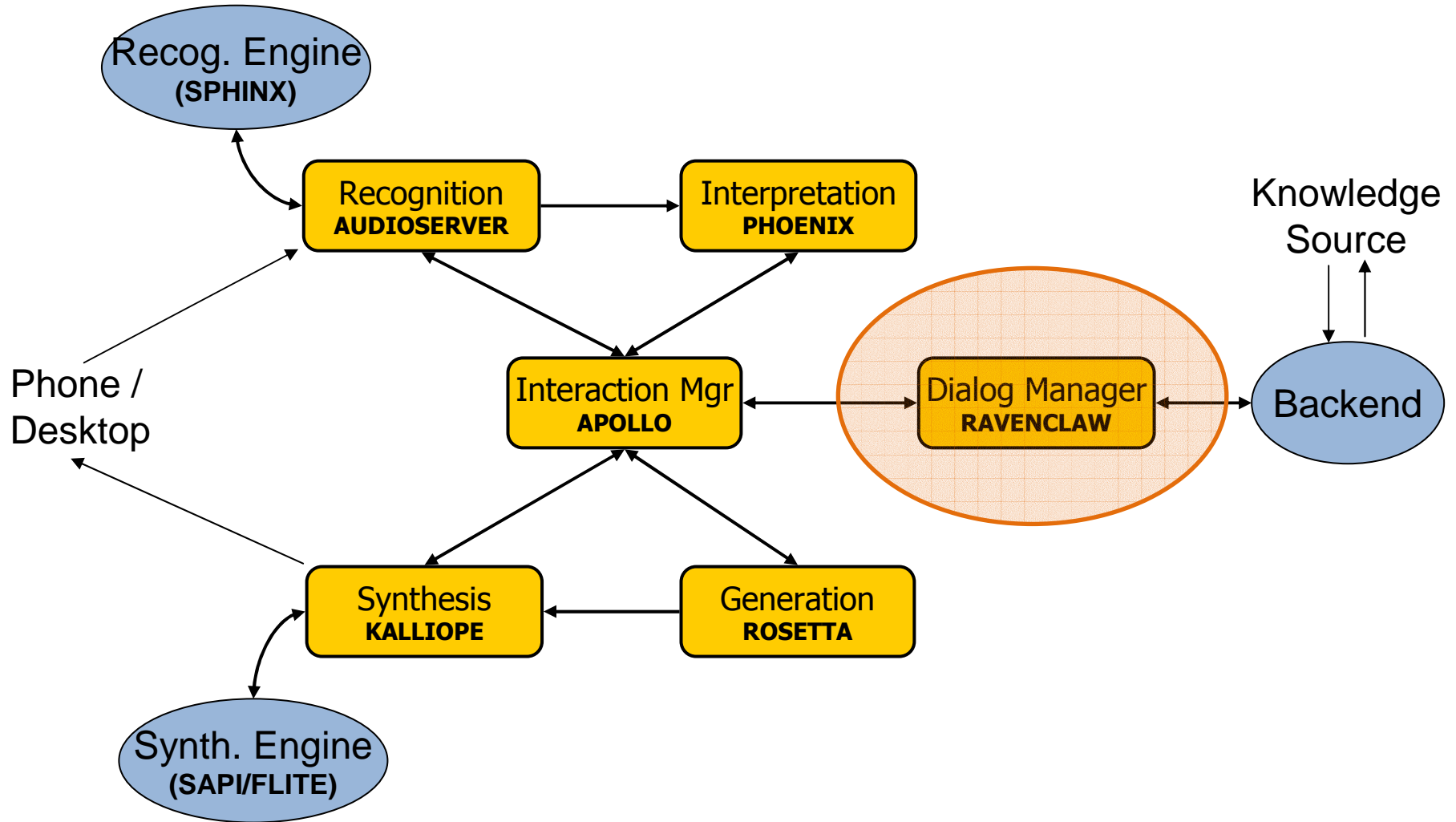
# *Speech Processing 15-492/18-492*

---

## Spoken Dialog Systems

- Details of *Olympus* modules
- Dialog Task Design

# The Olympus Architecture



# RavenClaw

- ◆ *Plan-based dialog manager*
- ◆ *Task-independent engine*
  - *core Olympus library*
  - *manage dialog by executing task specification*
  - *provides generic domain-independent behavior*
    - ⊗ *Help, repeat, ...*
    - ⊗ *Confirmation, non-understandings...*
- ◆ *Dialog Task Specification*
  - *dialog plan*
  - *interpretation context*

# RavenClaw Architecture

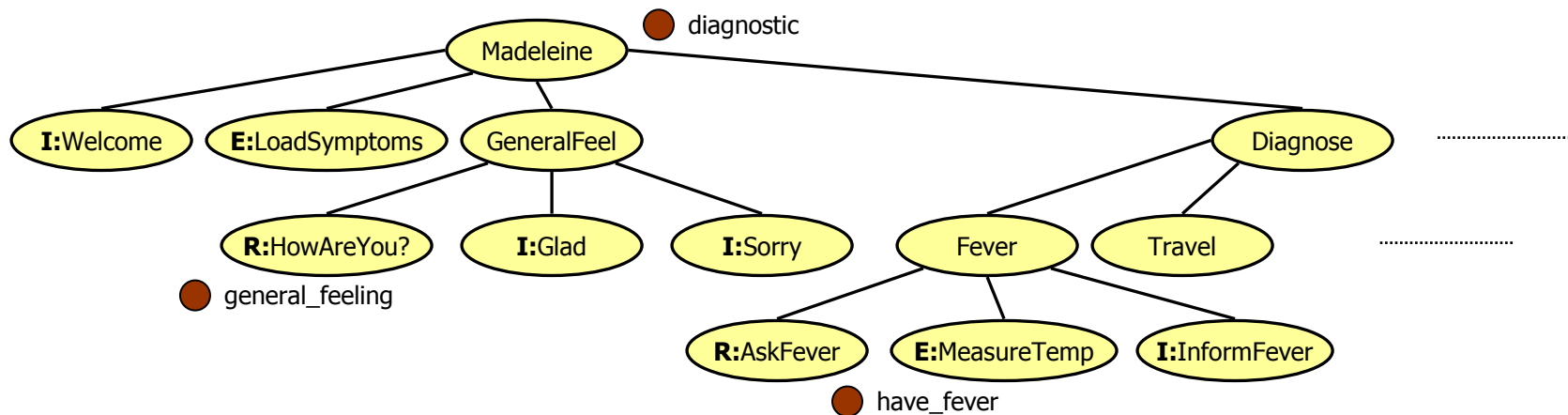
## *Task-independent Dialog Engine*

- ◆ *Manages dialog by executing the dialog task specification*
- ◆ *Provides many domain-independent conversational strategies*
- ◆ *Standard for most applications*
  - *No need to modify, just link shared library*

## *Dialog Task (Specification)*

- ◆ *Captures all domain-specific dialog (task) logic using a hierarchical description*
- ◆ *Unique to each application*
  - *Must be created for each application*
  - *Links to dialog engine library*

# RavenClaw: Dialog Task Specification



## ◆ *Tree of dialog agents*

- *Terminals: Inform, Request, Expect, Execute*
- *Non-terminals / Dialog agency: plans execution of child nodes*

## ◆ *Hierarchical Task Execution Network; each agent:*

- *Preconditions*
- *Success & failure criteria*
- *Trigger (focus) criteria*
- *Effects*

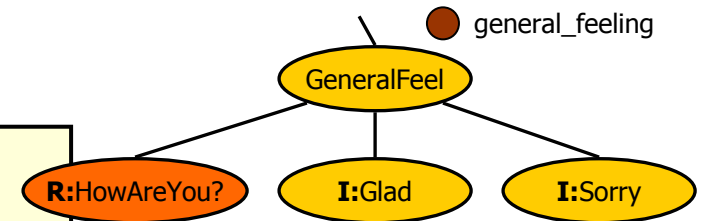
# Sample Task Specification Code

```
// /Madeleine/GeneralFeel
DEFINE_AGENCY(CGeneralFeel,
  DEFINE_CONCEPTS(
    STRING_USER_CONCEPT(general_feeling, none))
  DEFINE_SUBAGENTS(
    SUBAGENT(HowAreYou, CHowAreYou)
    SUBAGENT(Glad, CGlad)
    SUBAGENT(Sorry, CSorry))
  SUCCEEDS_WHEN(COMPLETED(Glad) || COMPLETED(Sorry)))

// /Madeleine/GeneralFeel/HowAreYou
DEFINE_REQUEST_AGENT(CHowAreYou,
  REQUEST_CONCEPT(general_feeling)
  GRAMMAR_MAPPING("![Yes]>good, ![FeelingGood]>good,"
    "![FeelingSoSo]>soso, ![FeelingBad]>bad"))

// /Madeleine/GeneralFeel/Glad
DEFINE_INFORM_AGENT(CGlad,
  PRECONDITION(C("general_feeling") == CString("good"))
  PROMPT("inform glad_youre_good")
  ON_COMPLETION(FINISH(/Madeleine)))

// /Madeleine/GeneralFeel/Sorry
DEFINE_INFORM_AGENT(CSorry,
  PRECONDITION(C("general_feeling") != CString("good"))
  PROMPT("inform sorry_youre_bad"))
```



# RavenClaw Task Specification Language (RCTSL)

- ◆ *(Pseudo-)declarative language*
  - *Defines concept types*
  - *Describes the task tree*
- ◆ *Set of C++ macros*
  - *Concept types and agents are classes*
  - *Can use pure C++ code if necessary*
  - *Need to be recompiled when modified*

# RCTSL Concepts

- ◆ *Concepts are effectively RCTSL variables*
  - *Store values for later use and manipulation*
- ◆ *Standard types*
  - *String, integer and bool*
- ◆ *User-defined types*
  - *Structures and arrays*
- ◆ *Two main categories:*
  - *System concepts*
    - ⊗ *Store internal values, database results, etc.*
  - *User concepts*
    - ⊗ *Capture entities obtained from the user*

# How User Concepts get Values

## ◆ GRAMMAR\_MAPPING *directive*

- *Defines which grammar slot(s) from Phoenix are assigned to an expected concept*

```
// /MyBus/PerformTask/GetQuerySpecs/RequestOriginAbe
DEFINE_REQUEST_AGENT( CRequestOriginPlace,
    REQUEST_CONCEPT(origin)
    PROMPT("request origin_place")
    GRAMMAR_MAPPING("[origin_place], ![Place]")
)
```

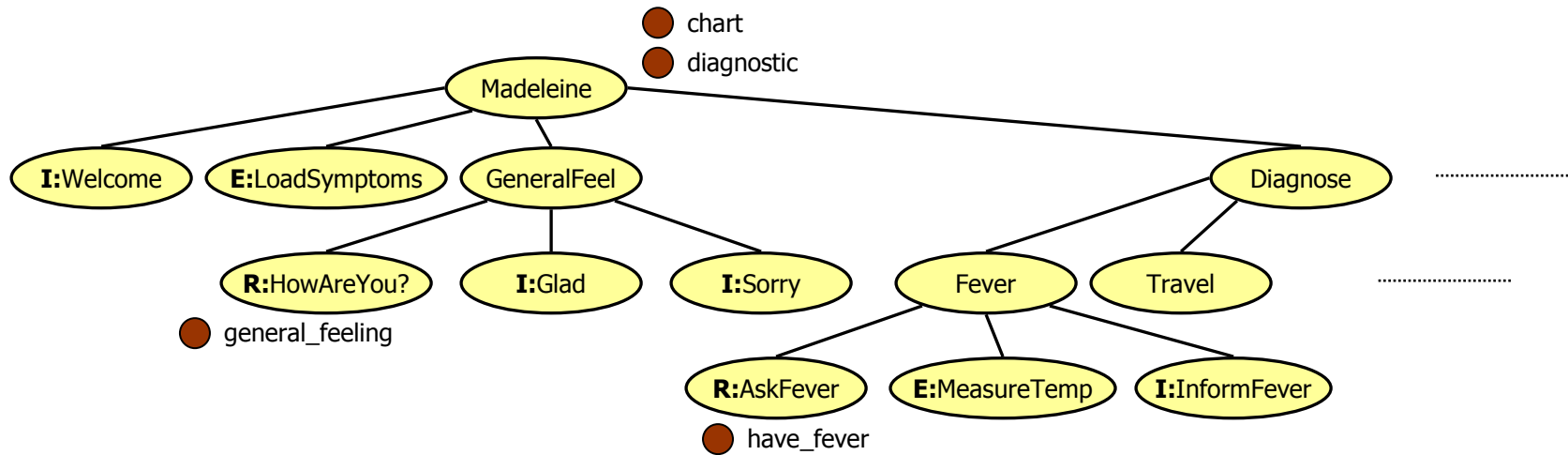
- *Maps parsed value from grammar (slot [origin\_place]) to concept origin*

# Specifying Binding Scope

- ◆ *Initiative can be controlled via binding scope*
  - *System vs. Mixed initiative*
- ◆ *Grammar mappings encode binding scope:*
  - *Special character before grammar slot name*
  - *Strict (!): bind only when request agent is active*
  - *Open (@): bind always*
  - *Default (∅): bind only when request agent's subtask is active*

```
// /MyBus/PerformTask/GetQuerySpecs/RequestOriginPlace
DEFINE_REQUEST_AGENT( CRequestOriginPlace,
    REQUEST_CONCEPT(origin)
    PROMPT("request origin_place")
    GRAMMAR_MAPPING("[origin_place], ![Place]")
)
```

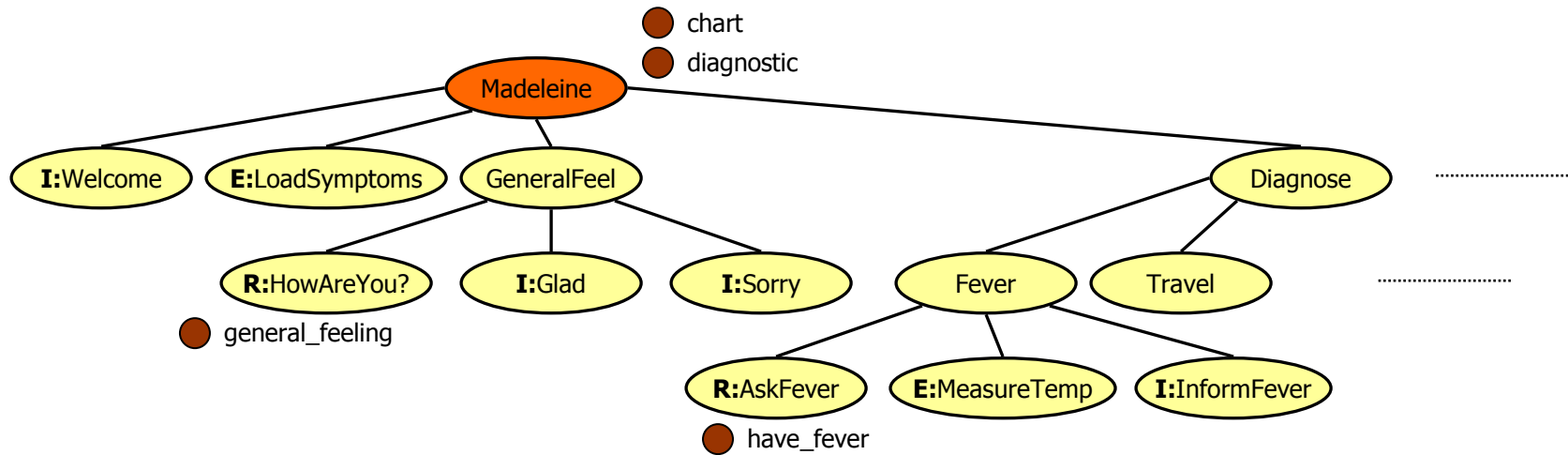
# RavenClaw Execution



**Dialog Stack**

**Expectation Agenda**

# RavenClaw Execution



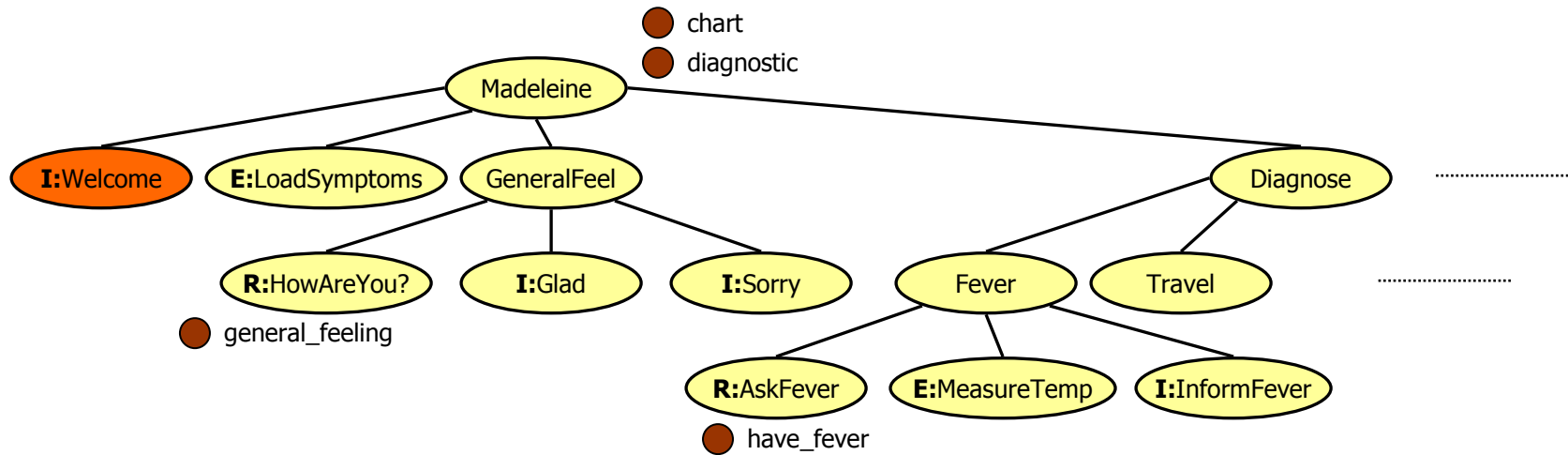
---

**Dialog Stack**

**Expectation Agenda**

**Madeleine**

# RavenClaw Execution

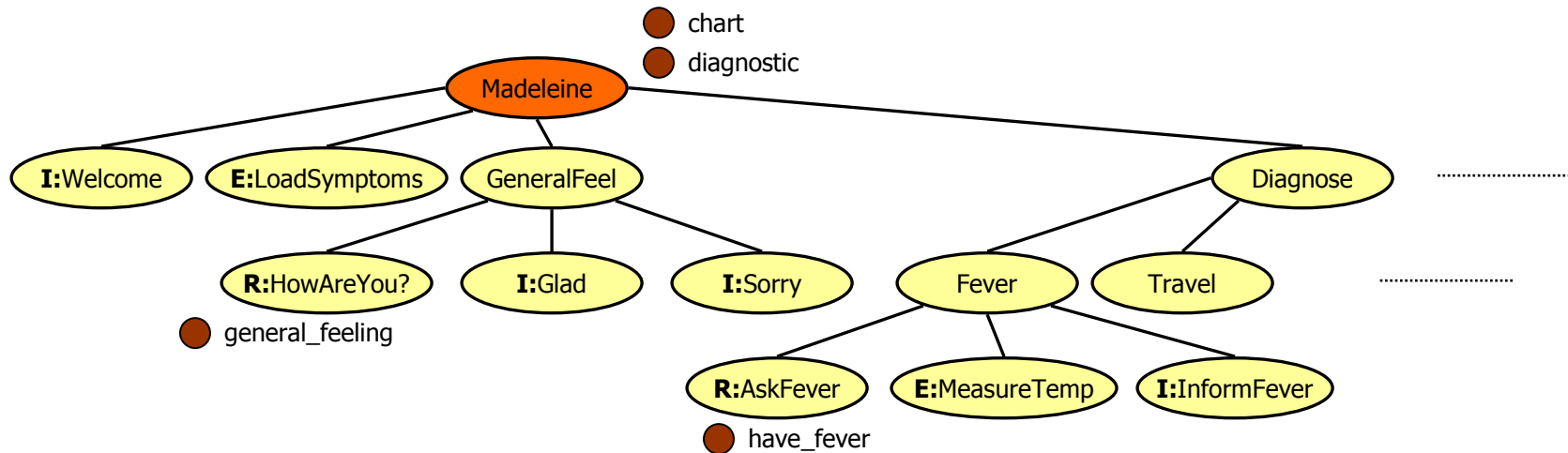


**Dialog Stack**

**Expectation Agenda**

Welcome
Madeleine

# RavenClaw Execution



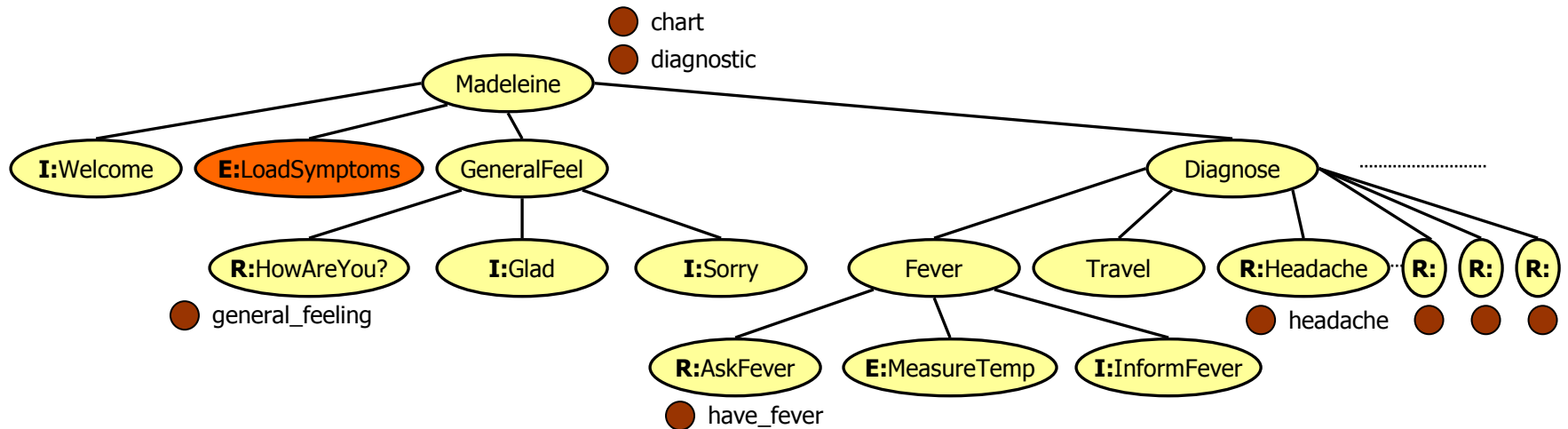
**Dialog Stack**

**Expectation Agenda**

Hi, this is Madeleine, the automated...

**Madeleine**

# RavenClaw Execution



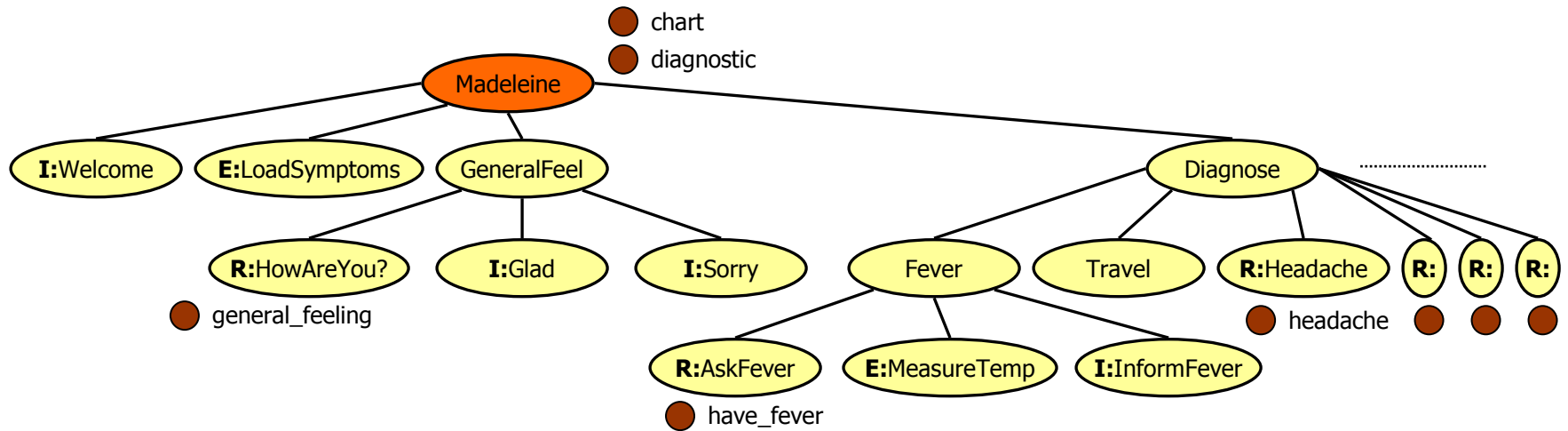
**Dialog Stack**

**Expectation Agenda**

Hi, this is Madeleine, the automated...

<b>LoadSymptoms</b>
<b>Madeleine</b>

# RavenClaw Execution



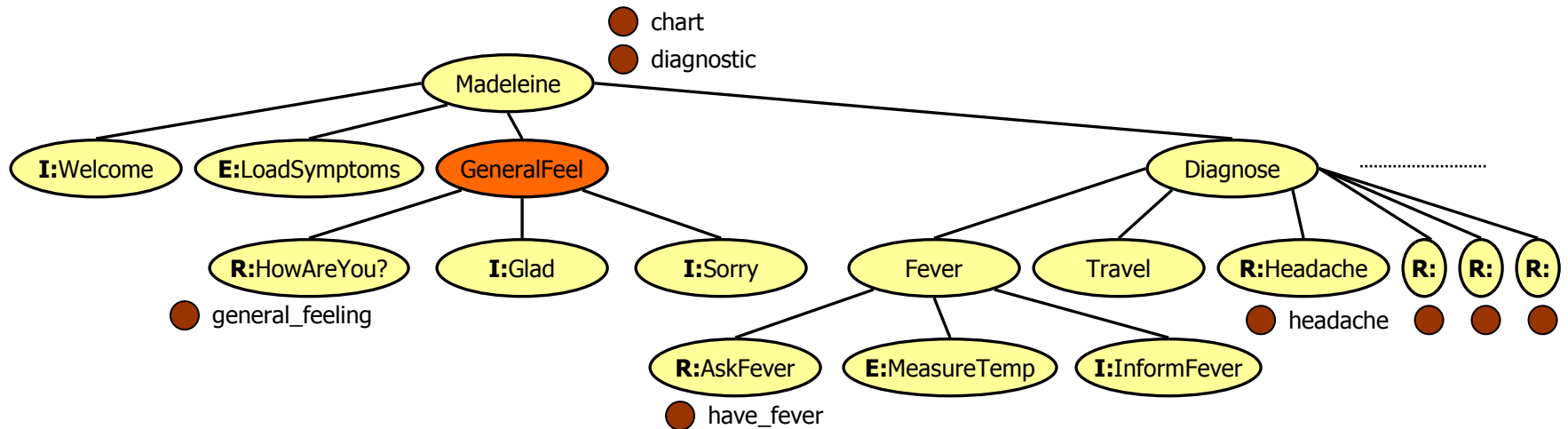
**Dialog Stack**

**Expectation Agenda**

Hi, this is Madeleine, the automated...

**Madeleine**

# RavenClaw Execution



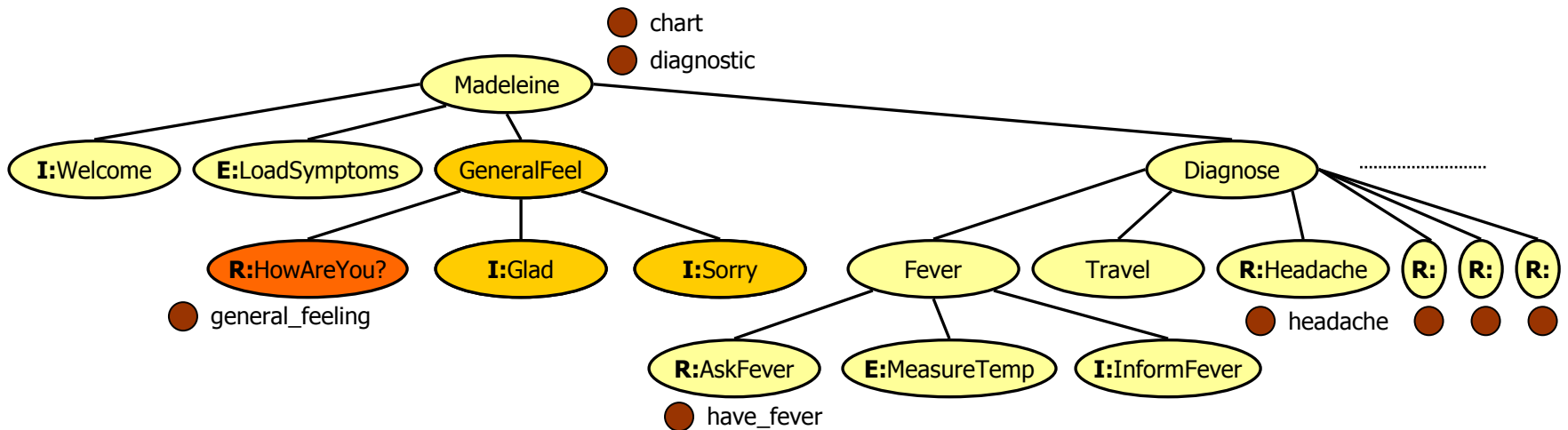
**Dialog Stack**

**Expectation Agenda**

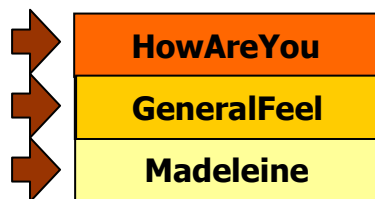
Hi, this is Madeleine, the automated...

<b>GeneralFeel</b>
<b>Madeleine</b>

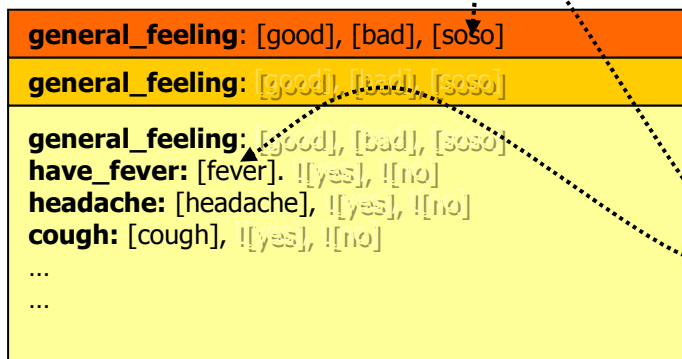
# RavenClaw Execution / Input Pass



## Dialog Stack



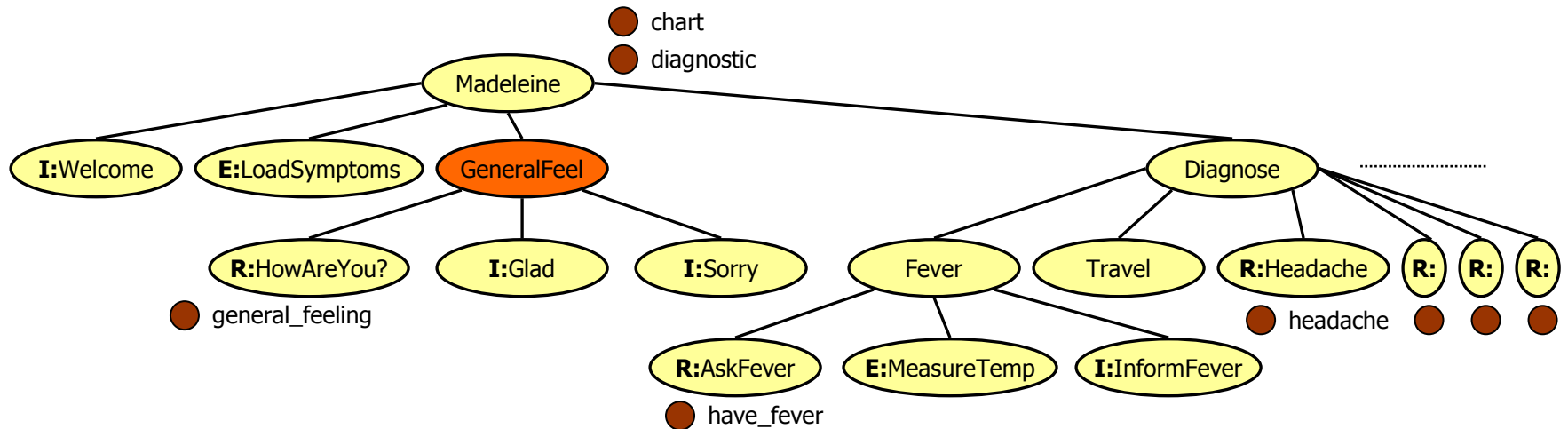
## Expectation Agenda



Hi, this is Madeleine, the automated...  
How are you feeling today?

*Not so good, I think I have a fever*  
 ◆ [soSo](not so good)  
 ◆ [fever](I think I have a fever)

# RavenClaw Execution



**Dialog Stack**

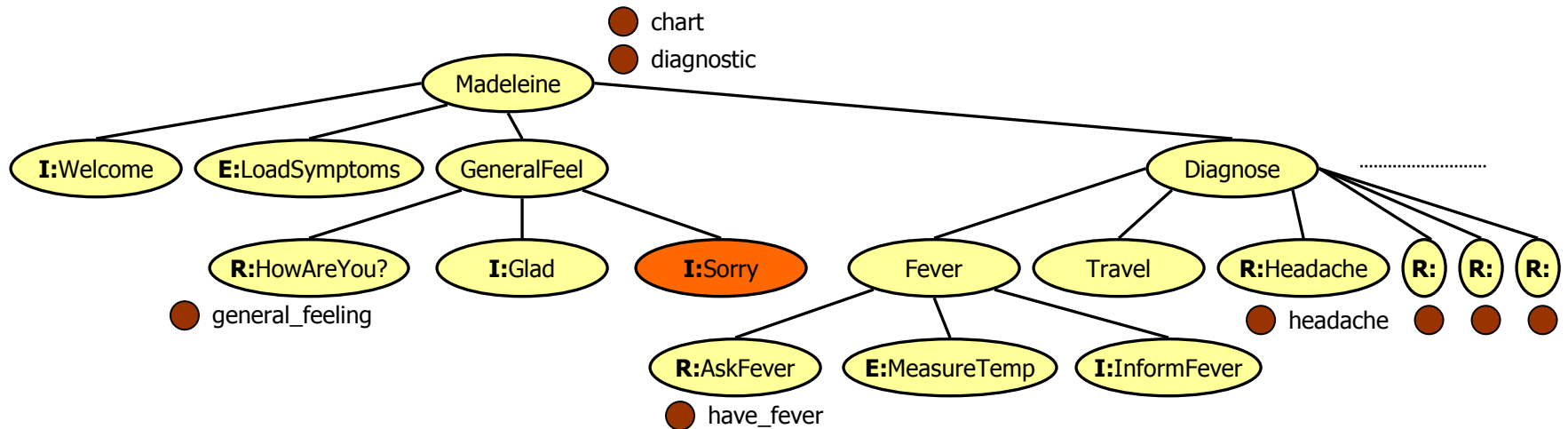
**Expectation Agenda**

Hi, this is Madeleine, the automated...  
How are you feeling today?

*Not so good, I think I have a fever*  
[soso](not so good)  
[fever](I think I have a fever)

<b>GeneralFeel</b>
<b>Madeleine</b>

# RavenClaw Execution



## Dialog Stack

## Expectation Agenda

Sorry
GeneralFeel
Madeleine

Hi, this is Madeleine, the automated...  
How are you feeling today?

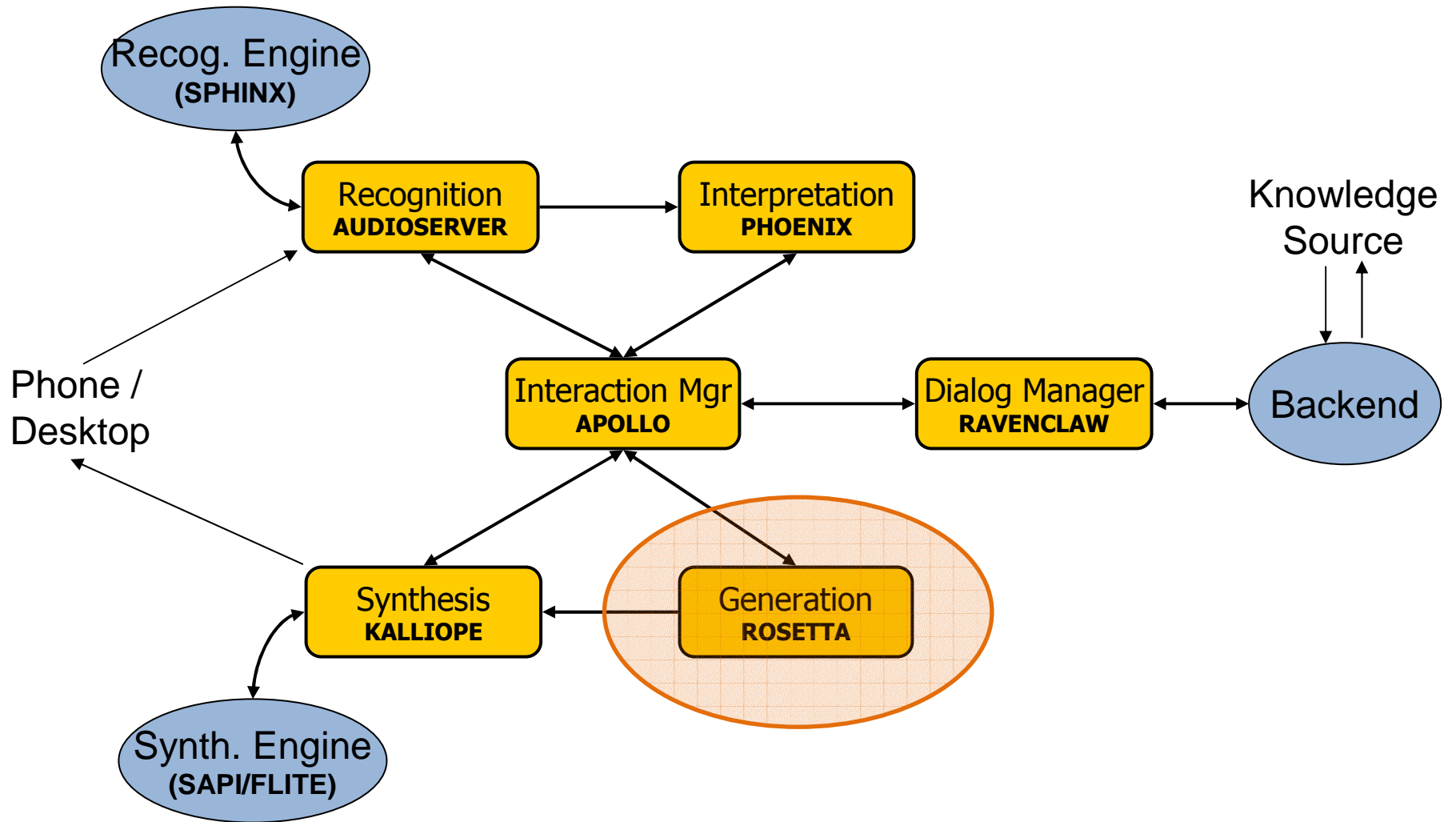
*Not so good, I think I have a fever*  
[soso](not so good)  
[fever](I think I have a fever)

Oh, I'm sorry to hear that...  
Let me take your temperature...

# RavenClaw – Other features

- ◆ *Transparently provides conversational skillset*
  - *Universal dialog mechanisms:*
    - ⊗ *Repeat, Quit, etc.*
  - *Help:*
    - ⊗ *Help!, What can I say?*
  - *Error handling:*
    - ⊗ *Explicit and implicit confirmations*
    - ⊗ *Strategies for recovering from non-understandings*
- ◆ *Dynamic dialog task generation*
- ◆ *Dynamic dialog control policy*

# The Olympus Architecture



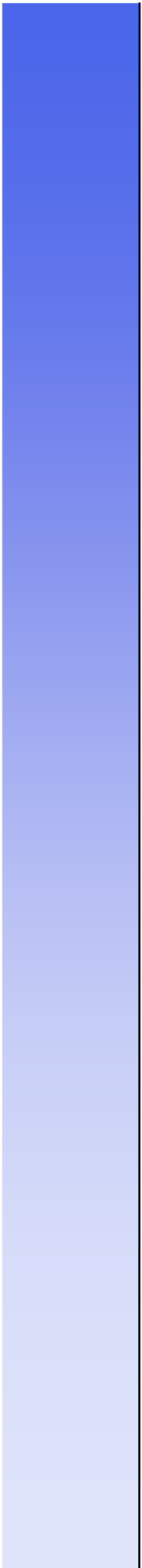
# Rosetta Language Generation

- ◆ *Template- and stochastic-based language generation*
  - *Input: (act, object, {slot=value})*
  - *Output: text (tagged with concepts)*
- ◆ *Takes semantic output from the dialog manager, generates corresponding surface forms*

```
# welcome to the system
"welcome" => "Welcome to RoomLine, the automated conference room.
             "reservation system."

# greet user
"greet_user" => ("Hello, <user_name>.",
                "Hi, <user_name>, good to hear from you again.")

# inform the user that the system has misunderstood the times (order)
"wrong_time_order" => sub {
    my %args = @_;
    my $time_interval_as_string =
        get_wrong_time_interval_as_string(\%args,
            "room_query.date_time.time");
    my $answer = "I'm sorry, I must have misunderstood the ."
                "time you needed the room. "
    $answer .= "I heard $time_interval_as_string. "
    return ["$answer So, let's see ... ",
            "$answer So, let's try this again ... ",
            "$answer So, let's try this once more ... "];
},
```



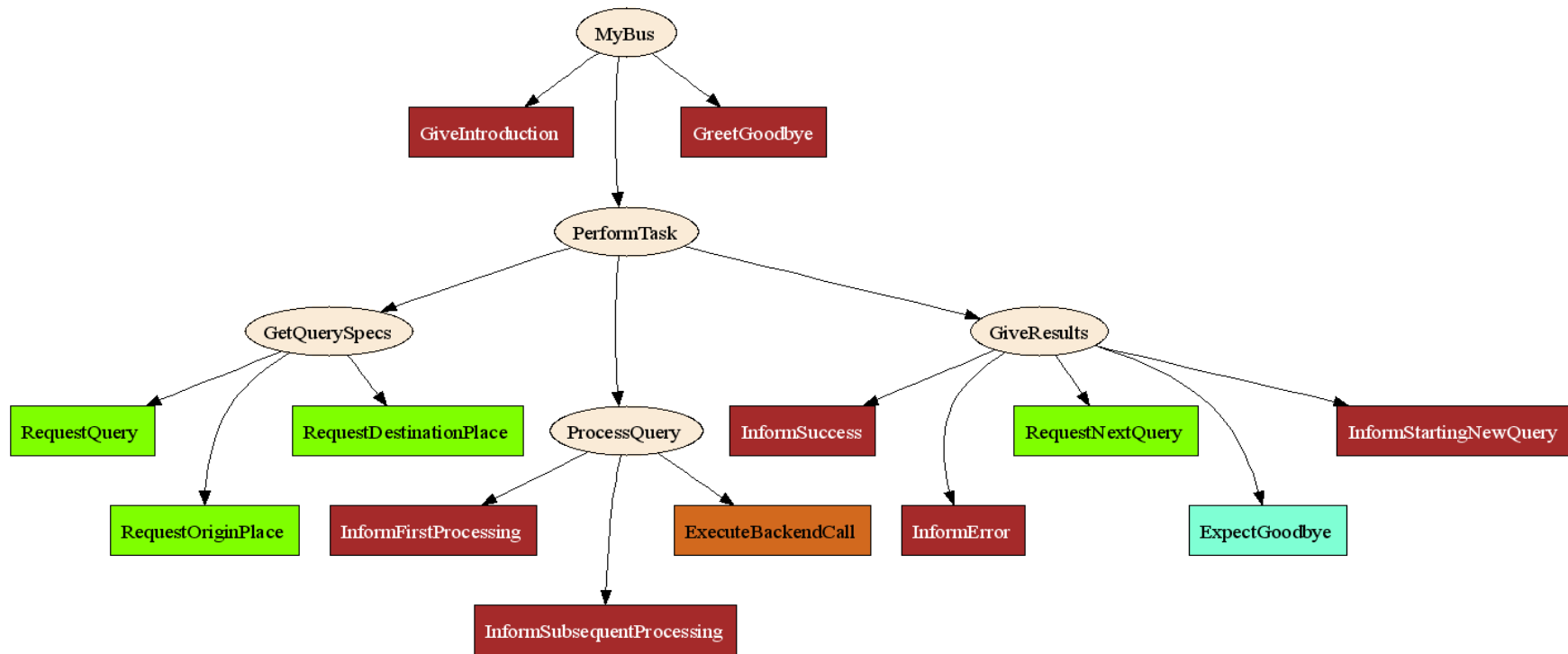
# Designing a Dialog Task

- ◆ *Good input and output language!*
  - *List expected user utterances*
    - ⊗ *Get **several** people to write example sentences, to improve coverage!*
    - ⊗ *Use to design grammar for system understanding*
  - *Write system prompts*
    - ⊗ ***Be concise!!** Nice written language does not translate well into spoken language...*

# Designing a Dialog Task

- ◆ *Structure the task specification*
  - *If it's a tree-based system, draw out the tree!*
  - *Define what information is needed/expected, and where*
- ◆ *Typical information-giving systems tend to have similar structure*
  - *greet – do task – goodbye*
    - ⊗ *do task: get info – process info – give answer*

# Example Dialog Task Tree



# Task Structure (Example 1)

S: Welcome to MyBus.

S: Which itinerary are you looking for?

U: I need to go to the airport.

S: Where are you leaving from?

U: Downtown.

S: Let me see.

S: There is a 28X leaving downtown at 10:15 AM. It will get to the airport at 11 AM.

S: You can say ...

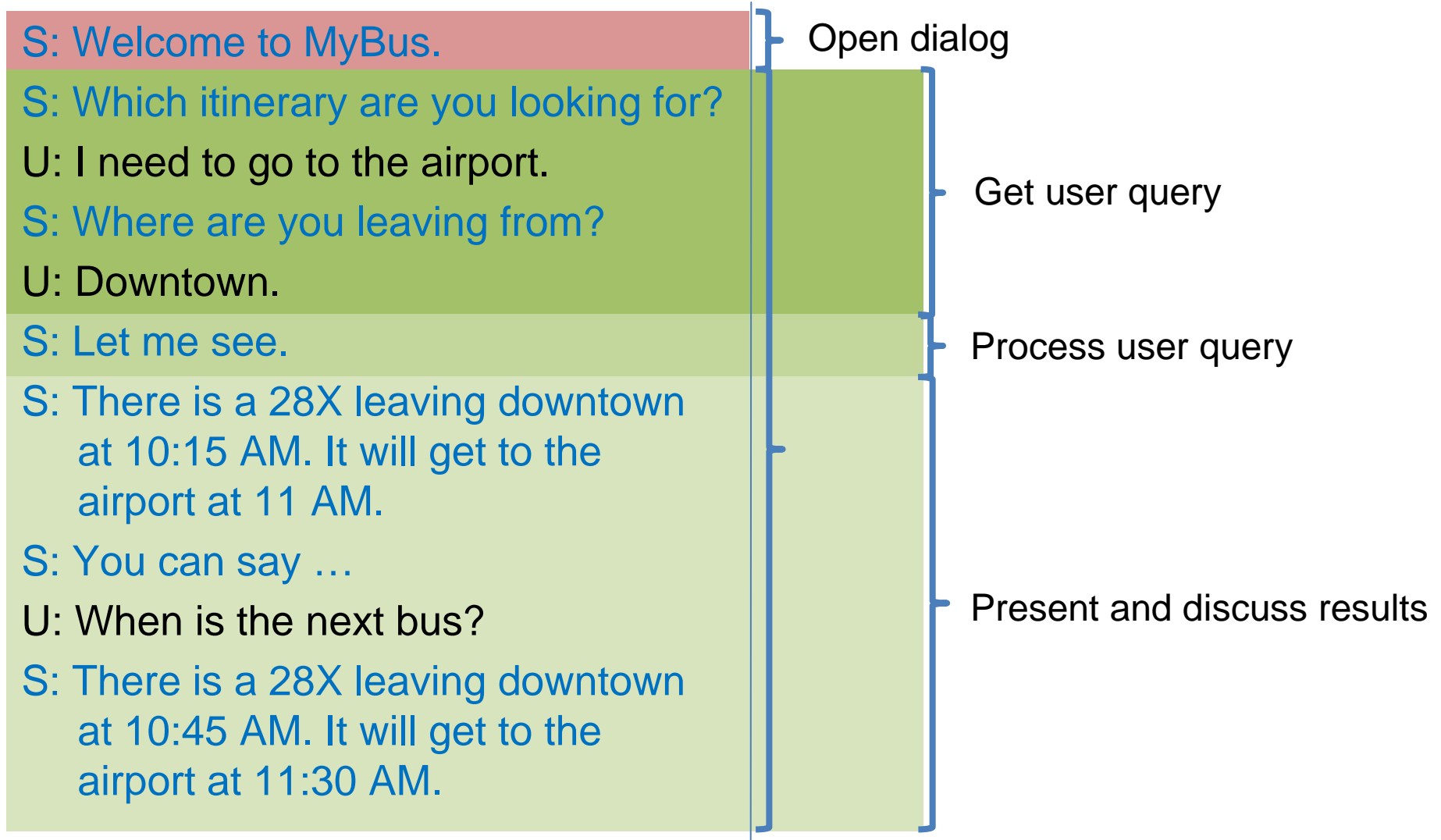
U: When is the next bus?

S: There is a 28X leaving downtown at 10:45 AM. It will get to the airport at 11:30 AM.

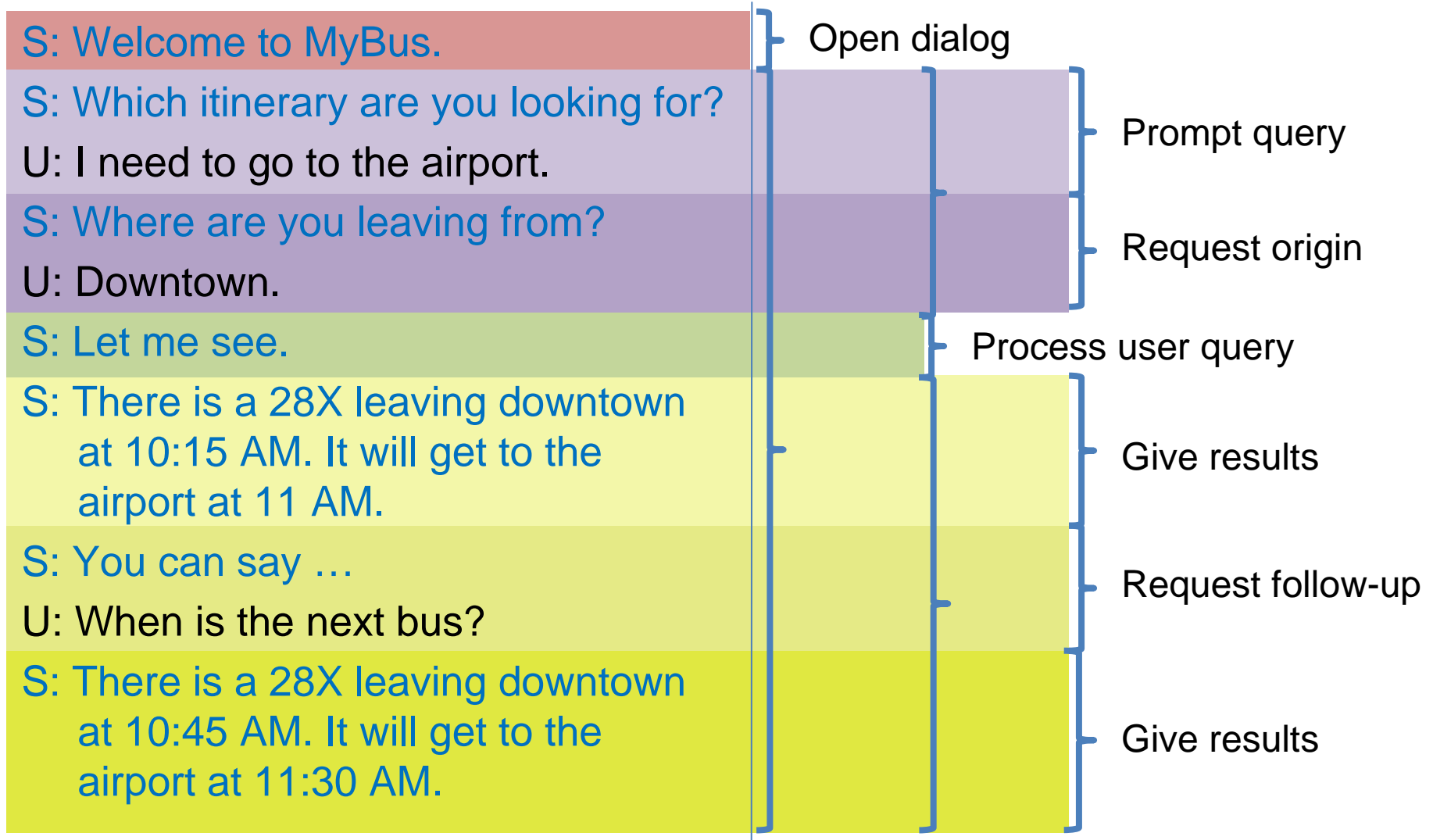
Open dialog

Perform task

# Task Structure (Example 1)



# Task Structure (Example 1)



# Task Structure (Example 2)

