Laurea Specialistica in Informatica
a.a. 2007-2008

Interazione Uomo - Macchina II:

Laboratorio di Interfacce Intelligenti

Fiorella de Rosis

Esercitazione 6

Simulazione di Dialoghi basati su ATN

Tutor esercitazione: Irene Mazzotta

Prerequisiti

Concetti di base sui metodi principali per la simulazione di dialoghi

Materiale

Dispense del corso, Unità 5 e 6 (reperibili sul sito web)

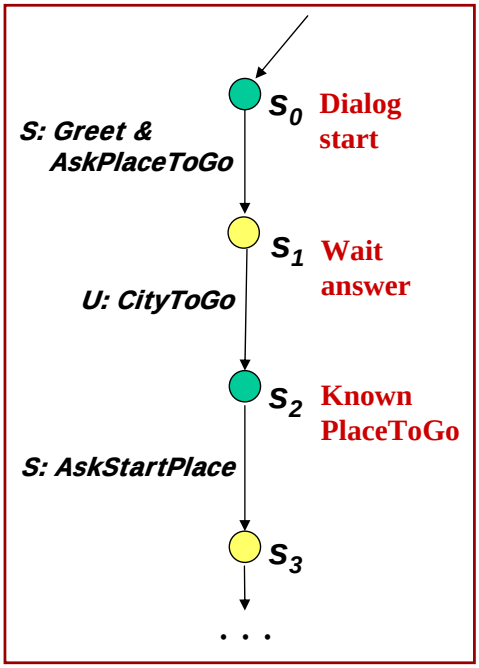
RAD Documentation:
http://www.cslu.ogi.edu/toolkit/docs/2.0/apps/rad/index.html

Obiettivo

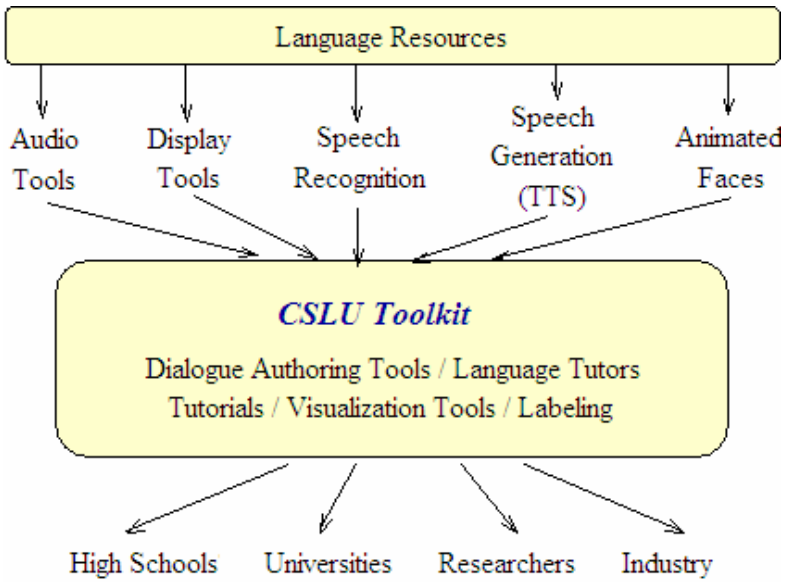
- Definire un semplice modello a stati finiti per la simulazione di un dialogo speech-based.
- Esperienza con RAD (Rapid Application Developer).

Dialoghi come Augmented Transition Networks (ATN)

- Una ATN è un diagramma di transizione fra gli stati.
- Ogni *stato* rappresenta il risultato dell'esecuzione di una mossa di dialogo nello stato precedente. Ad ogni stato è associata una lista di possibili mosse successive.
- La *transizione* fra gli stati è resa possibile dal verificarsi di una serie di condizioni, legate alla mossa eseguita dall'utente e al valore di alcuni registri.



CLSU ToolKit: a Platform for Research and Development of Spoken-Language Systems



Sviluppato da:
Centre for Spoken Language Understanding at the Oregon Graduate Institute of Science and Technology.

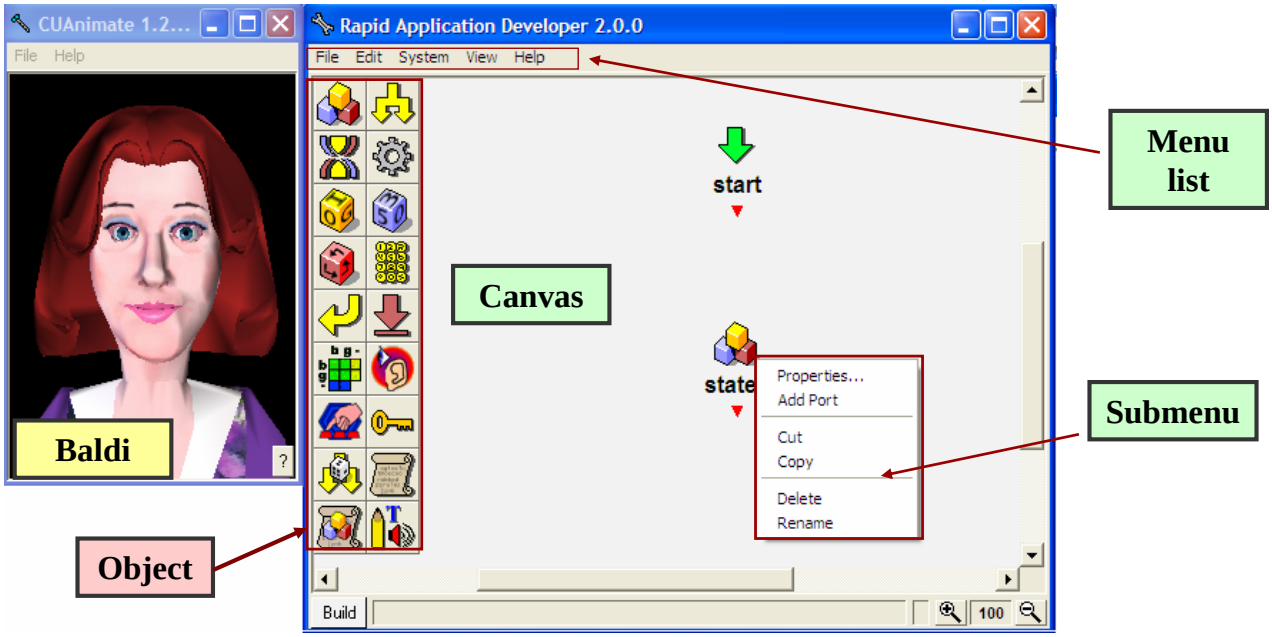
RAD is the dialogue management CLSU tool.

The CSLU RAD

- Rapid Application Developer (RAD) è un ambiente di programmazione visuale (GUI) per lo sviluppo di semplici sistemi di dialogo speech-based.
- Include:
  - TTS system Festival (University of Edimburgh);
  - speaker-independent speech recognition system.
- Supporta:
  - Baldi*, un agente animato (opzionale) .
- Documentazione di riferimento sul CSLU web site: http://www.cslu.ogi.edu/toolkit/docs/2.0/apps/rad/index.html

• Baldi permette di sincronizzare aspetti visivi con aspetti vocali. Si possono configurare molti aspetti grafici da usare nelle applicazioni .rad ma ciò non è oggetto di studio in questo corso.

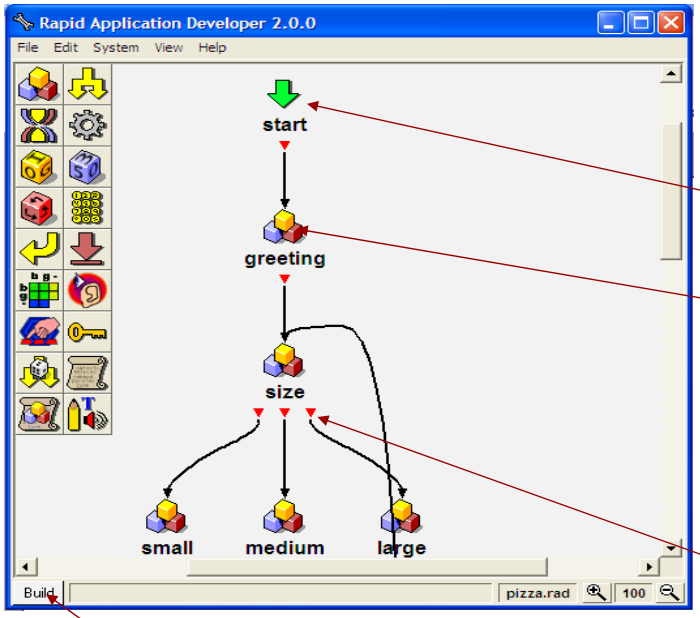
RAD components



Object: elementi per creare l'applicazione
Canvas: workspace per visualizzare la struttura del dialogue system.
Menu List: elenco di item a scelta (create, save, edit) sull'applicazione.
Submenu: elenco di item opzionali sul singolo Object.

RAD: an example

File\Examples\pizza.rad



Per creare la struttura di un dialogue system, trascinare gli oggetti nella canvas e collegarli fra di loro.

Tutte le applicazioni RAD cominciano con uno *Start Object*.

Il *Generic Object* è il più usato per lo Speech Synthesis and lo Speech Recognition. Permette di produrre system speech e di ramificare in base alla risposta dell'utente.

Ogni object ha almeno un *Output Port* (triangolo rosso) che rappresenta la risposta dell'utente.

Ogni applicazione RAD termina on un *Goodbye Object.*

Per eseguire l'applicazione:
Build e *Run* Buttons

Dal modello all'implementazione

Un applicazione RAD rispecchia il comportamento del sistema condizionatamente alle mosse dell'utente:

- **Gli stati, che rappresentiamo mediante *Objects*, appartengono all'insieme degli stati raggiunti dopo una mossa di U (cioè, *Object* ∈ AUM).**
- **La system move è intrinseca allo stato sottoforma di proprietà dell'*Object* (prompt).**
- **La user move è associata all'Output Port dell' *Object* sottoforma di words riconoscibile (vocabulary area).**

Ne consegue che,
essendo possibile associare allo stesso *Object*
sia la system move che le possibili user moves per quello stato,
il passaggio da uno stato all'altro avviene dopo il riconoscimento della userMove.

Pertanto, anche lo stato di *WaitAnswer* è intrinseco all'*Object.*

Riassumendo

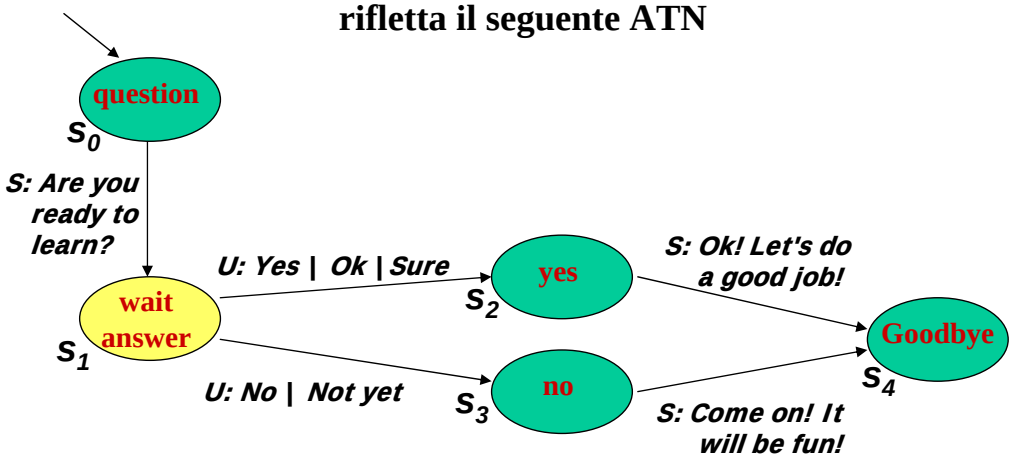
In un applicazione RAD:

- **Gli Objects (Stato) ∈ AUM.**
- **Gli archi sono privi di qualsiasi etichetta e connettono gli Object.**
- **La system move è la proprietà "*prompt*" dell'Object.**
- **La user move è associata all'Output Port dell' Object.**

Eventuali azione che il sistema può compiere, o condizioni che devono essere soddisfatte

A simple dialogue system: la struttura

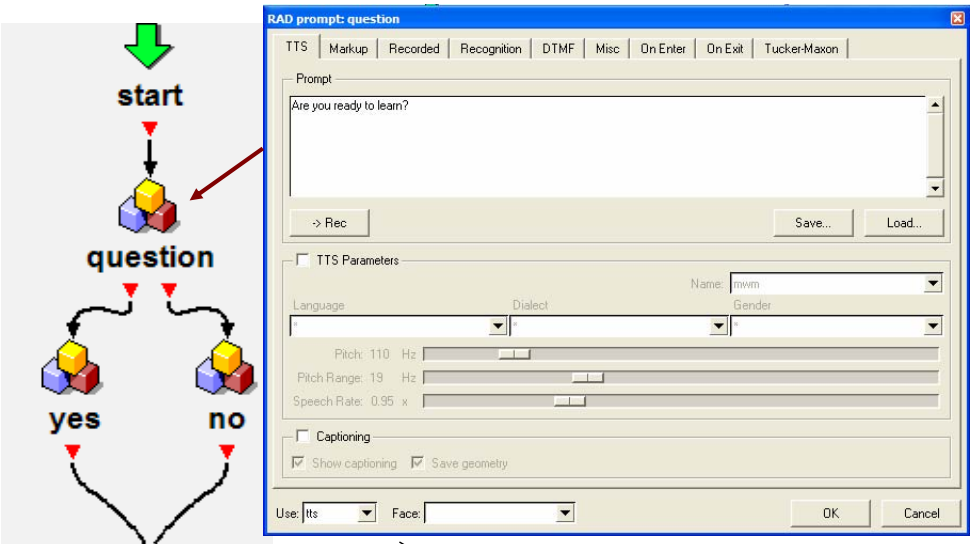
Creare la struttura di un semplice sistema di dialogo che rifletta il seguente ATN



Lineeguida:

- trascinare *3 generic objects* (rispettivamente per gli stati S_0, S_2, S_3) e un *goodbye object* nella Canvas-area ;
- aggiungere un *output port* allo stato S_0;
- connettere i port ai generic object (mediante trascinamento del port sull'object);
- rinominare i generic objects.

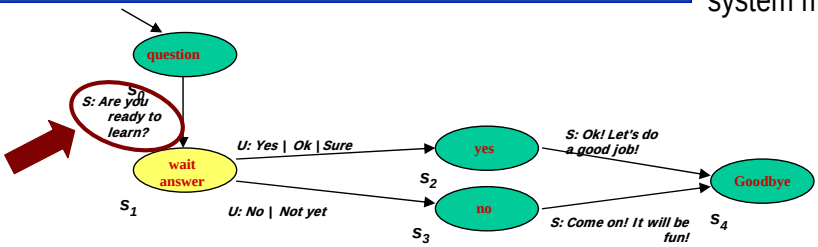
A simple dialogue system: speech synthesis



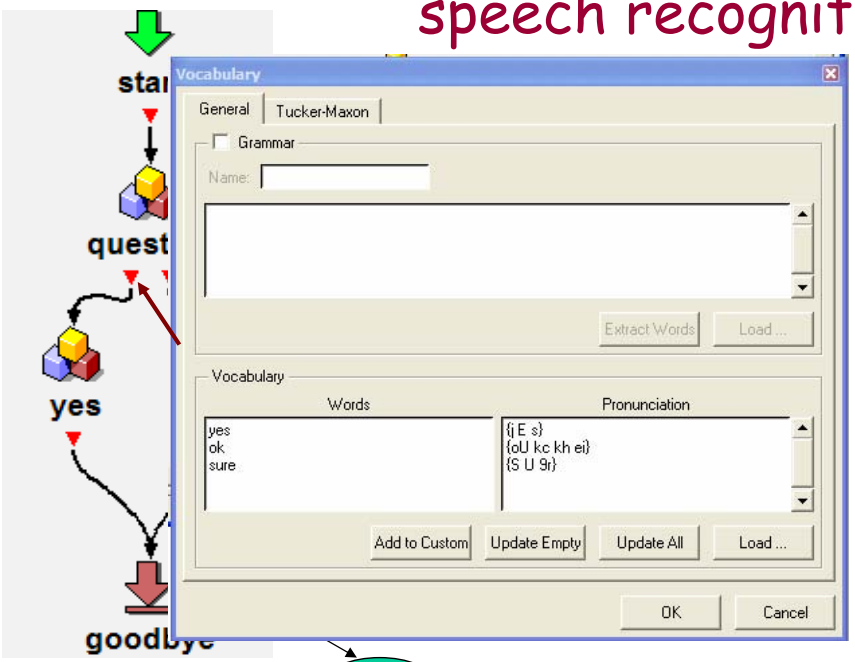
Per associare ai generic objects la corrispondente system move:

- aprire la window *properties* dell'object (tasto destro);
- inserire la system move nella *Prompt area* della *tab TTS*.

Ripetere per tutte le system moves.



A simple dialogue system: speech recognition

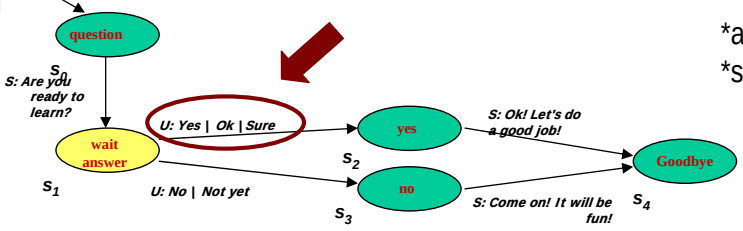


RAD riconosce le sole parole appartenenti ad un lessico precedentemente definito. L'*output port* è il punto di configurazione dello speech recognition. Per associare al port la user move:

- *open* the output port (double click);
- inserire la user move nella *Vocabulary area* della *tab General*.
- *Update All* per produrre la pronuncia nella *Pronounciation area.*

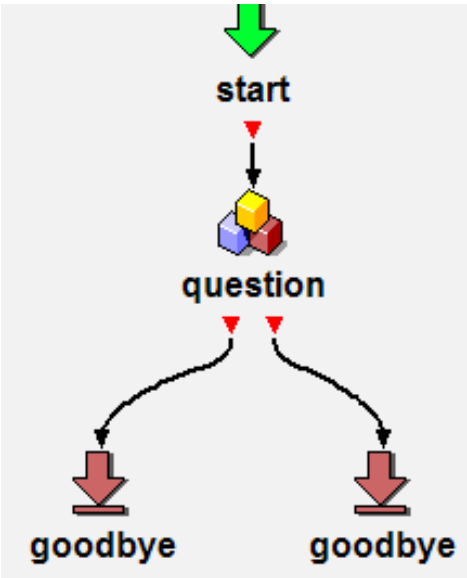
Ripetere adeguatamente per tutte le user moves.

<u>*Special words*</u>:
*any – qualsiasi sequenza
*sil – pausa.



A simple dialogue system: building and running

Lab1.rad



La struttura del dialogo non è unica ma dipende dall'analisi fatta e dall'ATN creato. Un'altra struttura RAD in grado di gestire lo stesso dialogo potrebbe essere la seguente:

Un'occhiata a File\Preferences*:*
RAD permette di settare proprietà globali, valide per l'intera applicazione ma anche proprietà specifiche per il singolo object. Per ulteriori info fare riferimento alla documentazione.

The TCL language

Tcl (Tool Command Language) è un linguaggio di programmazione creato da John K. Ousterhout at the University of California at Berkeley. Tcl è un linguaggio interpretato perciò il code è eseguito line by line.

A differenza del monologo, nel dialogo può essere necessario processare le informazioni ottenute, nonché tenere traccia del contesto del dialogo stesso.

RAD permette di eseguire linee di Tcl code durante il dialogo.
In particolare:

- quando il dialogo comincia (*System\Start of run action*);
- quando il dialogo finisce (*System\End of run action*);
- quando si processa un object (*OnEnter* tab in the preferences dialogue);
- quando si esce da un object (*OnExit* tab in the preferences dialogue);
- in speciali *objects* (es.: *action object*; *conditional object*).

The TCL language: cenni(1)

Approfondimenti riguardo l'uso di TCL per RAD sono disponibili nella documentazione.
Qui ci limitiamo a descrivere i comandi principali.

- Tutte le variabili contengono stringhe di caratteri ASCII. Non esistono i binary data. I numeri sono memorizzati come sequenza di digits. Non c'è differenza tra "123" e 123.
- Il nome delle variabili può contenere caratteri alfanumerici ma deve cominciare con un carattere alfabetico.
- Ci sono due tipi di variabili:
  - le simple variable hanno un solo valore.
    Set command: "set x y"
  - gli associative arrays hanno più valori.
    Set command: "set xarray(start) 10"
    "set xarray(end) 20".

The TCL language: cenni(2)

- Per ottentere il valore di una variabile si usa il "$" prima del nome della variabile stessa:
  - puts $x
  - puts $xarray(start)

  L'espressione "$x" può essere inclusa nel prompt di generic e goodbye objects.
- Supponiamo di avere un object con nome "XXX".
  "$XXX(recog)" si riferisce all'input dell'utente riconosciuto allo stato "XXX".
- Per incrementare il valore di una variabile "x": "incr x".
- TCL supporta conditional statement. Ma lo stesso obiettivo può essere raggiunto usando un conditional object.

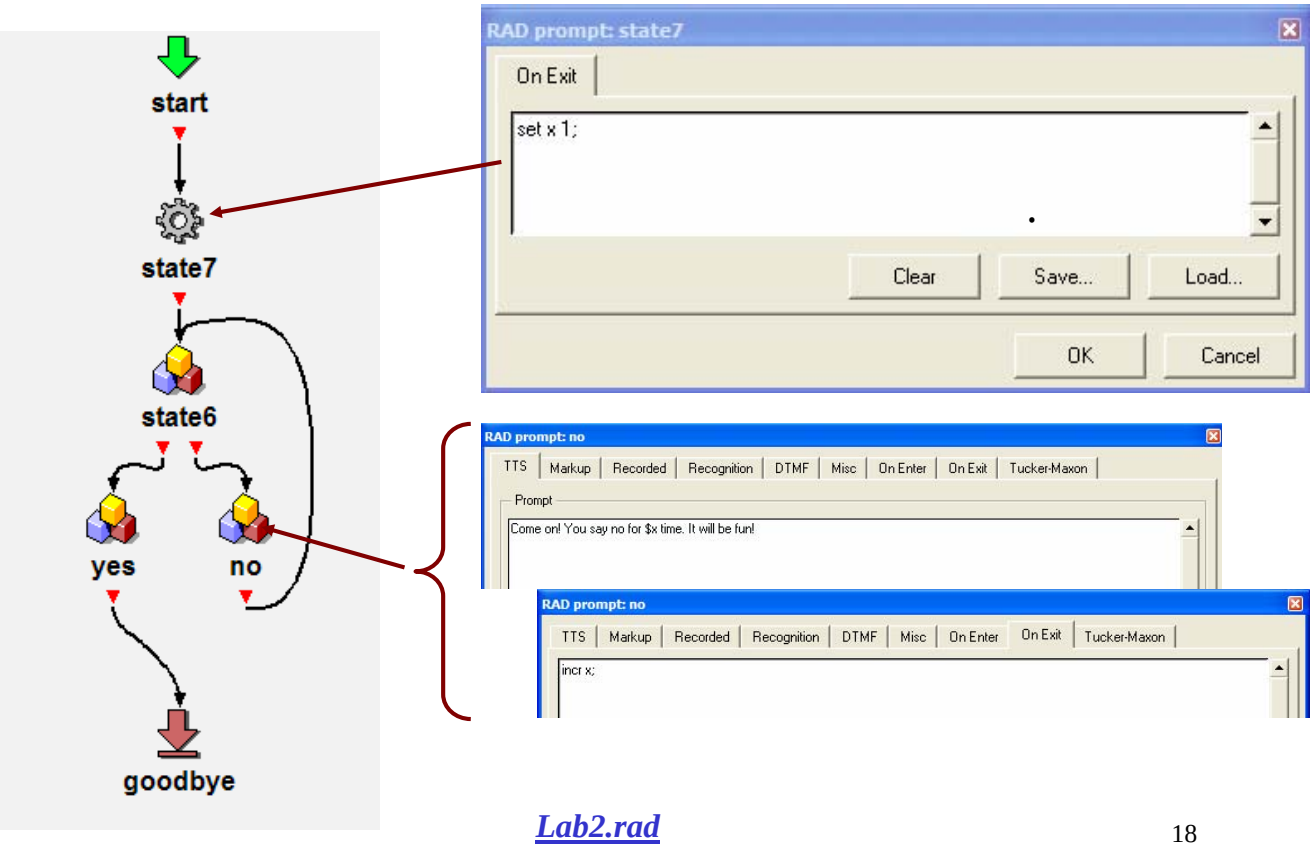
Esercizio (1)

Riprendiamo il nostro semplice sistema di dialogo.
Modifichiamolo in modo da avere un sistema di dialogo che:

- nel caso di risposta negativa, ripete la domanda, indicando il numero di volte in cui l'utente ha risposto "no";
- nel caso di risposta affermativa, termina.

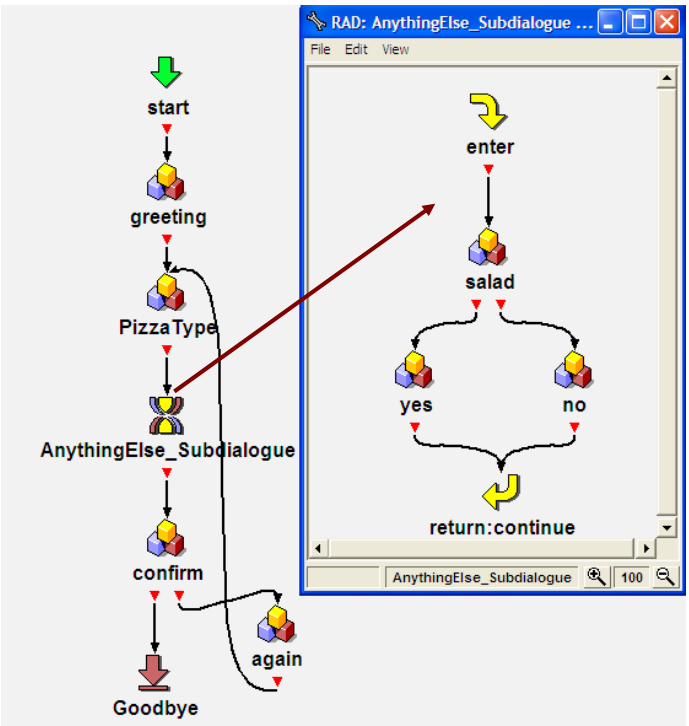
Forse può essere utile ricordare che:
Action object permette di settare facilmente le variabili;
incr x incrementa il valore della variabile x.

Possibile soluzione



Lab2.rad

Subdialogue

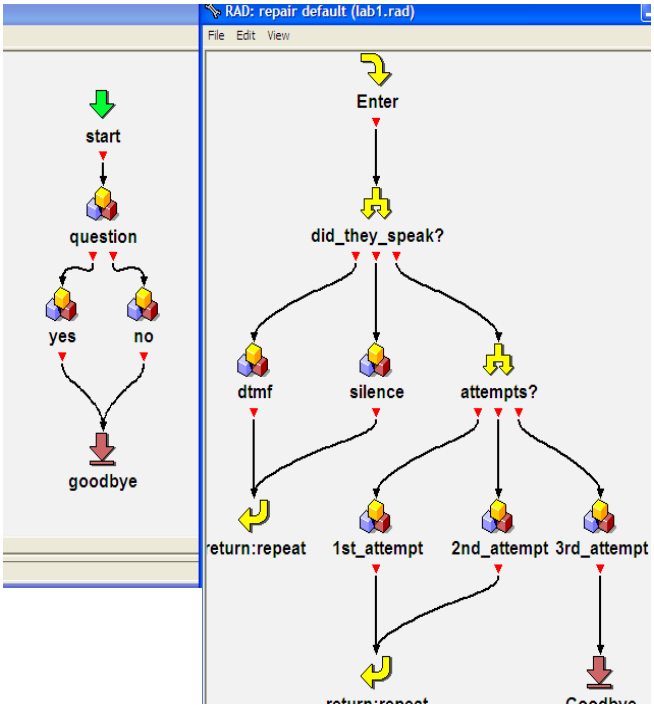


Subdialogues:

- *Mini-RAD canvases.*
- Subroutine che realizzano parte del dialogo che potrà essere ripetuto in diversi momenti.
  Un subdialogue, infatti, viene trattato come un normale object all'interno della canvas principale e può essere ripetuto più volte nella stessa struttura.
- Possono essere salvati ("Save Subdialogue Object" mediante right-click sull'oggetto) e possono essere inseriti in altre applicazioni ("Insert File" mediante right-click nella canvas area).

Lab3.rad

Subdialogue: the repair dialogue



Un repair dialogue è un particolare tipo di subdialogue progettato per riparare ad eventuali fallimenti nel riconoscimento oppure a risposte inaspettate dell'utente.

Viene usato automaticamente dal sistema quindi è già implicitamente incorporato nel sistema.

Default repair in RAD:

- invocato la prima volta risponde "sorry" e ripete la domanda;
- invocato la seconda volta risponde "I still don't understand" e riprova;
- invocato la terza volta rispondo "Sorry I give up" e chiude l'applicazione.

repair dialogue in lab1.rad

Recognition Grammar (1)

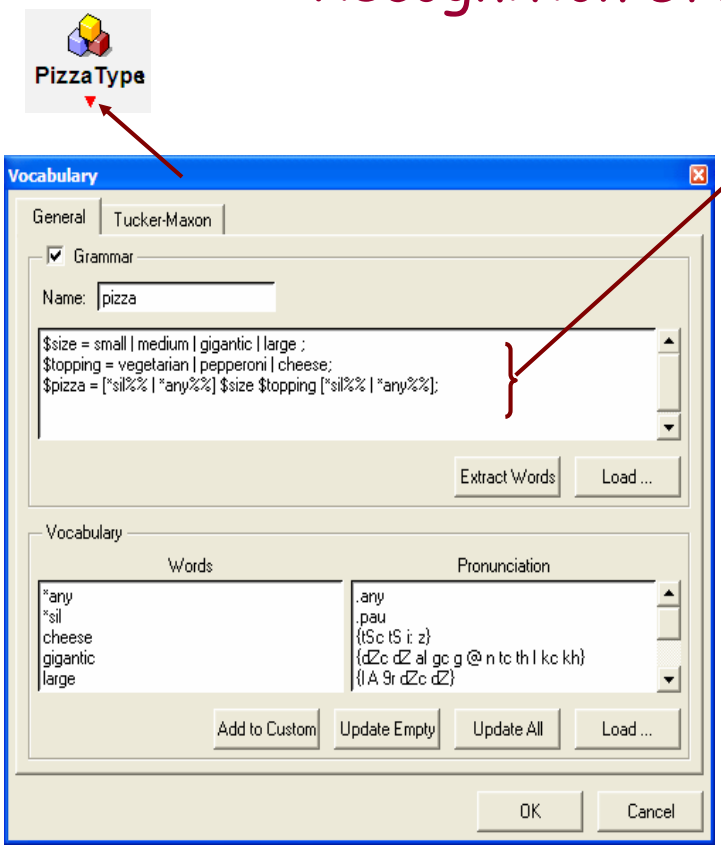
RAD non riconosce solo singole parole o frasi da una lista di scelte predefinite ma permette di definire semplici grammatiche a stati finiti da associare agli *output port.*

Commentiamo insieme il tutorial015.rad.
File\Examples\Tutorials\tutorial015.rad

Il tutorial è molto simile all'esempio iniziale (pizza.rad).
La differenza è nell'associazione, al recognition state, di una grammatica per riconosce il tipo di pizza ordinata.

Ad esempio, l'utente può rispondere con
A small pepperoni
A medium vegetarian
A large cheese.
Queste risposte saranno riconosciute dal sistema

Recognition Grammar (2)

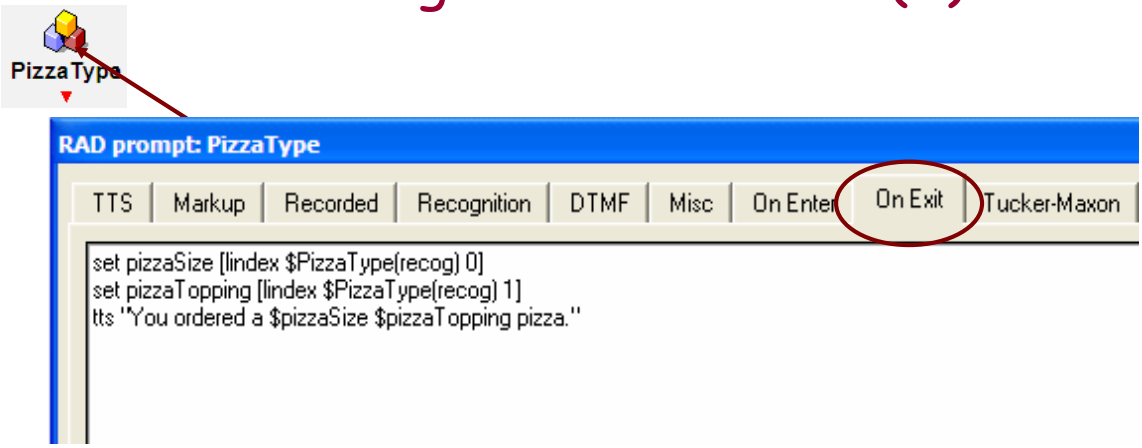


E' stata definita una nuova grammatica chiamata "pizza"
Composta da tre nuove variabili:
- la dimensione della pizza (*size*)
- il tipo di pizza (*topping*)
- la combinazione delle precedenti (*pizza*)

Chr	*Usage*
[]	Parti opzionali
< >	Parti ripetute 1 o più volte
{}	Parti ripetute 0 o più volte
|	or
%%	Dopo una word ad indicare che, anche se riconosciuta, la word non apparirà
%	Dopo una word ad indicare che la word sostituirà la prossima word riconosciuta.

Creata la grammatica, si estraggono i non terminali (*Extract Words* button) e si aggiorna la Pronounciation area (*Update All*).

Recognition Grammar (3)



Tcl code
Se il riconoscimento ha successo, verranno riconosciute due sole parole.
Le prime due linee di codice assegnano le due parole riconosciute a due variabili individuali.
La terza, usa le due variabili per formulare una mossa di conferma da parte del sistema

Tts è una RAD procedure usata nella action areas per forzare lo speech.

Esercizio (2)

Costruire una semplice applicazione Rad in grado di simulare il seguente dialogo:

S: Welcome to Irene's travel agency. I'm here to help you to organize your travels.
Which place do you want to go?
U: London
S: Which place do you start from?
U: Bari
S: What date do you need to do?
U: On June sixth
S: Do you prefer travelling with a regular or a low-cost company?
U: Regular
S: Which tarif do you prefer? Economic or business?
U: Economic
S: ok. Let me just see what sort of fare I can get you on this.
See you later.

Soluzione possibile

Riferimenti

CSL Toolkit - http://www.cslu.ogi.edu/toolkit/index.html