

# Rapid Dialogue Prototyping

- Rapid dialog prototyping is a significant part of the dialog model design.
- It is used to identify and model specificities of human-machine communication in real-life conditions.
- It is an iterative process.
- It must be flexible, allowing us to rapidly experiment with different dialog strategies.
- The result of the dialog prototyping process is a stabilized dialog model represented as a finite state automaton.

# Dialog-repair sub-system

- The objective of the practical session was to design a simple dialog model to interact with the user.
- The interaction is reduced to the processing of the following simple question to the user:

What is your favorite color?

- Interestingly enough, the difficulties arising in such a simple situation are quite representative of the ones occurring in more complex dialog tasks.

# What is your favorite color?

- Yellow!
- Well, I like violet blue purple.
- ahh hem [break] me I like I like rather colors hem hot colors.
- What! I haven't understood, what did you say?
- Sorry! Why you want to know that?
- Yeah, ok, what are the choice?
- Oh! That! But me, I wanted to go to the movies.
- ahhh [break] I never eat oranges!
- humm.

# Rapid dialog prototyping: the CSLU toolkit

The screenshot displays the Rapid Application Developer 2.0b2 interface, which is used for creating dialog-based applications. The main window is divided into several sections:

- System Messages (msg):** A window showing the system's output. It contains two messages:
  - 0 Bonjour! Bienvenue sur Rest'Info, le serveur d'information des restaurants de la ville de Martigny. Vous pouvez faire une recherche sur le type
  - 1 Combien environ comptez-vous dépenser par personne?
- Dialogue Flowchart:** A complex network of nodes and transitions representing the dialog logic. Nodes are labeled with numbers and names, such as 'start', '1\_message\_d'accueil', '2\_analyse\_de\_reponse', '7\_reponse\_incohérente', '8\_reponse\_ok', '9\_hors\_contexte', '10\_reponse\_pas\_comprise', '15\_resultats', '16\_arrêt\_de\_dialogue', '17\_pas\_de\_reponse', '18\_num\_reponses', '20\_plus\_de\_3\_reponses', '24\_avertissement', '26\_information', '25\_reservation', '27\_fin\_de\_dialogue', '29\_état\_final', '5.1\_modification', '18\_reservation', '5.2\_specification', '26\_transitions', '25\_transitions', '28\_continuation', and '29\_transitions'. Transitions are represented by arrows connecting these nodes.
- User Input (usr):** A window showing the user's input. It contains the text 'environ vingt francs'.
- Dialogue Manager Variables (str):** A panel on the right side of the interface. It contains various variables and their values, organized into sections:
  - Actual response:** Time: midday, Foodtype: rapide, Date: monday, Price: (empty), Localization: (empty).
  - Internal variable values:** A list of variables and their values: cIndicationChamps: 0, cIndicationValeurs: 0, cIncohérente: 0, cOK: 1, cHorsContexte: 0, cIntelligible: 0, dialogue context: Price.
  - Database search values:** hor5,hor8,hor11,hor14,hor17,hor20;cui9,cui10;0;.
  - Database results:** (empty).
  - Input mode:** keyboard (selected), recording, telephone, file.
  - Run mode:** run (selected), step.
  - Database mode:** www, local (selected).
  - Output mode:** none (selected), log file.

# Demo 1:

## Simple model

- **Features:** User's answers are categorized into 5 groups: OK, Repeat, Explain, Out-of-scope, Misunderstood.
- For each category, the system carry out some predefined action (message) in order to help the user.
- **Limitation:** There is no account for dialog history; in a more sophisticated approach, the system should be able to adapt its behavior according to the user mental representation.

## Demo 2:

### Enhanced model, improvement:

- Bring variability to the dialog control.
- If the same repair-state is consecutively reached twice, reformulation is used in order to bring some new information to the user.
- User's responses are not limited to strict commands – a garbage grammar model is used.
- Dialog repair is integrated as a subdialog in order to become a reusable component and to overcome some of the weaknesses of the finite-state model.

# Conclusion

- Designing an efficient dialogue prototyping tool
- Architecture: finite state automaton
- Implementing this model in a example
- Learning to handle the typical problems in such a dialogue model