UiO : University of Oslo

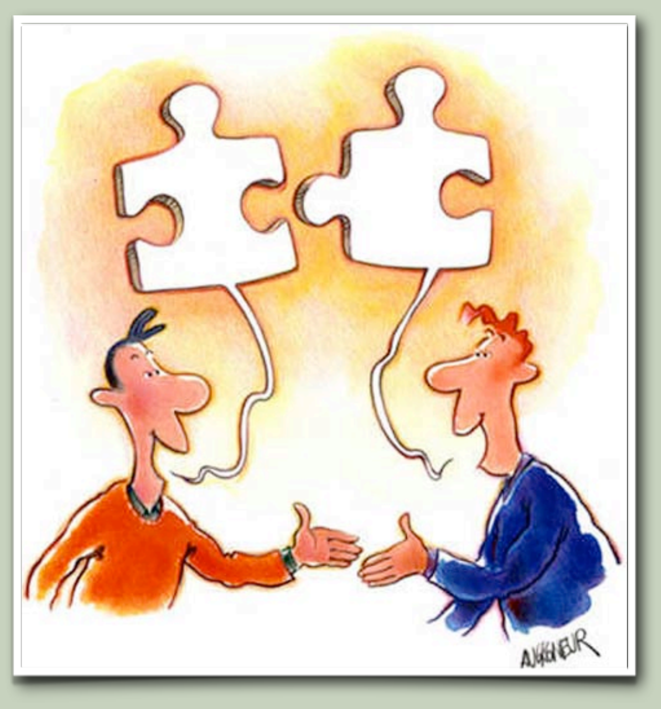
Dialogue management using a mixture of designed and learned policies

Pierre Lison

Logic & natural language group, Department of Informatics
University of Oslo, Norway.

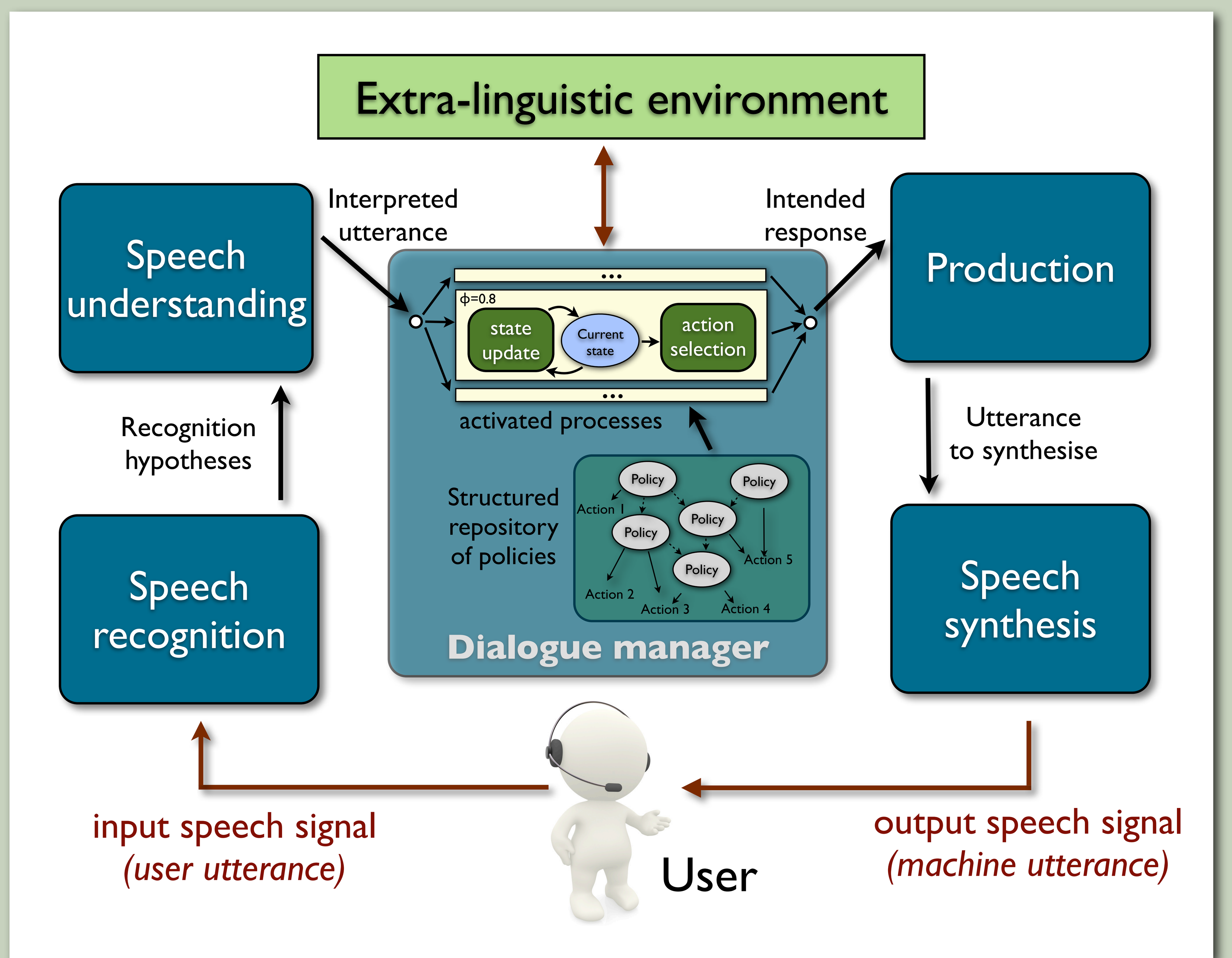
Introduction

- Natural language is a very intuitive means of interaction for humans. Can we build computer systems capable of conversing with humans using this communication medium?
- Key motivation: computers should *adapt* to their users (and speak their language) rather than the other way round!
- Such systems are called **spoken dialogue systems** (SDS for short)
- SDS are expected to play an ever-increasing role in our interactions with technology. Possible applications include, *inter alia*:
  - Phone-based information access and service delivery (e.g. travel booking, public transport information);
  - Speech-enabled software interfaces;
  - Intelligent tutoring systems for education;
  - Service robots in homes, schools, offices and hospitals;
  - Cognitive assistants and caretakers for the elderly.
- We are more specifically interested in **dialogue management**, which is the part of dialogue systems concerned with *decision-making*



- Role of dialogue management: *decide* what to do/say at each point of the interaction, given a collection of goals and a set of observations (interaction history, external context)
- Dialogue management is thus responsible for dynamically controlling the *flow* of the interaction.
- It is a *hard* problem! SDS must typically operate with:
  - multiple, mutually competing objectives (trade-offs);
  - numerous social/pragmatic conversational constraints;
  - high levels of uncertainty (due to e.g. error-prone speech recognition, partial knowledge of the environment, etc.).
- SDS should also be able to *adapt* their conversational behaviour depending on user- and context-specific factors
- Two main paradigms in the literature on dialogue management:
  - **design-based** approaches rely on handcrafted dialogue policies
  - **optimisation-based** approaches automatically extract optimal policies from experience using reinforcement learning

Approach



- Depending on the particular problem at hand, different types of policies might be appropriate
- We developed a dialogue management algorithm operating with *multiple, interconnected policies*
- Policies are defined in a modular way, resulting in a *mixture* of different (designed or learned) policies
- Policies are combined **hierarchically** and **concurrently**:
  - *Hierarchical combination*: one policy is able to call another by executing an *abstract action*
  - *Concurrent combination*: several policies are executed in parallel upon receiving a new observation

Algorithm 1 Multi-policy execution

let o be a new observation
initialise a^* with void value and $U(a^*) = -\infty$

for all $p \in \mathcal{P}$ **do**
  $s'_p \leftarrow \text{update}_p(s_p, o)$
  $a^*_p \leftarrow \text{argmax}_{a_p} Q_p(s'_p, a_p)$
  $U(a^*_p) \leftarrow \phi_p \cdot Q_p(s'_p, a^*_p)$

  if $U(a^*_p) > U(a^*)$ **then**
    if a^*_p is an abstract action **then**
      fork new process q with policy from a^*_p
      $\text{parents}(q) \leftarrow \{p\} \cup \text{parents}(p)$
      $\phi_q \leftarrow \phi_p$
      add q to $\mathcal{P}$
    else
      $a^* \leftarrow a^*_p$
      $\text{trace}(a^*) \leftarrow \{p\} \cup \text{parents}(p)$
    end if
  end if
end for

redistribute activation values $\{\phi_p : \phi_p \in \mathcal{P}\}$
prune processes with $\phi_p < \epsilon_{min}$ and w/o children
return a^*

- Dialogue policies are decomposed in two components:
  - **State update** takes as input the current state s and a new observation o, and outputs the update state s'
  - **Action selection** takes the updated state s' as input, outputs the optimal action a* to execute (if any)

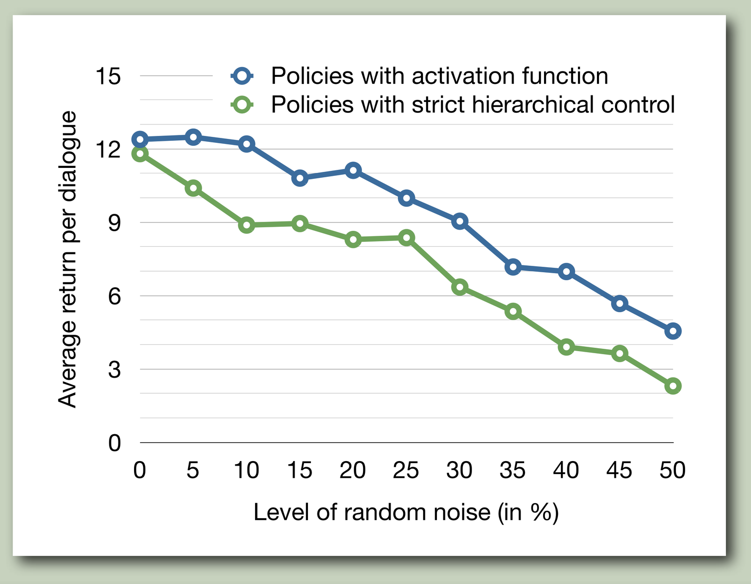
Comments on Algorithm 1:

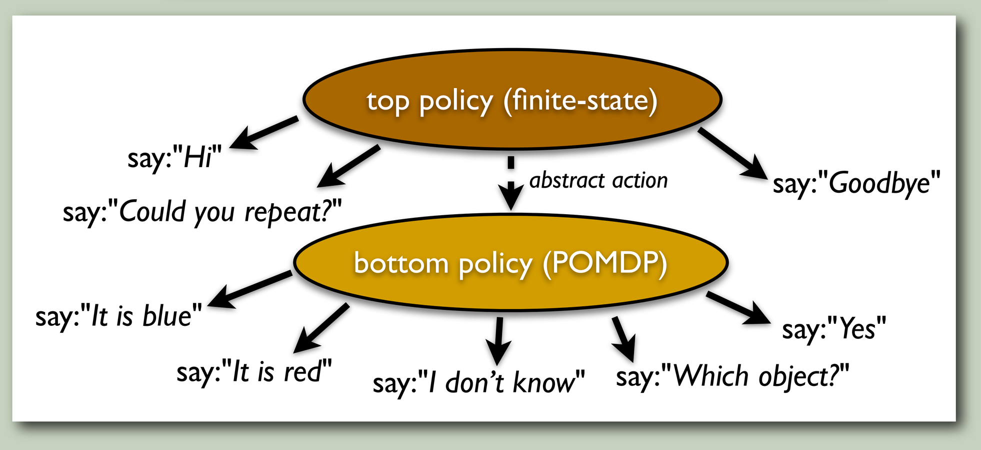
- $\mathcal{P}$: set of concurrent processes, with each $p \in \mathcal{P}$ associated with a policy, a state s_p and activation value ϕ_p
- $Q_p(s,a)$: local utility of executing action a in state s according to p
- Algorithm 1 searches for the optimal action a^* over the processes in $\mathcal{P}$
- *Abstract action* = action pointing to a policy instead of a concrete action
- For details on the redistribution of the activation mass, simply ask me!

- Key idea: associate an **activation value** ϕ to each policy
- The ϕ values indicate the current "*focus of attention*"(i.e. which part of the interaction model is most salient)
- The ϕ values are redistributed after each turn
- Allows for "*soft*" control of multi-policy execution

Evaluation

- Experiment with a small dialogue domain: (simulated) visual learning task between a human and a robot in a shared scene
  - The scene includes objects with properties such as color or shape
  - The human asks questions related to these properties, and then confirms or corrects the robot answers
  - Uncertainty in the linguistic inputs and in the visual perception
- Domain modelled with 2 interconnected policies:
  - Top policy (finite-state controller) handles the general interaction
  - Bottom policy (POMDP) answers the object-related queries
- Goal of experiment: compare the performance of Algorithm 1 with a hierarchical control mechanism (top policy blocked until bottom releases its turn), using a handcrafted user simulator and various levels of noise
- Results demonstrate that activation values are beneficial for dialogue management with multiple policies, esp. in presence of noise





Conclusion

- We introduced a new approach to dialogue management based on **multiple, interconnected policies** weighted by *activation values*
- Activation values are updated after each turn to reflect which part of the interaction is in focus
- Future work will focus on:
  - formalising the *activation mass redistribution*
  - introducing a *shared state* for all policies
  - applying *reinforcement learning* to learn model parameters on multiple policies

Want to know more? Check out my papers at:
http://folk.uio.no/plison
Or email me at: plison@ifi.uio.no