CLASSiC

D1.3: POMDP Learning for ISU Dialogue Management

Paul Crook, James Henderson, Oliver Lemon, Xingkun Liu

Distribution: Public

CLASSiC
Computational Learning in Adaptive Systems for Spoken Conversation
216594 Deliverable 1.3

February 2010

Project funded by the European Community under the Seventh Framework Programme for Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	216594
Project acronym	CLASSiC
Project full title	Computational Learning in Adaptive Systems for Spoken Conversation
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2008 / 36 Months

Security	Public
Contractual date of delivery	M24 = February 2010
Actual date of delivery	February 2010
Deliverable number	1.3
Deliverable title	D1.3: POMDP Learning for ISU Dialogue Management
Type	Prototype
Status & version	Final 1.0
Number of pages	15 (excluding front matter)
Contributing WP	1
WP/Task responsible	Heriot-Watt University
Other contributors	University of Geneva, Edinburgh University
Author(s)	Paul Crook, James Henderson, Oliver Lemon, Xingkun Liu
EC Project Officer	Philippe Gelin
Keywords	POMDP, ISU, Dialogue Management

The partners in CLASSiC are:		
	Heriot Watt University	HWU
	University of Edinburgh HCRC	EDIN
	University of Cambridge	UCAM
	University of Geneva	GENE
	Ecole Superieure d'Electricite	SUPELEC
	France Telecom/ Orange Labs	FT

For copies of reports, updates on project activities and other CLASSiC -related information, contact:

The CLASSiC Project Co-ordinator
Dr. Oliver Lemon
Heriot Watt University
School of Mathematical and Computer Sciences (MACS)
Heriot Watt University
Edinburgh , EH14 4AS
United Kingdom
o.lemon@hwac.uk
Phone +44 (131) 451 3782 - Fax +44-131-451-3327

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.classic-project.org`

Contents

Executive Summary

This document is a short report to accompany the Prototype Deliverable D1.3, due at month 24 of the CLASSiC project. This prototype is a enhancement of the DIPPER Information State Update Dialogue Manager with POMDP planning capabilities.

This work is the result of a collaboration between Heriot-Watt, Edinburgh University, and the University of Geneva.

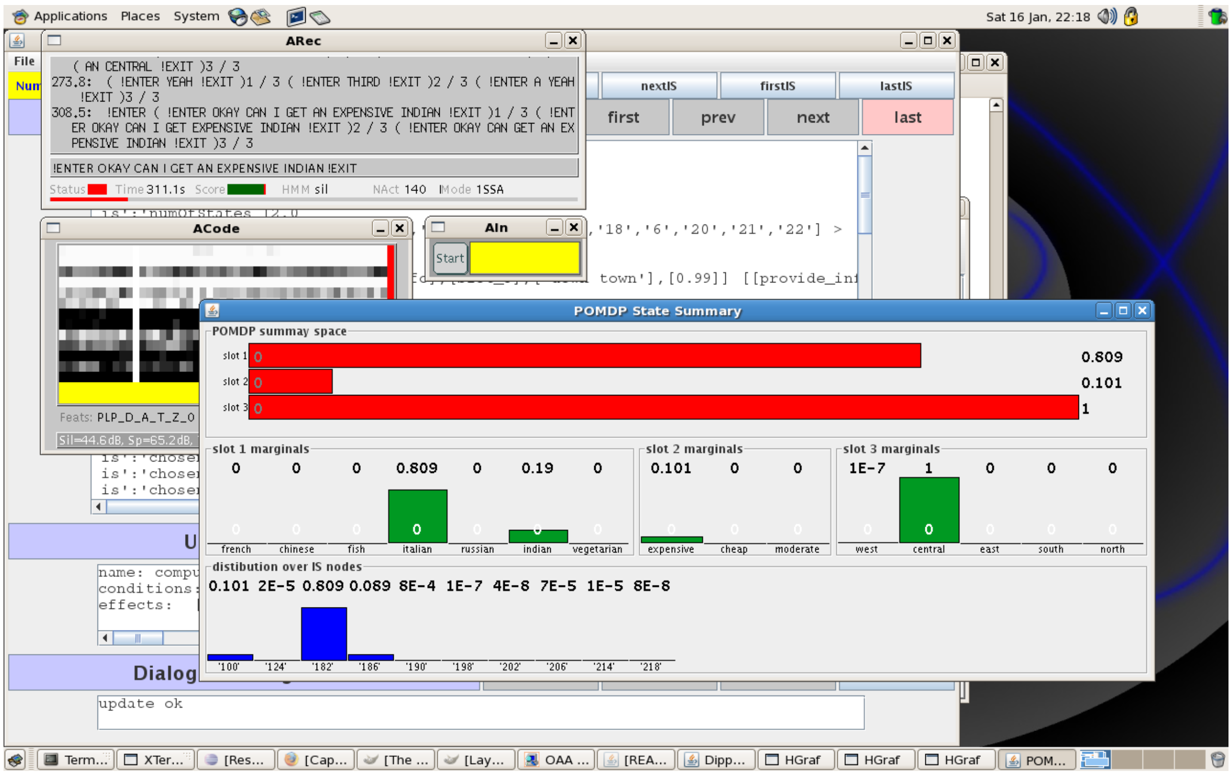
Early evaluation results for the work presented here were presented in CLASSiC project Deliverable D6.3 (June 2009). Integration with new generalization methods is presented in CLASSiC Deliverable D1.2 (October 2009).

Elements of this work have been published as [1, 2], and several publications are in preparation [3, 4].

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a formal framework for making decisions under uncertainty. Recent research in spoken dialogue systems (SDS) has used POMDPs for dialogue management [5, 6]. These systems represent the uncertainty about the dialogue history using a probability distribution over dialogue states, known as the POMDP's belief state, and they use approximate POMDP inference procedures to make dialogue management decisions.

This deliverable presents a framework which allows an upgrade route from existing Information State Update (ISU) SDS which use MDPs, to POMDP SDS using the MM-POMDP approach as proposed by [1]. The resulting system represents uncertainty about the dialogue history using a probability distribution over dialogue states, as shown graphically in Figure 1. Here we see the central window "POMDP State Summary" representing a probability distribution where the system believes that the user wants a central italian restaurant (the green bars in Figure 1), but where there is some uncertainty about the food type and price-range: there is some probability that the user wants indian food, and a small probability that they are looking for an expensive restaurant.



```
   ( AN CENTRAL !EXIT )3 / 3
273.8:  ( !ENTER YEAH !EXIT )1 / 3 ( !ENTER THIRD !EXIT )2 / 3 ( !ENTER A YEAH
    !EXIT )3 / 3
308.5:  !ENTER ( !ENTER OKAY CAN I GET AN EXPENSIVE INDIAN !EXIT )1 / 3 ( !ENT
   ER OKAY CAN I GET EXPENSIVE INDIAN !EXIT )2 / 3 ( !ENTER OKAY CAN GET AN EX
   PENSIVE INDIAN !EXIT )3 / 3
```

```
!ENTER OKAY CAN I GET AN EXPENSIVE INDIAN !EXIT
Status  Time 311.1s  Score  HMM sil  NAct 140  IMode 1SSA
```

Figure 1: POMDP - DIPPER probability distribution over dialogue states (IS nodes) and slot-values at runtime

1.1 ISU Dialogue Systems

The "Information State Update" (ISU) approach to dialogue [7, 8] employs rich representations of dialogue context for flexible dialogue management.

> "The term *Information State* of a dialogue represents the information necessary to distinguish it from other dialogues, representing the cumulative additions from previous actions in the dialogue, and motivating future action" [8].

Technically, Information States represent dialogue context as a large set of features, *e.g.* speech acts, tasks, filled information slots (*e.g.* destination=Paris), confirmed information slots, speech recognition confidence scores, etc. Update rules then formalise the ways that information states or contexts change as the dialogue progresses. Each rule consists of a set of *applicability conditions* and a set of *effects*. The applicability conditions specify aspects of the Information State that must be present for the rule to be appropriate. Effects are changes that are made to the information state when the rule has been applied. For full details see [8, 9].

Within this report reference is made to various "DIPPER" systems. The DIPPER framework [9] is a general framework for building ISU spoken dialogue systems. Using this framework three specific instances of DIPPER systems were implemented. These specific instances are referred to as DIPPER-MDP, DIPPER-QMDP and DIPPER-POMDP. Section 3 gives details of these three systems.

1.2 DIPPER-MDP, DIPPER-QMDP

Previous work has demonstrated effective learning of MDP dialogue management policies using reinforcement learning (RL) within the ISU (DIPPER) framework. [10] demonstrated the DIPPER-MDP system learning a MDP policy to perform a slot filling task in fewer moves that a comparable hand coded policy. This work has then been extended by creating a DIPPER-QMDP system using the mixture model POMDP (MM-POMDP) framework [1] to maintain a representation of alternative dialogue states and then plan across these using the QMDP approximation proposed by [11].

The work presented in this report further extends that presented by [1] in that it demonstrates learning of *POMDP policies* within the MM-POMDP framework. [1] approximates the POMDP planning process by using a MDP policy (previously learnt by DIPPER-MDP) which is consulted for *each* possible state of the dialogue and then a weighted vote is taken across the set actions which have been suggested. In this report the set of possible states of the dialogue is considered *in its totality* (as represented by the POMDP belief state) and a single action is chosen that is optimal for the current belief distribution.

1.3 Partition based POMDP SDSs

A significant body of recent work on POMDP SDSs can be grouped under the heading of "partition based" systems [6, 12, 13]. Such systems are distinguished by the grouping of states into *partitions* such that the POMDP belief distribution can be represented by a minimum number of partitions, with all those states contained within one partition sharing the same probability mass[1]. This method, as outlined by [12], minimises the computational effort required for each POMDP update such that the system can be scaled to real application domains. To further minimise computational effort this approach normally starts with a single partition containing all states, which is subsequently split as observations are made.

[1]Each state's share being determined by its prior.

The work presented in this report takes an alternative approach to minimise the computational effort required and thus is also scalable to real application domains. By representing the POMDP transition update as deterministic updates, see [1, 4], and starting with all the probability mass concentrated in a single state, we can constrain the number of dialogue states that need to be considered at each time step to a small handful out of the potentially infinite number of states that could exist. Since our dialogue states are ISU states, see Section 1.1 above, the set of possible dialogue states that this approach can represent is as rich as partition based methods.

1.4 Factorisation of Dialogue State Space

In POMDP approaches to SDSs the POMDP belief space is commonly factorised [14, 15, 12, 13], something that is also true in this work.

Factorisation is employed to exploit structure of the domain, *e.g.* known or assumed conditional independence relations, and thus obtain computationally efficient update algorithms.

Currently one of the most dominant factorisations [12, 13] in the field of POMDP SDSs is that set out originally by [15] where the state is factored into the *long-term user goal* s_u, *user action* a_u and *dialogue history* s_d – see Figure 2. In this factorisation the *user action model* which predicts a'_u is conditionally independent of the dialogue history. This approach allows relatively simple "user models" which lend themselves to being hand-crafted. The disadvantage is that the simplifying assumptions commonly used (*e.g.* the user's goal remains constant during a dialogue) can reduce the ability of such models to reproduce the full complexity found in the data. In contrast, the factorisation used in this report does not assume independence between the user action model and the dialogue history, see Figure 3.

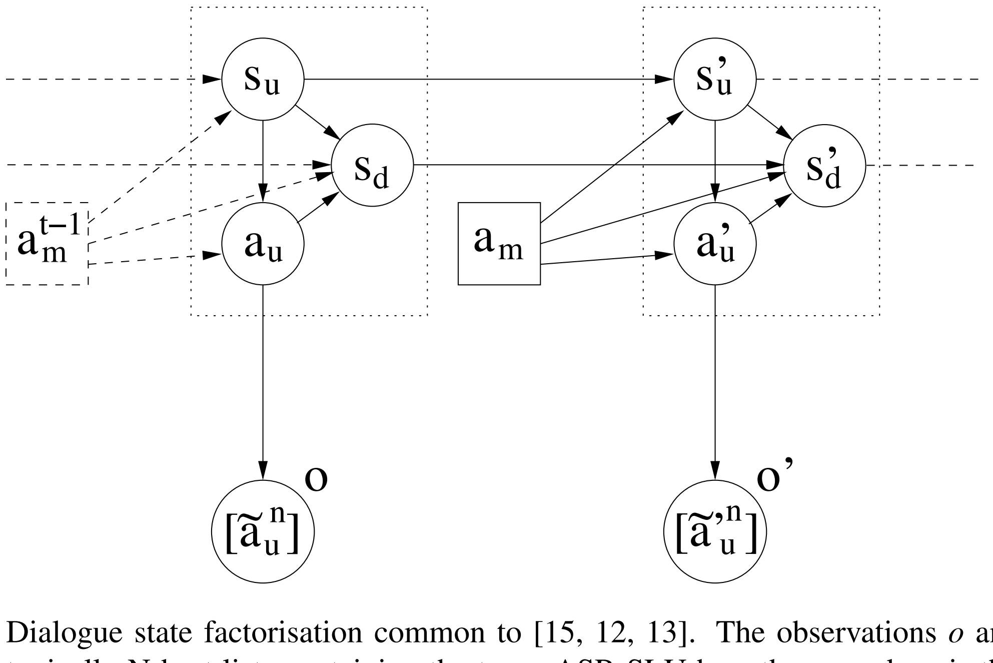


Figure 2: Dialogue state factorisation common to [15, 12, 13]. The observations o and o' are typically N-best lists containing the top n ASR-SLU hypotheses and a_m is the action selected by the SDS.

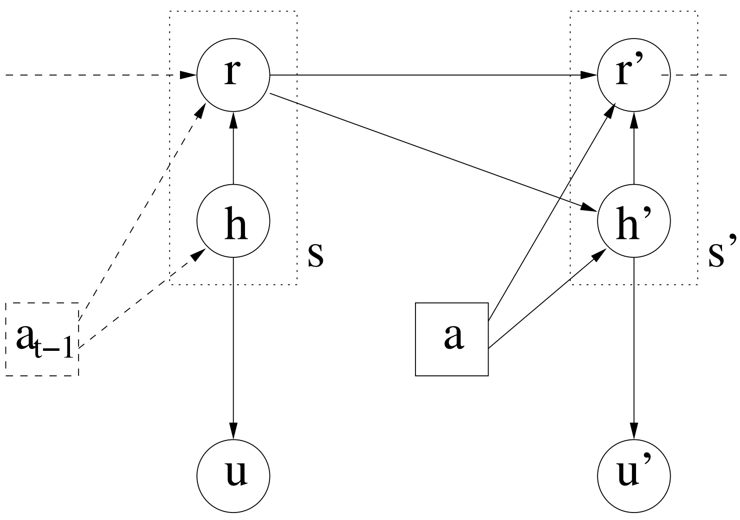


Figure 3: DIPPER-QMDP and DIPPER-POMDP factorised belief.

In our approach the *user model* is potentially more complex and hence more difficult to engineer by hand. However, our approach is amenable to building data driven models. Given either a sufficiently large dialogue corpus or a framework to generalise from smaller corpora (see Section 3.2), the user transition model required by our factorisation can directly use previous spoken dialogue corpora to predict the likelihood of user actions. Using corpus data directly should allow for accurate prediction of real user actions.

The ISU POMDP approach set out here is also maintaining uncertainty over a different informational view of the dialogue compared to [15, 12, 13]. Where [15, 12, 13] primarily try to reason (in a probabilistic sense) about the unknown variable which is the user's goal, our approach is reasoning over the uncertainty in the *shared* information in the dialogue. Hence our approach starts knowing we are in the dialogue state of knowing nothing about the users requirements, while [15, 12, 13] typically starts knowing that the dialogue state is in an initial state and the user's goal variable is a distribution over all possible user goals.

1.5 Policy Optimisation

Given a complete model of observation and transition dynamics an exact solution to a POMDP, *i.e.* the optimal policy, can be computed *off-line* [16]. However such approaches tend not to scale well due to computational intractability. Even approximate off-line algorithms for solving POMDPs, such as Perseus [17], can only be applied to "toy" SDS problems [15].

In this work we use belief space compression (also referred to as the *summary POMDP method*) as proposed by [18] to constrain the size of the state and action space over which the policy has to be learnt. We also adopt an *on-line* approach training the policy over many thousands of dialogue using a *simulated user* with user goals [19, 20, 2]. This is similar to the approach used by [12]. However, our basic system uses temporal difference updates (using a hierarchical form of SARSA(λ) [21]), and a further version of the DIPPER-POMDP system has been developed which uses new RL generalisation techniques (such as Kalman filtered RL), as described in Deliverable D1.2.

2 Application Domain

The application domain used in this work is that of a "slot filling" SDS. Specifically we use a restaurant search service for a fictional town. Users are expected to fill three slots; food type, location and price range. When the dialogue manager considers that it has sufficient confidence in the slot values that it has perceived it then presents the user with a restaurant recommendation that matches the specified constraints.

An illustrative dialogue is given below:

System: Welcome to the restaurant system. How may I help you today?
User: I'm looking for an inexpensive French restaurant in the northern area of the city.
System: Okay, French food. Which area of the city would you like, for example central or west?
User: North please.
System: Did you say expensive?
User: No, cheap.
System: Did you say cheap?
User: Yes.
System: Okay you are looking for a cheap French restaurant in the north of the city. Only one result matches your criteria, it is ...

3 DIPPER Spoken Dialogue Systems

This work uses three DIPPER based SDSs. This report focus on the performance of the DIPPER-POMDP system. Two other systems, the DIPPER-QMDP and DIPPER-MDP systems, were used to provide a comparative baseline against which a full POMDP based system can be measured (see D6.3). Both DIPPER-QMDP and DIPPER-MDP systems have been described previously by [1] and [10] respectively. For completeness the essential details of all three systems are set out here.

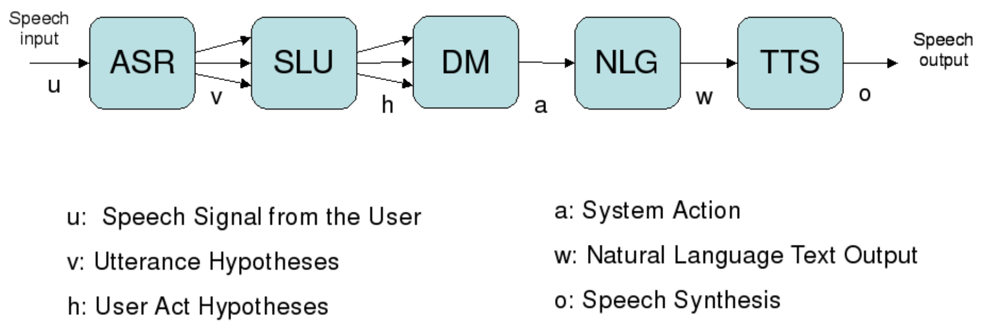


Figure 4: Outline of DIPPER systems

All three systems follow the basic outline shown in Figure 4. They have many components in common including:

- Automatic Speech Recogniser (ASR): ATK/HTK [22].
- Spoken Language Understanding (SLU): two parsers were used. The grammar-based Grammatical Framework (GF) Parser [23] and a statistical parser [24]. The GF parser is used preferentially with

the system falling back to the statistical parser should GF be unable to provide a parse.

- Dialogue Manager: DUDE (rule based system) coupled either with *(i)* REALL (a platform for hierarchical reinforcement learning) [25] or *(ii)* the new Reinforcement Learning libraries presented in D1.2.
- Text To Speech (TTS): CereProc [26].

The differences between the systems are; (i) the number of ASR/SLU hypotheses that are considered by each system, (ii) how each system internally represents the state of the dialogue and (iii) the policies followed by each system. These differences are summarised in Table 1 with more details given below.

System	**ASR/SLU hypotheses used**	**Dialogue state representation**	**Policy**
DIPPER-POMDP	top N	MM-POMDP (tree) and summary space	learnt policy based on summary space
DIPPER-QMDP	top N	MM-POMDP (tree)	weighted sum of MDP policy over all leaves
DIPPER-MDP	top 1	single state history	learnt MDP policy (also several hand coded policies)

Table 1: Main differences between systems

In [1] the general form of a MM-POMDP belief distribution is defined as set of probability distributions associated with the MDP dialogue states. In DIPPER-QMDP and DIPPER-POMDP each dialogue state distribution is taken as a single point mass, that is to say there is a single real value p_i in the range 0 to 1 which is associated with each dialogue state r_i (at time t). This results in the update form as presented by [1], *i.e.*:

$$p'_{i'} = \zeta \frac{1}{P(h')} P(h' \mid u')\, P(h' \mid r_i, a)\, p_i \tag{1}$$

Where $p'_{i'}$ is the probability associated with dialogue state $r'_{i'}$ at time step $t+1$, $P(h'|u')$ is the probability of the user's hypothesised act h' given the observed utterance u', $P(h'|r_i,a)$ is the user model, *i.e.* an estimate of likelihood of the user's hypothesised act h' given the previous dialogue state r_i and the machine's last action a, $P(h')$ is a prior over hypothesised user acts and ζ is a normalising constant.

3.1 The "Upgrade Path" from MDPs to POMDPs

The MM-POMDP approach as described above has been designed to allow a simple "upgrade path" between existing MDP SDSs and POMDP approaches to SDSs. The only additional components are the user transition model $P(h'|r,a)$ (see Section 3.2), the prior over hypothesised user acts $P(h')$, and the ASR-SLU confidence scores $P(h'|u')$. This upgrade exploits the efficient representation of uncertainty in MDP state representations, but adds the capacity to exploit multiple ASR-SLU hypotheses.

3.2 Internal User Transition Model

The first step in building a data driven user transition model is to define a representation for dialogues that captures the features that are required for accurate probability estimation whilst allowing generalisation of corpus data to cover previously unseen examples.

The MM-POMDP approach requires a user transition model, $P(h'|r,a)$ which estimates the likelihood of a hypothesised user action h' given the previous system action a and dialogue state r.

We consider that the hypothesised user action h' can be factorised into two components, a user semantic act with unspecified slot value arguments h'_x and an associated set of slot value arguments h'_w (which can be the empty set if h'_x doesn't require any slot values), *i.e.*

$$\begin{aligned} P(h'|r,a) &= P(h'_x, h'_w|r,a) && (2)\\ &= P(h'_w|h'_x,r,a)P(h'_x|r,a) && (3) \end{aligned}$$

The user's semantic act h'_x is a reaction to the machine action a. Thus given the dialogue state r and the user's action h'_x it appears a reasonable to assume that the slot values h'_w associated with the user's act are independent of a, *i.e.*

$$P(h'|r,a) \approx P(h'_w|h'_x,r)P(h'_x|r,a) \qquad (4)$$

The term $P(h'_x|r,a)$ defines a distribution over underspecified user semantic acts while $P(h'_w|h'_x,r)$ defines a distribution over slot values.

Both $P(h'_w|h'_x,r)$ and $P(h'_x|r,a)$ can be derived from dialogue system corpora. In deriving such a distribution, slot values for the hypothesised user act h'_x which are contain in the corpora can be ignored and thus the data is much less thinly spread. We can further improve the generalisation by noting that the terms r and a are still fully specified down to the level of slot values. For determining which user action is likely to be selected the *relationship* between the slot values seen in the dialogue history and those in the machine's current action are more important than the *specific* slot values that occur in a dialogue [4].

3.3 ASR-SLU hypotheses

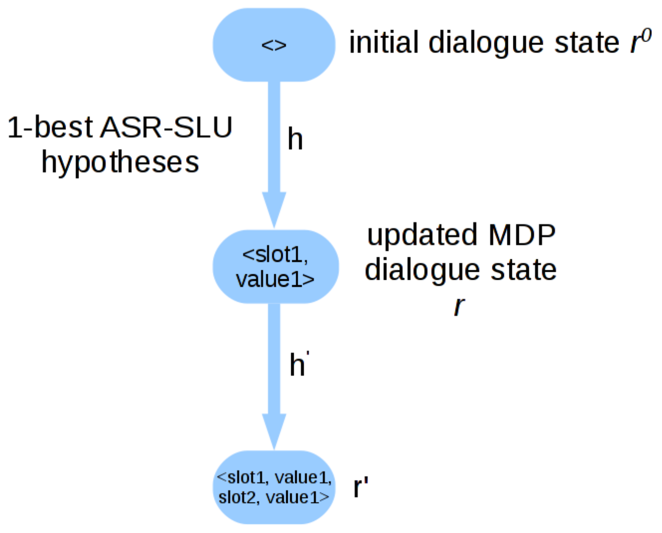
The POMDP and QMDP systems both accept an N-best list of hypotheses from the ASR-SLU. An additional "something-else" hypothesis $\hat{h}$ is also generated (provided it has non-zero probability) which represents an estimate of the likelihood that none of the ASR-SLU hypotheses are correct, see Figure 6. This "something-else" hypothesis ensures that after each user dialogue act a small amount of system belief remains associated with the previous dialogue states, *i.e.* there remains a small belief that nothing was done. Maintaining non-zero belief in previous dialogue states allows the system to fall back if its current dominant beliefs failed to be confirmed by later dialogue moves - see the final paragraph in Section 3.4 below.

The MDP system only uses the first-best hypothesis from the ASR/SLU and does not generate a "something-else" hypothesis.

3.4 State Representation

All three systems maintain a *master space* [18] that provides a detailed view of the state of the dialogue. For these three ISU-based systems the master space consists of a record of the last K dialogue acts. It includes a record of the ASR for the hypothesised user act, a likelihood score for that user act generated by the internal user transition model (see Section 3.2) and a record of the filled and confirmed slot values associated with each step of the dialogue.

For the MDP, which considers only the top ASR-SLU hypothesis, this master space is represented by a single sequence of dialogue states, see Figure 5. For the QMDP and POMDP, which both accept N-best+1 hypotheses the master space is represented as a branching tree structure, as shown in Figure 6.



<>
initial dialogue state r0
1-best ASR-SLU hypotheses
h
<slot1, value1>
updated MDP dialogue state r
h'
<slot1, value1, slot2, value1>
r'

Figure 5: MDP dialogue states and history.

The only significant difference in maintenance of the master space between the MDP and QMDP/POMDP is how confirmations rejected by users are handled. The MDP un-fills any slots that a user rejects. This is a reasonable strategy since the MDP maintains only a single view of the dialogue state. The QMDP/POMDP does not un-fill slots. Instead, rejecting a confirmation causes a shift of probability mass between branches of the dialogue state tree. This shift in probability is based on the likelihood of a user rejecting the confirmation as estimated by the QMDP/POMDP's internal user transition model. Because the QMDP/POMDP does not un-fill slots, it is important that some belief in previous dialogue states is maintained. If belief in such states becomes zero it is not possible for DIPPER-POMDP or DIPPER-QMDP systems to fall back to believing in that state. The maintenance of such states is afforded by the "something-else" hypothesis $\hat{h}$ which is added to the N-best list and which indicates overall certainty in the ASR-SLU hypotheses - see Section 3.3.

3.5 State and Action Spaces

The dialogue manager's *policy* is responsible for the mapping of dialogue state, or belief over dialogue states, into dialogue actions. The master space representation provides a very detailed view of the current state of the dialogue. Unfortunately it is too detailed to generalise a dialogue management policy over. To allow policies to be learnt the master space is mapped onto a smaller *summary space* [18]. The mappings used in our work are set out in Table 2. These mappings effectively generalise the dialogue management policy over large portions of the master space.

Table 2 sets out the policy actions available to the DIPPER-POMDP, DIPPER-QMDP and DIPPER-MDP systems. The MDP and QMDP systems use the the same set of four actions. DIPPER-POMDP's actions

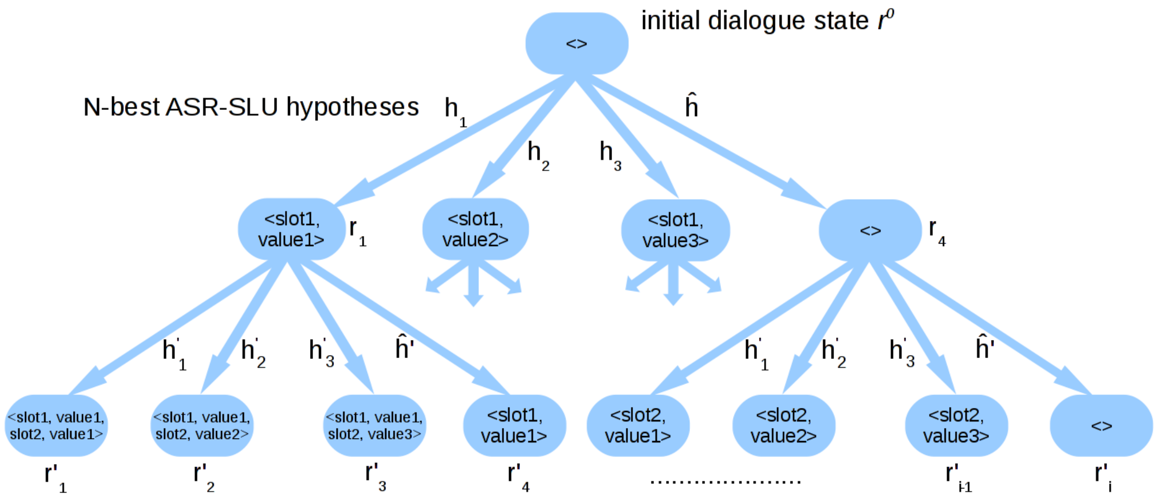


Figure 6: POMDP *belief distribution* (and history) compactly represented using MDP dialogue states as leaves of a tree.

set is necessarily extended as its state space can indicate a degree of uncertainty as to whether individual slots are filled.

The policy actions are *summary space actions* [18] and have to be mapped onto fully instantiated actions in the master space. For example the MDP action `Implicit_Confirm+Ask_A_Slot` could for example be mapped onto "`Implicit_Confirm_Slot_1_Value_Italian+Ask_Slot_2`" or "`Implicit_Confirm_Slot_1_Value_Greek+Ask_Slot_3`". The mapping is context dependent and is carried out using as set out in Table 3.

3.6 Policy Learning

Policy learning consists of the exploration of the policy space, whose dimensionality is determined by the number of features in the state space and the number of actions available to the system. The mappings from master space to summary space and from summary space actions to master space actions (Tables 2 and 3) facilitates the grouping of both dialogue states and actions. This is necessary to reduce the computational complexity of learning the policy.

The DIPPER-MDP summary space is a single vector with 6 binary values corresponding to the current filled and confirmed state of each slot. Both hand-coded policies and learnt policies exist which map points in this compact space to appropriate actions [10].

For both DIPPER-QMDP and DIPPER-POMDP the master space (which can be represented in the form of a tree, see Figure 6 and Section 3.4 for details) represents a *belief distribution* over the states of the dialogue. Theoretically, given good transition and observation models, the belief distribution in the master space should be Markovian and it would be possible to learn optimal policies for this space. In practice the size and dimensionality of the master space belief distribution makes learning such policies impractical [18]. DIPPER-QMDP and DIPPER-POMDP use two different approaches to making policy learning a tractable problem.

DIPPER-QMDP uses an approach suggested by [11] of treating the belief space as representing multiple

System	Summary Space State	Summary Space Actions

POMDP	summary state of the form: $\langle P(F^1 = f^1_{max}), P(F^2 = f^2_{max}), P(F^3 = f^3_{max})$ $P(G^1 = f^1_{max}), P(G^2 = f^2_{max}), P(G^3 = f^3_{max}),$ $d_{len}\rangle$
	
	where $f^l_{max} = \arg\max_{v \in V^l} \sum_{i \in I} p_i \delta(f^l_i = v)$, $P(F^l = f^l_{max}) = \sum_{i \in I} p_i \delta(f^l_i = f^l_{max})$, $P(G^l = f^l_{max}) = \sum_{i \in I} p_i \delta(g^l_i = f^l_{max})$, V^l is the set of possible values for slot l, I is the set of leaf nodes of the current dialogue state tree, p_i is the probability associated with leaf node i, and d_{len} is the length of the dialogue.
		`Ask_Slot_1`
		`Ask_Slot_2`
		`Ask_Slot_3`
		`Implicit_Confirm+Ask_Slot_1`
		`Implicit_Confirm+Ask_Slot_2`
		`Implicit_Confirm+Ask_Slot_3`
		`Explicit_Confirm`
		`Present_Result_&_Close`

QMDP	$\lvert I \rvert$ vectors of the form: $\langle \delta(f^1_i \neq unset), \delta(f^2_i \neq unset), \delta(f^3_i \neq unset),$ $\delta(g^1_i \neq unset), \delta(g^2_i \neq unset), \delta(g^3_i \neq unset)\rangle$
	
	each leaf node has an associated probability p_i.
		`Ask_A_Slot`
		`Implicit_Confirm+Ask_A_Slot`
		`Explicit_Confirm`
		`Present_Result_&_Close`

MDP	Single node of the form: $\langle \delta(f^1_1 \neq unset), \delta(f^2_1 \neq unset), \delta(f^3_1 \neq unset),$ $\delta(g^1_1 \neq unset), \delta(g^2_1 \neq unset), \delta(g^3_1 \neq unset)\rangle$	`Ask_A_Slot`
		`Implicit_Confirm+Ask_A_Slot`
		`Explicit_Confirm`
		`Present_Result_&_Close`

Table 2: Summary space state and action set of DIPPER-MDP, DIPPER-QMDP and DIPPER-POMDP systems. f^l_i and g^l_i are the filled and grounded values associated with slot l and leaf node m. $\delta(Z)$ is a Kronecker delta function which takes values $\delta(Z) = 1$ if Z is true, otherwise $\delta(Z) = 0$.

System	Action Mapping
POMDP	`Ask_Slot_`l_q ⇒ `Ask_Slot_`l_q `Implicit_Confirm+Ask_Slot_`l_q ⇒ `Implicit_Confirm_Slot_`l_c`_Value_`v_c `+ Ask_Slot_`l_q where $v_c = f^{l_c}_{max}$ is a most probable *slot value* recently mentioned by the user (*i.e.* appears at least once in the set of hypothesised user acts immediately proceeding this system action), is not associated with the slot l_q being asked for and has the *maximum filled* marginal probability (which is greater than zero), *i.e.* $l_c = \arg\max_{l \in L^h} P(F^l = f^l_{max})$ where L^h is the set of slots containing recently mentioned values excluding slot l_q and where $P(F^{l_c} = f^{l_c}_{max}) > 0$ (see Table 2 for explanation of terms). `Explicit_Confirm` ⇒ `Explicit_Confirm_Slot_`l_c`_Value_`v_c where $v_c = f^{l_c}_{max}$ is the most probable *slot value* that is associated with the slot l_c which has the *minimum grounded* marginal probability and a filled marginal probability greater than zero, *i.e.* $l_c = \arg\min_{l \in L} P(G^l = f^l_{max})$ and $P(F^{l_c} = f^{l_c}_{max}) > 0$ (see Table 2 for explanation of terms).
QMDP	Requires Q-values to be defined over all *master space actions* – see Equation 5. For MDP policies which only define Q-values over summary space actions a mapping of Q-values to the master space is required; see discussion in Section 3.6.
MDP	`Ask_A_Slot` ⇒ `Ask_Slot_`l_q where l_q is instantiated as the first *unfilled* slot in the current master space state. (The ordering of slots to fill is determined by the system designer). `Implicit_Confirm+Ask_A_Slot` ⇒ `Implicit_Confirm_Slot_`l_c`_Value_`v_c `+ Ask_Slot_`l_q Confirm the first slot value v_c which has been filled but *not* grounded in the current master space state. l_c is the slot associated with v_c. l_q is instantiated as the first *unfilled* slot in the list of slots to fill. (The ordering of slot values is determined by the slots-to-fill order which is specified by the system designer). `Explicit_Confirm` ⇒ `Explicit_Confirm_Slot_`l_c`_Value_`v_c Confirm the first slot value v_c which has been filled but *not* grounded in the current master space state. l_c is the slot associated with v_c.

Table 3: Mapping of actions from summary to master space. Note that if the conditions for mapping a particular action from summary to master space cannot be met then that summary space action cannot be chosen. This is enforced by the hierarchical SHARSHA reinforcement learning algorithm employed [25].

MDP problems. The leaves of the state space tree (Figure 6), which represent the possible states of the dialogue, are treated as *I* independent DIPPER-MDP dialogues. A vote is take over the leaves, in the form of a weighted summation of action values (Q-values), in order to decide the action to take next, see Equation 5.

$$Q(a,b) \approx \sum_{i \in I} p_i Q_{\text{MDP}}(a, r_i) \tag{5}$$

Where a is a master space action to be voted on, b is the current belief distribution and r_i is the MDP dialogue state associated with leaf node i. The weighting used, p_i, is the probability associated with the leaf node i in the master space dialogue state tree.

Note that the vote is taken over *master space actions* and thus requires a MDP policy which defines Q-values for all master space actions. In cases where the MDP policy only defines Q values over summary space actions (*e.g.* for the convenience afforded to policy learning by the reduced action space), a mapping is required from the summary space action Q-values to the master space action Q-values. One plausible mapping is to map the Q-value of a summary space action onto all of the master space actions that it could represent, *e.g.* all possible master space `Ask_Slot_`*x* actions are assigned the Q-value for the summary space action `Ask_A_Slot`. We note that such a mapping is equivalent to summing Q-values over *summary space actions* in Equation 5. Other more sophisticated mappings are possible but tend to introduce additional parameters, and there is no sound theoretical basis for determining their values.

The DIPPER-POMDP policy maps the whole belief distribution to actions. This is in contrast to the DIPPER-QMDP approach of treating the belief space as being composed of a parallel set of independent dialogues. To make policy mapping tractable the dimensionality of the belief space is reduced by using a summary state space of the form shown in Table 2. This reduces a large joint probability distribution across all of the possible slot values to a set of six marginal probabilities for the slot values. This is an approximation to the master space belief distribution in that it lumps together large regions of the space and many details of the joint distribution are lost. This representation is, however, sufficient to demonstrate the advantages of POMDP spoken dialogue systems, such as accumulation of evidence. This was tested in D6.3.

A version using new generalisation techniques has also been tested in simulation. This work is reported in D1.2.

3.7 Data driven ASR-SLU noise model

The ASR-SLU noise model produces an N-best list for each action generated by the user simulation. It simulates error rates found in real data and the coincidence of errors between items in the N-best list. It also generates ASR-scores for each of the items in the N-best list, drawn from realistic score distributions.

In training DIPPER-POMDP our aim is to find policies that efficiently trade-off certainty against dialogue length. The ideal policy would present information that matches the *every* user's requirements with the absolute minimum of dialogue moves. An important consideration in attempting to train such policies is for the system to learn the level of certainty it requires in each slot value. As the level of certainty will be derived from repeated observations obtained via a noisy ASR-SLU channel and from the ASR-scores, it is important that the system is trained using a noise model and score distributions that closely match the real ASR-SLU with which it will be used. This is equivalent to training a robot on as a realistic simulation as is possible. To this end an analysis of existing SDS corpus data was undertaken and a data-driven probabilistic noise model was derived, see [2].

4 Software

The prototype DIPPER POMDP system consists of the following software components:

- multi-state version of the DIPPER dialogue manager (java), adapted to provide POMDP summary spaces
- internal user transition model derived from dialogue corpus data (C-code with OAA wrapper)
- interface to REALL Reinforcement Learning platform (Java OAA wrapper)
- interface to SUPELEC RL library (see D1.2, JNI interface)
- training simulation environment; including ASR-noise model (java) [2].

References

[1] James Henderson and Oliver Lemon. Mixture Model POMDPs for Efficient Handling of Uncertainty in Dialogue Management. In *Proceedings of ACL*, 2008.

[2] Paul A. Crook and Oliver Lemon. Accurate probability estimation of hypothesised user acts for POMDP approaches to dialogue management. In *Proceedings of the 12th Annual Research Colloquium of the special-interest group for computational linguistics in the UK and Ireland (CLUKI 2009)*, April 2009.

[3] Paul Crook. A principled framework for data driven user transition models in POMDP dialogue management. *ACM Transaction in Speech and Language Processing*, 2010. (in preparation).

[4] Paul A. Crook, Oliver Lemon, James Henderson, and Xingkun Liu. Combining POMDP approaches with ISU dialogue management. *Computer Speech and Language*, 2010. (in preparation).

[5] JD Williams and SJ Young. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech and Language*, 21(2):231–422, 2007.

[6] SJ Young, J Schatzmann, K Weilhammer, and H Ye. The Hidden Information State Approach to Dialog Management. In *ICASSP 2007*, 2007.

[7] Peter Bohlin, Robin Cooper, Elisabet Engdahl, and Staffan Larsson. Information states and dialog move engines. *Electronic Transactions in AI*, 3(9), 1999.

[8] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340, 2000.

[9] Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, pages 115–124, Sapporo, 2003.

[10] Oliver Lemon and Xingkun Liu. Dialogue policy learning for combinations of noise and user simulation: transfer results. In *SIGdial*, 2007.

[11] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, USA, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

[12] S. Young, M. Gašić, S. Keizer, F. Mairesse, B. Thomson, and K. Yu. The Hidden Information State model: a practical framework for POMDP based spoken dialogue management. *Computer Speech and Language*, 2009. To appear.

[13] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 2009. To appear.

[14] B. Zhang, Q. Cai, J. Mao, and B. Guo. Planning and acting under uncertainty: A new model for spoken dialogue system. In *Proc 17th Conf on Uncertainty in AI*, Seattle, 2001.

[15] Jason Williams, Pascal Poupart, and Steve Young. Factored partially observable markov decision processes for dialogue management. In *Workshop on Knowledge and Reasoning in Practical Dialog Systems, International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[16] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI'94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 1023–1028, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[17] MTJ Spaan and N Vlassis. Perseus: randomized point-based value iteration for POMDPs. Technical Report IAS-UVA-04-02, Universiteit van Amsterdam, 2004.

[18] Jason Williams and Steve Young. Scaling Up POMDPs for Dialog Management: The "Summary POMDP" Method. In *Proc. ASRU*, 2005.

[19] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The Knowledge Engineering Review*, 21:97–126, 2006.

[20] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye, and S. Young. Agenda-based User Simulation for Bootstrapping a POMDP Dialogue System. In *Proceedings of HLT/NAACL*, 2007.

[21] Richard Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.

[22] Steve Young. ATK: An Application Toolkit for HTK, Version 1.6. Technical report, Cambridge University Engineering Department, 2007.

[23] A. Ranta. Grammatical framework. a type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189, 2004.

[24] Ivan Meza-Ruiz, Sebastian Riedel, and Oliver Lemon. Accurate statistical spoken language understanding from limited development resources. In *ICASSP 08*, 2008.

[25] Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander. Hierarchical Reinforcement Learning of Dialogue Policies in a development environment for dialogue systems: REALL-DUDE. In *Proceedings of Brandial, the 10th SemDial Workshop on the Semantics and Pragmatics of Dialogue, (demonstration systems)*, 2006.

[26] Cereproc. http://www.cereproc.com/.