

Interspeech 2009  
Tutorial T-8  
6 September, afternoon session

## Statistical approaches to dialogue systems

Jason D. Williams, AT&T

Steve Young, Cambridge University

Blaise Thomson, Cambridge University

# Outline

**Introduction**

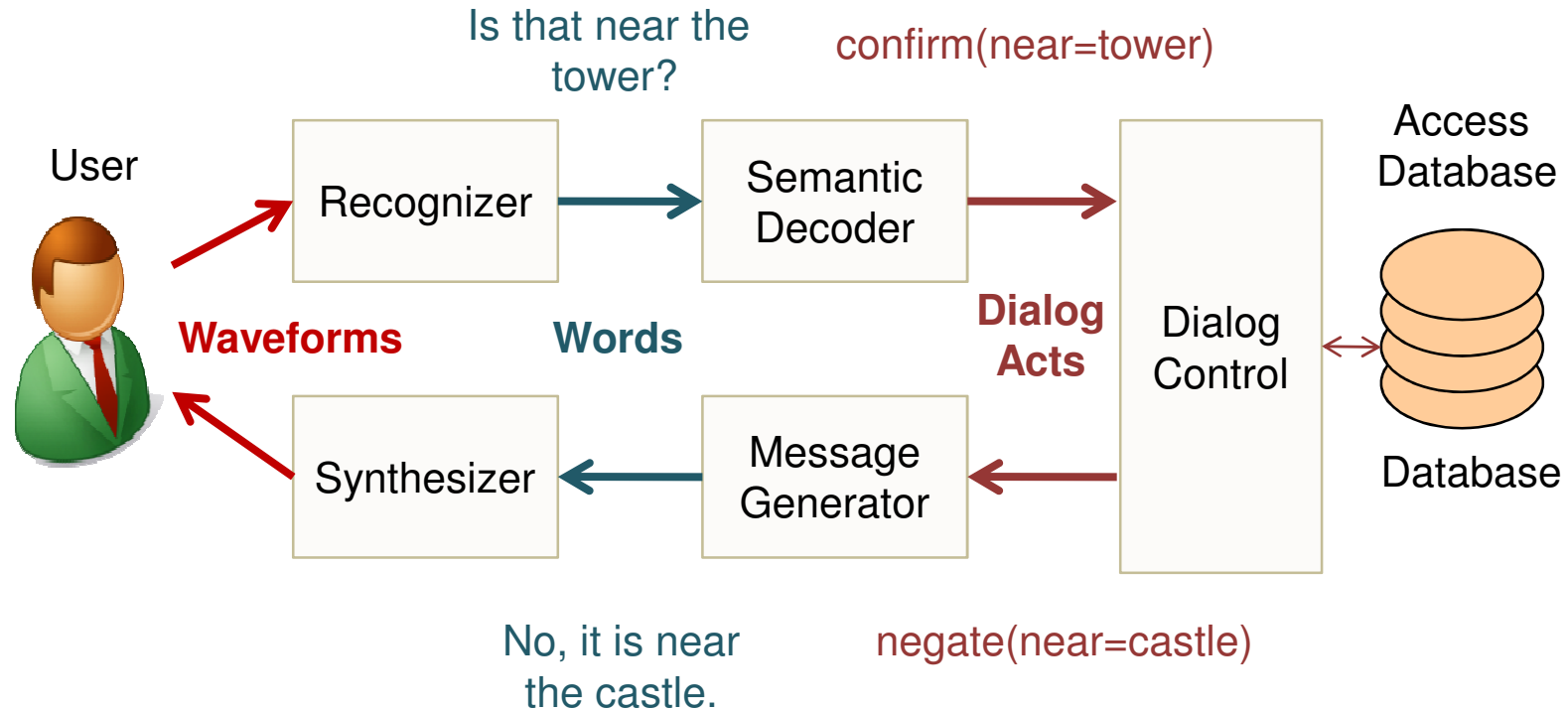
**Tracking / Belief Monitoring**

**Action Selection / Policy Optimization**

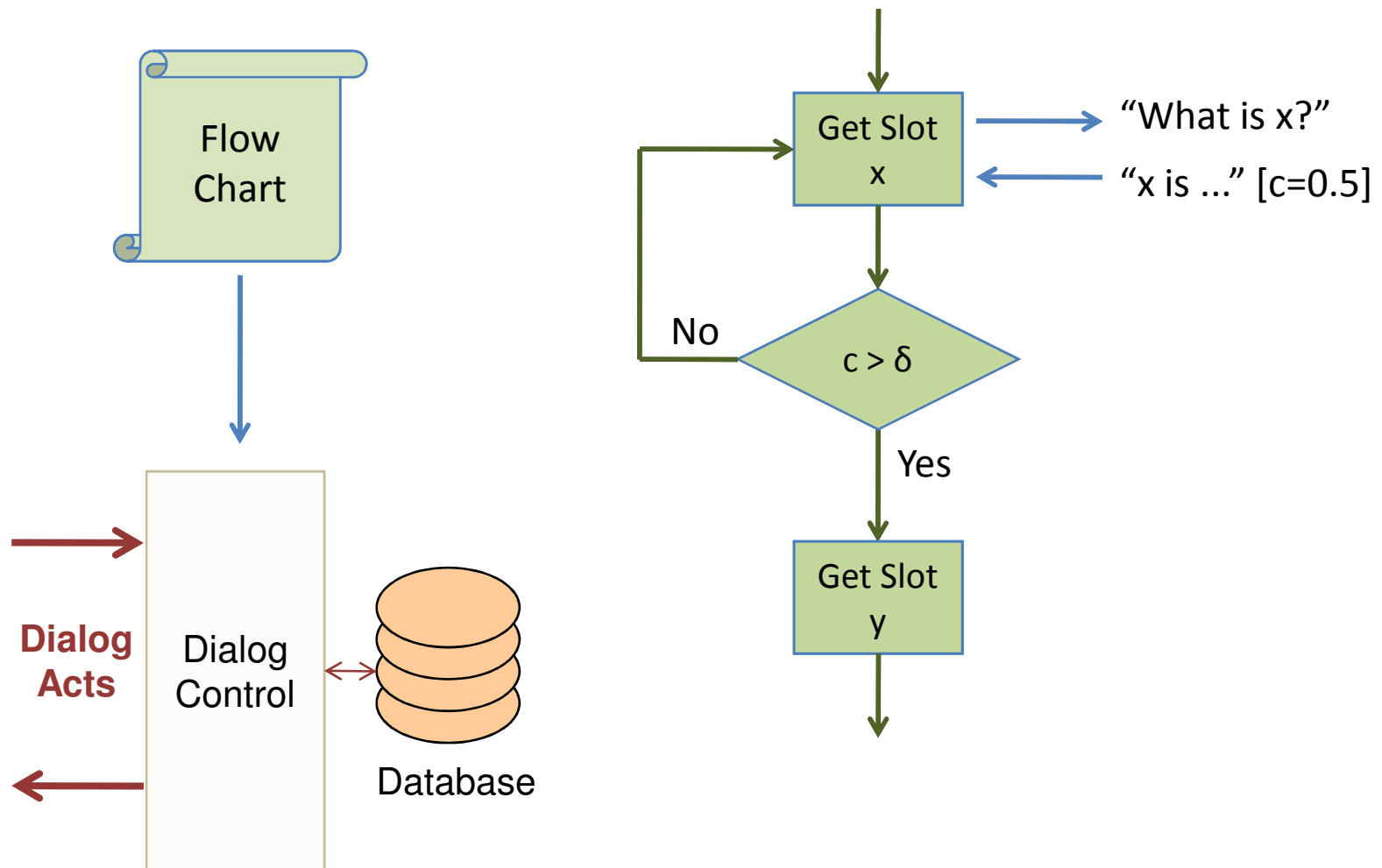
**Practical Systems**

# Introduction

# Architecture of a Spoken Dialogue System



# Dialog Control: the Traditional Approach



# Problems with Traditional Approach

- no mechanism for handling uncertainty
- difficult to integrate models of human behavior
- no objective measure of “goodness”
- no parameterization of decision logic

## Result:

- development manually intensive hence expensive
- systems are fragile, especially in noisy conditions
- unable to adapt either short term or long term

# The Statistical Approach

## Tracking / Belief Monitoring

- incorporate user goals into a hidden dialog state
- hidden state is called the *belief state*.
- use Bayesian inference to track belief state

## Action Selection / Policy Optimization

- define a parametric mapping from belief state to action
- mapping is called the *policy*
- use reinforcement learning to optimize policy

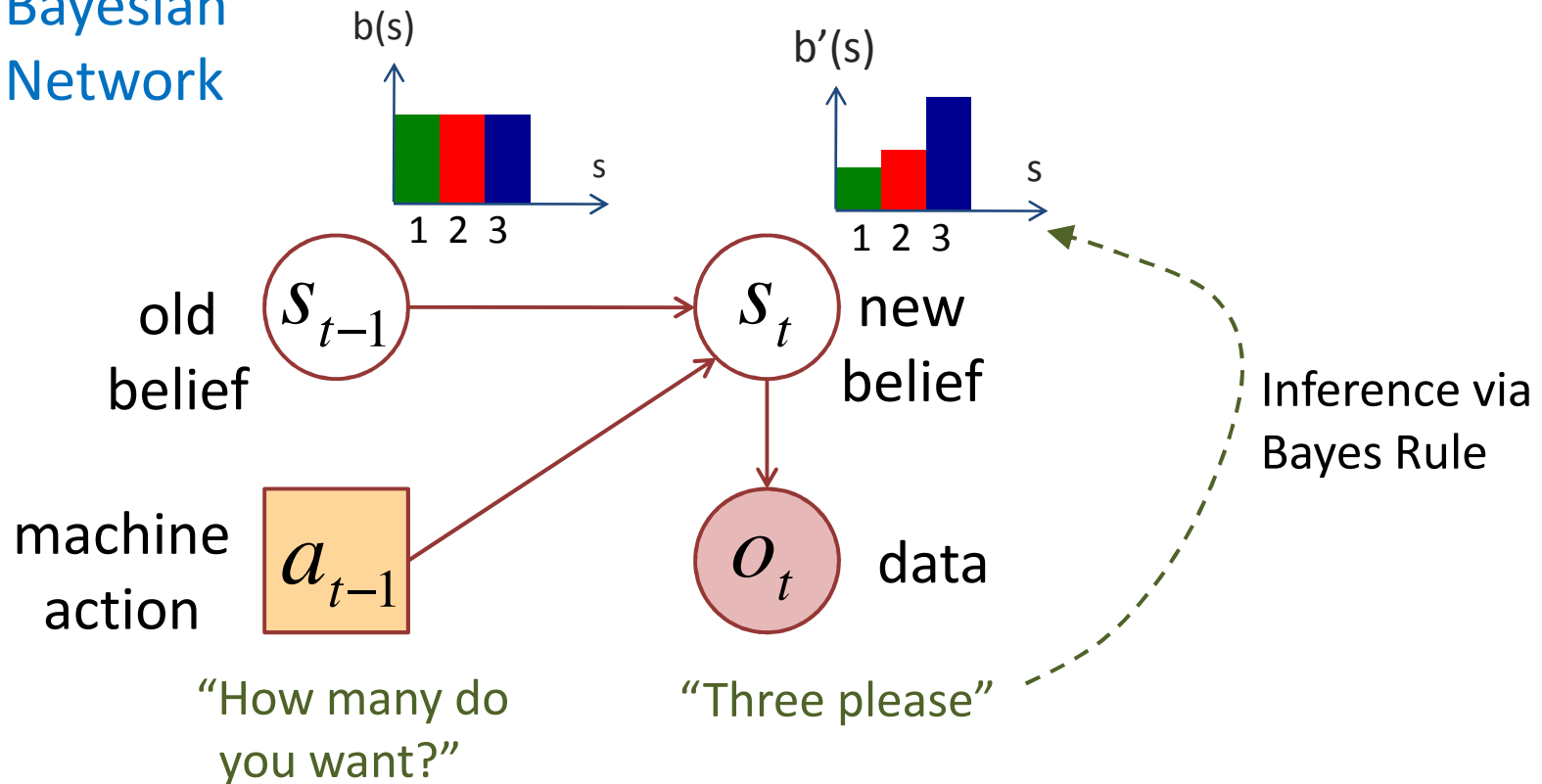
# Belief Monitoring

$$P(\text{belief} \mid \text{data}) = \frac{P(\text{data} \mid \text{belief})P(\text{belief})}{P(\text{data})}$$

Reverend Thomas Bayes  
(1702-1761)



## Bayesian Network



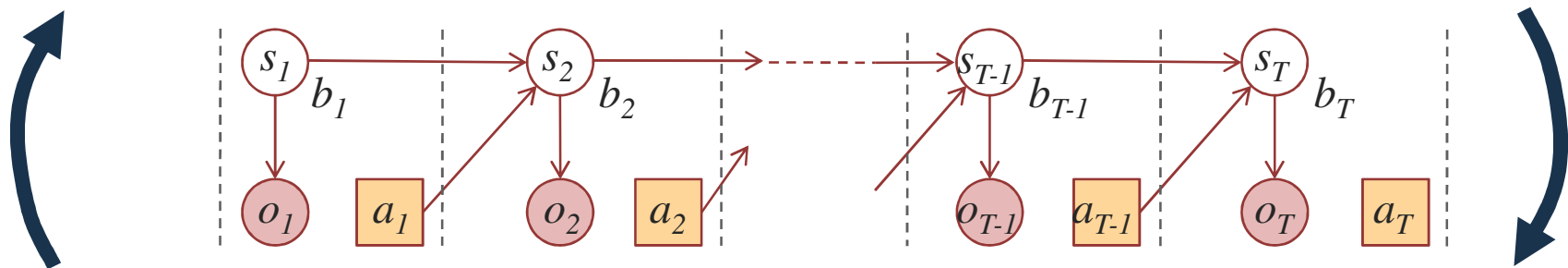
# Policy Optimization

$$V^*(b) = \max_a \left\{ r(b, a) + \sum_{o'} P(o' | b, a) V^*(b') \right\}$$

Richard E Bellman  
(1920-1984)



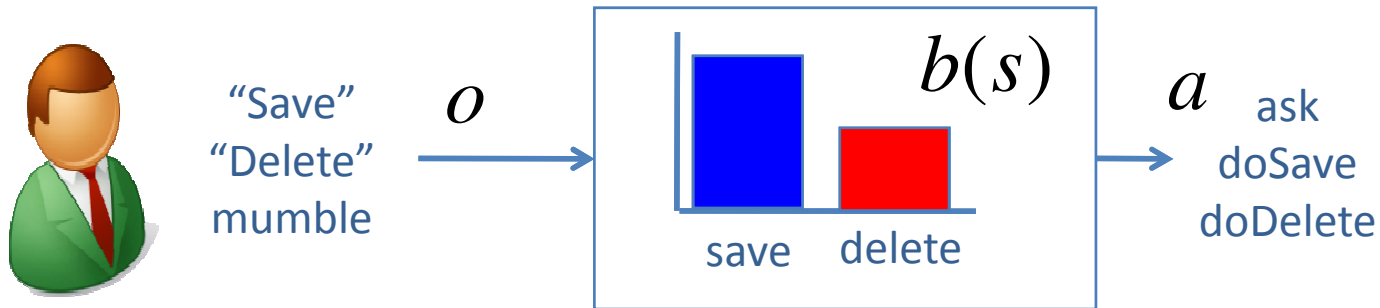
$$\text{Reward} = R(b_1, a_1) + R(b_2, a_2) + \dots + R(b_{T-1}, a_{T-1}) + R(b_T, a_T)$$



$$\text{Policy} \quad \pi(b_1) \rightarrow a_1 \quad \pi(b_2) \rightarrow a_2 \quad \pi(b_{T-1}) \rightarrow a_{T-1} \quad \pi(b_T) \rightarrow a_T$$

Reinforcement Learning

# A Simple Two State Example



Observation Probability

$$P(o' | s', a)$$

eg

"Save"	0.7
"Delete"	0.1
mumble	0.2

$$P(o' | save, ask)$$

Transition Probability

$$P(s' | s, a)$$

	save	delete
save	1.0	0.0
delete	0.0	1.0

$$P(s' | s, ask)$$

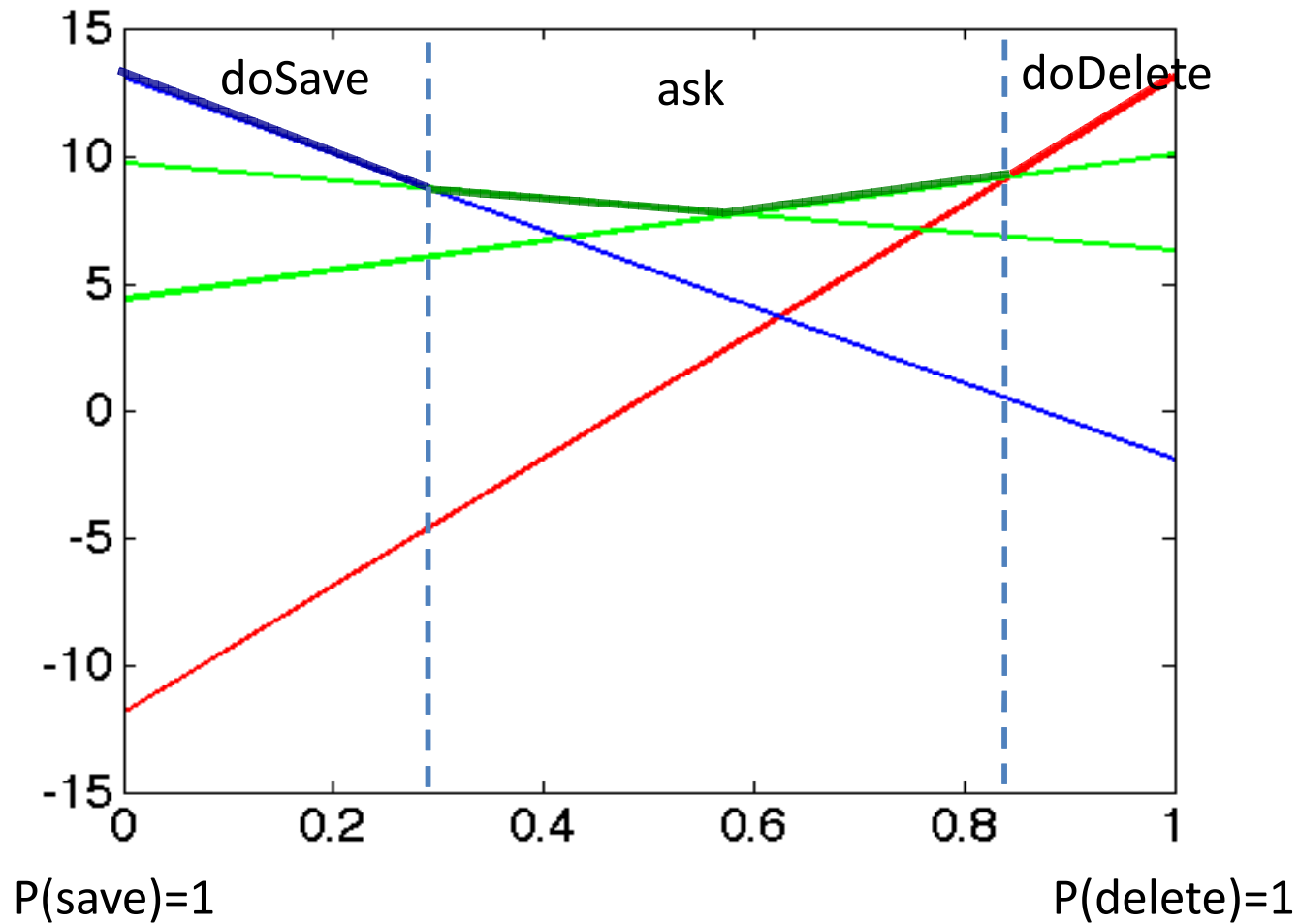
Reward Function

$$R(s, a)$$

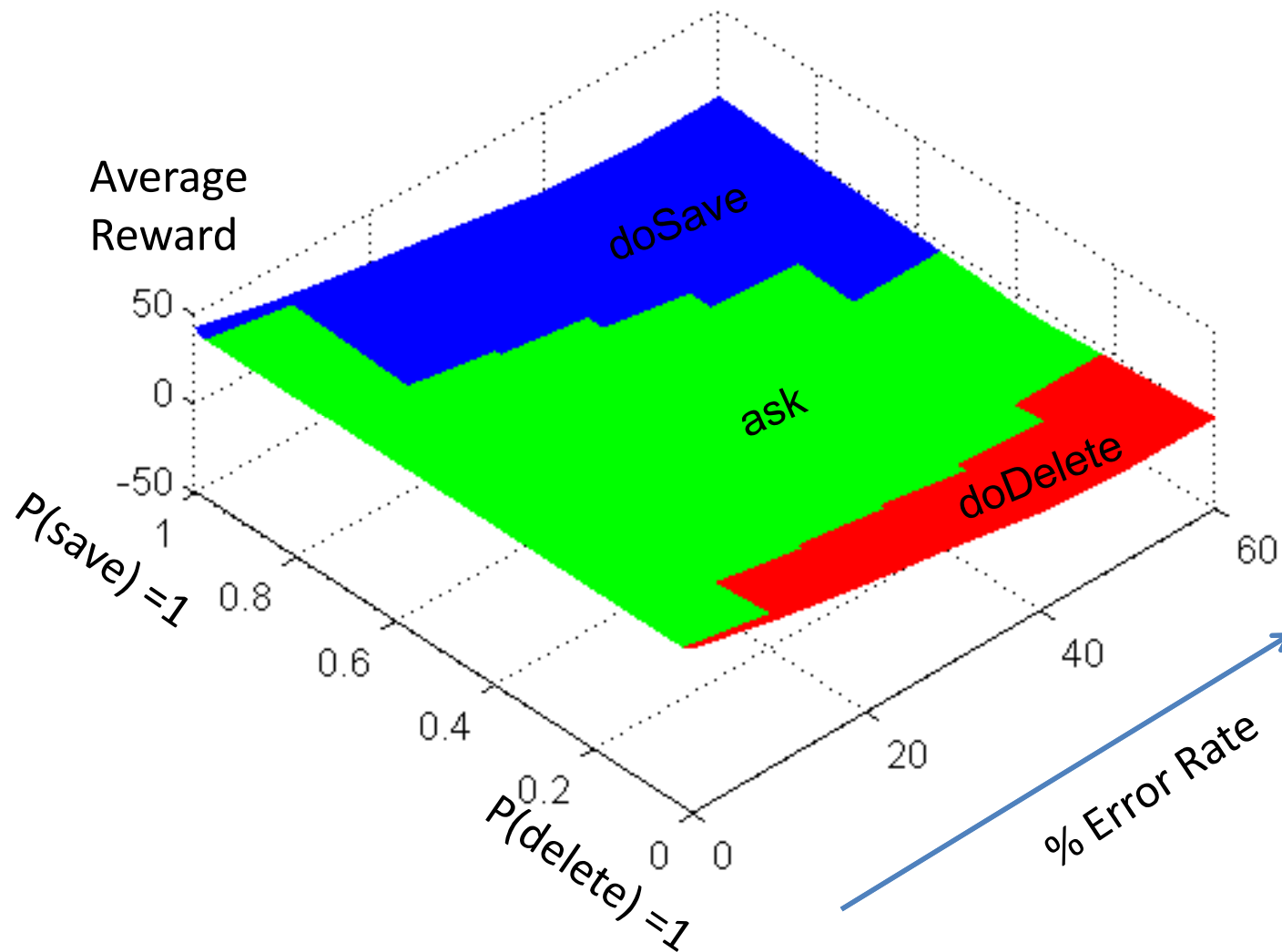
	save	delete
ask	-1	-1
doSave	+5	-10
doDelete	-20	+5

# Policy Value Function at 30% Error Rate

Average Return



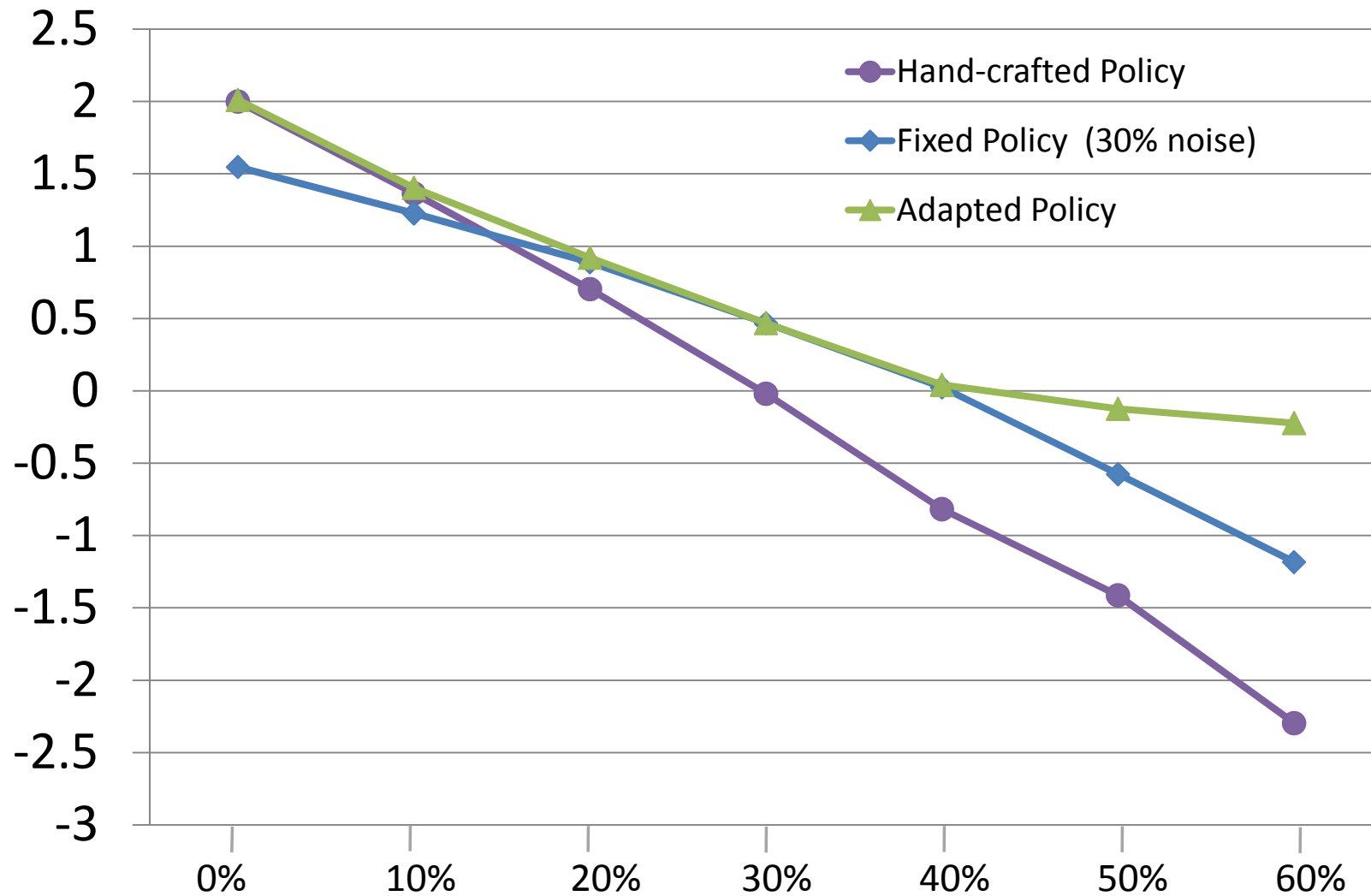
# Policy Value Function vs Error Rate



# Example Dialog

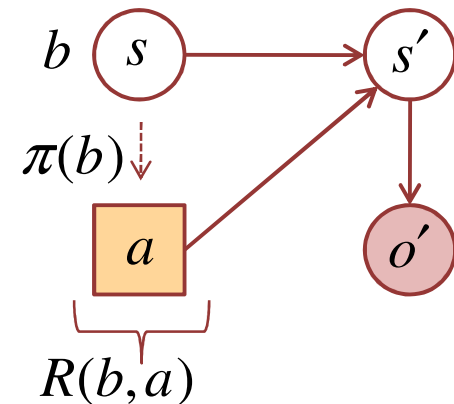
action	observation	belief [ save delete]	reward	
		[0.65 0.35]		Prior for save vs delete
ask	mumble	[0.65 0.35]	-1	No ASR output
ask	“Delete”	[0.28 0.72]	-1	ASR Correct
ask	mumble	[0.28 0.72]	-1	No ASR output
ask	“Save”	[0.65 0.35]	-1	ASR Error
ask	“Delete”	[0.28 0.72]	-1	ASR Correct
ask	“Delete”	[0.08 0.92]	-1	ASR Correct
doDelete			+5	Correct action taken

# Performance: Average Reward vs Error Rate



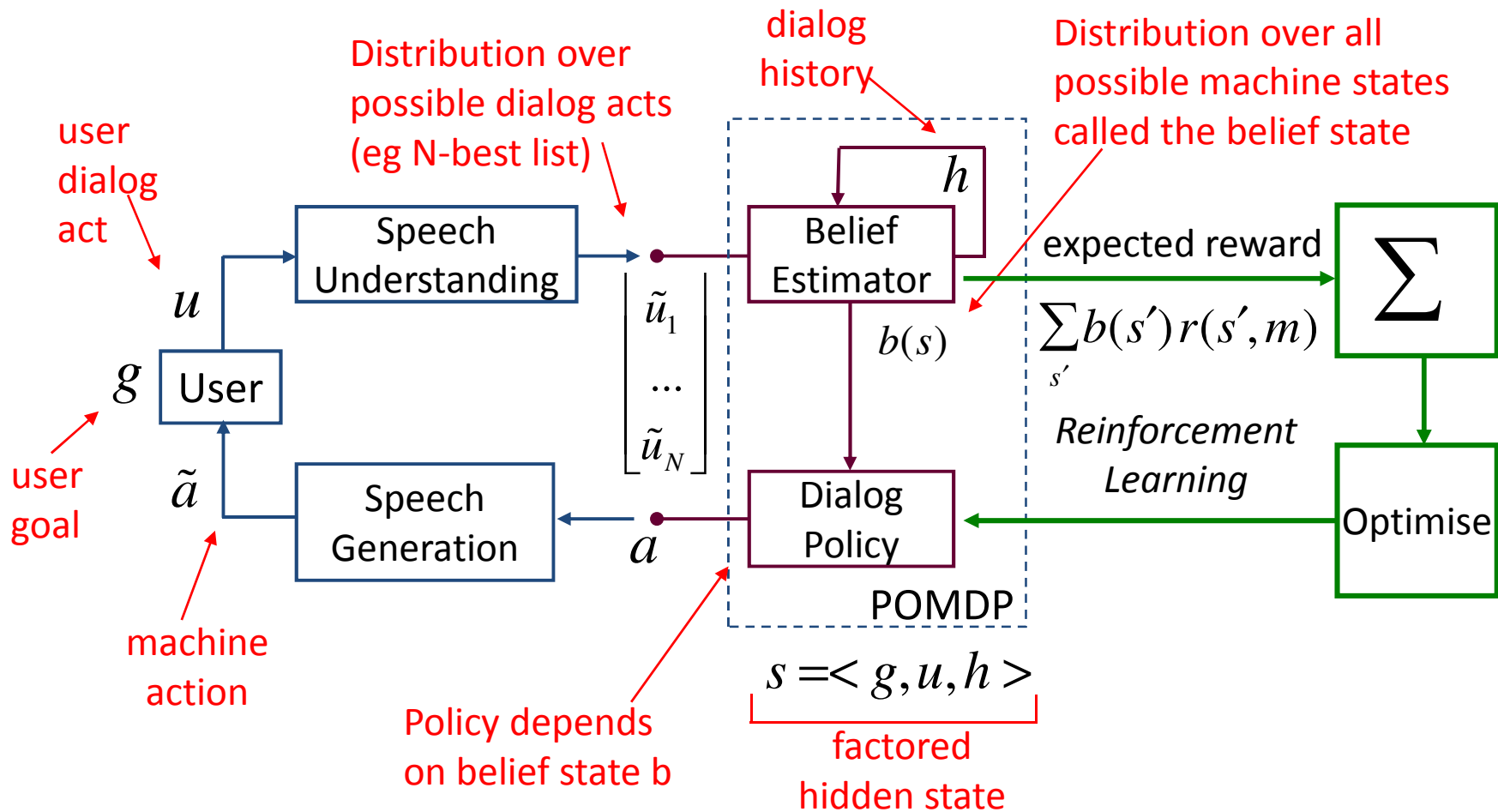
## Partially Observable Markov Decision Processes (POMDPs)

- Belief represented by distributions over states and updated from observations by Bayesian inference
- Objectives defined by the accumulation of rewards
- Policy which maps beliefs into actions and which can be optimized by reinforcement learning



- Principled approach to handling uncertainty and planning
- However, scaling to real-world problems is challenging

# Summary: POMDP-based Dialog Systems



# Tracking / Belief Monitoring

# Basic symbols

Symbol	Meaning	Examples
$a$	Machine action ( <u>known</u> )	ask(origin) "Where from?"
$o$	<u>Observed</u> output from ASR or other sensors	BOSTON~0.5 AUSTIN~0.2
$s$	<u>Everything</u> that is hidden from the machine	goal=flight(austin,london) from-status=confirmed say: "austin"
$b(s)$	<u>Distribution</u> over hidden variables	goal=flight(austin,london) from-status=confirmed say: "austin" $b = 0.3$ $o = 0.2$ $o = 0.1$

# Goal of belief tracking

Aim of this section:

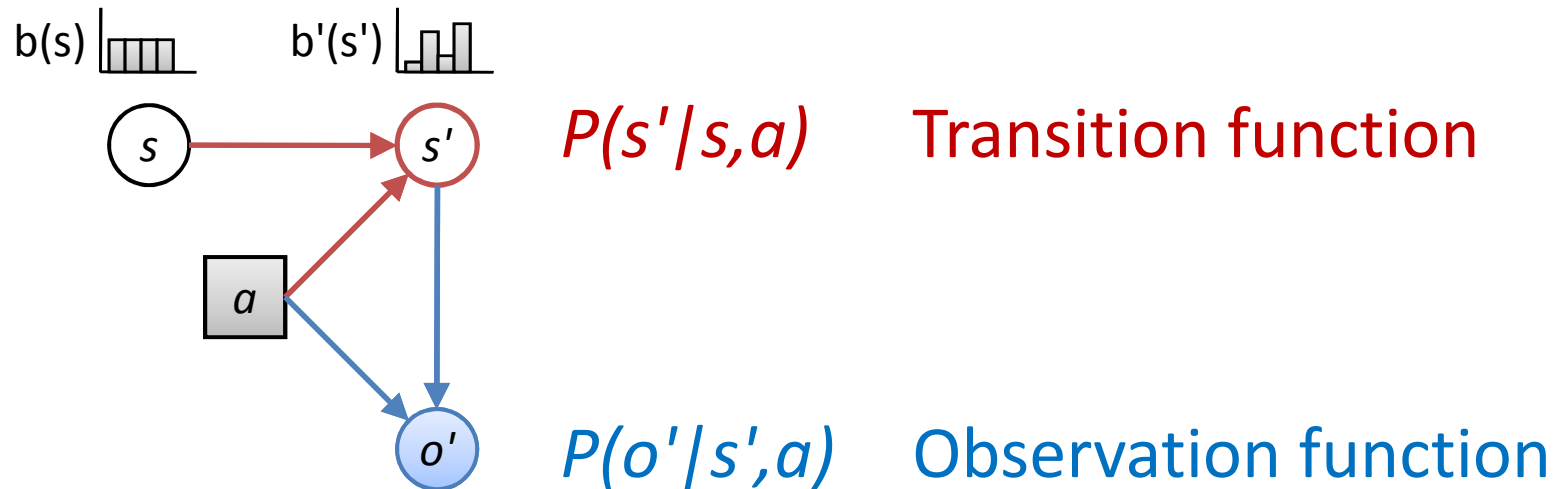
how to compute  $b'(s')$

given

$b(s), a, o,$

and whatever we know about the domain

## 2 core modeling assumptions



Key idea:  $s$  includes sufficient information about history for Markov assumption to hold

## Derivation of the core update equation

$$\begin{aligned} b'(s') &= P(s' | b, a, o') \\ &= \frac{P(o' | s', a, b) P(s' | a, b)}{P(o' | a, b)} \\ &= \frac{P(o' | s', a) \sum_s P(s' | a, b, s) P(s | a, b)}{P(o' | a, b)} \\ &= \frac{P(o' | s', a) \sum_s P(s' | s, a) b(s)}{P(o' | a, b)} \\ &= \eta \cdot P(o' | s', a) \sum_s P(s' | s, a) b(s) \end{aligned}$$

*Leslie Kaelbling, Michael Littman and Anthony Cassandra. Planning and Acting in Partially Observable Stochastic Domains. Artificial Intelligence, Vol. 101, 1998.*

## Notes on the core update equation

new belief state      normalizing constant      observation function      transition function      old belief state

$$b'(s') = \eta \cdot P(o' | s', a) \sum_s P(s' | s, a) b(s)$$

Synthesizes:

- Current belief over hidden states  $b(s)$
- How we think  $s$  changes to  $s'$  (conditioned on  $a$ )
- How likely  $o'$  is given  $s'$  and  $a$

# SDS-POMDP model: A useful decomposition

$$s = (g, u, h)$$

Symbol	Meaning	Examples
$g$	User's goal	flight(austin,london)
$u$	User's (true, hidden) action	state(to(london)) "I'm flying to London"
$h$	Relevant dialog history drawing on (hidden) user actions and system actions	from-status=stated to-status=not-stated

*Jason D. Williams and Steve Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. Computer Speech and Language 21(2): 393-422.*

# SDS-POMDP transition function

$$P(s' | s, a) = P(g', u', h' | g, u, h, a) \\ = P(g' | g, u, h, a) P(u' | g', g, u, h, a) P(h' | u', g', g, u, h, a)$$

Assume

Assume

Assume

$$= \underbrace{P(g' | g, a)}_{\text{goal model}} \cdot \underbrace{P(u' | g', h, a)}_{\text{user action model}} \cdot \underbrace{P(h' | u', g', h, a)}_{\text{history model}}$$

How user goal changes

Estimate from annotated dialogs

How user actions are generated

Estimate from annotated dialogs

What elements of dialog history to track

Dialog designer writes this; usually has functional form

# SDS-POMDP observation function

$$P(o' | s', a) = P(o' | g', u', h', a)$$

|  
Assume



$$P(o' | s', a) = \underbrace{P(o' | u', a)}$$

**ASR model**

How user actions (and system choice of language model) produce ASR output

Estimate from transcribed recognitions

# SDS-POMDP update equation

$$b'(s') = \eta \cdot P(o' | s', a) \sum_s P(s | s, a) b(s)$$

$$b'(g', u', h') = \eta \cdot P(o' | u', a) \sum_{g, u, h} P(g' | g, a) P(u' | g', h, a) P(h' | u', g', h, a) b(g, u, h)$$

$$= \eta \cdot P(o' | u', a) \sum_h P(u' | g', h, a) P(h' | u', g', h, a) \sum_g P(g' | g, a) \sum_u b(g, u, h)$$

$$= \eta \cdot P(o' | u', a) \sum_h P(u' | g', h, a) P(h' | u', g', h, a) \sum_g P(g' | g, a) b(g, h)$$

$$b'(g', h') = \eta \cdot \underbrace{\sum_{u'} P(o' | u', a)}_{\text{new belief state}} \underbrace{\sum_h P(u' | g', h, a)}_{\text{ASR model}} \underbrace{P(h' | u', g', h, a)}_{\text{user action model}} \underbrace{\sum_g P(g' | g, a)}_{\text{dialog history model}} \underbrace{b(g, h)}_{\text{old belief state}}$$

new belief state

ASR model

dialog history model

old belief state

normalizing constant

user action model

user goal model

## SDS-POMDP update: worked example

$$b'(g', h') = \eta \cdot \sum_{u'} P(o' | u', a) \underbrace{\sum_h P(u' | g', h, a)}_{\text{drop dialog history}} \underbrace{P(h' | u', g', h, a)}_{\text{drop dialog history}} \sum_g P(g' | g, a) b(g, h)$$

for clarity, drop dialog history

$$b'(g') = \eta \cdot \sum_{u'} P(o' | u', a) P(u' | g', a) \sum_g P(g' | g, a) b(g)$$

## Toy dialog problem - voicemail

g  $\in$  { save, delete }

u  $\in$  { req-save, req-delete, mumble }

a  $\in$  { ask, do-save, do-delete }

o  $\in$  { req-save, req-delete }

(No confidence scores, for now)

# Goal model

$$P(g'|g,a)$$

		$g'$	
		save	delete
$g$	save	1.0	0.0
	delete	0.0	1.0

$a \in \{ \text{ask} \}$

Asking the user what they want doesn't change what they want.

		$g'$	
		save	delete
$g$	save	0.7	0.3
	delete	0.7	0.3

$a \in \{ \text{do-save, do-delete} \}$

Saving/deleting moves to the next message; the user's goal for the next message is sampled from a prior

# User action model

$$P(u' | g', a)$$

		$u'$		
		req-save	req-delete	mumble
$g'$	save	0.8	0.0	0.2
	delete	0.0	0.8	0.2

Same for all values of  $a$

80% of the time the user states their goal (honestly);  
20% of the time they mumble something out of grammar

Note that  $a = do-save$  and  $a = do-delete$  advance to the next message and ask the user what they would like to do on that message

# ASR model

$$P(o' | u', a)$$

		$o'$	
		save	delete
$u'$	req-save	0.9	0.1
	req-delete	0.1	0.9
	mumble	0.5	0.5

Same for all values of  $a$

When the user says something in grammar, ASR is 90% accurate.  
When the user says something out of grammar, ASR yields a random result.

$b = [save=0.7, delete=0.3]$   $a=ask$   $o=save$

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	Norm	$b'(g')$
save	req-save	save	0.9	0.8	1	0.7	0.504	0.803	0.914
	req-save	del	0.9	0.8	0	0.3	0	0	
	req-del	save	0.1	0	1	0.7	0	0	
	req-del	del	0.1	0	0	0.3	0	0	
	mumble	save	0.5	0.2	1	0.7	0.070	0.111	
	mumble	del	0.5	0.2	0	0.3	0	0	
del	req-save	save	0.9	0	0	0.7	0	0	0.086
	req-save	del	0.9	0	1	0.3	0	0	
	req-del	save	0.1	0.8	0	0.7	0	0	
	req-del	del	0.1	0.8	1	0.3	0.024	0.038	
	mumble	save	0.5	0.2	0	0.7	0	0	
	mumble	del	0.5	0.2	1	0.3	0.030	0.048	

$b = [save=0.7, delete=0.3]$   $a=ask$   $o=delete$

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	Norm	$b'(g')$
save	req-save	save	0.1	0.8	1	0.7	0.056	0.151	0.339
	req-save	del	0.1	0.8	0	0.3	0	0	
	req-del	save	0.9	0	1	0.7	0	0	
	req-del	del	0.9	0	0	0.3	0	0	
	mumble	save	0.5	0.2	1	0.7	0.070	0.188	
	mumble	del	0.5	0.2	0	0.3	0	0	
del	req-save	save	0.1	0	0	0.7	0	0	0.661
	req-save	del	0.1	0	1	0.3	0	0	
	req-del	save	0.9	0.8	0	0.7	0	0	
	req-del	del	0.9	0.8	1	0.3	0.216	0.581	
	mumble	save	0.5	0.2	0	0.7	0	0	
	mumble	del	0.5	0.2	1	0.3	0.030	0.081	

## Perfect ASR (for in-grammar speech) : o= delete

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	Norm	$b'(g')$
save	req-save	save	0	0.8	1	0.7	0	0	0.206
	req-save	del	0	0.8	0	0.3	0	0	
	req-del	save	1	0	1	0.7	0	0	
	req-del	del	1	0	0	0.3	0	0	
	mumble	save	0.5	0.2	1	0.7	0.070	0.206	
	mumble	del	0.5	0.2	0	0.3	0	0	
del	req-save	save	0	0	0	0.7	0	0	0.794
	req-save	del	0	0	1	0.3	0	0	
	req-del	save	1	0.8	0	0.7	0	0	
	req-del	del	1	0.8	1	0.3	0.240	0.706	
	mumble	save	0.5	0.2	0	0.7	0	0	
	mumble	del	0.5	0.2	1	0.3	0.030	0.081	

*Perfect ASR + user never mumbles: o= delete;*

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	Norm	$b'(g')$
save	req-save	save	0	1	1	0.7	0	0	0.0
	req-save	del	0	1	0	0.3	0	0	
	req-del	save	1	0	1	0.7	0	0	
	req-del	del	1	0	0	0.3	0	0	
	mumble	save	0.5	0	1	0.7	0	0	
	mumble	del	0.5	0	0	0.3	0	0	
del	req-save	save	0	0	0	0.7	0	0	1.0
	req-save	del	0	0	1	0.3	0	0	
	req-del	save	1	1	0	0.7	0	0	
	req-del	del	1	1	1	0.3	0.3	1.0	
	mumble	save	0.5	0	0	0.7	0	0	
	mumble	del	0.5	0	1	0.3	0	0	

## Better ASR model

Include confidence score

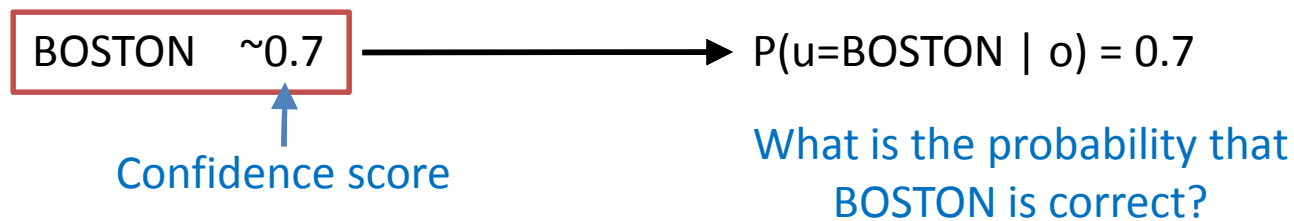
- Confidence score indicates reliability of ASR result
- Including it yields more accurate belief updates

Use all of the N-Best list

- For difficult recognition tasks, 10-20% of the time, the correct ASR result is in position  $N > 1$
- Making use of the full N-Best list increases the likelihood that the correct answer will attain high belief

## Confidence score

For our purposes, a confidence score is a prediction of the probability that an ASR result is correct:



However, our framework calls for a generative model:

$$P(o|u) \longrightarrow P(\text{BOSTON} \sim 0.7 \mid u=BOSTON) = ?$$

Of all the recognitions and confidence scores that could have been produced, what is the likelihood of producing this one, given that the user said BOSTON?

## Solution – apply Bayes' rule

$$P(o | u) = \frac{P(u | o)P(o)}{P(u)} \quad \text{Apply Bayes' rule}$$

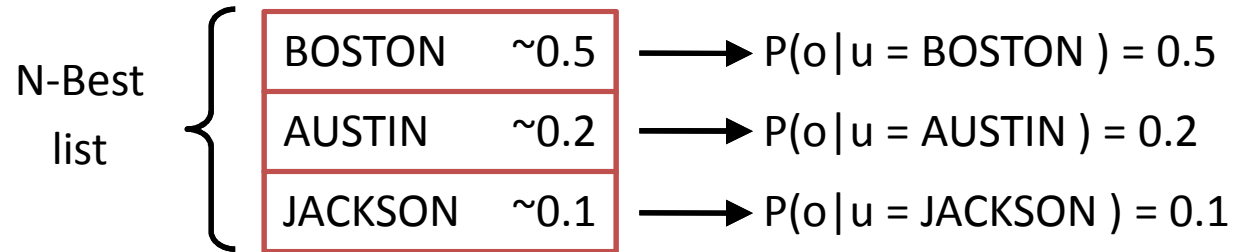
$$\propto \frac{P(u | o)}{P(u)} \quad \begin{array}{l} P(o) \text{ is constant for all } u; \\ P(o) \text{ can be merged into} \\ \text{normalization constant } \eta \end{array}$$

$$\approx P(u | o) \quad \begin{array}{l} \text{Assume } P(u) \text{ is uniform –} \\ \text{or at least much more} \\ \text{uniform than } P(u/o) \end{array}$$

*S. Young, J. Schatzmann, K. Weilhammer and H. Ye. (2007). "The Hidden Information State Approach to Dialog Management." ICASSP 2007, Honolulu, Hawaii.*

# Incorporate N-Best list

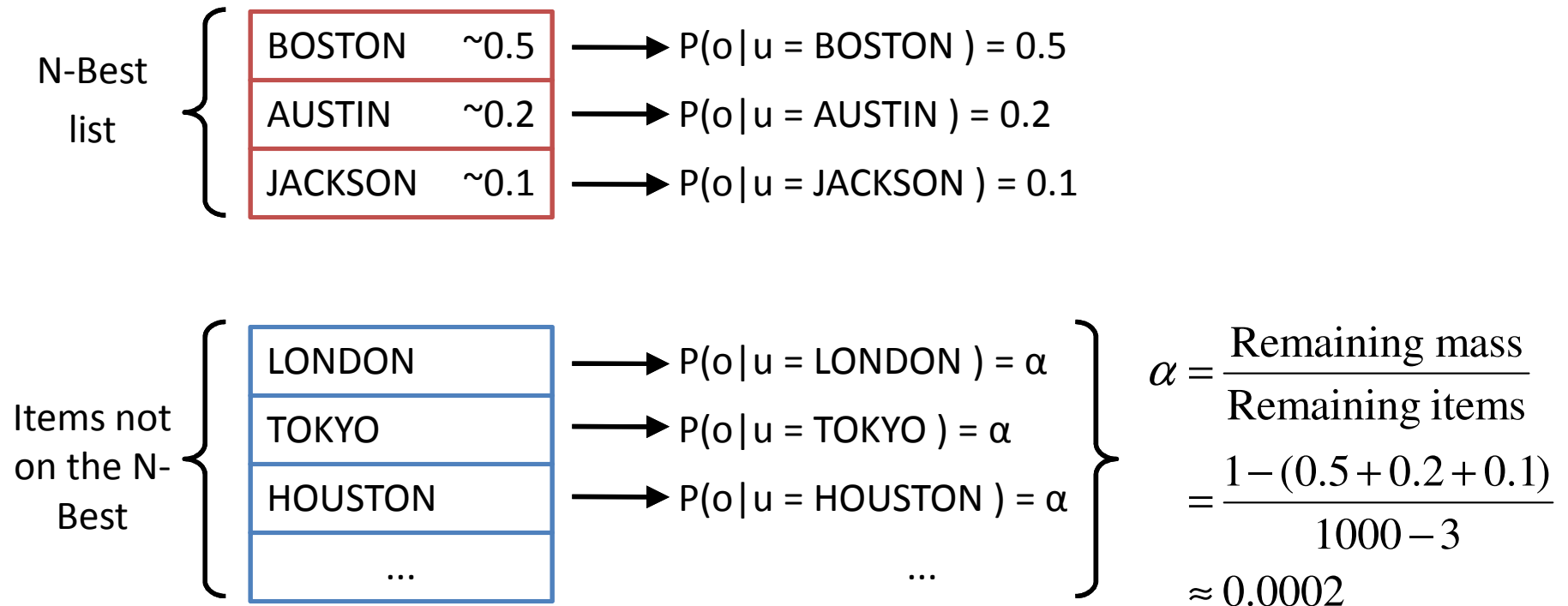
Example: recognition on 1000 cities



*Jason D. Williams and Suhrid Balakrishnan. 2009. Estimating probability of correctness for ASR N-Best lists. Proc SIGDIAL, London, United Kingdom.*

# Incorporate N-Best list

Example: recognition on 1000 cities



Distribute remaining mass uniformly over unseen elements

Jason D. Williams and Suhrid Balakrishnan. 2009. Estimating probability of correctness for ASR N-Best lists. Proc SIGDIAL, London, United Kingdom.

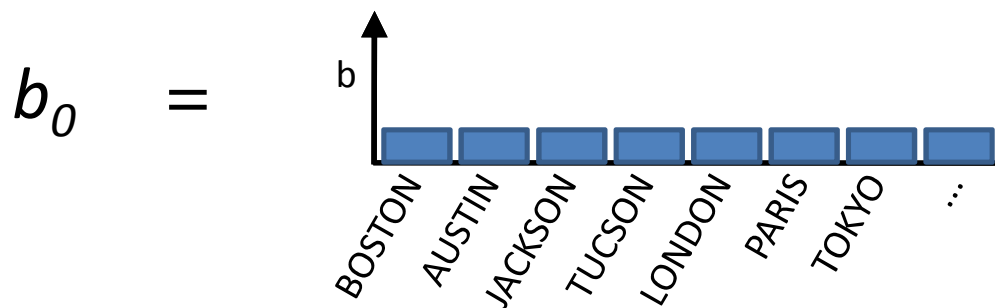
## N-Best model illustration: weather info

$g \in$  1000 cites; fixed throughout dialog

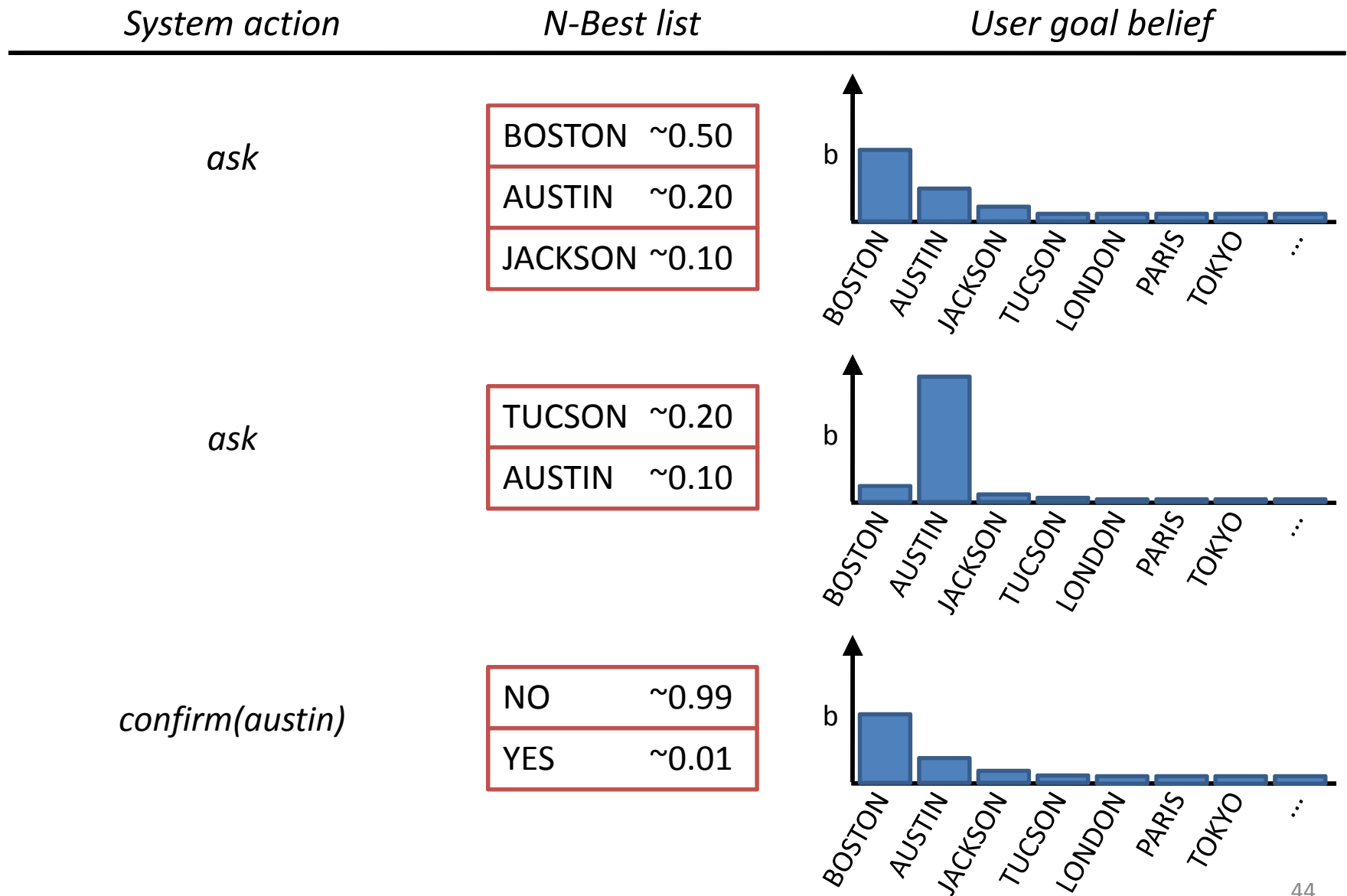
$u \in$  CITIES + { yes, no }

$o \in$  N-Best lists of elements of  $u$ , with conf scores

$a \in$  { ask, confirm(CITY), play-weather(CITY) }



# N-Best model illustration



## Scalability problems

SDS-POMDP requires  $O(|G|^2)$  computations

Consider a small domain:

from	1000 values
to	1000 values
time	1000 values
date	1000 values

$$\left. \vphantom{\begin{matrix} \text{from} & 1000 \text{ values} \\ \text{to} & 1000 \text{ values} \\ \text{time} & 1000 \text{ values} \\ \text{date} & 1000 \text{ values} \end{matrix}} \right\} |G| = 1000^4 = 10^{12}$$

For this simple problem,  $|G|^2 = 10^{24}$

We need a response in  $< 1$  s

$O(10^{24})$  impossible in real time !

## 2 methods for efficient belief monitoring

1. **M-Best:** Constrain aspects of the model such that un-observed goals can be tracked en-masse
2. **Factorization:** Decompose the network as much as possible; apply approximate inference techniques from the Bayesian network literature

## 2 main assumptions for M-Best approaches

1. Assume user goal is fixed:

$$P(g' | g, a) = \delta(g', g) = \begin{cases} 1, & g = g' \\ 0, & g \neq g' \end{cases}$$

(This can be relaxed slightly, but arbitrary changes in the persistent hidden state are not supported)

2. Elements not on the N-Best list are equally confusable

For  $u'_{(i)}$  and  $u'_{(j)}$  not on the N - Best list,

$$P(o' | u'_{(i)}, a) = P(o' | u'_{(j)}, a)$$

## One M-Best implementation: Partitions

$$\mathbf{G} = \{g_1, \dots, g_N\}$$

$$\mathbf{Q} = \{q_1, \dots, q_M\}$$

$q_i$  is a *partition* containing 1 or more user goals

$N$  is fixed;  $M$  varies

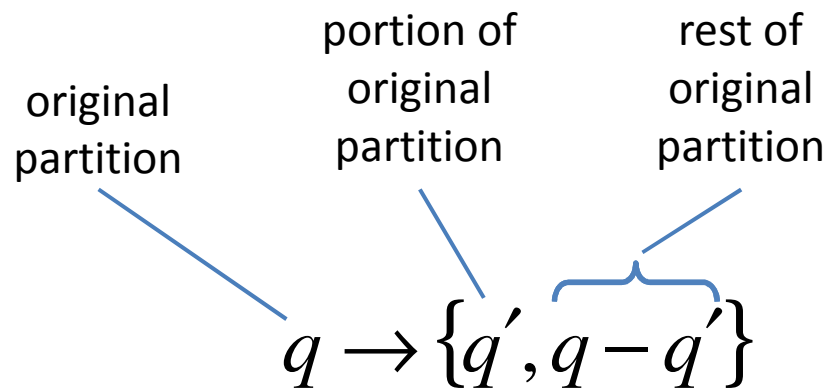
Every goal  $g_i$  is a member of exactly 1 partition  $q_j$

Every partition contains at least one goal

*S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson and K. Yu (2009). "The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management." Computer Speech and Language, To appear.*

# Belief refinement

To track a new partition, separate it using *belief refinement*



$$b(q') = \frac{b_0(q')}{b_0(q)} b(q)$$

$$b(q - q') = \frac{b_0(q) - b_0(q')}{b_0(q)} b(q)$$

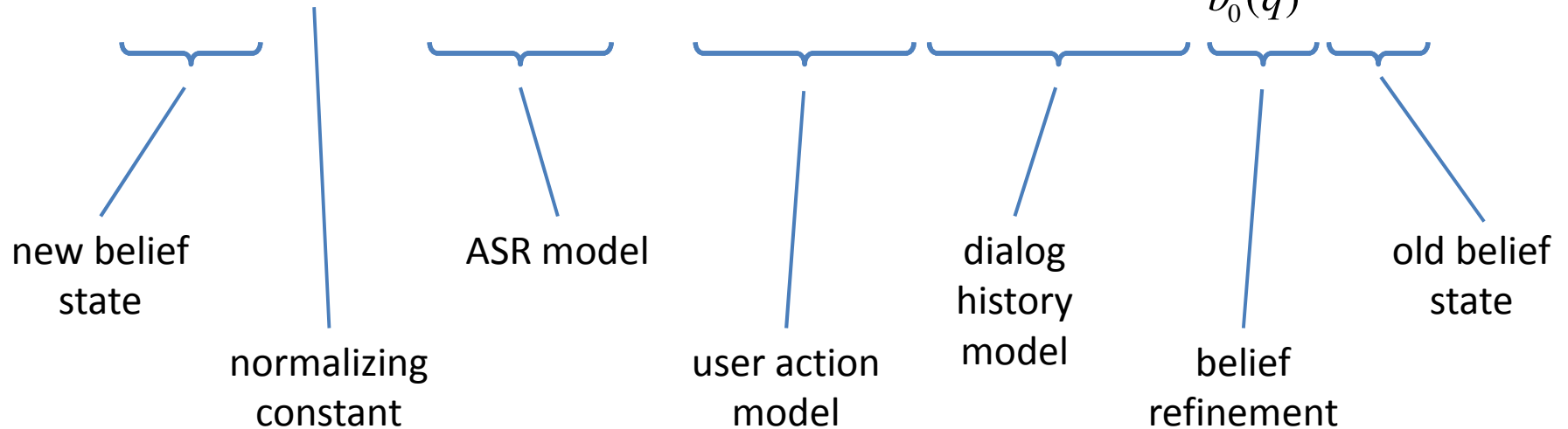
# Partition update equation

$$b'(g', h') = \eta \cdot \sum_{u'} P(o' | u', a) \sum_h P(u' | g', h, a) P(h' | u', g', h, a) \underbrace{\sum_g P(g' | g, a)}_{\text{goal is fixed; goal model drops out}} b(g, h)$$



goal is fixed; goal model drops out

$$b(q', h') = \eta \cdot \sum_{u'} P(o' | u', a) \sum_h P(u' | q', h, a) P(h' | u', q', h, a) \frac{b_0(q')}{b_0(q)} b(q, h)$$



## Partition update example

100 cities

$g = ( \text{from-city, to-city} )$

$|G| = 100 * 99 = 9900$

- Can't fly from X to X

Assume simple user action model

## Partition update example

```
from: (all)
to:   (all)
prior: 1.0   belief: 1.0
```

## Partition update example

from: (all)
to: (all)
prior: 1.0      belief: 1.0

*"Where to?"*

BOSTON	~0.5
AUSTIN	~0.2

# Partition update example

*"Where to?"*

BOSTON	~0.5
AUSTIN	~0.2

from: (all)  
to: x { boston, austin }  
prior: 0.98 belief: 0.98

from: (all)  
to: boston  
prior: 0.01 belief: 0.01

from: (all)  
to: austin  
prior: 0.01 belief: 0.01

# Partition update example

*"Where to?"*

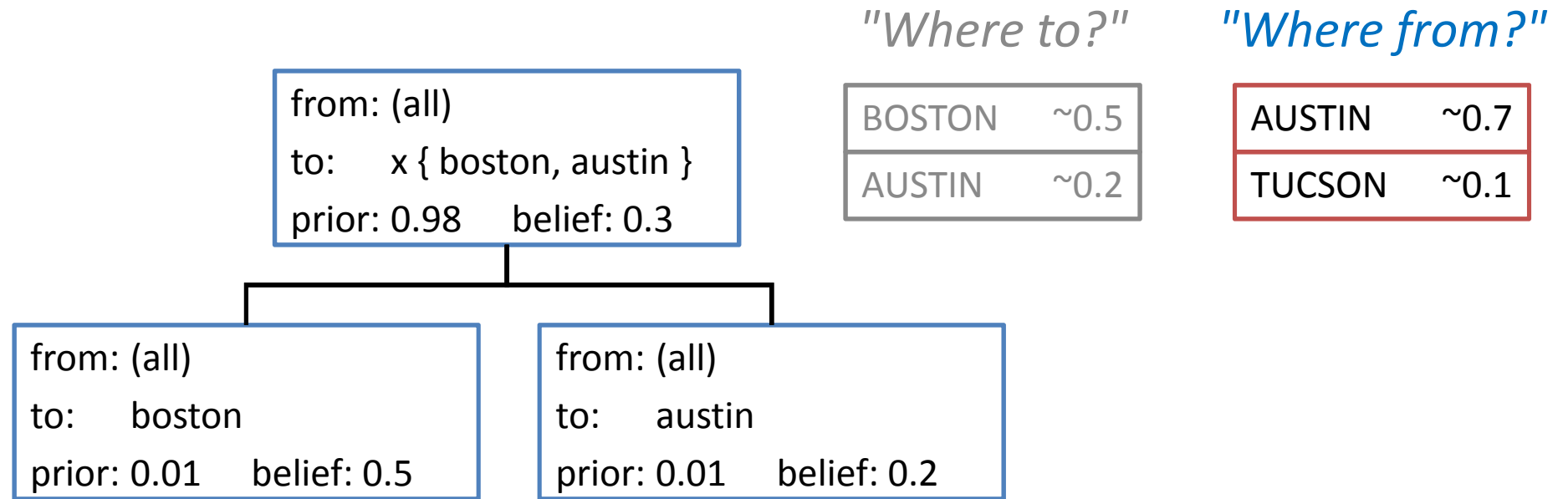
BOSTON	~0.5
AUSTIN	~0.2

from: (all)  
to: x { boston, austin }  
prior: 0.98 belief: 0.3

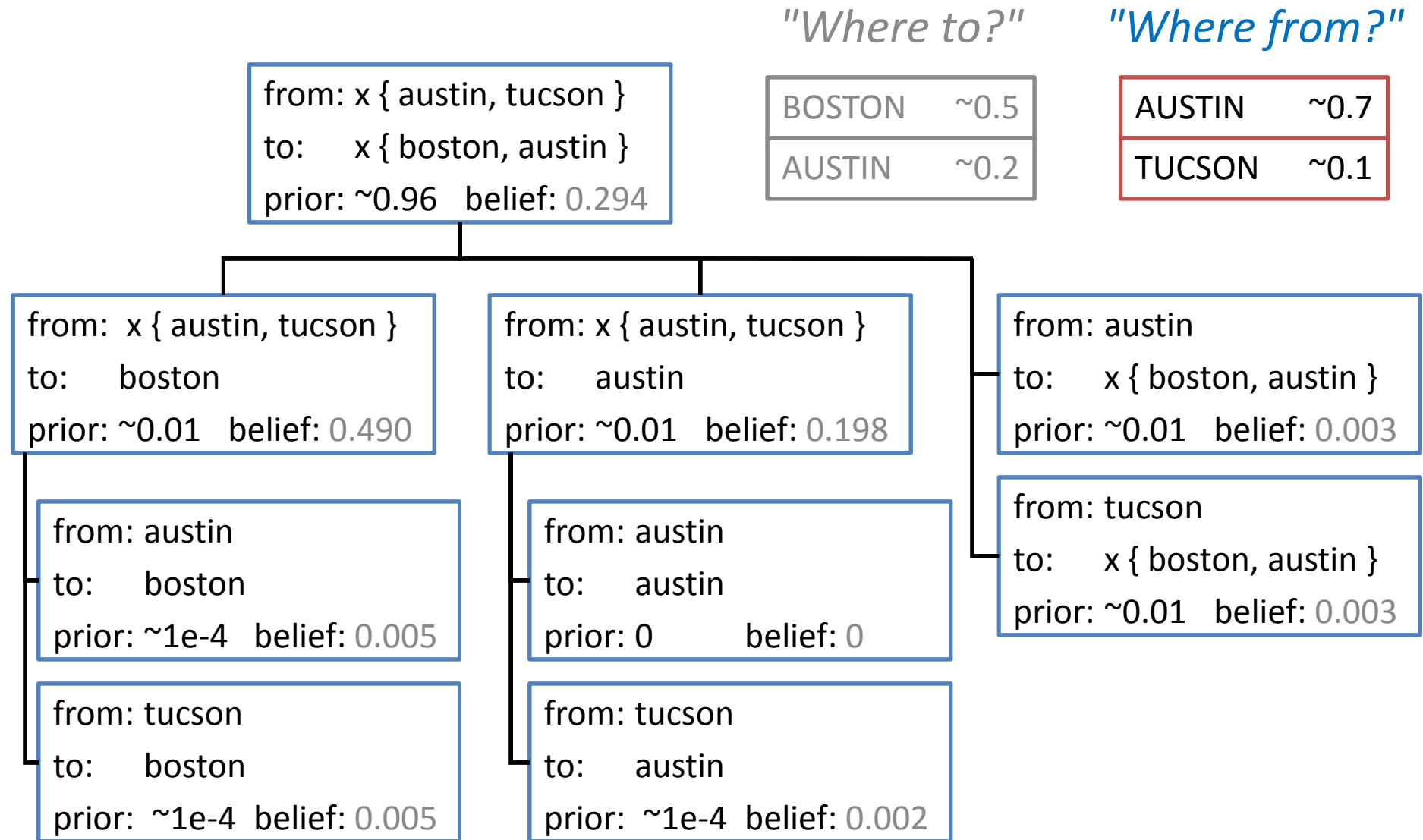
from: (all)  
to: boston  
prior: 0.01 belief: 0.5

from: (all)  
to: austin  
prior: 0.01 belief: 0.2

# Partition update example



# Partition update example



# Partition update example

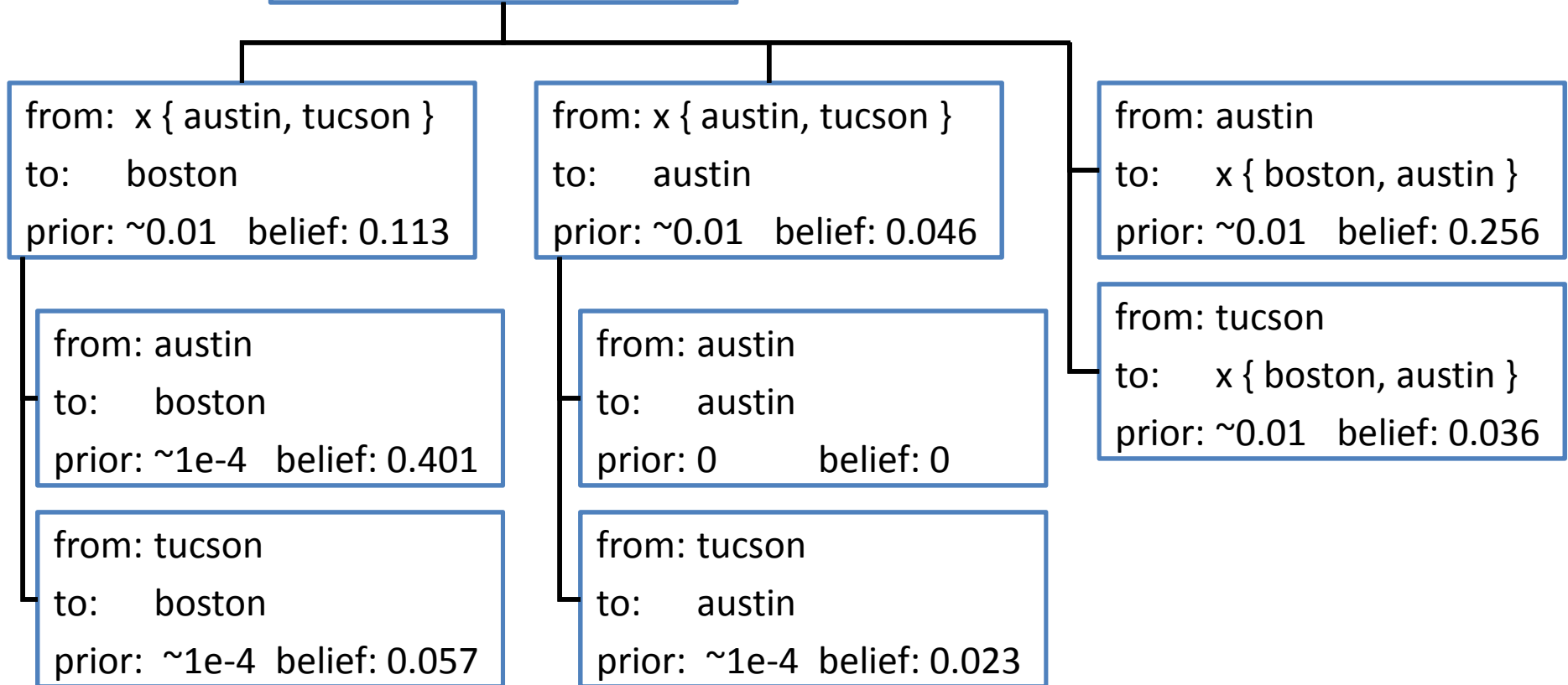
"Where to?"

"Where from?"

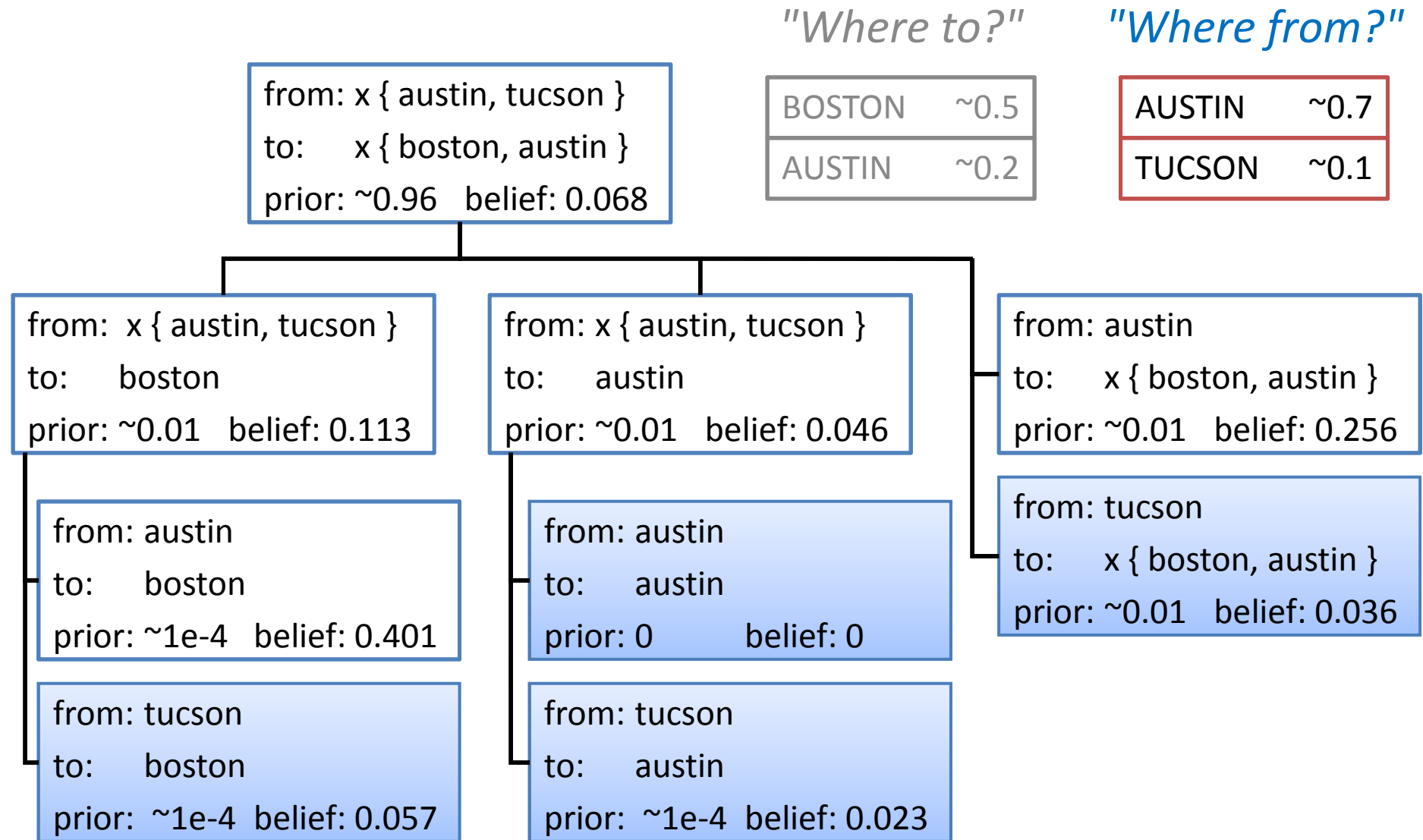
BOSTON	~0.5
AUSTIN	~0.2

AUSTIN	~0.7
TUCSON	~0.1

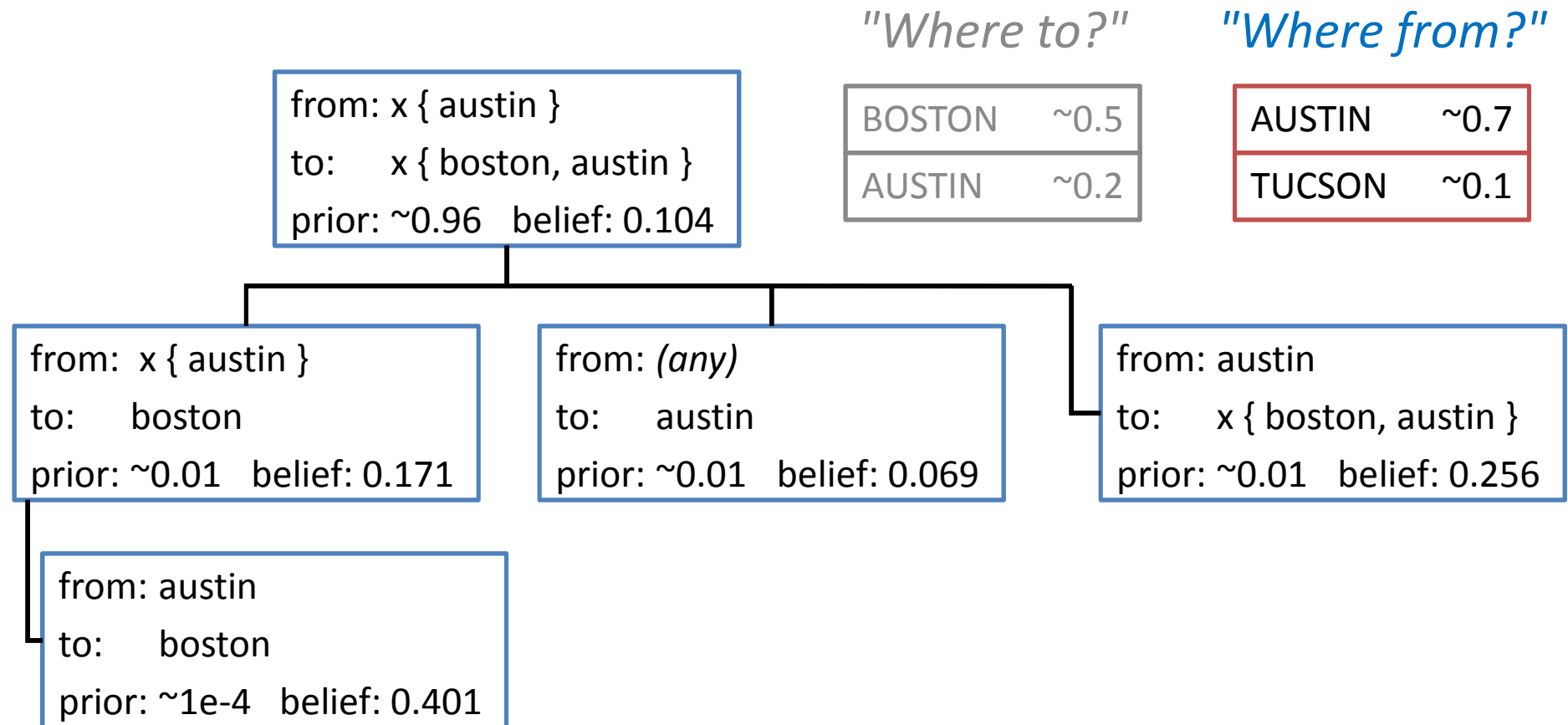
from: x { austin, tucson }  
 to: x { boston, austin }  
 prior: ~0.96 belief: 0.068



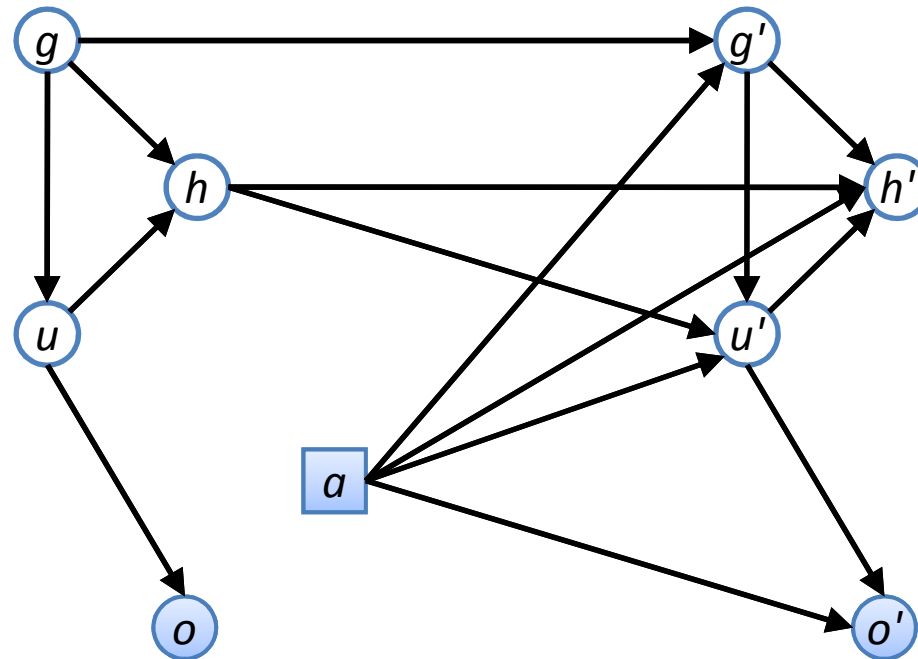
# Partition update example



# Partition update example



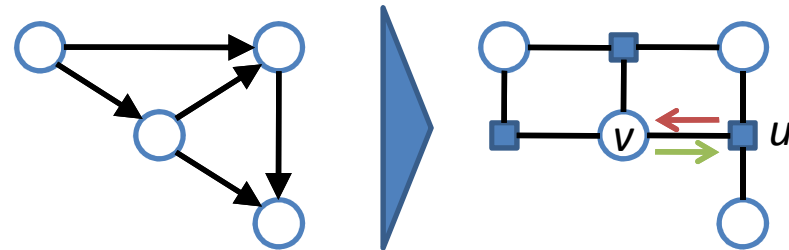
# Network-based approaches



Idea: Apply general purpose Bayes Network inference techniques  
*Approximate* inference can be much faster than exact  
Examples: loopy belief propagation and particle filters

# Loopy belief propagation

1. Re-write network as a *factor graph*



2. Instantiate *messages* – 2 for each edge

$$\mu_{v \leftarrow u}(x_v) = 1 \quad \leftarrow$$

$$\mu_{v \rightarrow u}(x_v) = 1 \quad \rightarrow$$

3. Iteratively updates message (next slide)

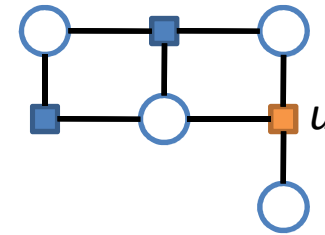
4. Compute estimated marginals

$$b_v(x_v) = \eta_v \prod_{u \in N(v)} \mu_{v \leftarrow u}(x_v)$$

*B. Thomson and S. Young (2009). "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems." Computer Speech and Language, To appear.*

# Loopy belief propagation

repeat until convergence:



choose a factor  $u$  to update

for each variable  $v$  connected to  $u$ :

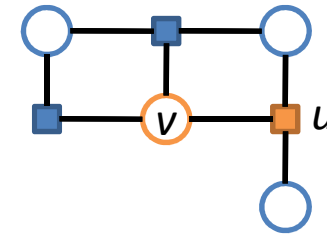
$$\forall x_v \quad \mu_{v \leftarrow u}(x_v) = \sum_{\mathbf{x}'_u : x'_v = x_v} f_u(\mathbf{x}'_u) \prod_{v^* \in \mathbf{N}(u) - v} \mu_{v^* \rightarrow u}(x'_{v^*})$$

for each factor  $u' \neq u$  connected to  $v$

$$\forall x_v \quad \mu_{v \rightarrow u'}(x_v) = \prod_{u^* \in \mathbf{N}(v) - u'} \mu_{v \rightarrow u^*}(x_v)$$

# Loopy belief propagation

repeat until convergence:



choose a factor  $u$  to update

for each variable  $v$  connected to  $u$ :

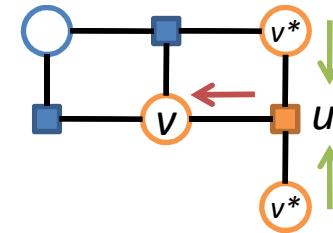
$$\forall x_v \quad \mu_{v \leftarrow u}(x_v) = \sum_{\mathbf{x}'_u : x'_v = x_v} f_u(\mathbf{x}'_u) \prod_{v^* \in \mathbf{N}(u) - v} \mu_{v^* \rightarrow u}(x'_{v^*})$$

for each factor  $u' \neq u$  connected to  $v$

$$\forall x_v \quad \mu_{v \rightarrow u'}(x_v) = \prod_{u^* \in \mathbf{N}(v) - u'} \mu_{v \rightarrow u^*}(x_v)$$

# Loopy belief propagation

repeat until convergence:



choose a factor  $u$  to update

for each variable  $v$  connected to  $u$ :

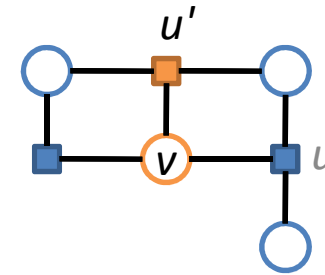
$$\forall x_v \quad \mu_{v \leftarrow u}(x_v) = \sum_{\mathbf{x}'_u : x'_v = x_v} f_u(\mathbf{x}'_u) \prod_{v^* \in \mathbf{N}(u) - v} \mu_{v^* \rightarrow u}(x'_{v^*})$$

for each factor  $u' \neq u$  connected to  $v$

$$\forall x_v \quad \mu_{v \rightarrow u'}(x_v) = \prod_{u^* \in \mathbf{N}(v) - u'} \mu_{v \rightarrow u^*}(x_v)$$

# Loopy belief propagation

repeat until convergence:



choose a factor  $u$  to update

for each variable  $v$  connected to  $u$ :

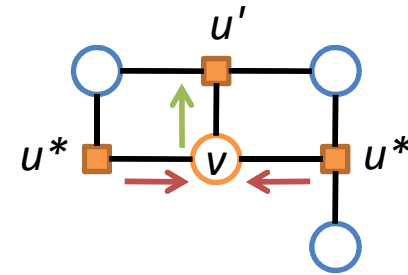
$$\forall x_v \quad \mu_{v \leftarrow u}(x_v) = \sum_{\mathbf{x}'_u : x'_v = x_v} f_u(\mathbf{x}'_u) \prod_{v^* \in \mathbf{N}(u) - v} \mu_{v^* \rightarrow u}(x'_{v^*})$$

for each factor  $u' \neq u$  connected to  $v$

$$\forall x_v \quad \mu_{v \rightarrow u'}(x_v) = \prod_{u^* \in \mathbf{N}(v) - u'} \mu_{v \rightarrow u^*}(x_v)$$

# Loopy belief propagation

repeat until convergence:



choose a factor  $u$  to update

for each variable  $v$  connected to  $u$ :

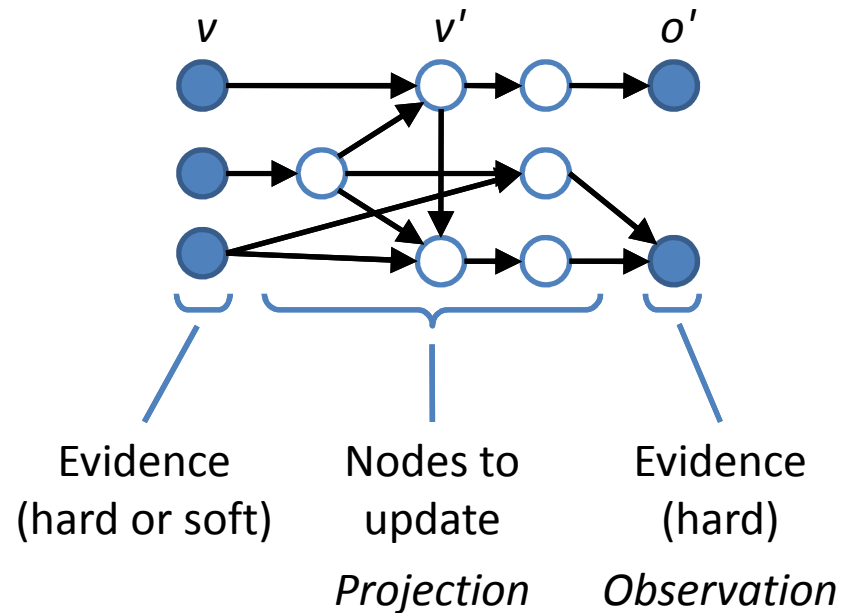
$$\forall x_v \quad \mu_{v \leftarrow u}(x_v) = \sum_{\mathbf{x}'_u : x'_v = x_v} f_u(\mathbf{x}'_u) \prod_{v^* \in \mathbf{N}(u) - v} \mu_{v^* \rightarrow u}(x'_{v^*})$$

for each factor  $u' \neq u$  connected to  $v$

$$\forall x_v \quad \mu_{v \rightarrow u'}(x_v) = \prod_{u^* \in \mathbf{N}(v) - u'} \mu_{v \leftarrow u^*}(x_v)$$

# Particle filters

Insight: Network to update has this *family* of structure:

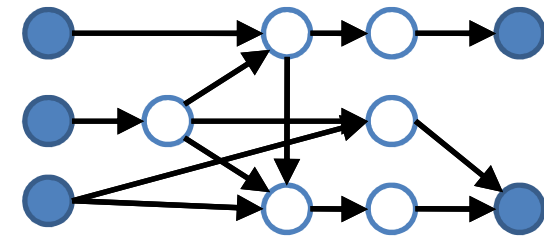


This *projection – observation* structure is well-suited to filtering methods, such as particle filters

*Jason D. Williams. 2007. Using Particle Filters to Track Dialogue State. Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan.*

# Particle filters

for  $i = [1, N]$  particles:



$x_v^* \sim b_v, \forall v$  in previous timestep

$x_v^i \sim P(v' | x_1^*, \dots, x_V^*, x_1^i, \dots, x_{v-1}^i), \forall v$  in this timestep

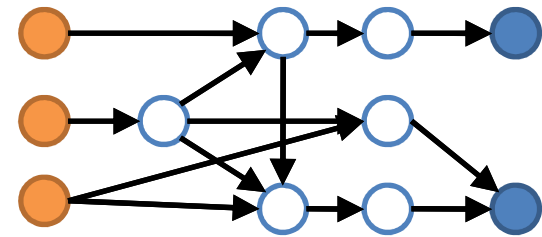
$w^i = P(o' | x_1^i, \dots, x_V^i)$

compute estimated marginals

$$b_v(x_v) = \eta_v \sum_{i: x_v^i = x_v} w^i, \forall v \text{ in this timestep}$$

# Particle filters

for  $i = [1, N]$  particles:



$$x_v^* \sim b_v, \forall v \text{ in previous timestep}$$

$$x_v^i \sim P(v' \mid x_1^*, \dots, x_V^*, x_1^i, \dots, x_{v-1}^i), \forall v \text{ in this timestep}$$

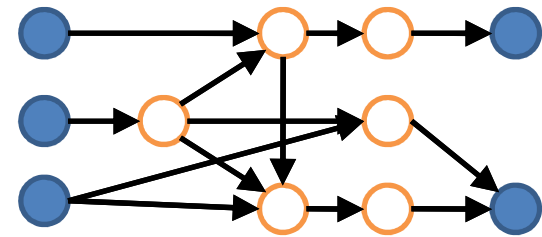
$$w^i = P(o' \mid x_1^i, \dots, x_V^i)$$

compute estimated marginals

$$b_v(x_v) = \eta_v \sum_{i: x_v^i = x_v} w^i, \forall v \text{ in this timestep}$$

# Particle filters

for  $i = [1, N]$  particles:



$x_v^* \sim b_v, \forall v$  in previous timestep

$x_v^i \sim P(v' | x_1^*, \dots, x_V^*, x_1^i, \dots, x_{v-1}^i), \forall v$  in this timestep

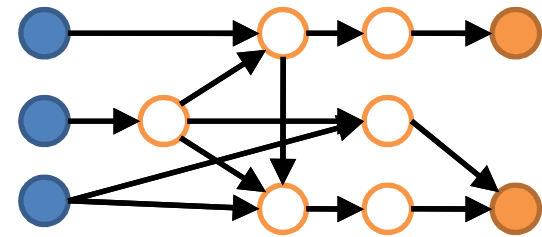
$w^i = P(o' | x_1^i, \dots, x_V^i)$

compute estimated marginals

$$b_v(x_v) = \eta_v \sum_{i: x_v^i = x_v} w^i, \forall v \text{ in this timestep}$$

# Particle filters

for  $i = [1, N]$  particles:



$x_v^* \sim b_v, \forall v$  in previous timestep

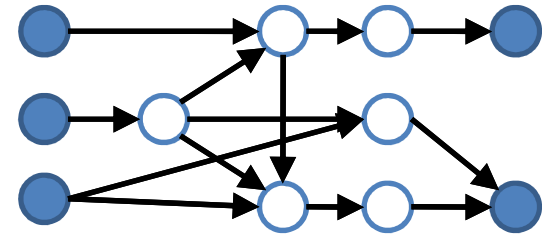
$x_v^i \sim P(v' | x_1^*, \dots, x_V^*, x_1^i, \dots, x_{v-1}^i), \forall v$  in this timestep

$$w^i = P(o' | x_1^i, \dots, x_V^i)$$

compute estimated marginals

$$b_v(x_v) = \eta_v \sum_{i: x_v^i = x_v} w^i, \forall v \text{ in this timestep}$$

# Particle filters



for  $i = [1, N]$  particles:

$x_v^* \sim b_v, \forall v$  in previous timestep

$x_v^i \sim P(v' | x_1^*, \dots, x_V^*, x_1^i, \dots, x_{v-1}^i), \forall v$  in this timestep

$w^i = P(o' | x_1^i, \dots, x_V^i)$

compute estimated marginals

$$b_v(x_v) = \eta_v \sum_{i: x_v^i = x_v} w^i, \forall v \text{ in this timestep}$$

# Summary: efficient belief monitoring

	M-Best	Factorization
Efficiency gain	<ul style="list-style-type: none"><li>• Track only important dialog states</li></ul>	<ul style="list-style-type: none"><li>• Model only important dependencies</li></ul>
Speed vs. accuracy parameter	<ul style="list-style-type: none"><li>• Number of dialog states to track</li></ul>	<ul style="list-style-type: none"><li>• Accuracy of update (# of iterations/particles)</li></ul>
Form of belief state	<ul style="list-style-type: none"><li>+ Full joint over dialog states</li></ul>	<ul style="list-style-type: none"><li>- Marginal over each variable</li></ul>
Dependencies modeled	<ul style="list-style-type: none"><li>+ Models all state variable dependencies</li></ul>	<ul style="list-style-type: none"><li>- Adding dependencies adds computation</li></ul>
Transition dynamics	<ul style="list-style-type: none"><li>- Very limited</li></ul>	<ul style="list-style-type: none"><li>+ Arbitrary transition dynamics</li></ul>

Recent work has sought to combine these two approaches:

*B. Thomson and S. Young (2009). "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems." Computer Speech and Language, To appear.*

## Summary: belief monitoring

Goal: compute  $b'(s')$  given  $b(s)$ ,  $a$ , and  $o$

SDS-POMDP:  $s = (g, u, h)$

Confidence scores: assume  $P(o|u) = P(u|o)$

Real-time operation requires either constraints (like partitions) or approximations (like network methods)

## Next up (after the break)

So far,  $a$  has been given

After the break, we'll tackle how to choose  $a$  given  $b(s)$

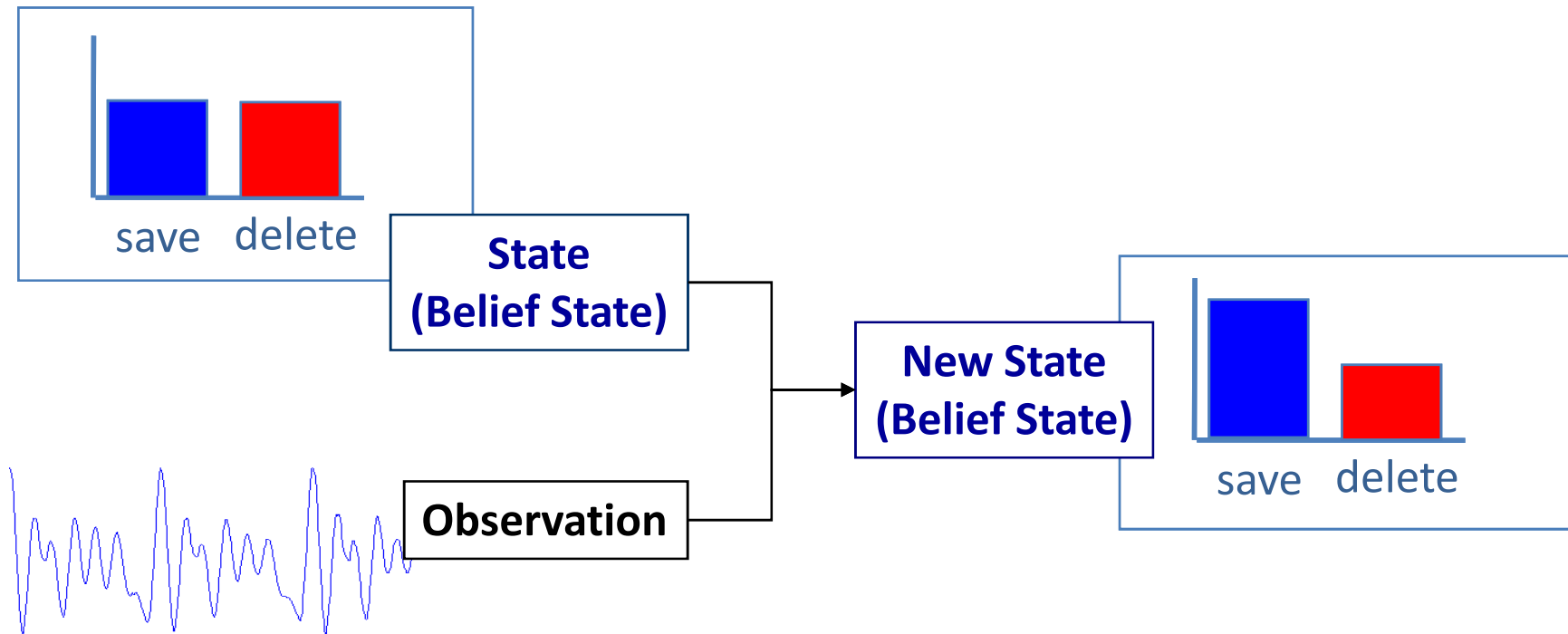
$$\pi(b) = a$$

# Action Selection / Policy Optimization

# Action Selection – Outline

1. Introduction
2. Model-based learning
3. Online learning
4. Function Approximation
5. Application to dialogue systems

# Action Selection – States & Transitions



$$(b, o) \longrightarrow T \longrightarrow b'$$

# Introduction – Actions & Policies



# Introduction – Rewards

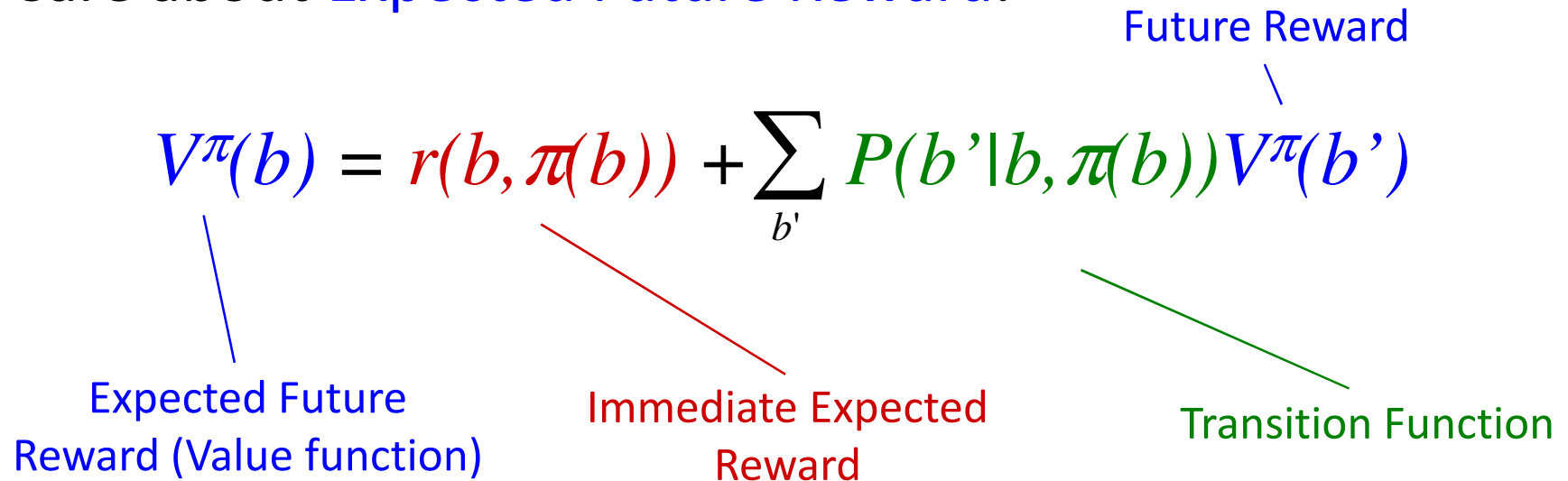
System is rewarded every turn based on:

- Environment state
- Action taken

save	delete	
-1	-1	ask
+5	-10	doSave
-20	+5	doDelete

# Introduction – What is the objective?

Care about Expected Future Reward:

$$V^\pi(b) = r(b, \pi(b)) + \sum_{b'} P(b'|b, \pi(b)) V^\pi(b')$$


Expected Future Reward (Value function)

Immediate Expected Reward

Transition Function

Future Reward

Choose  $\pi$  to maximise  $V^\pi(b_0)$ :

Start State

## Introduction – The MDP special case

- Define a finite set of options for beliefs
- Define transitions by hand
- Rewards a function of belief state



Simplifies policy learning – we will start with MDPs

# Model-based learning – An MDP example

## States

Definitely Delete  
Probably Delete  
Don't know  
Probably Save  
Definitely Save

## Transitions

T(ask)	DSave	PSave	?	PDel	DDel
DSave	0.8	0.1	0.1	0	0
PSave	0.5	0.2	0.1	0.1	0.1
?	0.2	0.2	0.2	0.2	0.2
PDel	0.1	0.1	0.1	0.2	0.5
DDel	0	0	0.1	0.1	0.8

## Actions

ask  
doSave  
doDelete

## Rewards

DSave	PSave	?	PDel	DDel	
-1	-1	-1	-1	-1	ask
4	-2	-8	-16	-20	doSave
-20	-16	-8	-2	4	doDelete

# Model-based learning – An MDP example

## Transitions

T(ask)	DSave	PSave	?	PDel	DDel
DSave	0.8	0.1	0.1	0	0
PSave	0.5	0.2	0.1	0.1	0.1
?	0.2	0.2	0.2	0.2	0.2
PDel	0.1	0.1	0.1	0.2	0.5
DDel	0	0	0.1	0.1	0.8

A transition **matrix**  
for each action:  $T_a$

## Rewards

DSave	PSave	?	PDel	DDel	
-1	-1	-1	-1	-1	ask
4	-2	-8	-16	-20	doSave
-20	-16	-8	-2	4	doDelete

A reward **vector**  
for each action:  $r_a$

## Model-based learning – Getting $V$ given $\pi$

Aim to maximise Expected Future Reward:

$$V^\pi(b) = r(b, \pi(b)) + \sum_{b'} P(b'|b, \pi(b)) V^\pi(b')$$

Replace with matrix notation (finite states):

$$V^\pi = r^\pi + T^\pi V^\pi$$

$$r^\pi(b) = r(b, \pi(b))$$

$$T^\pi(b, b') = P(b'|b, \pi(b))$$

Solve using matrix inverse

# Model-based learning – Optimising $\pi$

## 1. Starting with any policy $\pi$

DSave	ask	doSave	doDelete
PSave	ask	doSave	doDelete
?	ask	doSave	doDelete
PDel	ask	doSave	doDelete
DDel	ask	doSave	doDelete

## 2. Calculate Value function

$$V^\pi = r^\pi + T^\pi V^\pi$$

# Model-based learning – Optimising $\pi$

3a. See what happens for each state if:

*We start with ANY action*

*THEN follow the policy*

**FIRST TURN**

PSave	ask	doSave	doDelete
-------	-----	--------	----------

**THEN**

DSave	ask	doSave	doDelete
PSave	ask	doSave	doDelete
?	ask	doSave	doDelete
PDel	ask	doSave	doDelete
DDel	ask	doSave	doDelete

3b. Define  $Q(b,a)$  as the expected reward

$$Q^\pi(b,a) = r(b,a) + \sum_{b'} P(b'|b,a) V^\pi(b')$$

## Model-based learning – Optimising $\pi$

4. If  $Q^\pi(b,a) > V^\pi(b)$  then:

- $a$  is a better action than  $\pi(b)$
- Make a new policy  $\pi'$  taking the best action:

$$\pi'(b) = \arg \max_a Q^\pi(b,a)$$

- This will always improve things:

$$V^{\pi'}(b) \geq V^\pi(b)$$

## Model-based learning – Optimising $\pi$

5. Repeat the process until there is no change
  - *Calculate Value function*
  - *See what happens for other actions (calc  $Q$ )*
  - *Choose new policy based on this*

This is always an optimal policy!

# Model-based learning – Summary

- Two important stages:
  - Estimating Value function
  - Improving policy
  
- Two problems with the algorithm:
  - Need to know transition probabilities (model-based)
    - Want to do learning online
  - Need small (finite) number of belief states
    - Want to use function approximation

## Online learning – The idea

We had:

$$Q^\pi(b, a) = r(b, a) + \sum_{b'} P(b'|b, a) V^\pi(b')$$

In terms of expectation and values at time  $t$ :

$$Q^\pi(b_t, a_t) = E_\pi(r_t + Q^\pi(b_{t+1}, a_{t+1}))$$

# Online learning – The idea

We had:

We don't have this!

$$Q^\pi(b, a) = r(b, a) + \sum_{b'} \cancel{P(b'|b, a)} V^\pi(b').$$

In terms of expectation and values at time  $t$ :

We can't calculate this!

$$Q^\pi(b_t, a_t) = \cancel{E_\pi} (r_t + Q^\pi(b_{t+1}, a_{t+1})).$$

But we do have lots of samples:

$b_1, a_1, r_1, b_2$

$b_2, a_2, r_2, b_3$

$b_3, a_3, r_3, b_4$

...

## Online learning – The idea

We want:

$$Q^\pi(b_t, a_t) = E_\pi(r_t + Q^\pi(b_{t+1}, a_{t+1})),$$

in terms of samples:

$$Q^\pi(b_1, a_1) \approx r_1 + Q^\pi(b_2, a_2)$$

$$Q^\pi(b_2, a_2) \approx r_2 + Q^\pi(b_3, a_3)$$

...

$$Q^\pi(b_t, a_t) \approx r_t + Q^\pi(b_{t+1}, a_{t+1}).$$

Optimise for  $Q^\pi(b_i, a_j)$  using standard techniques!

## Online learning – Solving for Q

$$Q^\pi(b_1, a_1) \approx r_1 + Q^\pi(b_2, a_2)$$

$$Q^\pi(b_2, a_2) \approx r_2 + Q^\pi(b_3, a_3)$$

...

METHOD 1 – Least Squares:

$Q^\pi(b_t, a_t)$  are variables to be estimated

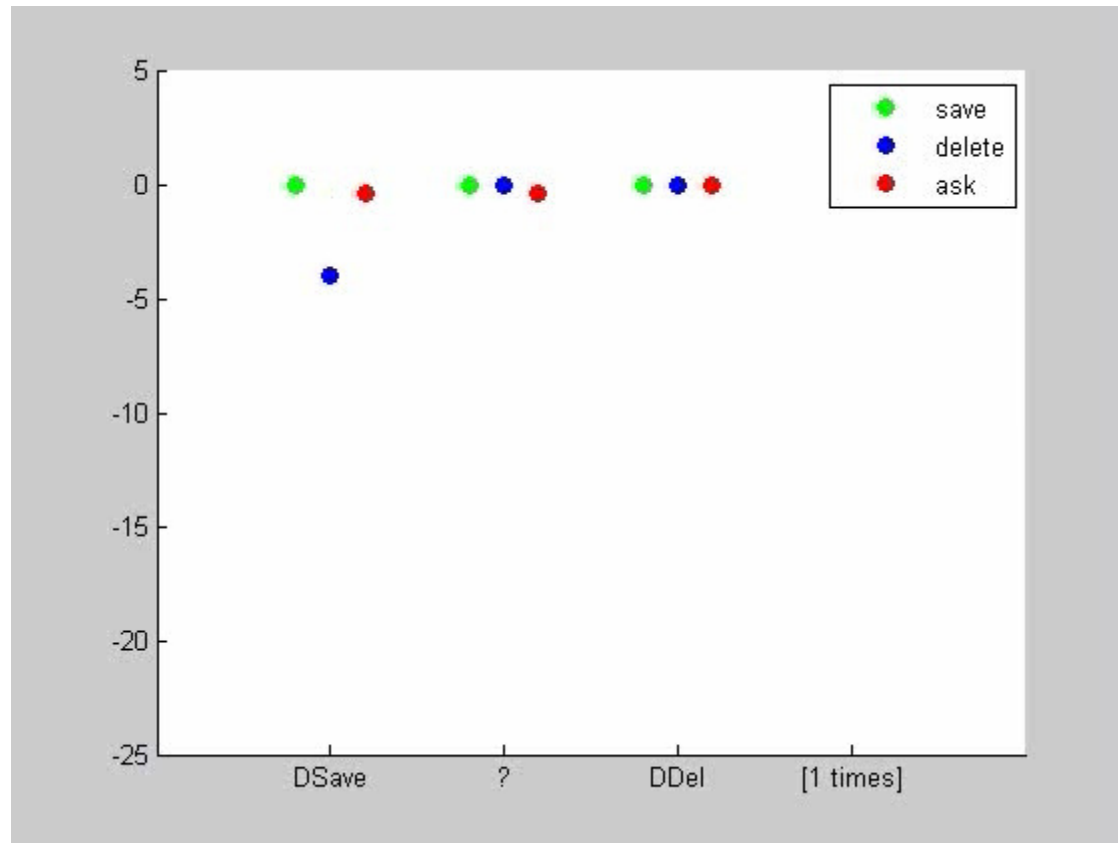
METHOD 2 – Iterative:

$$Q^\pi(b_t, a_t) \leftarrow Q^\pi(b_t, a_t) + \alpha [r_t + Q^\pi(b_{t+1}, a_{t+1}) - Q^\pi(b_t, a_t)]$$

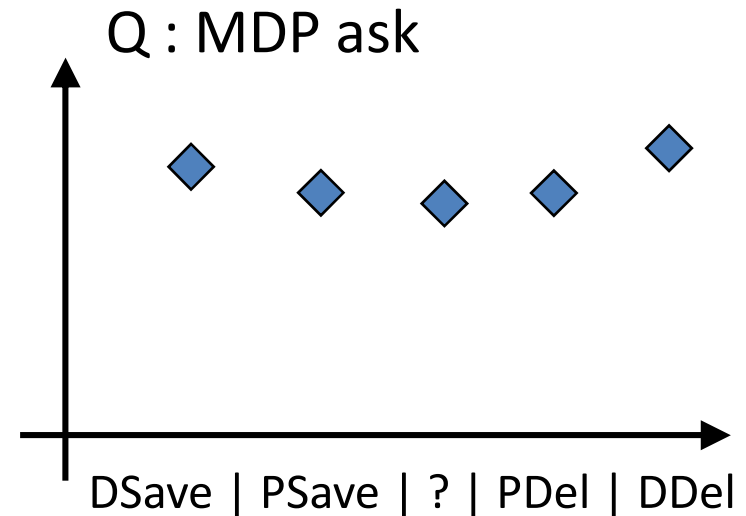
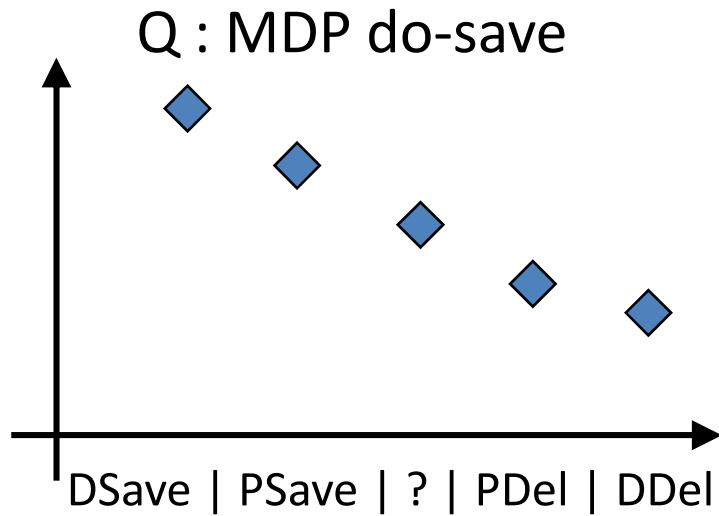
# Online learning – The algorithm

1. Follow currently optimal policy
  - Take a random action with probability  $\epsilon$  (Exploration)
2. Estimate  $Q$  using samples
  - Iterative version: SARSA(0)
  - Least squares: LS-SARSA(0)
  - SARSA = State, Action, Reward, State, Action
3. Update policy using best action given by  $Q$
4. Repeat, gradually decreasing  $\epsilon$ .

# Online learning – An example

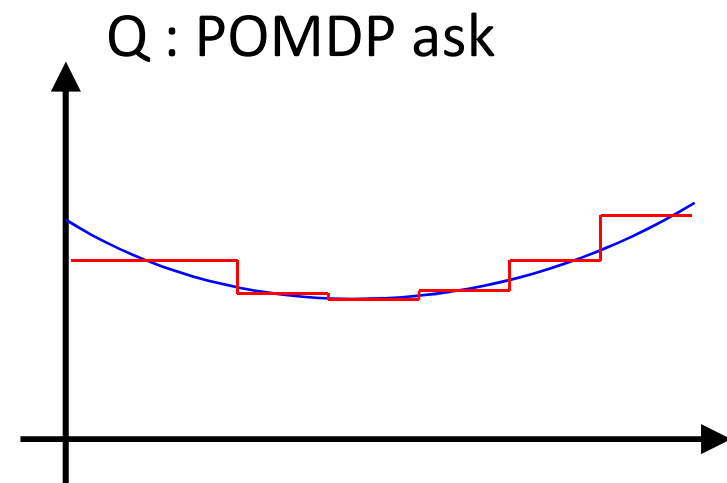
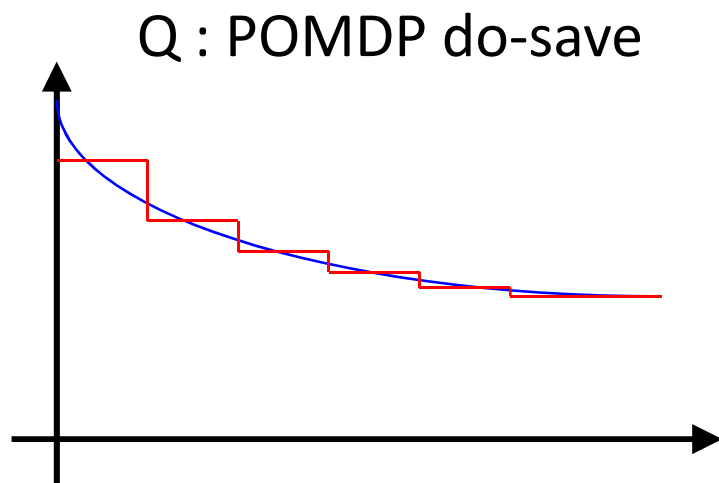
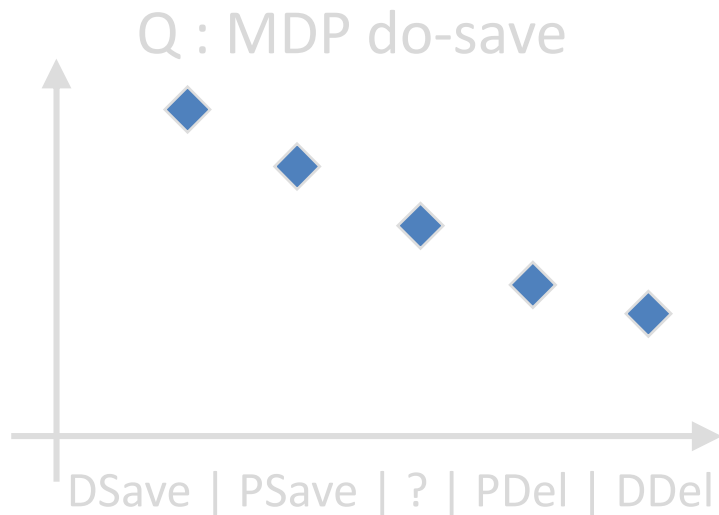


# Function approximation – The POMDP problem



*R. Sutton and A. Barto. (1998). "Reinforcement Learning: An Introduction". MIT Press*

# Function approximation – The POMDP problem



# Function approximation – Formalities

General linear form is: Summarising function  
(result is a vector)

$$Q^\pi(b, a) = \phi_a(b) \cdot \theta$$

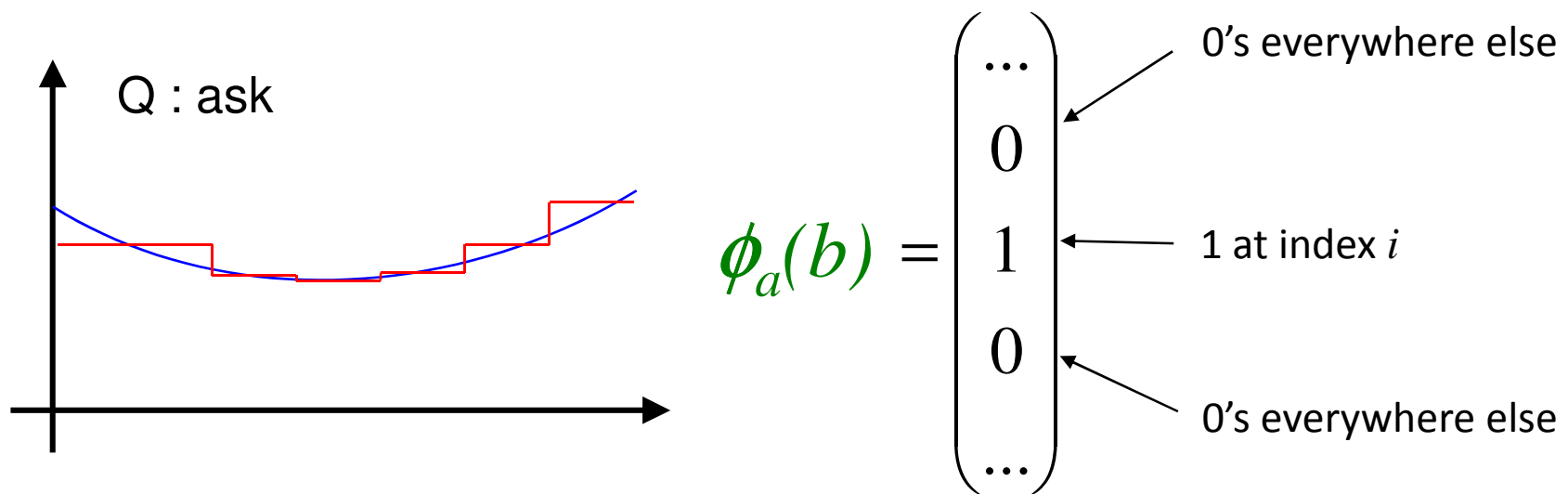
Parameter vector

# Function approximation – Formalities

Discrete approximation uses:

$$Q^\pi(b, a) = \phi_a(b) \cdot \theta = \theta_i$$

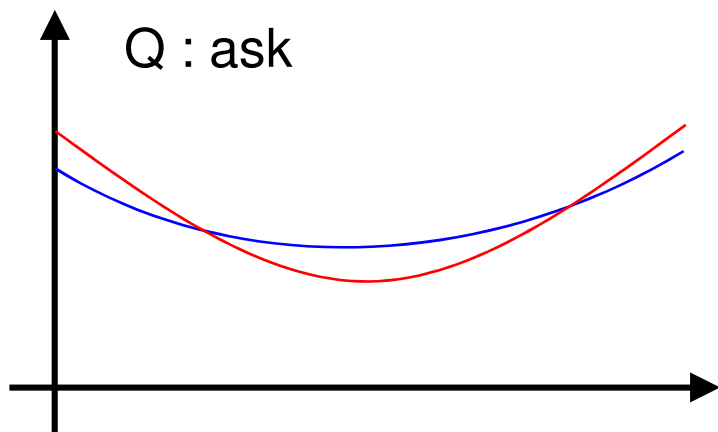
Where each (region, action) is given an index  $i$



# Function approximation – Formalities

Can use other approximations (e.g. quadratic):

$$Q^\pi(b, a) = \phi_a(b) \cdot \theta = \theta_{a,1} + b \cdot \theta_{a,2} + b^2 \cdot \theta_{a,2}$$



$$\phi_a(b) = \begin{pmatrix} \dots \\ 1 \\ b \\ b^2 \\ \dots \end{pmatrix}$$

0's for other actions

## Function approximation – The details

In the SARSA algorithms we had:

$$Q^\pi(b_1, a_1) \approx r_1 + Q^\pi(b_2, a_2)$$

$$Q^\pi(b_2, a_2) \approx r_2 + Q^\pi(b_3, a_3)$$

...

Replacing with function approximations gives:

$$\phi_{a_1}(b_1) \cdot \theta \approx r_1 + \phi_{a_2}(b_2) \cdot \theta$$

$$\phi_{a_2}(b_2) \cdot \theta \approx r_2 + \phi_{a_3}(b_3) \cdot \theta$$

...

## Function approximation – Solving for $\theta$

$$\phi_{a_1}(b_1) \cdot \theta \approx r_1 + \phi_{a_2}(b_2) \cdot \theta$$

$$\phi_{a_2}(b_2) \cdot \theta \approx r_2 + \phi_{a_3}(b_3) \cdot \theta$$

...

METHOD 1 – Least Squares:

$\theta$  are variables to be estimated

METHOD 2 – Iterative:

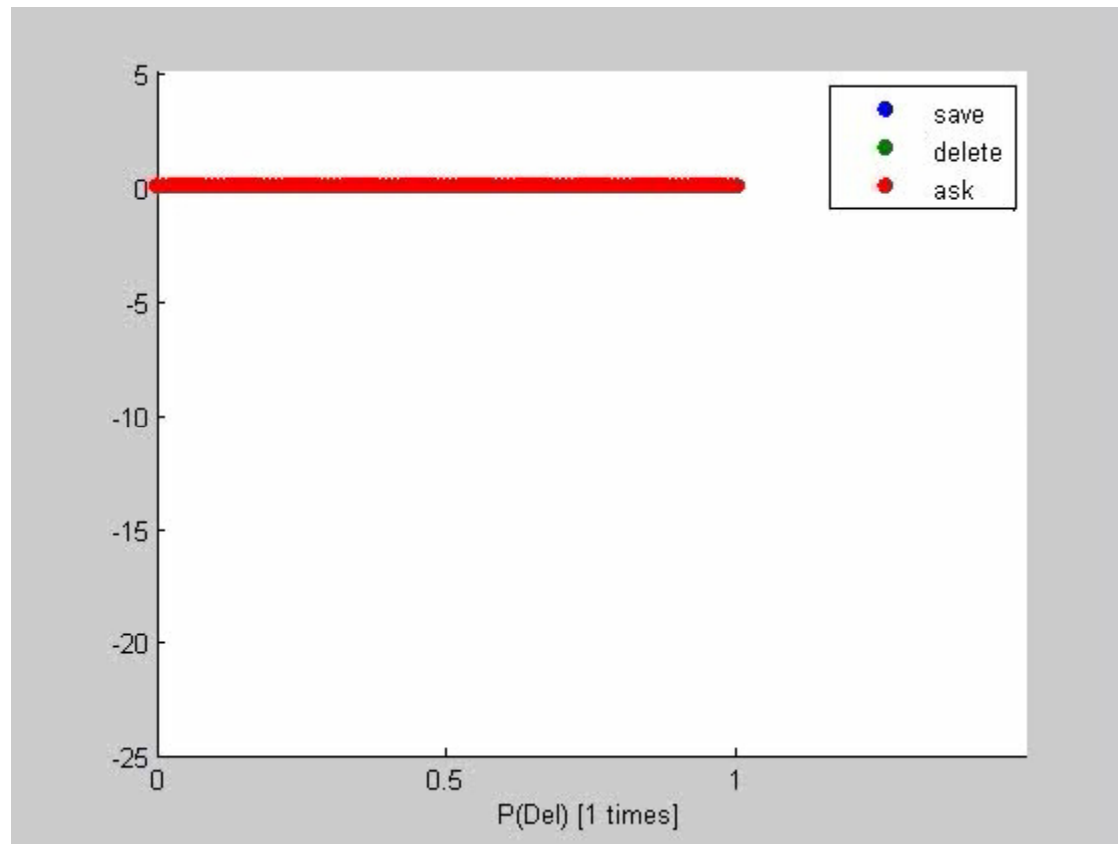
$$\theta \leftarrow \theta +$$

$$\alpha [r_t + \phi_{a_{t+1}}(b_{t+1}) \cdot \theta - \phi_{a_t}(b_t) \cdot \theta] \phi_{a_t}(b_t)$$

# Function approximation – The algorithm

1. Follow currently optimal policy
  - Take a random action with probability  $\epsilon$  (Exploration)
2. Estimate  $\theta$  using samples
  - Iterative version: SARSA(0)
  - Least squares: LS-SARSA(0)
3. Update policy using best action given by  $Q$
4. Repeat, gradually decreasing  $\epsilon$ .

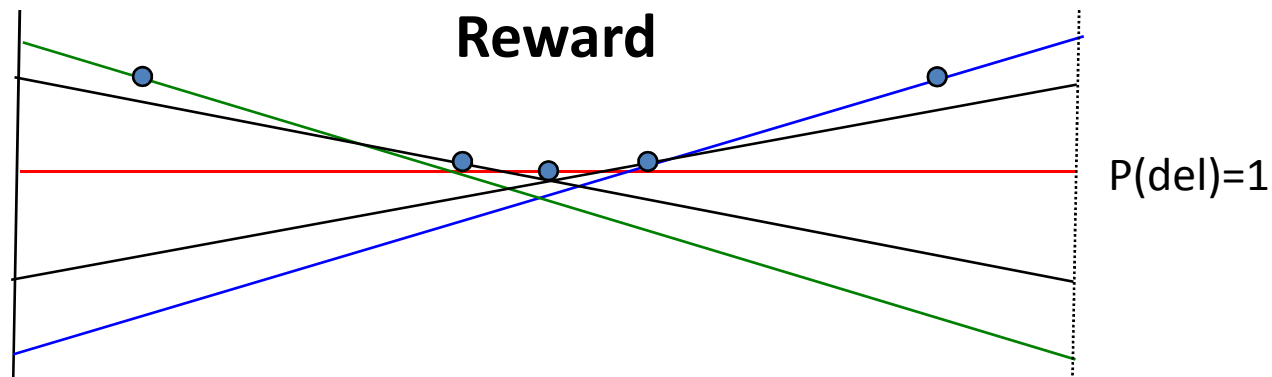
# Function approximation – An example



## Other learning techniques – POMDP approaches

Assumption of finite observations & finite look-ahead →

POMDP value function is piecewise linear



Build policy from this - Point Based Value Iteration (PBVI)

Implementations for dialogue:

Composite Summary Point Based Value Iteration (CSPBVI)

Heuristic Search Value Iteration (HSVI)

## Other learning techniques – Policy Gradients

Aim is to get probabilistic  $\pi$  to maximise:

$$V^\pi(b) = r(b, \pi(b)) + \sum_{b'} P(b'|b, \pi(b)) V^\pi(b')$$

Parameterise  $\pi$  according to  $\theta$ . Directly estimate:

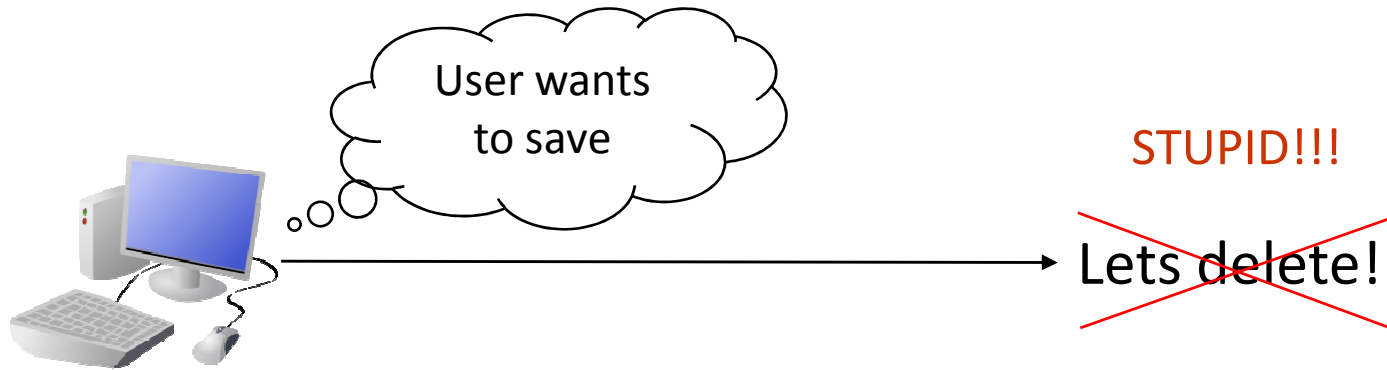
$$\frac{d}{d\theta} V^\pi(b)$$

Update using gradient descent:

$$\theta \leftarrow \theta + \alpha \frac{d}{d\theta} V^\pi(b)$$

# Application – Summary actions

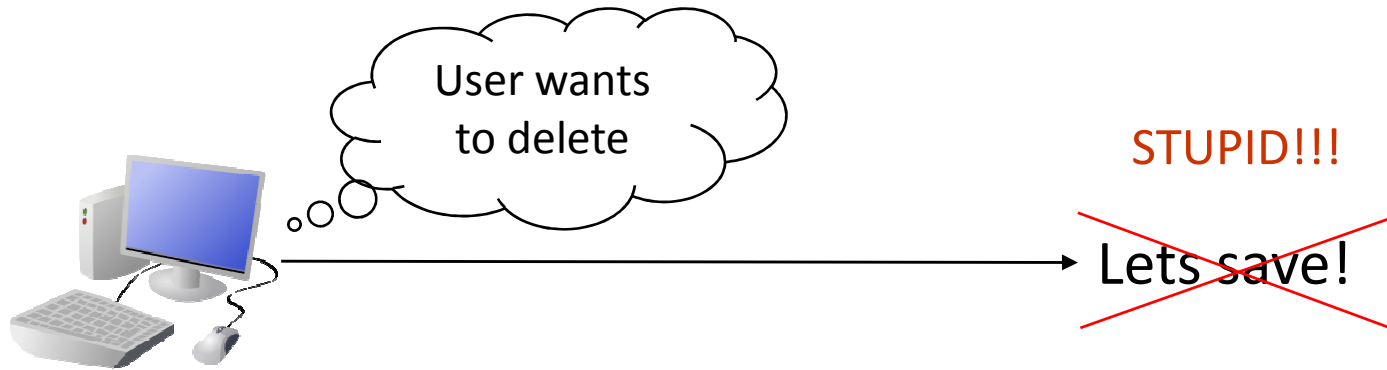
Some things are obvious!



*Jason D. Williams and Steve Young. (2005). Scaling Up POMDPs for Dialog Management: The "Summary POMDP" Method. Proc IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), San Juan, Puerto Rico, USA.*

# Application – Summary actions

Some things are obvious!



Replace “do-save” and “do-delete” with one action:  
Do whatever is the most likely

Formally still an MDP – actions simply different

## Application – “Best of both worlds”

Businesses need to be **sure** the policy:

Doesn't do something stupid

Follows business rules



Hi! My name's Bill Gates.  
Could you transfer all my  
money to A/C ....



**VERY BAD!**



Sure! It's  
done!

Policy learning should learn this from rewards

But sometimes it doesn't

And we can't have the system “exploring”

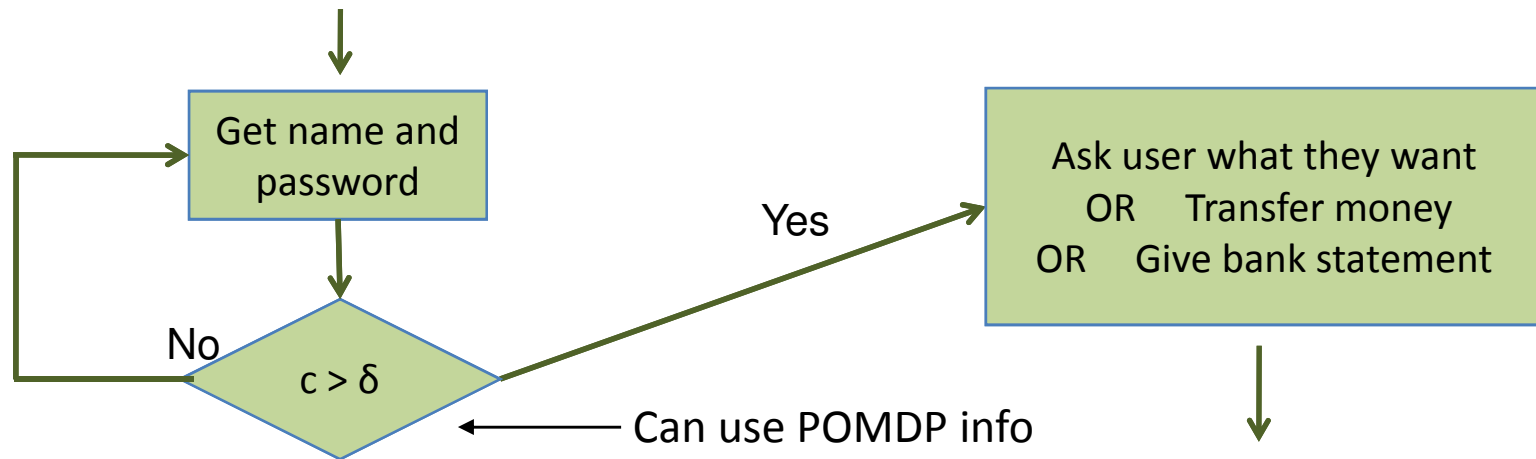
# Application – “Best of both worlds”

Augment belief state with position in a flow chart

Restrict actions to those allowed by a system designer

Constrains policy search – allows more complexity

Dialogue designers can incorporate good design

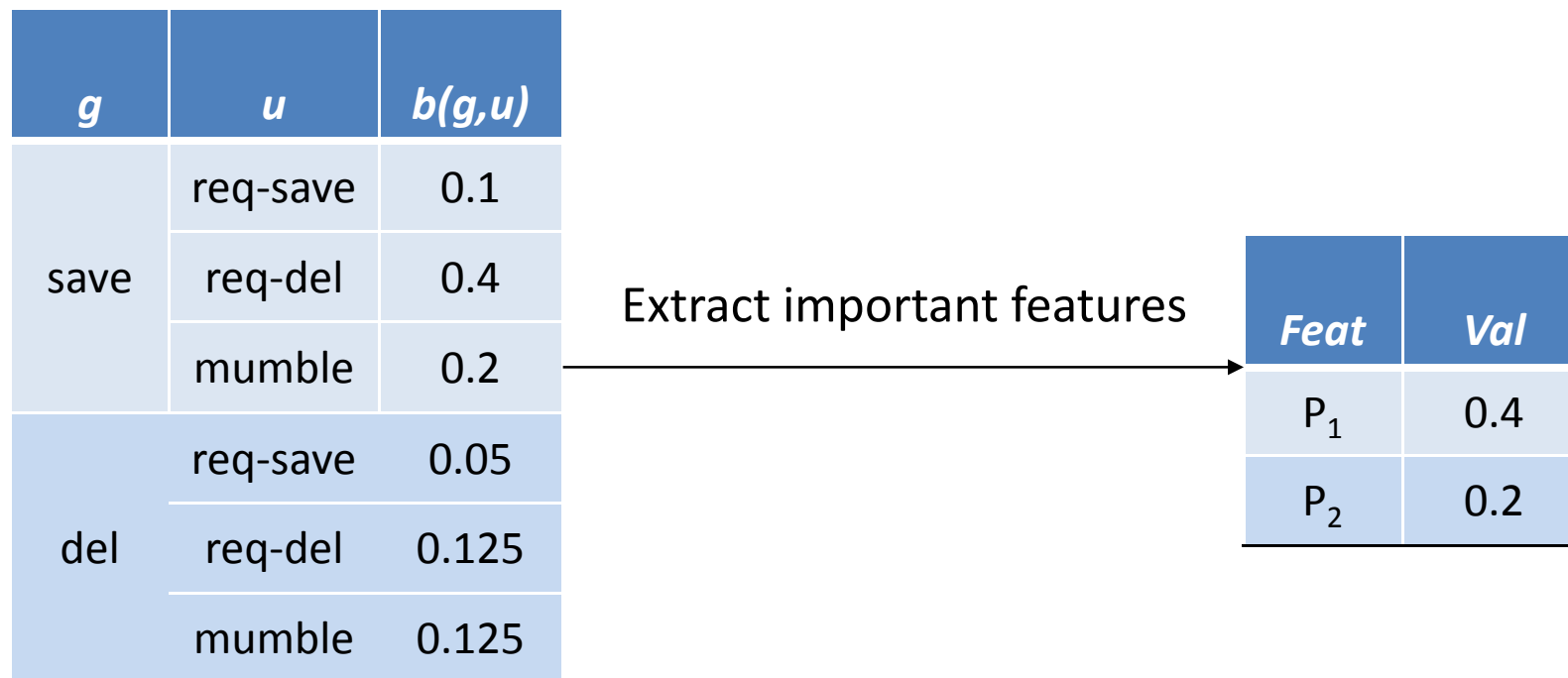


*Jason D. Williams. (2008). The best of both worlds: Unifying conventional dialog systems and POMDPs. Proc Interspeech, Brisbane, Australia.*

# Application – Summary features (for M-best)

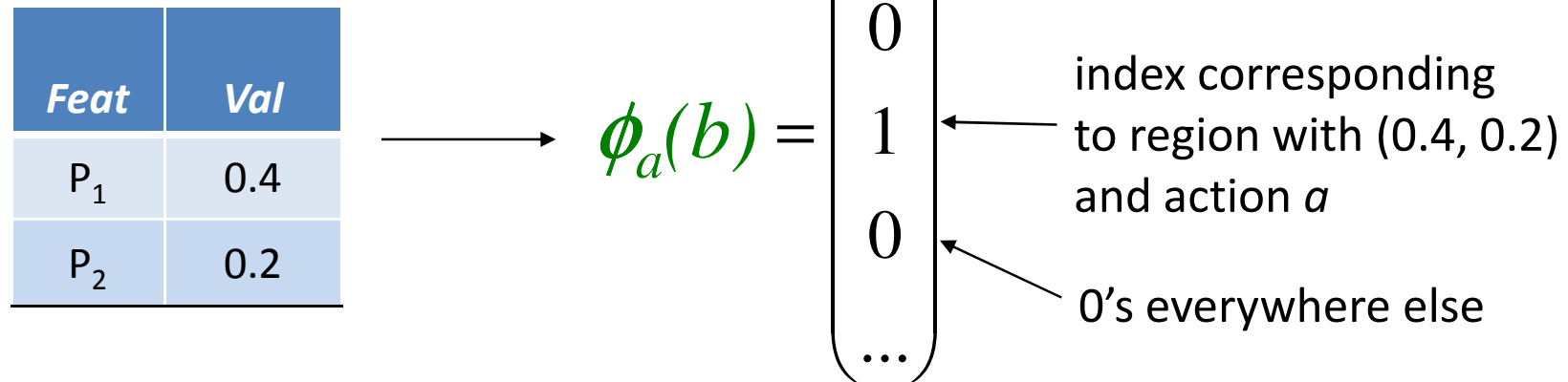
For M-best approaches:

- Probability of most likely hypothesis is important
- The probability of second most likely
- Possibly also entropy / accumulated probabilities



## Application – Summary features (for M-best)

Then discretise using a grid:



Apply learning with summary actions:

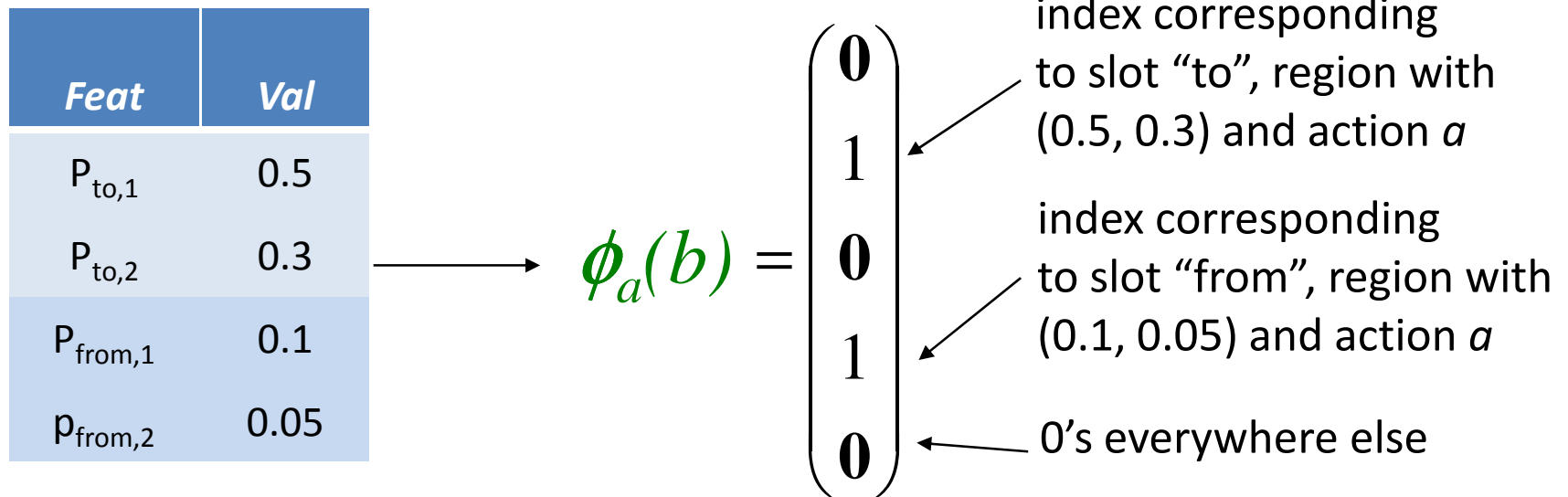
- “Do” action converts to the relevant action on the most likely hypothesis
- “Ask” action asks for more info

*S. Young (2006). "Using POMDPs for Dialog Management." IEEE/ACL Workshop on Spoken Language Technology (SLT 2006), Aruba*

# Application – Summary features (for factorised)

For factorised approaches:

- Need to use probabilities / entropies of each slot
- Too many values to discretise joint space
- Consider effect of each slot independently



*B. Thomson and S. Young (2009). "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems." Computer Speech and Language, To appear.*

## Action Selection – Summary

Generally, learning algorithms alternate between:

- Estimating Value Function
- Updating Policy

Applications to dialogue use:

- Function approximation
- Summary features
- Summary actions

Business rules can be included by restricting actions and keeping extra states

## Action Selection – Note

Many algorithms for POMDPs:

- SARSA / LS-SARSA / MCMC (Zhou et al., Young et al)
- CSPBVI (Williams and Young)
- HSVI (Kim et al)
- Policy gradient / Natural Actor Critic (Thomson et al.)
- Dynamic Decision Network (Bui et al.)
- Least Squares Policy Iteration (Li, et al.)
- Q-MDP (Henderson & Lemon)

# Practical Systems

## Implementation Approaches to POMDP-based SDS

Key issues:

- method of approximating belief space
- design of summary state and action spaces
- summary -> master space mapping
- policy representation
- optimization method
- user simulation

# Approximating the belief space

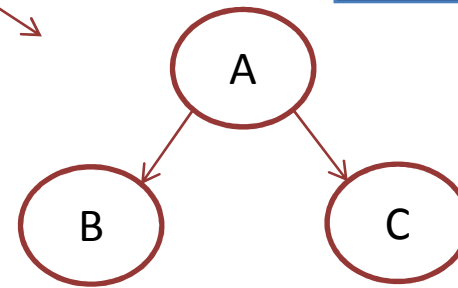
Full Joint Distribution

$P(A,B,C)$

N-Best

$P(A=a_1, B=b_2, C=c_4)=0.65$   
 $P(A=a_1, B=b_1, C=c_4)=0.21$   
 $P(A=a_1, B=b_1, C=c_1)=0.04$   
 $P(A=a_1, B=b_1, C=c_1)=0.01$   
.....

Factor

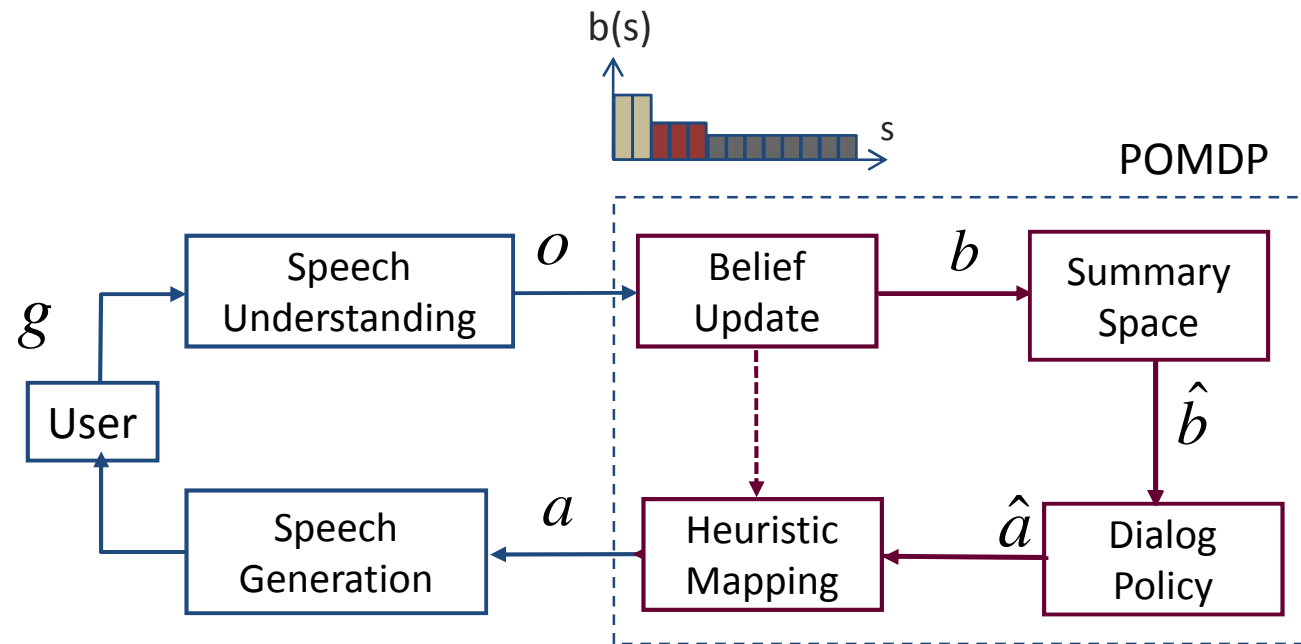


$P(B|A)P(C|A)P(A)$

DIPPER Mixture Model (Edinburgh)  
Hidden Information State (Cambridge)  
Frame-based System (POSTECH)

BUDS System (Cambridge)  
DDN POMDP (Twente)  
Particle Filter System (AT&T)

# Architecture of the Hidden Information State System



Two key ideas:

- States are grouped into equivalence classes called partitions and belief updating is applied to partitions rather than states
- Belief space is mapped into a much simpler summary space for policy implementation and optimization

*Partially Observable Markov Decision Processes for Spoken Dialog Systems, Williams & Young (CSL 2007)*  
*The Hidden Information State Approach to Dialog Management, Young et al (ICASSP 2007)*

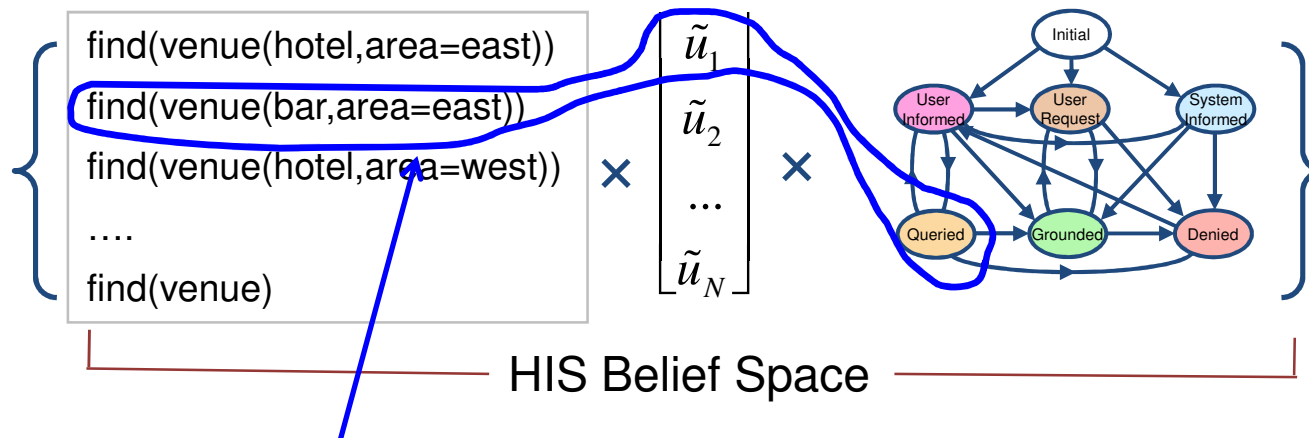
# The HIS Belief Space

Each state is composed of three factors:

$$s = \langle g, u, h \rangle$$

User Goal      User Act      Dialog History

User goals are grouped into partitions

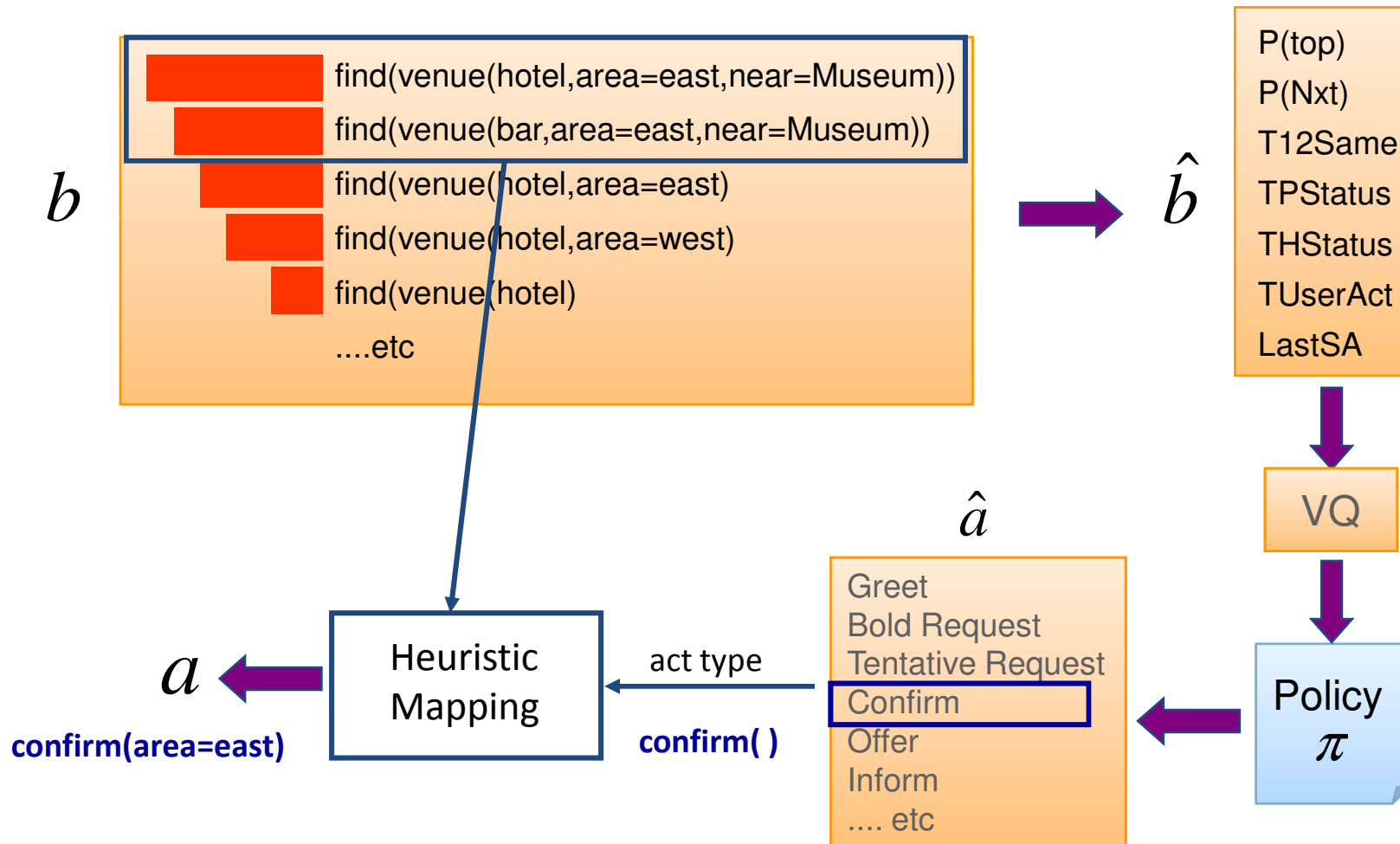


Beliefs update is limited to the most likely members of this set.

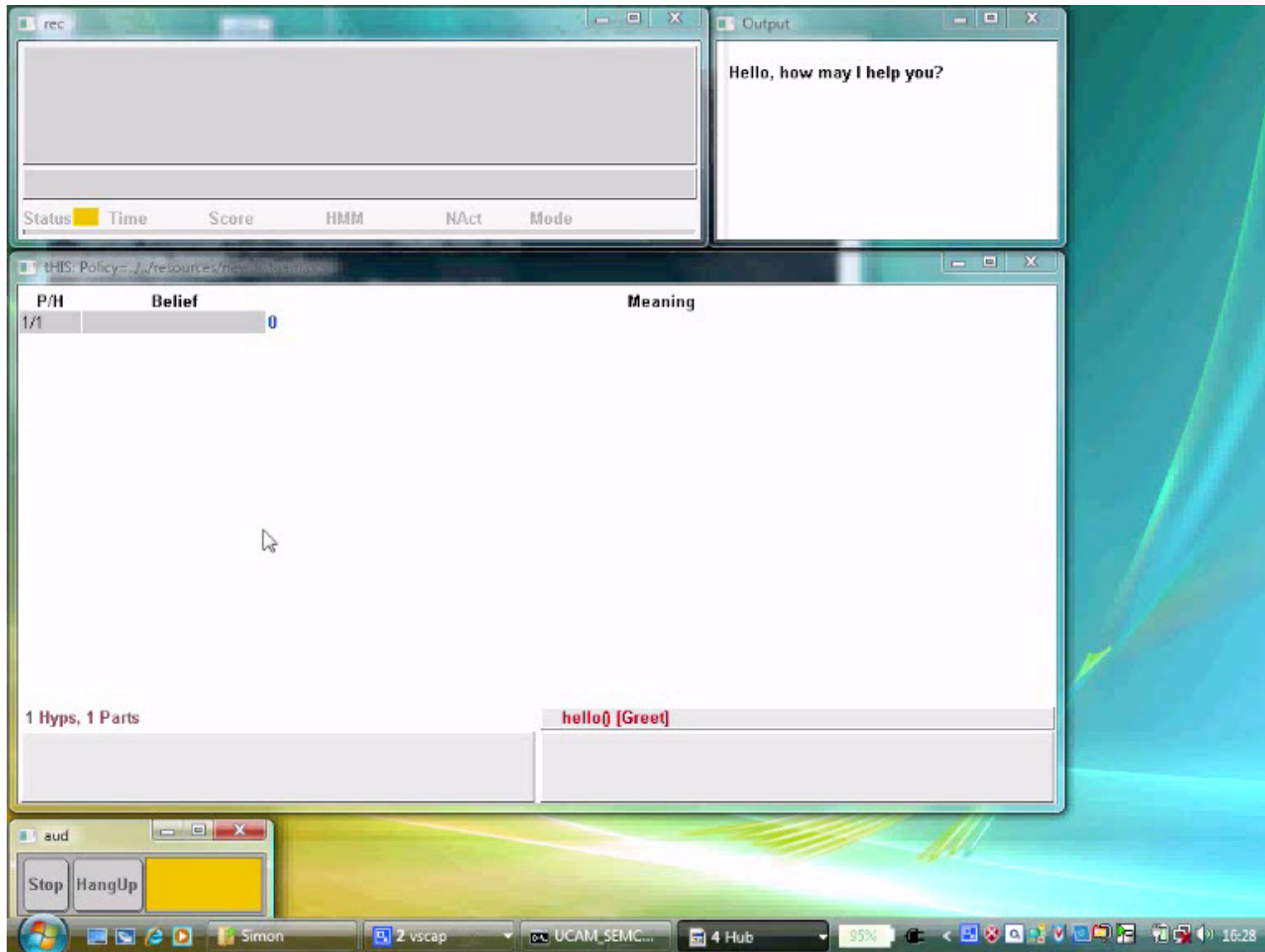
*The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management, Young et al (CSL 2009)*

# Master <-> Summary State Mapping

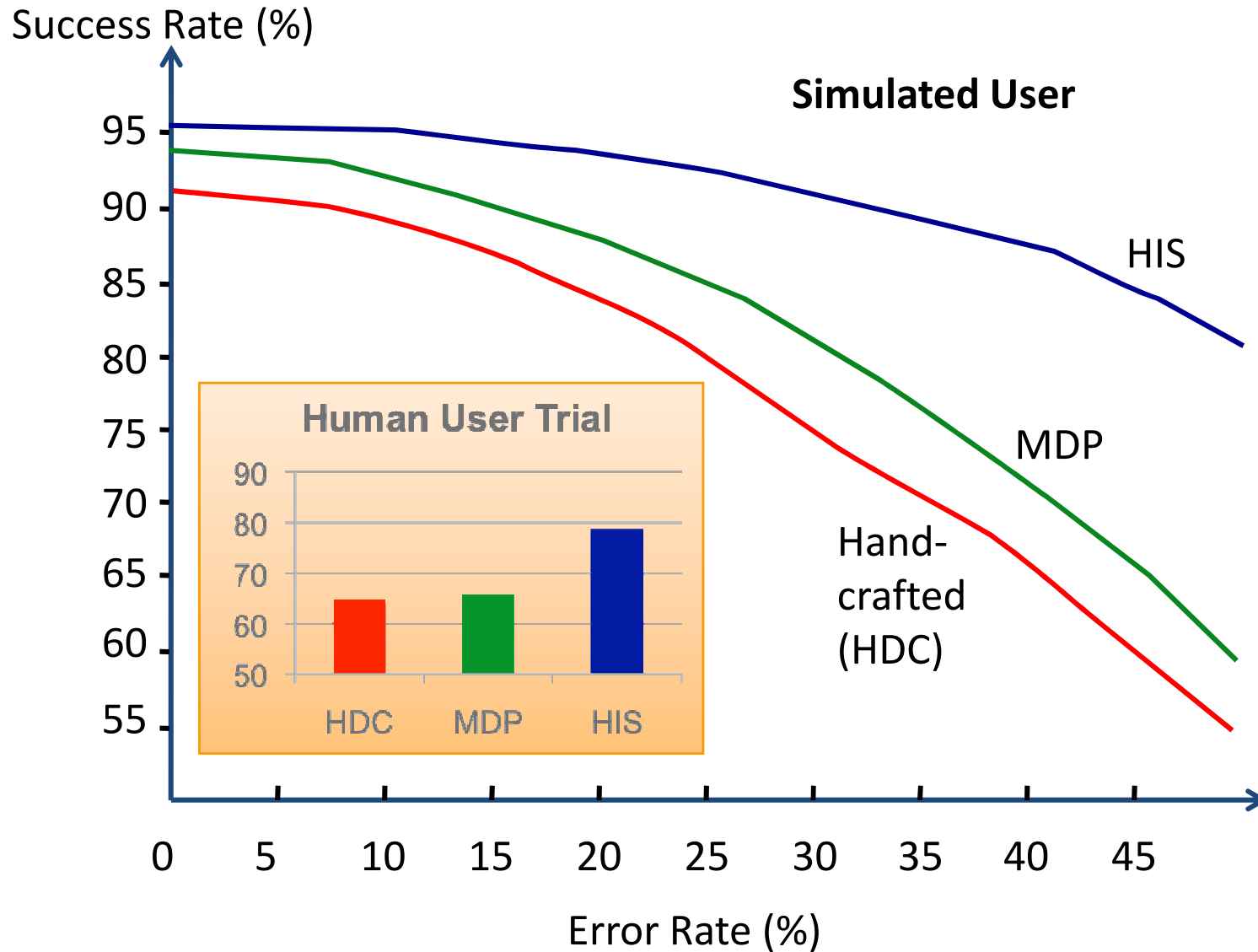
Master space is mapped into a reduced summary space:



# HIS System Demo



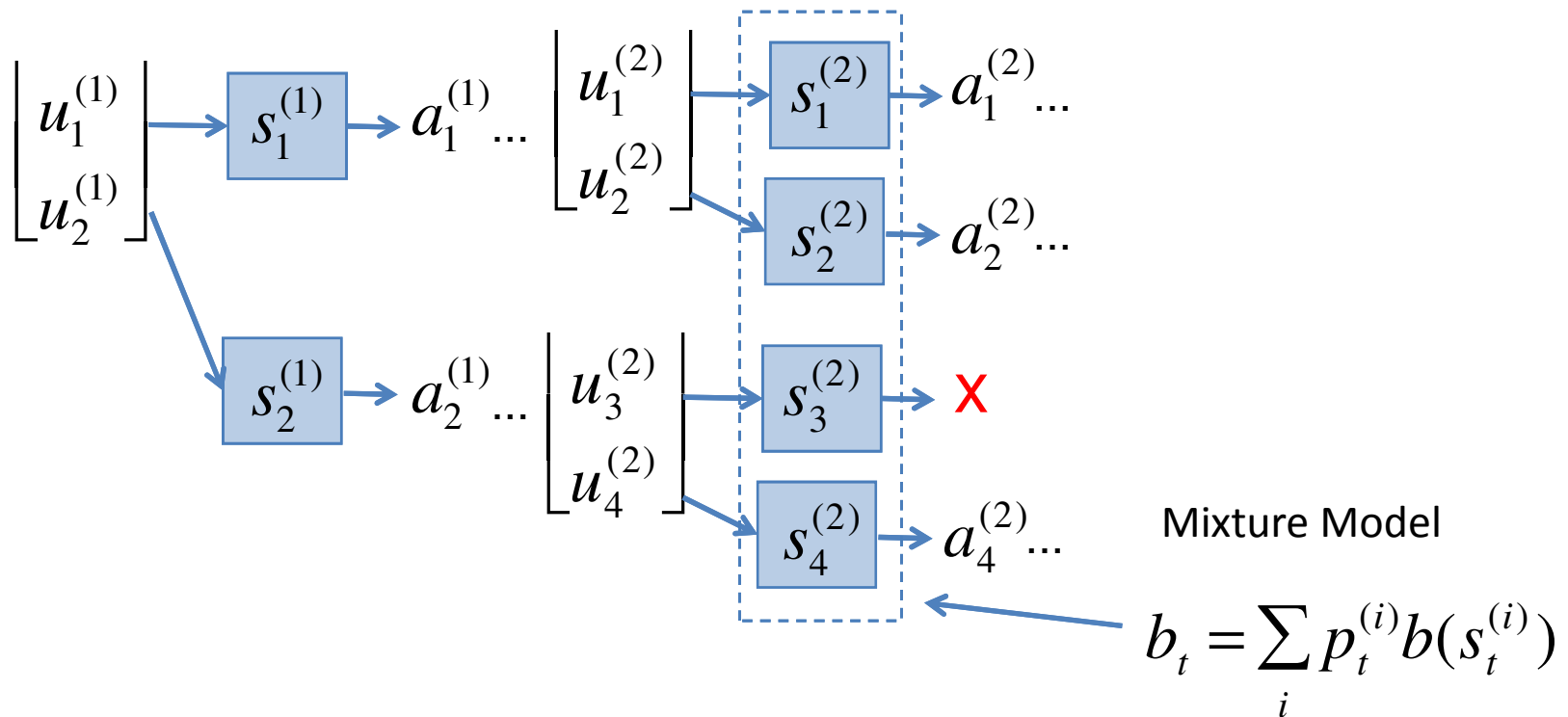
# HIS Performance in Noise



# Beam-search Interpretation of N-Best Approach

Primary source of uncertainty is generated by multiple recognition outputs

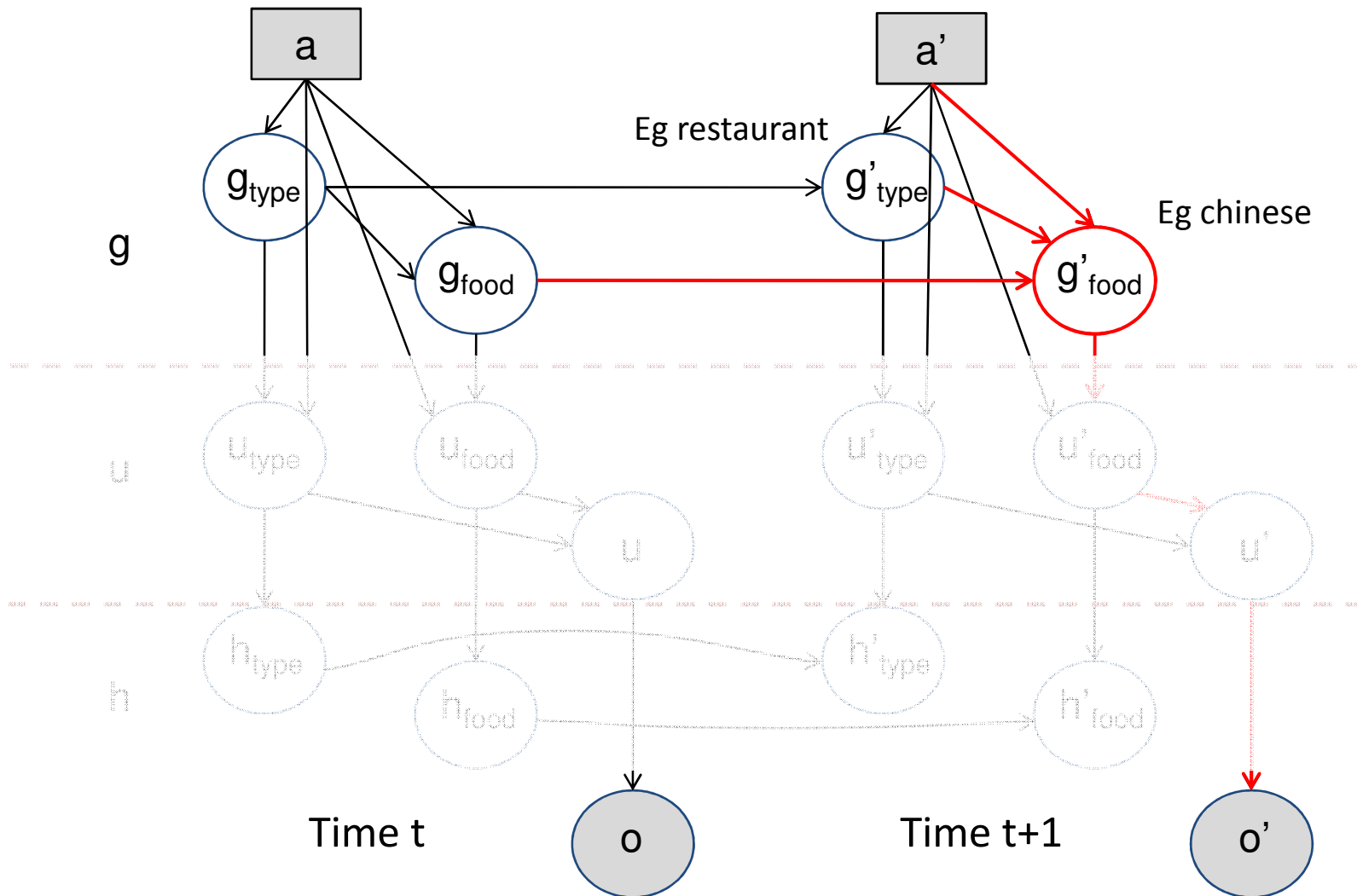
With Alternative input hypotheses



*Mixture Model POMDPs for Efficient handling of Uncertainty in Dialogue Management, Henderson and Lemon (ACL-HLT, 2008)*

# Modeling Belief with Dynamic Bayesian Networks

Decompose state into DBN, retaining only essential conditional dependencies



*Bayesian Update of Dialogue State for Robust Dialogue Systems, Thomson et al (ICASSP, 2008)*

# Policy Optimization in the BUDS System

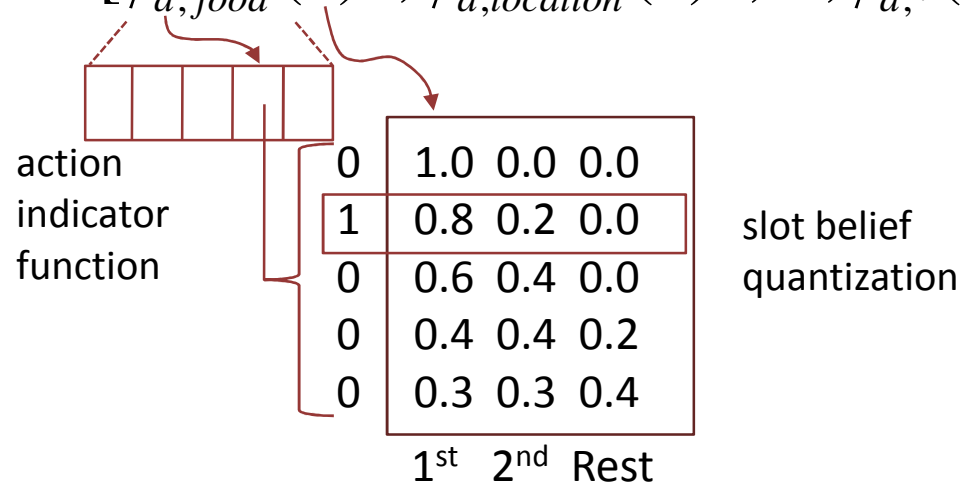
Summary space now depends on forming a simple characterization of each individual slot.

Define policy as a parametric function and optimize wrt  $\theta$  using Natural Actor Critic algorithm.

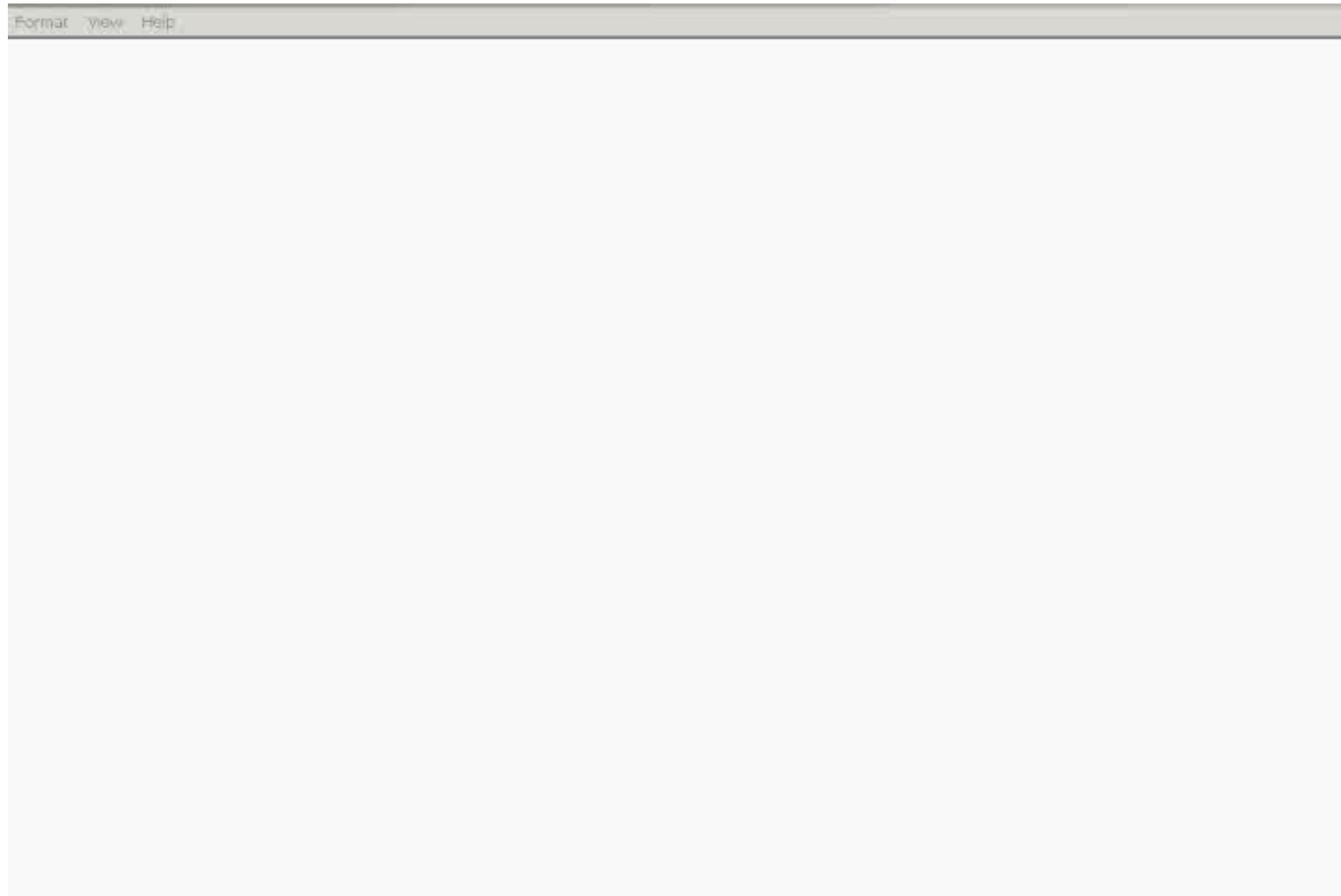
$$\pi(a | b, \theta) = \frac{e^{\theta \cdot \phi_a(b)}}{\sum_{a'} e^{\theta \cdot \phi_{a'}(b)}}$$

Each action dependent basis function is separated out into slot-based components, eg

$$\phi_a(b)^\top = [\phi_{a,food}(b)^\top, \phi_{a,location}(b)^\top, \dots, \phi_{a,*}(b)^\top]$$

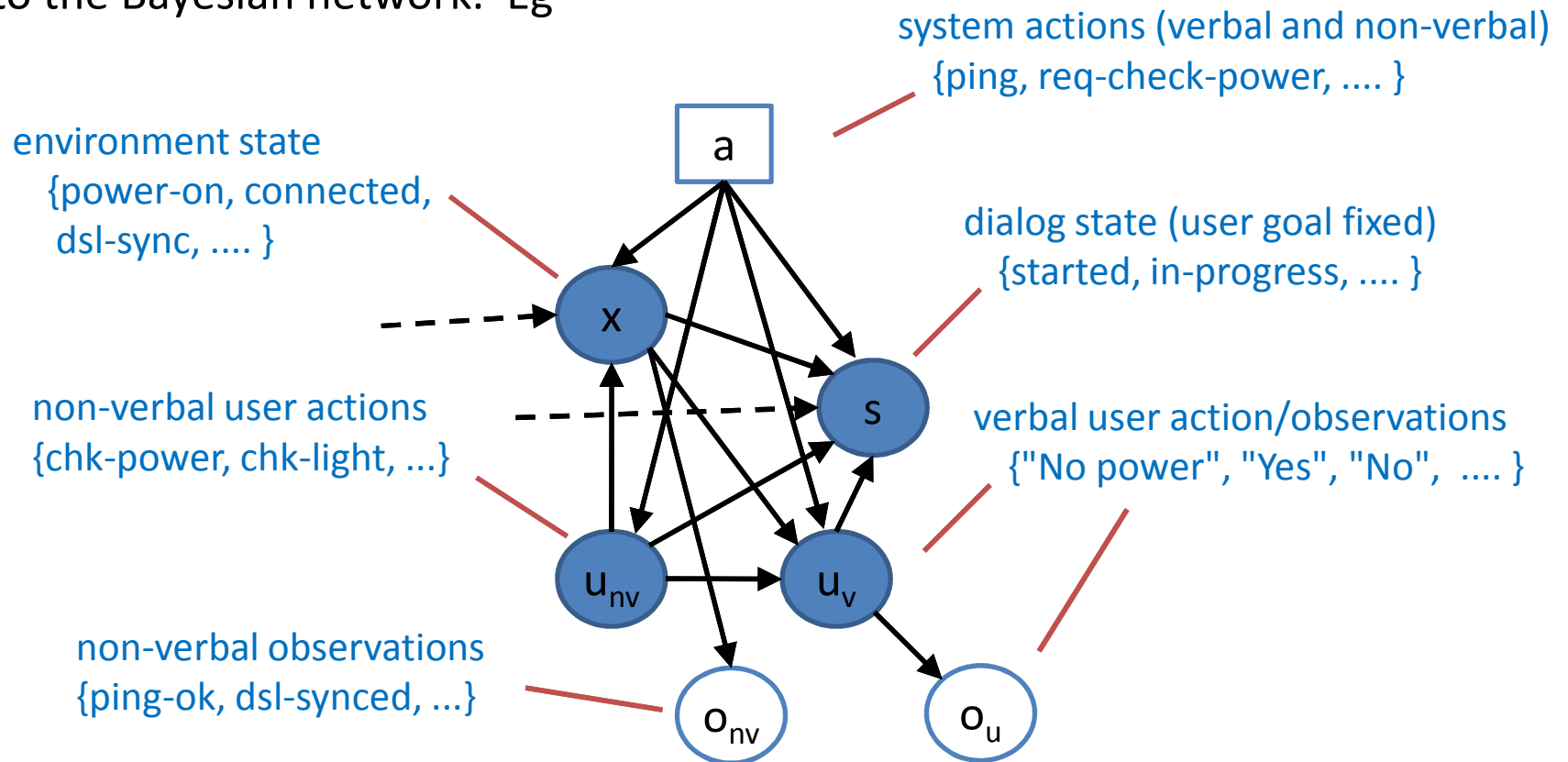


# BUDS System Demo



# AT&T's Trouble-shooting System

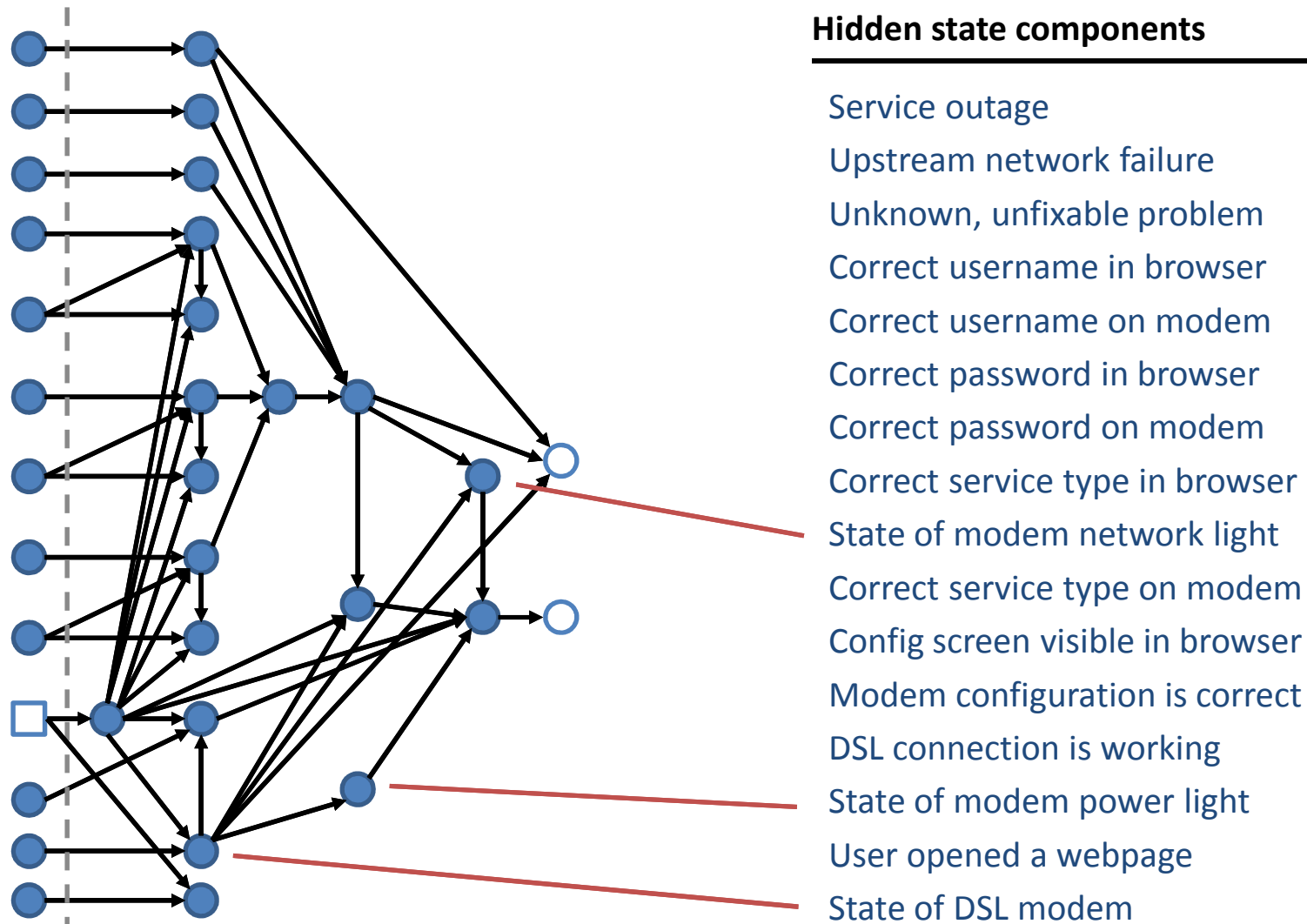
In some applications such as help-lines for DSL modem faults, there are additional sources of uncertainty. These can be easily incorporated into the Bayesian network. Eg



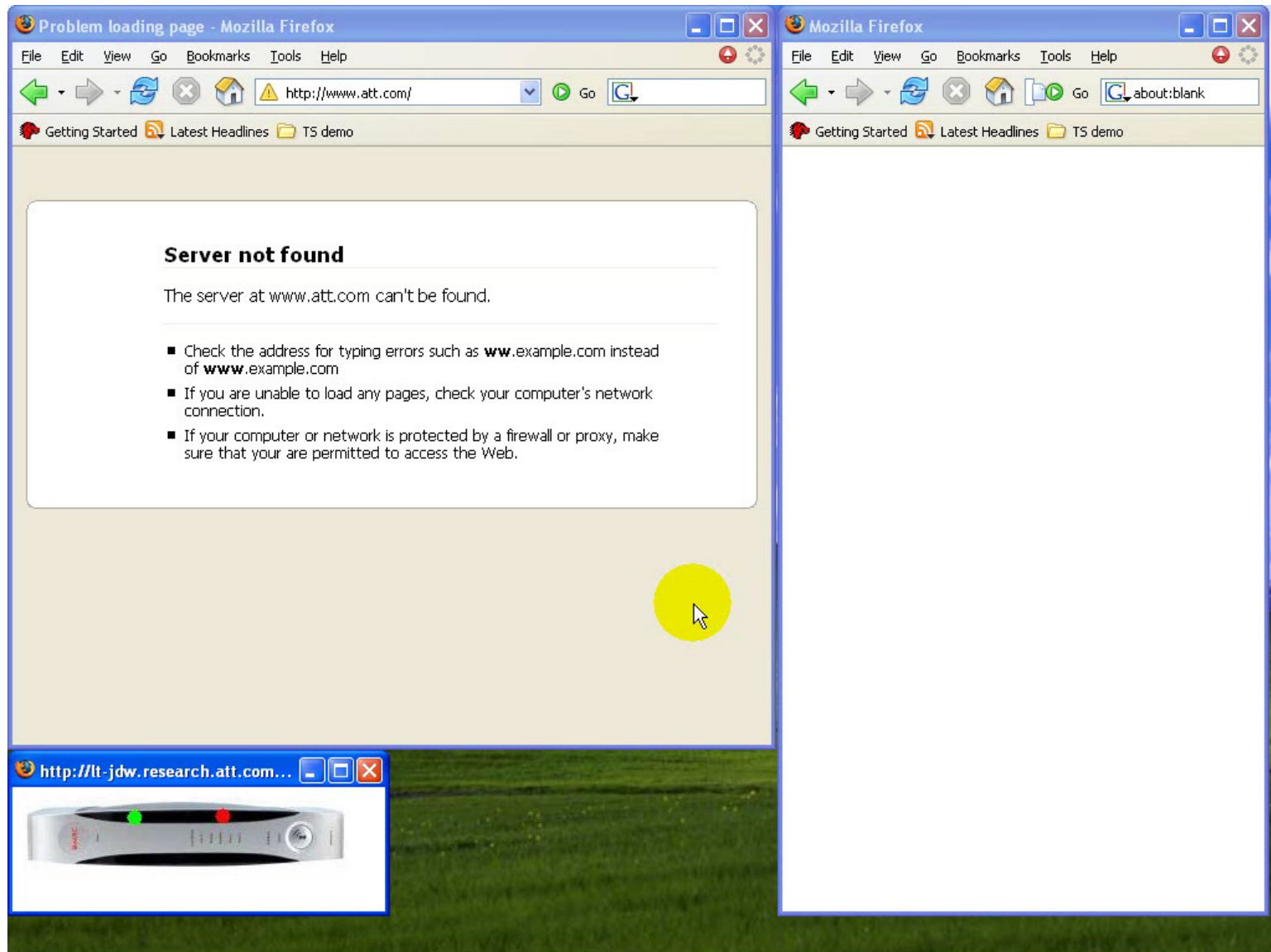
*Applying POMDPs to dialog systems in the troubleshooting domain, Williams (Proc W'Shop Bridging the Gap, ACL, 2007)*

Statistical approaches to dialogue systems : Williams, Young, and Thomson

# DSL troubleshooting SDS as a Bayesian network



# Demonstration of AT&T troubleshooting system



# Comparison of Approaches

## N-Best

- Easy to integrate existing symbolic logic components.
- No state transition dynamics.
- Migration path for existing systems.

## Factorisation

- Full state transition dynamics.
- Mathematically clean.
- Complexity issues limit ability to model dependencies

But note that there are many variants in the POMDP literature left to explore

# Summary

The POMDP framework provides the potential for building dialogue systems which

- are robust to user uncertainty and recognition errors
- are designed to optimize specific objectives
- can be trained automatically on data
- can adapt both in the short and the long term

The key foundations are

- tracking via Bayesian belief monitoring
- policy optimization via Reinforcement learning

# Future Challenges

Early results are promising but research is needed

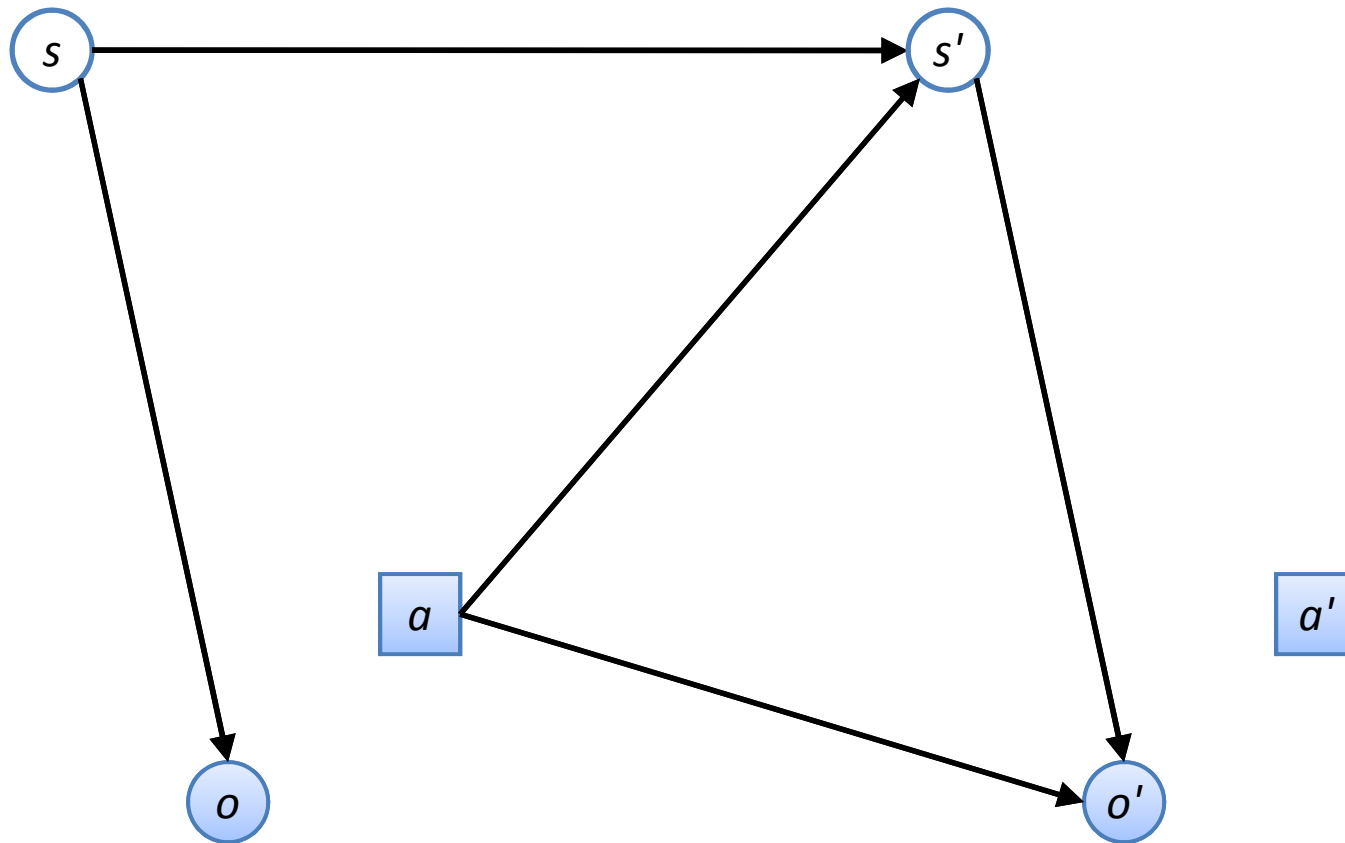
- to develop scalable solutions which can handle very large networks in real time
- to incorporate more detailed linguistic capabilities and symbolic logic e.g. spatial reasoning, set operations, quantification, etc
- to understand how to integrate different modalities: speech, gesture, emotion, etc
- to understand how to migrate these approaches into industrial systems.

# Bibliography

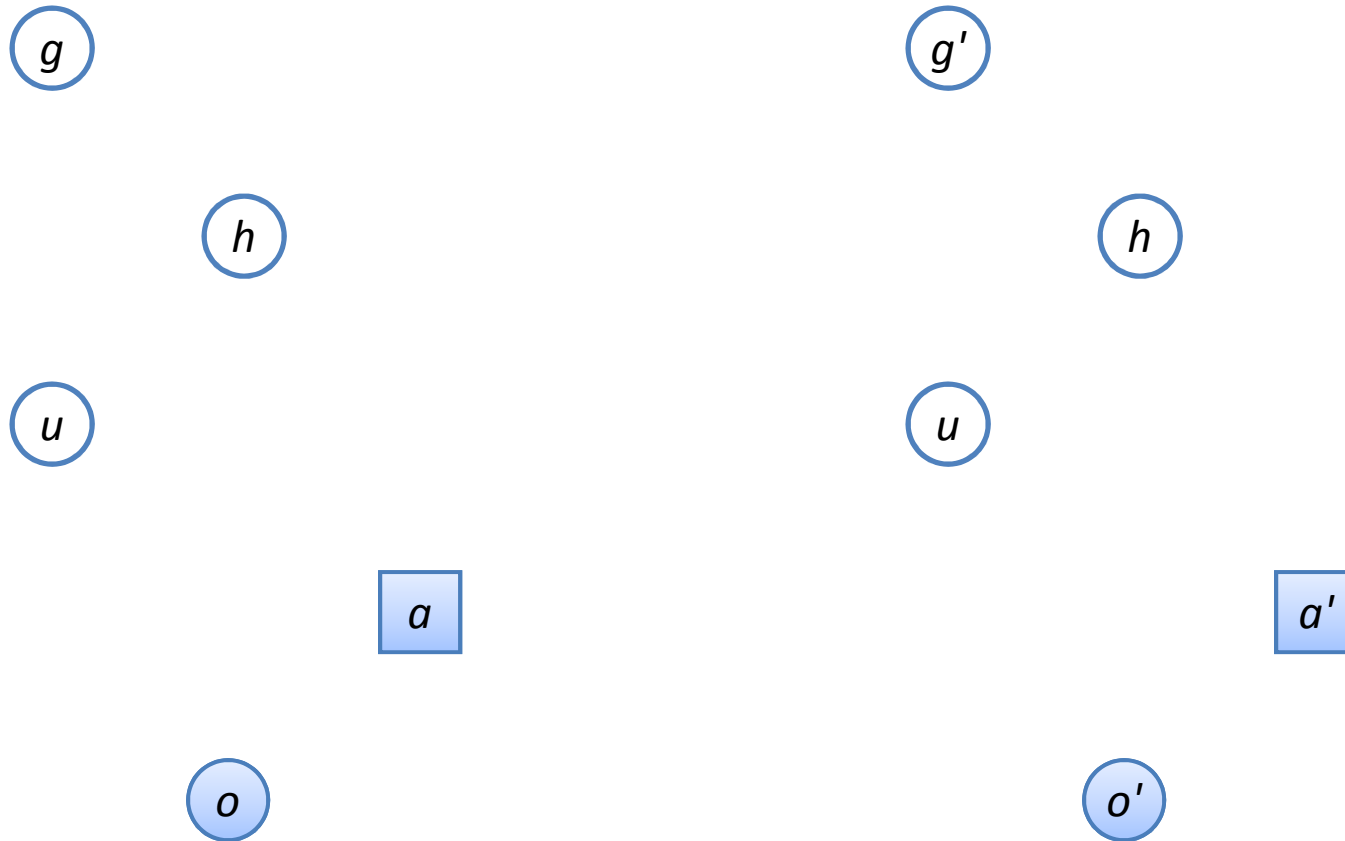
<http://www.research.att.com/~jdw/sdsbib.html>

END

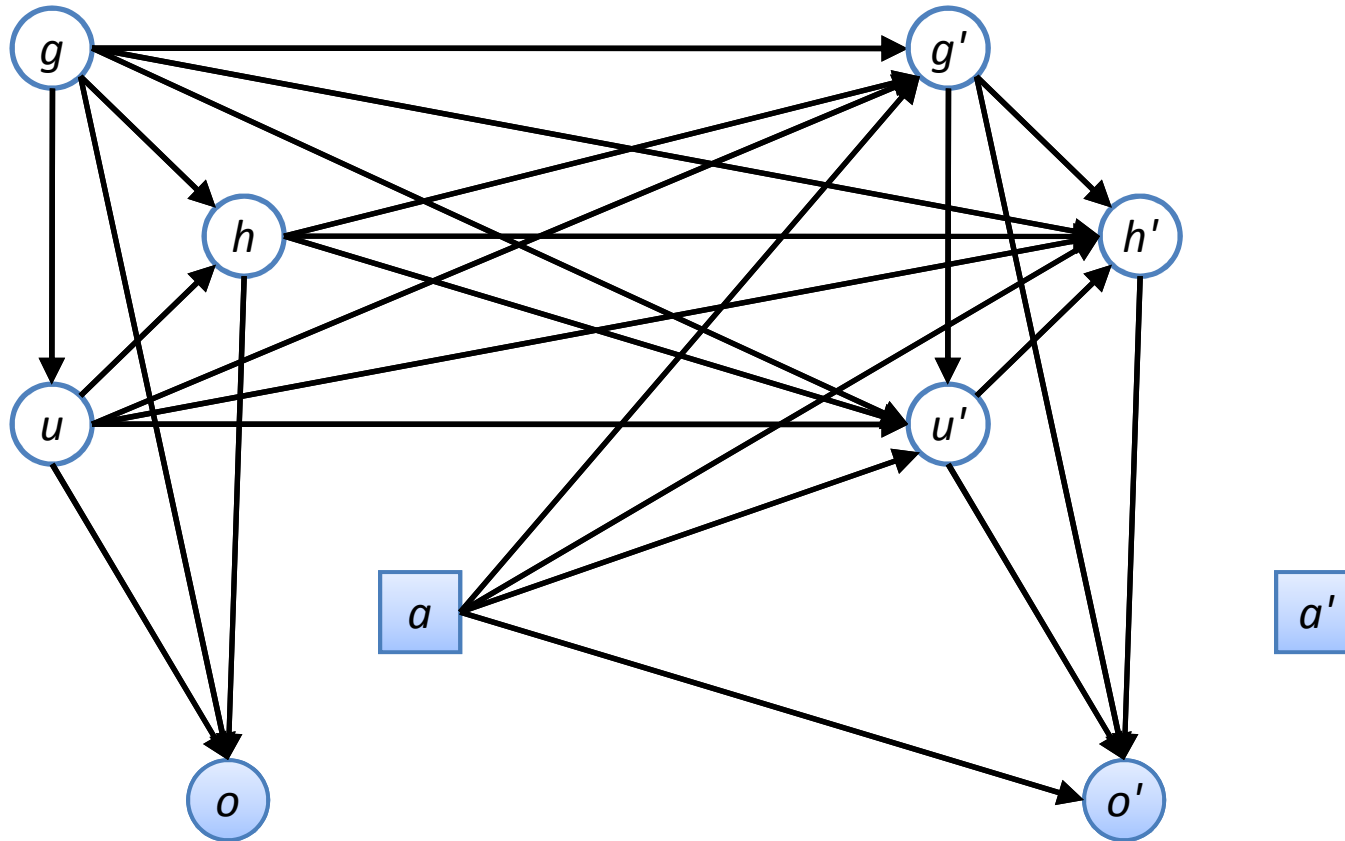
# SDS-POMDP as a graphical model



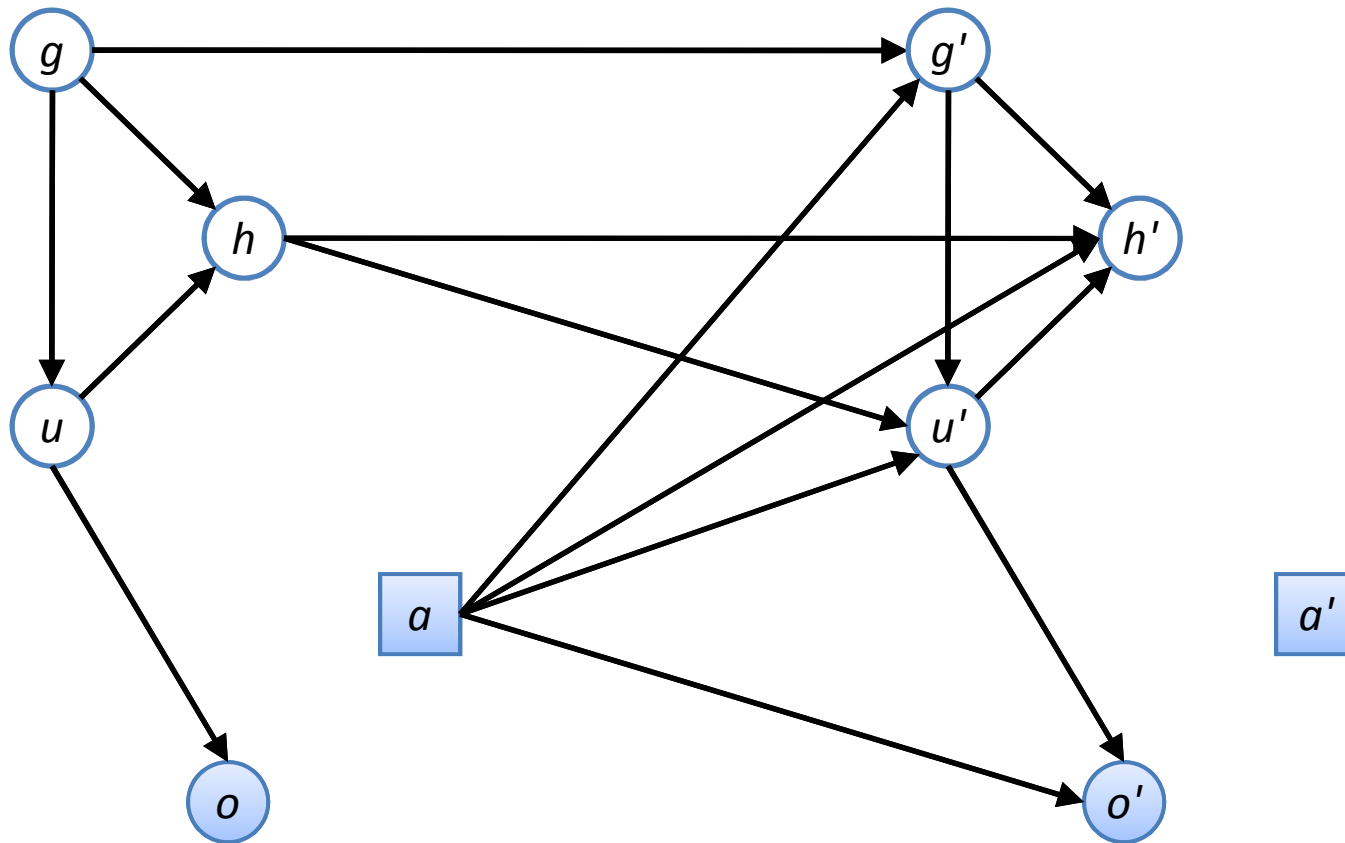
# SDS-POMDP as a graphical model



# SDS-POMDP as a graphical model



# SDS-POMDP as a graphical model



# M-Best: intuition

$a = ask$   $o =$

BOSTON	~0.5
AUSTIN	~0.2
JACKSON	~0.1

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston	0.5	1	1	0.001	5e-4	0.5
austin	austin	austin	0.2	1	1	0.001	2e-4	0.2
jackson	jackson	jackson	0.1	1	1	0.001	1e-4	0.1
tucson	tucson	tucson	2e-4	1	1	0.001	2e-7	2e-4
london	london	london	2e-4	1	1	0.001	2e-7	2e-4
paris	paris	paris	2e-4	1	1	0.001	2e-7	2e-4
tokyo	tokyo	tokyo	2e-4	1	1	0.001	2e-7	2e-4
...	...	...	...	...	...	...	...	...

# M-Best: intuition

$a = ask$   $o =$

BOSTON	~0.5
AUSTIN	~0.2
JACKSON	~0.1

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston	0.5	1	1	0.001	5e-4	0.5
austin	austin	austin	0.2	1	1	0.001	2e-4	0.2
jackson	jackson	jackson	0.1	1	1	0.001	1e-4	0.1
ELSE	ELSE	ELSE	2e-4	1	1	0.997	2e-4	0.2

$a = ask \quad o =$

BOSTON	~0.5
AUSTIN	~0.2
JACKSON	~0.1

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston	0.5	1	1	0.001	5e-4	0.5
austin	austin	austin	0.2	1	1	0.001	2e-4	0.2
jackson	jackson	jackson	0.1	1	1	0.001	1e-4	0.1
tucson	tucson	tucson	2e-4	1	1	0.001	2e-7	2e-4
london	london	london	2e-4	1	1	0.001	2e-7	2e-4
paris	paris	paris	2e-4	1	1	0.001	2e-7	2e-4
tokyo	tokyo	tokyo	2e-4	1	1	0.001	2e-7	2e-4
...	...	...	...	...	...	...	...	...

Only non-zero rows shown

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston				0.5		0.5
austin	austin	austin				0.2		0.2
jackson	jackson	jackson				0.1		0.1
tucson	tucson	tucson				2e-4		2e-4
london	london	london				2e-4		2e-4
paris	paris	paris				2e-4		2e-4
tokyo	tokyo	tokyo				2e-4		2e-4
...	...	...	...	...	...	...	...	...

Only non-zero rows shown



$a = ask \quad o =$

TUCSON	~0.2
AUSTIN	~0.1

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston	7e-4	1	1	0.5	3.5e-4	0.017
austin	austin	austin	0.1	1	1	0.2	2e-2	0.971
jackson	jackson	jackson	7e-4	1	1	0.1	7e-5	0.003
tucson	tucson	tucson	0.2	1	1	2e-4	4e-5	0.002
london	london	london	7e-4	1	1	2e-4	1.4e-7	6.8e-6
paris	paris	paris	7e-4	1	1	2e-4	1.4e-7	6.8e-6
tokyo	tokyo	tokyo	7e-4	1	1	2e-4	1.4e-7	6.8e-6
...	...	...	...	...	...	...	...	...

Only non-zero rows shown

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	boston	boston				0.017		0.017
austin	austin	austin				0.971		0.971
jackson	jackson	jackson				0.003		0.003
tucson	tucson	tucson				0.002		0.002
london	london	london				6.8e-6		6.8e-6
paris	paris	paris				6.8e-6		6.8e-6
tokyo	tokyo	tokyo				6.8e-6		6.8e-6
...	...	...	...	...	...	...	...	...

Only non-zero rows shown



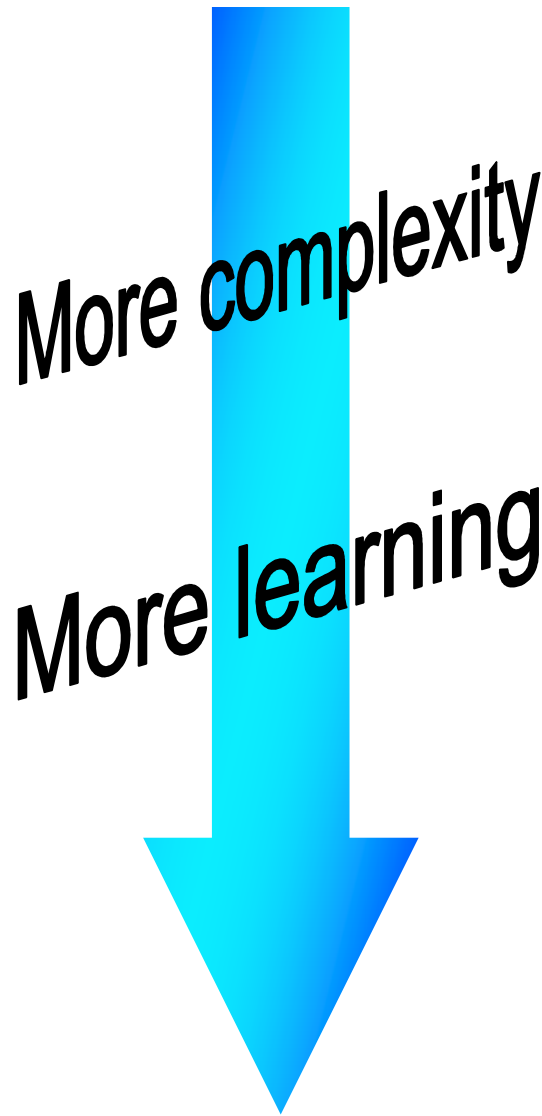
$a = \text{confirm}(\text{austin})$   $o =$

NO	~0.99
YES	~0.01

$g'$	$u'$	$g$	$P(o'   u', m)$	$P(u'   g', m)$	$P(g'   g, m)$	$b(g)$	=	$b'(g')$
boston	no	boston	0.99	1	1	0.017	1.68e-2	0.441
austin	yes	austin	0.01	1	1	0.971	9.71e-3	0.254
jackson	no	jackson	0.99	1	1	0.003	2.97e-3	0.078
tucson	no	tucson	0.99	1	1	0.002	1.98e-3	0.052
london	no	london	0.99	1	1	6.8e-6	6.73e-6	1.76e-4
paris	no	paris	0.99	1	1	6.8e-6	6.73e-6	1.76e-4
tokyo	no	tokyo	0.99	1	1	6.8e-6	6.73e-6	1.76e-4
...	...	...	...	...	...	...	...	...

Only non-zero rows shown

# Policy learning – Comparison



## MDP

- Hand-crafted belief update
- Hand-crafted single-utterance confidence thresholds

## Grid-based POMDP

- Probabilistic belief update
- Hand-crafted belief thresholds

## Function-approximated POMDP

- Probabilistic belief update
- Learned belief thresholds