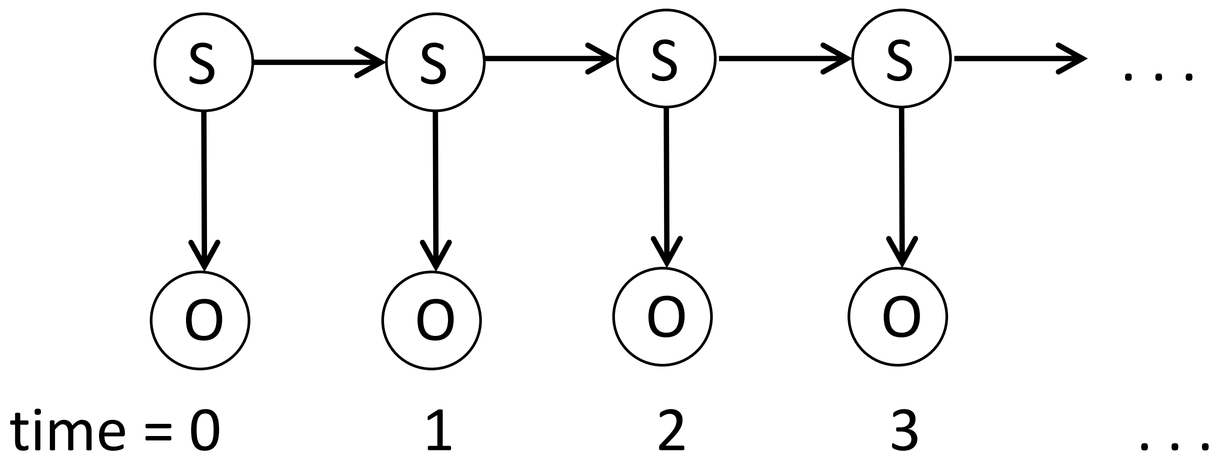
Learning Analytics

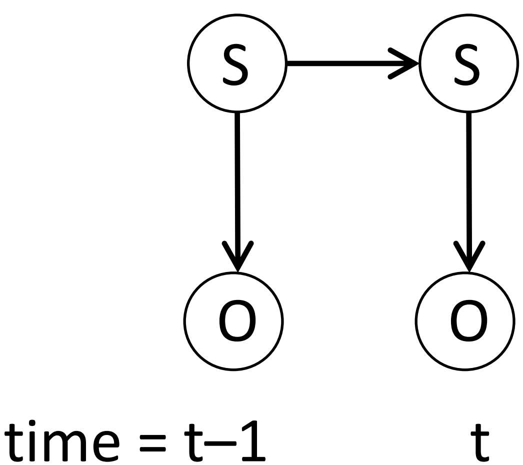
Dr. Bowen Hui
Computer Science
University of British Columbia Okanagan

Recall: Inference Over Time

Mimic:

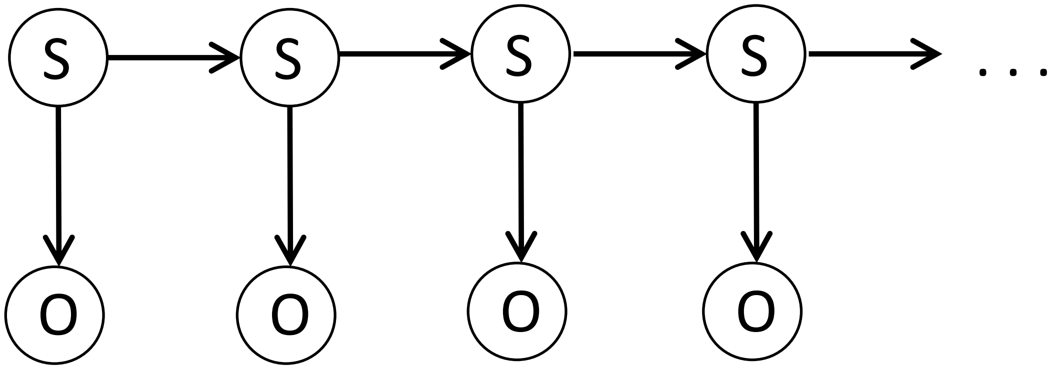


Setup:



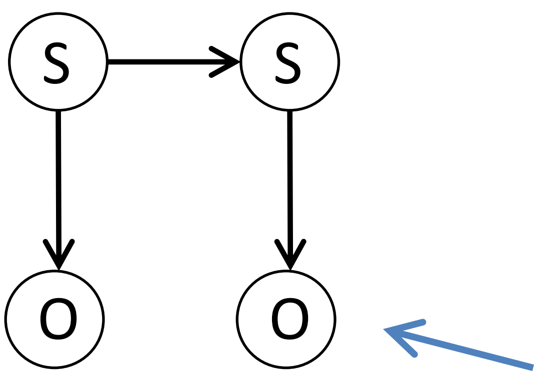
Recall: Inference Over Time

Mimic:



time = 0 1 2 3 . . .

Start, time=1:



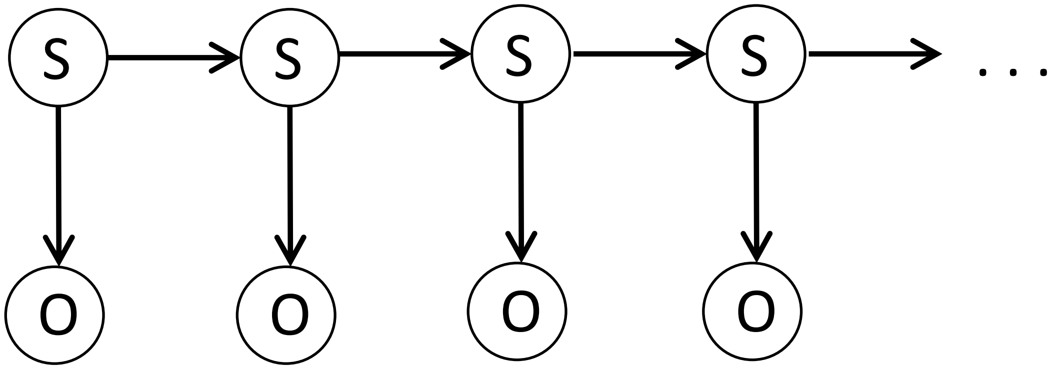
time = 0 1

Observe user behaviour

Enter evidence

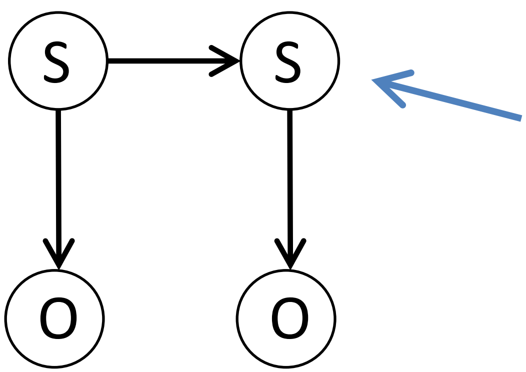
Recall: Inference Over Time

Mimic:



time = 0 1 2 3 . . .

Start, time=1:



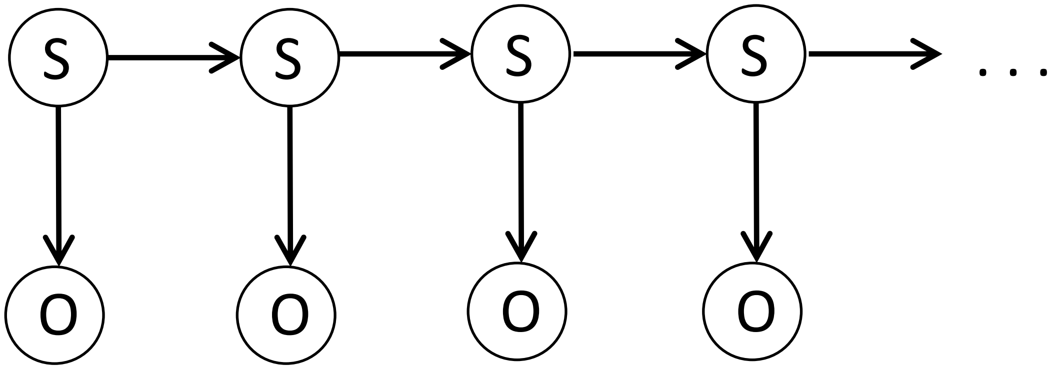
Compute marginal of interest

Get: $Pr(S_1)$

time = 0 1

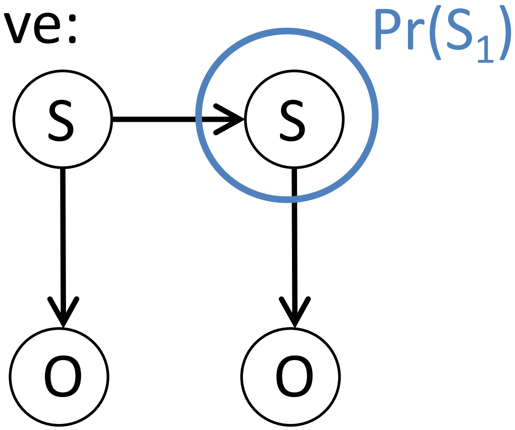
Inference Over Time

Mimic:



time = 0 1 2 3 . . .

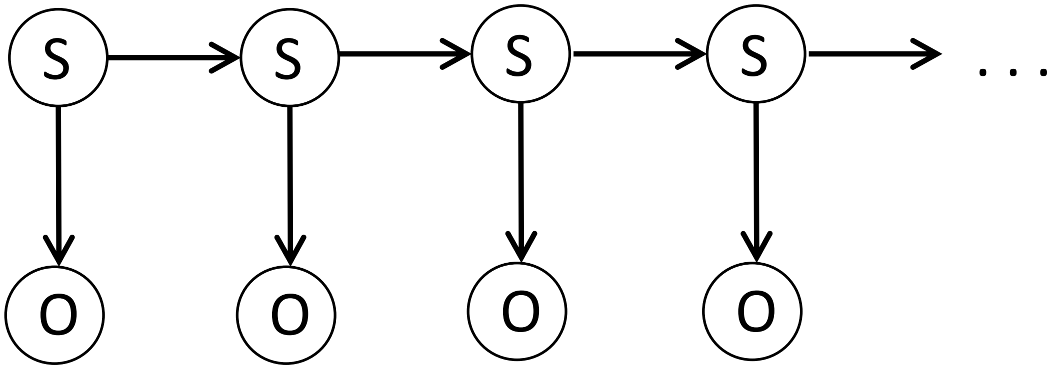
We have:



time = 0 1

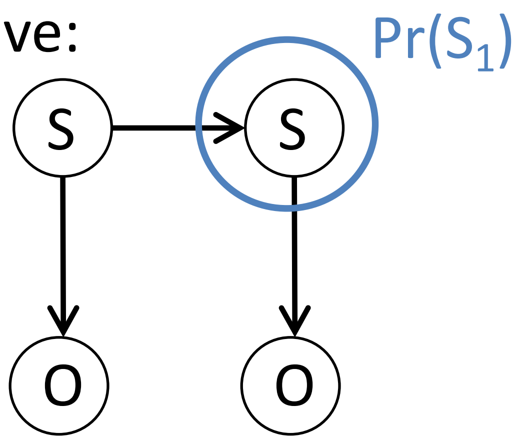
Inference Over Time

Mimic:



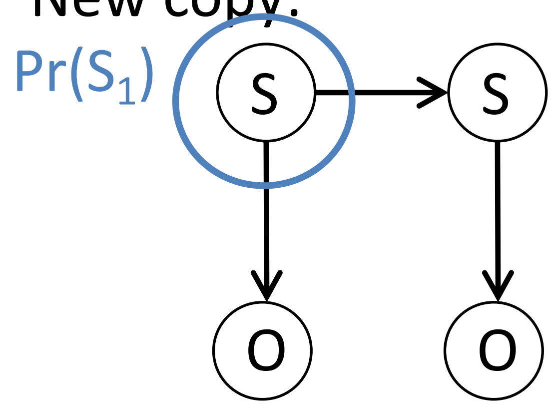
time = 0 1 2 3 . . .

We have:



time = 0 1

New copy:



time = t–1 t

A Closer Look: Value of Offering Hints

Image taken from MasterCluster.software

- Hints are helpful when you read them
- How to estimate whether user is reading hints?

Model Intuitions

- If you read hints:
  - Time hint box stays opened is about your average reading time for the sentence displayed
- If you don't read hints:
  - Time hint box stays opened is very short or very long (relative to average reading time)

- If you read this hint, you'll probably read the next hint; and vice versa

Model Intuitions

Pr(TimeOpen | Read)

- If you read hints:
  - Time hint box stays opened is about your average reading time for the sentence displayed
- If you don't read hints:
  - Time hint box stays opened is very short or very long (relative to average reading time)

- If you read this hint, you'll probably read the next hint; and vice versa

$\Pr(\text{Read}_t \mid \text{Read}_{t-1})$

Defining Model Variables

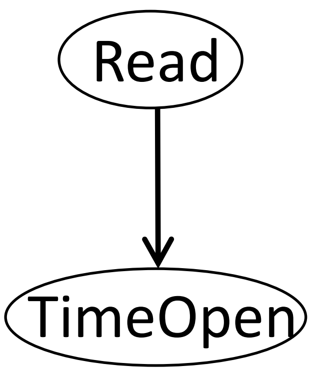
- Read = false, true
  - User is either going to read hints or not

Defining Model Variables

- Read = false, true
  - User is either going to read hints or not
- TimeOpen:
  - Too short = hint box closed soon after popped up
  - Too long = hint box left opened and ignored
  - On task = hint box is being read and closed when done

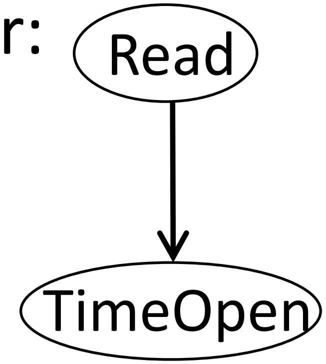
Defining Model Variables

- Read = false, true
  - User is either going to read hints or not
- TimeOpen:
  - Too short = hint box closed soon after popped up
  - Too long = hint box left opened and ignored
  - On task = hint box is being read and closed when done
- Model so far:



Defining Observation Function

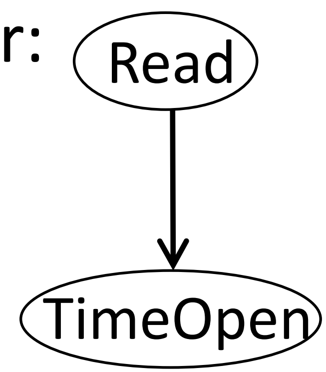
- Pr(TimeOpen | Read)
  - If you read hints: Time hint box stays opened is about your average reading time for the sentence displayed
  - If you don't read hints: Time hint box stays opened is very short or very long (relative to average reading time)
- Model so far:



	TimeOpen = …		
Read	Too short	On task	Too long
false			
true			

Defining Observation Function

- Pr(TimeOpen | Read)
  - If you read hints: Time hint box stays opened is about your average reading time for the sentence displayed
  - If you don't read hints: Time hint box stays opened is very short or very long (relative to average reading time)
- Model so far:

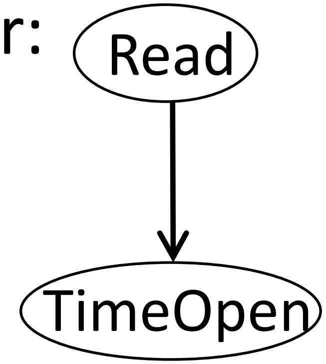


	TimeOpen = …		
Read	Too short	On task	Too long
false			
true	0.1	0.8	0.1

User is generally reading, with a small chance of either closing the box too quickly or ignoring it

Defining Observation Function

- Pr(TimeOpen | Read)
  - If you read hints: Time hint box stays opened is about your average reading time for the sentence displayed
  - If you don't read hints: Time hint box stays opened is very short or very long (relative to average reading time)
- Model so far:

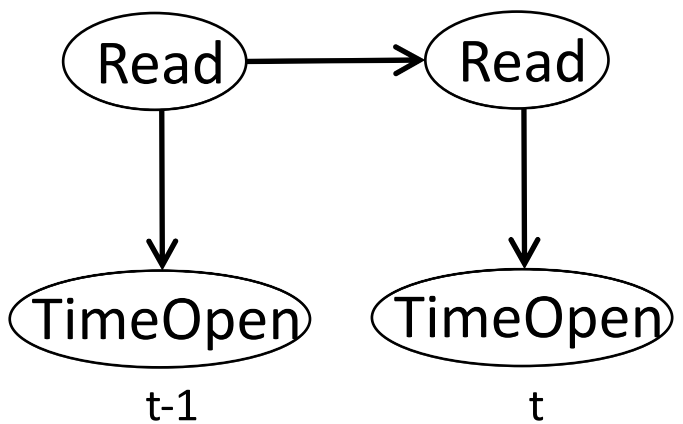


	TimeOpen = …		
Read	Too short	On task	Too long
false	0.7	0.1	0.2
true	0.1	0.8	0.1

User tends to close box, sometimes ignores box, but is rarely on task

Defining Transition Function

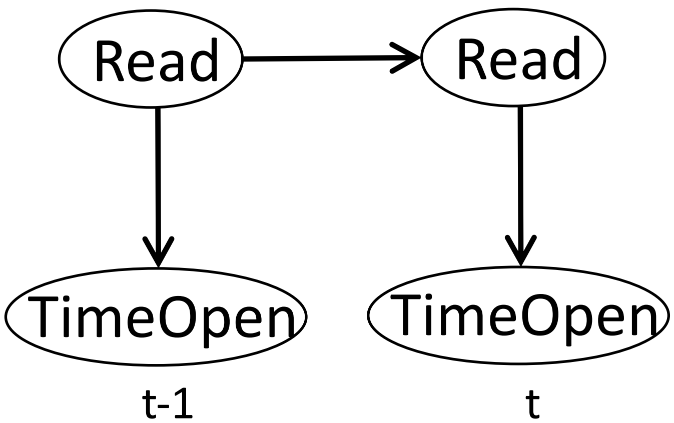
- $\Pr(Read_t \mid Read_{t-1})$
  - If you read this hint, you'll probably read the next hint; and vice versa
- Model so far:



	Read _t = …	
Read_t-1	false	true
false		
true		

Defining Transition Function

- $Pr(Read_t \mid Read_{t-1})$
  - If you this hint, you'll probably read the next hint, and vice versa
- Model so far:



	Read _t = …	
Read_t-1	false	true
false	0.8	0.2
true	0.1	0.9

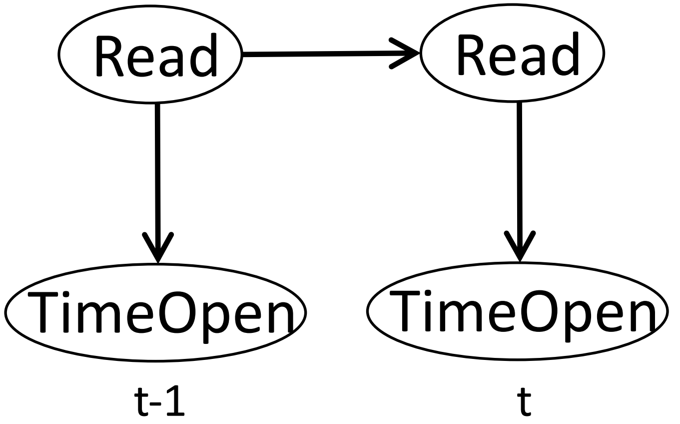
Some noise added

Defining Prior Distribution

- Pr(Read)
  - How likely is the average user to read hints?

- Model so far:

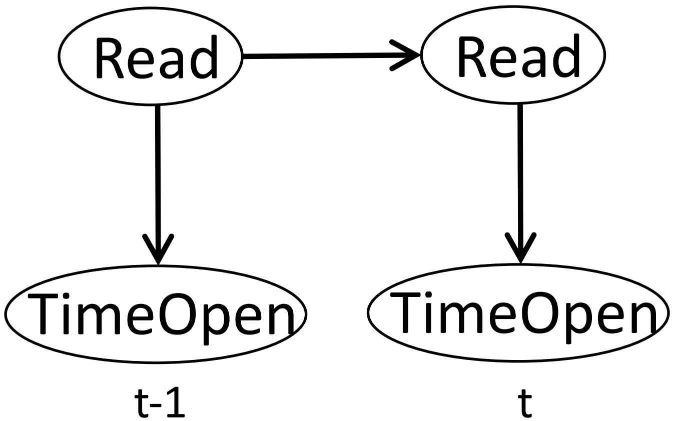
Read = …	
false	true



Defining Prior Distribution

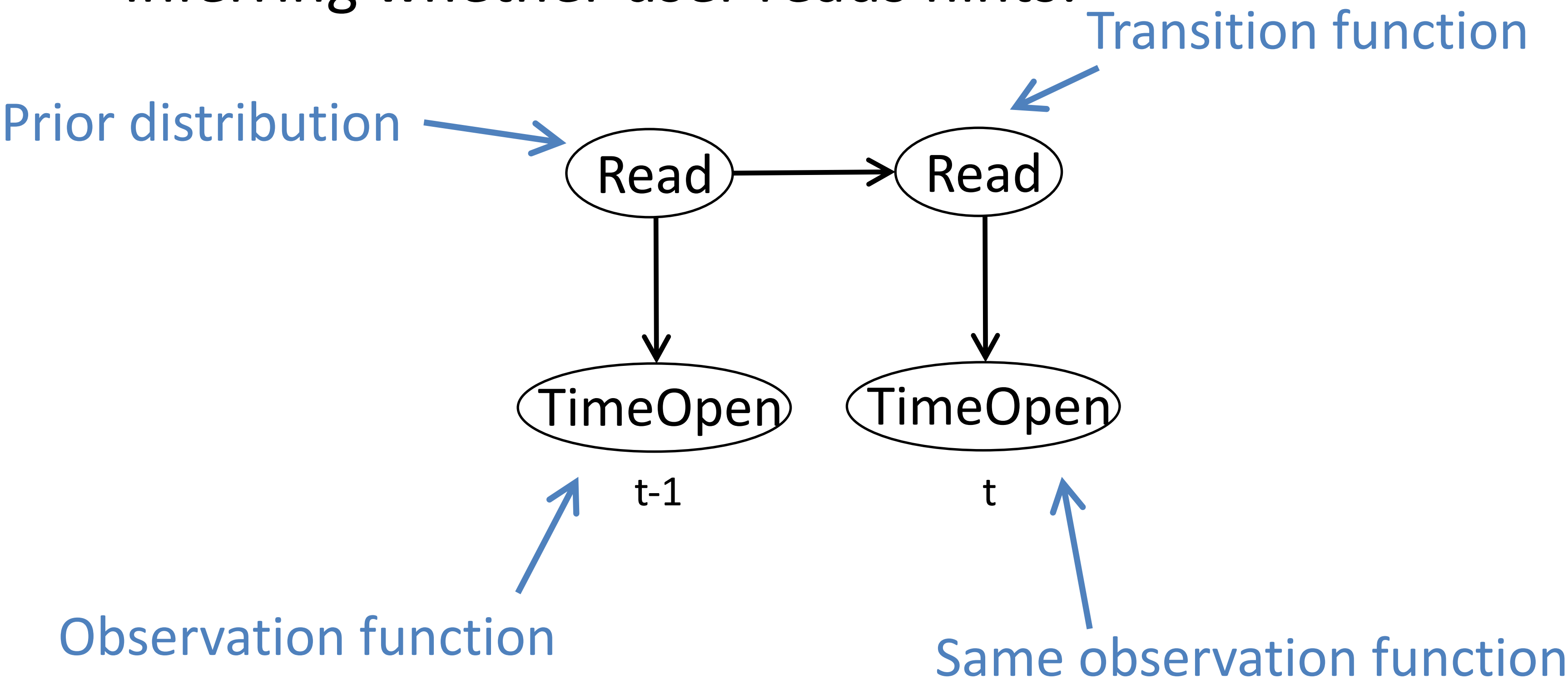
- Pr(Read)
  - How likely is the average user to read hints?
  - No information: assign uniform distribution
- Model so far:

Read = …	
false	true
0.5	0.5



Recap Model

- Inferring whether user reads hints:



Implementation in BNT/Matlab

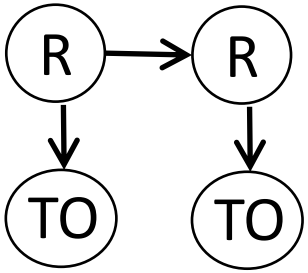
- Use editor to save scripts into .m files
- Easier to re-run scripts
- Can also define functions

- Example, inside mk_hints.m:
```
function DBN = mk_hints
. . .
DBN = . . .                % whatever you intend to return
```

- Later, at the prompt:
```
>> myDbn = mk_hints;
```

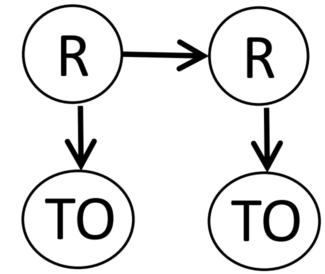
Inside mk_hints.m

```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss    = length( names );
DBN   = names;
```



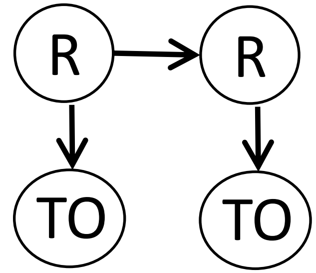
Inside mk_hints.m

```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss    = length( names );
DBN   = names;

% intra-stage dependencies
intrac = {...
'Read', 'TimeOpen'};
[intra, names] = mk_adj_mat( intrac, names, 1 );
DBN = names;   % potentially re-ordered names
```

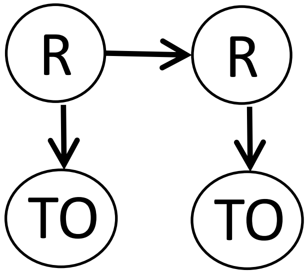


Inside mk_hints.m

```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss    = length( names );
DBN   = names;

% intra-stage dependencies
intrac = {...
'Read', 'TimeOpen'};
[intra, names] = mk_adj_mat( intrac, names, 1 );
DBN = names;   % potentially re-ordered names

%inter-stage dependencies
interc = {...
'Read', 'Read'};
inter = mk_adj_mat( interc, names, 0 );
```

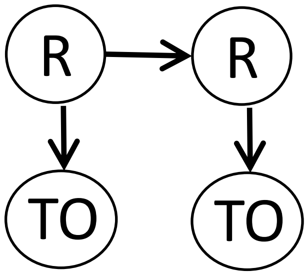


Inside mk_hints.m



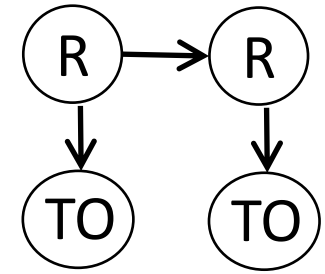

```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss    = length( names );
DBN   = names;

% intra-stage dependencies
intrac = {...
'Read', 'TimeOpen'};
[intra, names] = mk_adj_mat( intrac, names, 1 );
DBN = names;   % potentially re-ordered names

%inter-stage dependencies
interc = {...
'Read', 'Read'};
inter = mk_adj_mat( interc, names, 0 );

% observations
onodes = [ find(cellfun(@isempty, strfind(names,'TimeOpen'))==0) ];
```

Inside mk_hints.m




```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss    = length( names );
DBN   = names;

% intra-stage dependencies
intrac = {...
'Read', 'TimeOpen'};
[intra, names] = mk_adj_mat( intrac, names, 1 );
DBN = names;   % potentially re-ordered names

%inter-stage dependencies
interc = {...
'Read', 'Read'};
inter = mk_adj_mat( interc, names, 0 );

% observations
onodes = [ find(cellfun(@isempty, strfind(names,'TimeOpen'))==0) ];

% discretize nodes
Q      = 2;      % two hidden states
O      = 3;      % three observable states
ns     = [Q O];
dnodes = 1:ss;
```

Inside mk_hints.m




```
names = {'Read', 'TimeOpen'};   % easier to refer to later
ss     = length( names );
DBN    = names;

% intra-stage dependencies
intrac = {...
'Read', 'TimeOpen'};
[intra, names] = mk_adj_mat( intrac, names, 1 );
DBN = names;   % potentially re-ordered names

%inter-stage dependencies
interc = {...
'Read', 'Read'};
inter = mk_adj_mat( interc, names, 0 );

% observations
onodes = [ find(cellfun(@isempty, strfind(names,'TimeOpen'))==0) ];

% discretize nodes
Q      = 2;      % two hidden states
O      = 3;      % three observable states
ns     = [Q O];
dnodes = 1:ss;

% define equivalence classes
ecl1 = [1 2];
ecl2 = [3 2];   % node 4 is tied to node 2
```

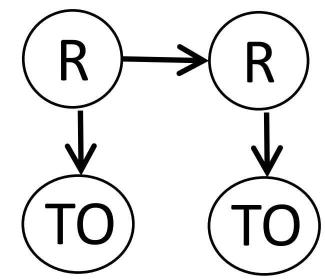
Inside mk_hints.m (cont.)

```
% create the dbn structure based on the components defined above
bnet = mk_dbn( intra, inter, ns, ...
  'discrete', dnodes, ...
  'eclass1', ecl1, ...
  'eclass2', ecl2, ...
  'observed', onodes, ...
  'names', names );
DBN  = bnet;
```

Last step to creating the DBN structure

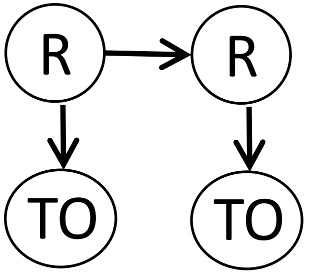
Inside mk_hints.m (cont.)

```
Read0    = 1;
TimeOpen = 2;
Read1    = 3;
```



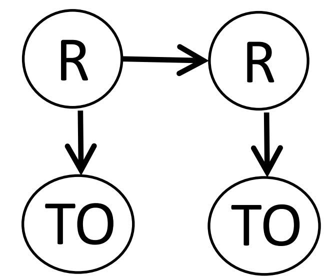
Inside mk_hints.m (cont.)

```
Read0    = 1;
TimeOpen = 2;
Read1    = 3;

% prior, Pr(Read0)
cpt = normalize( ones(Q,1) );
bnet.CPD{Read0} = tabular_CPD( bnet, Read0, 'CPT', cpt );
```



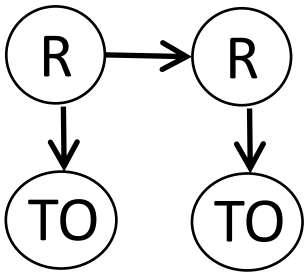
Inside mk_hints.m (cont.)

```
Read0    = 1;
TimeOpen = 2;
Read1    = 3;

% prior, Pr(Read0)
cpt = normalize( ones(Q,1) );
bnet.CPD{Read0} = tabular_CPD( bnet, Read0, 'CPT', cpt );

% transition function, Pr(Read_t|Read_t-1)
% R0   R1=false, true
%  false  0.8      0.2
%  true   0.1      0.9
cpt = [.8 .1 .2 .9];
bnet.CPD{Read1} = tabular_CPD( bnet, Read1, 'CPT', cpt );
```



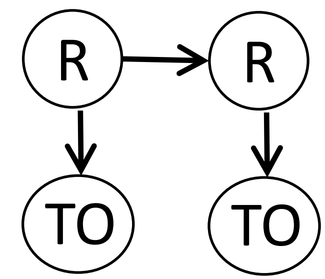
Inside mk_hints.m (cont.)

```
Read0    = 1;
TimeOpen = 2;
Read1    = 3;

% prior, Pr(Read0)
cpt = normalize( ones(Q,1) );
bnet.CPD{Read0} = tabular_CPD( bnet, Read0, 'CPT', cpt );

% transition function, Pr(Read_t|Read_t-1)
% R0   R1=false, true
%  false  0.8      0.2
%  true   0.1      0.9
cpt = [.8 .1 .2 .9];
bnet.CPD{Read1} = tabular_CPD( bnet, Read1, 'CPT', cpt );

% observation function, Pr(TimeOpen_t|Read_t)
% R      time=short, onTask, long
% false  0.7          0.1     0.2   % user tends to close box and not ignore it
% true   0.1          0.8     0.1   % user will be reading
cpt = [.7 .1 ...
       .1 .8 ...
       .2 .1];
bnet.CPD{TimeOpen} = tabular_CPD(bnet, TimeOpen, 'CPT', cpt );
```

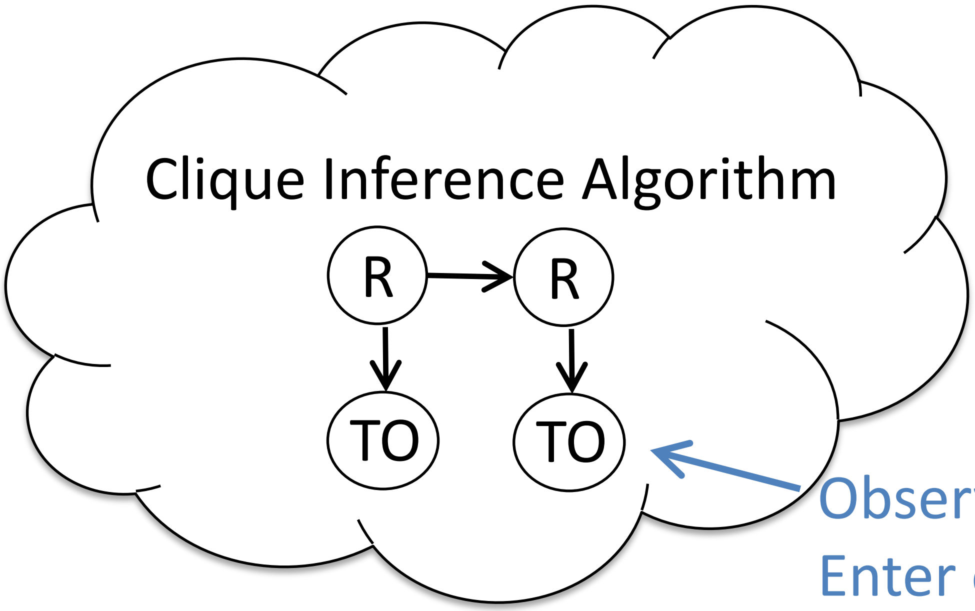


Inside mk_hints.m (cont.)

```
Read0    = 1;
TimeOpen = 2;
Read1    = 3;

% prior, Pr(Read0)
cpt = normalize( ones(Q,1) );
bnet.CPD{Read0} = tabular_CPD( bnet, Read0, 'CPT', cpt );

% transition function, Pr(Read_t|Read_t-1)
% R0   R1=false, true
%  false  0.8     0.2
%  true   0.1     0.9
cpt = [.8 .1 .2 .9];
bnet.CPD{Read1} = tabular_CPD( bnet, Read1, 'CPT', cpt );

% observation function, Pr(TimeOpen_t|Read_t)
% R      time=short, onTask, long
% false  0.7          0.1     0.2   % user tends to close box and not ignore it
% true   0.1          0.8     0.1   % user will be reading
cpt = [.7 .1 ...
       .1 .8 ...
       .2 .1];
bnet.CPD{TimeOpen} = tabular_CPD(bnet, TimeOpen, 'CPT', cpt );

DBN = bnet;
```

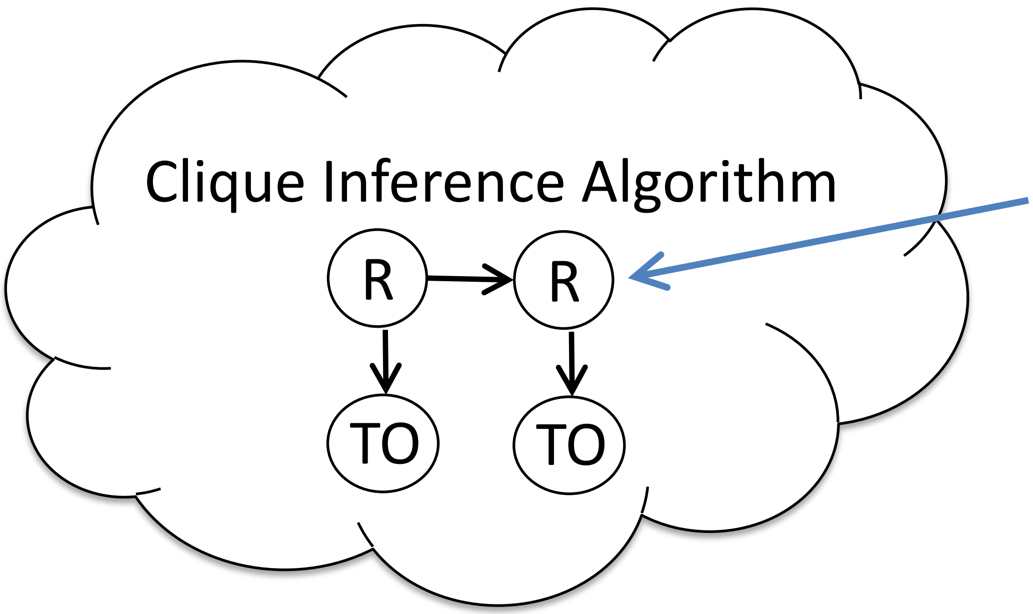


Simulation Setup



Observe user behaviour
Enter evidence

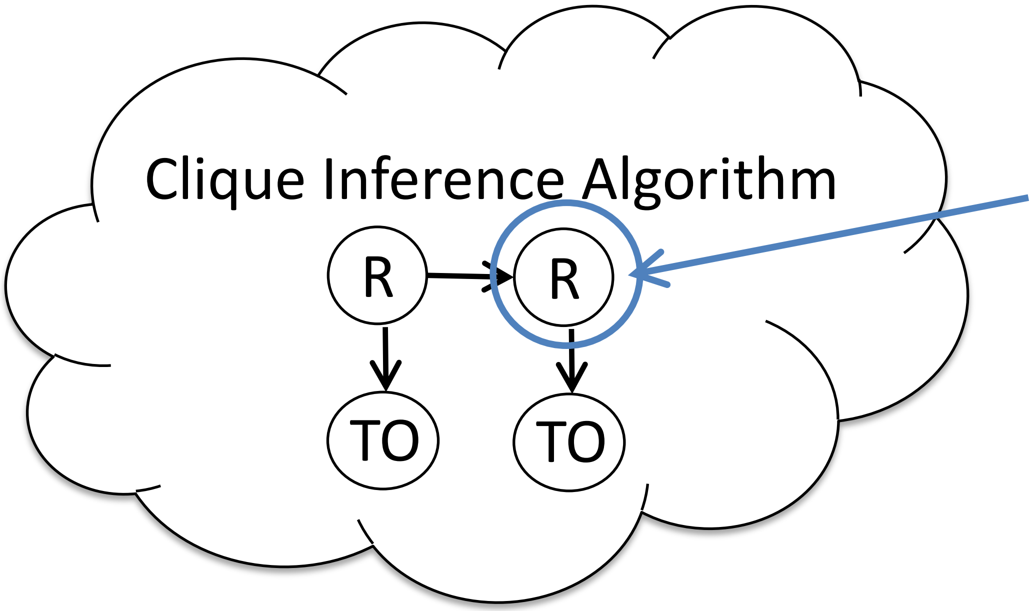
Simulation Setup



Compute marginal of interest

Get: $\Pr(Read_t)$

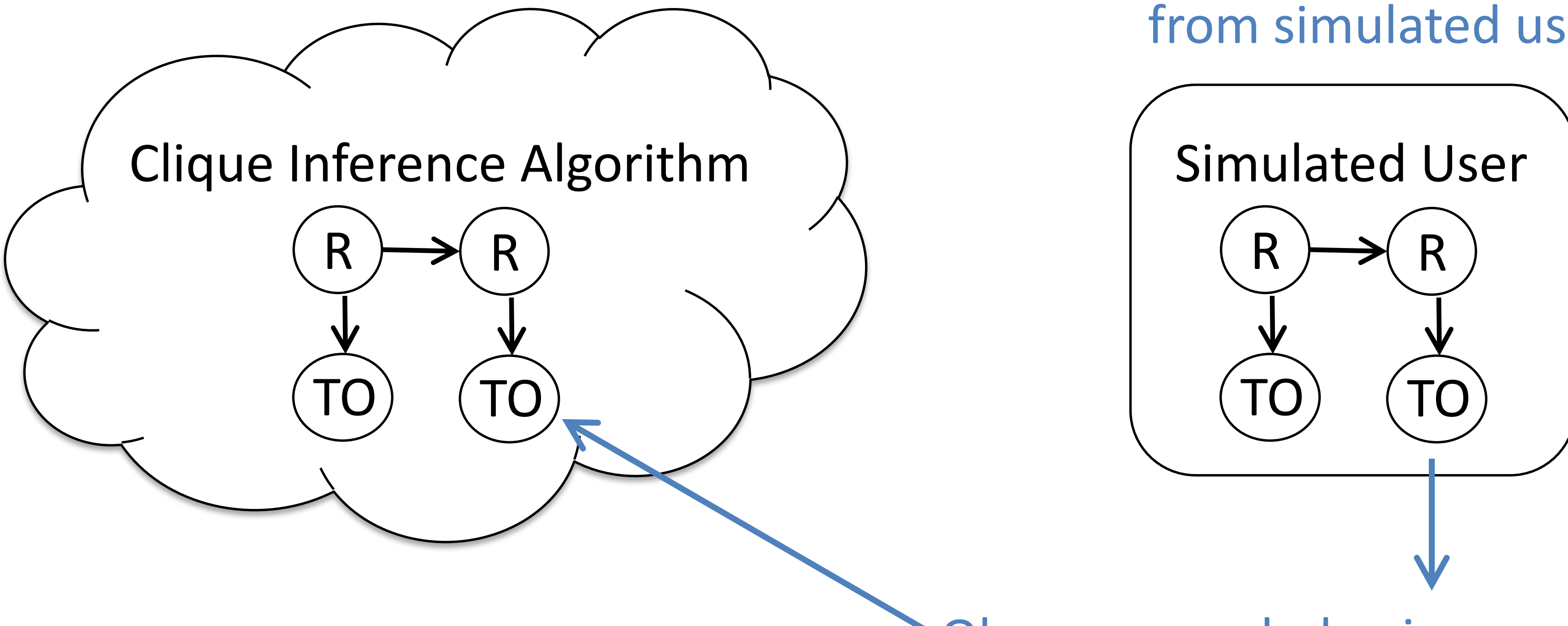
Simulation Setup



Get updated belief:
$Pr(Read_t)$

Simulation Interaction

Sample evidence
from simulated user



Observe user behaviour
Enter evidence

Separate File: sim_hints.m

```
% setup inference process
%
% sample series of evidence in advance, say 10
%
% t=0: prRead is 0.5 according to our model
%
% t=1:
%     enter first piece of evidence
%     update belief by computing marginal prRead
%
% for t=2 to t=T:
%     enter evidence at t
%     update belief by computing marginal prRead
```

Inside sim_hints.m

% setup inference process

```
function prRead = sim_hints( dbn, ex )
% function prRead = sim_hints( dbn, ex )
% ARGS: dbn = dynamic bayes net model specified by BNT syntax
%       ex  = a specific setting used to generate evidence
%

engine = bk_inf_engine( dbn );   % set up inference engine
T = 10;                          % define number of time steps in problem
```

Inside sim_hints.m

% sample series of evidence in advance, say 10

```
if ex == 1,
  ev = sample_dbn( dbn, T);
  evidence = cell( 2, T);
  onodes   = dbn.observed;
  evidence( onodes, : ) = ev( onodes, : ); % all cells besides onodes are empty
```

Case 1:
Random evidence

Case 2:
Fixed evidence

Case 3:
Controlled randomness

Inside sim_hints.m

% sample series of evidence in advance, say 10

```
if ex == 1,
  ev = sample_dbn( dbn, T);
  evidence = cell( 2, T);
  onodes    = dbn.observed;
  evidence( onodes, : ) = ev( onodes, : ); % all cells besides onodes are empty
elseif ex == 2,
  evidence = cell( 2, T);
  for ii=1:T,
    evidence{2,ii} = 2;
  end;
```

Case 1:
Random evidence

Case 2:
Fixed evidence

Case 3:
Controlled randomness

Recall: TimeOpen has 3 values

Inside sim_hints.m

% sample series of evidence in advance, say 10

```
if ex == 1,
  ev = sample_dbn( dbn, T);
  evidence = cell( 2, T);
  onodes    = dbn.observed;
  evidence( onodes, : ) = ev( onodes, : ); % all cells besides onodes are empty
elseif ex == 2,
  evidence = cell( 2, T);
  for ii=1:T,
    evidence{2,ii} = 2;
  end;
else
  readval = 2;
  evidence = sampleHint_seq( dbn, readval, T );
end;
evidence
```

Case 1:
Random evidence

Case 2:
Fixed evidence

Case 3:
Controlled randomness

Recall: Read has 2 values

Inside sim_hints.m

% t=0: prRead is 0.5 according to our model

```
% setup results to be stored
belief = [];
subplot( 1, 1, 1 );     % setup plot for graph

% at t=0, no evidence has been entered, so the probability is same as the
% prior encoded in the DBN itself
%
prRead = get_field( dbn.CPD{ dbn.names('Read') }, 'cpt' );
belief = [belief, prRead(2)];
plot( belief );
```

Setup

Get prRead from model

Plot it

Inside sim_hints.m

% t=1:

% enter first piece of evidence

% update belief by computing marginal prRead

```
% at t=1: initialize the belief state
%
[engine, ll(1)] = dbn_update_bel1(engine, evidence(:,1));

marg = dbn_marginal_from_bel(engine, 1);
prRead = marg.T;
belief = [belief, prRead(2)];
plot( belief );
```

Update belief

Get prRead from model

Plot it

Inside sim_hints.m

% for t=2 to t=T:

% enter evidence at t

% update belief by computing marginal prRead

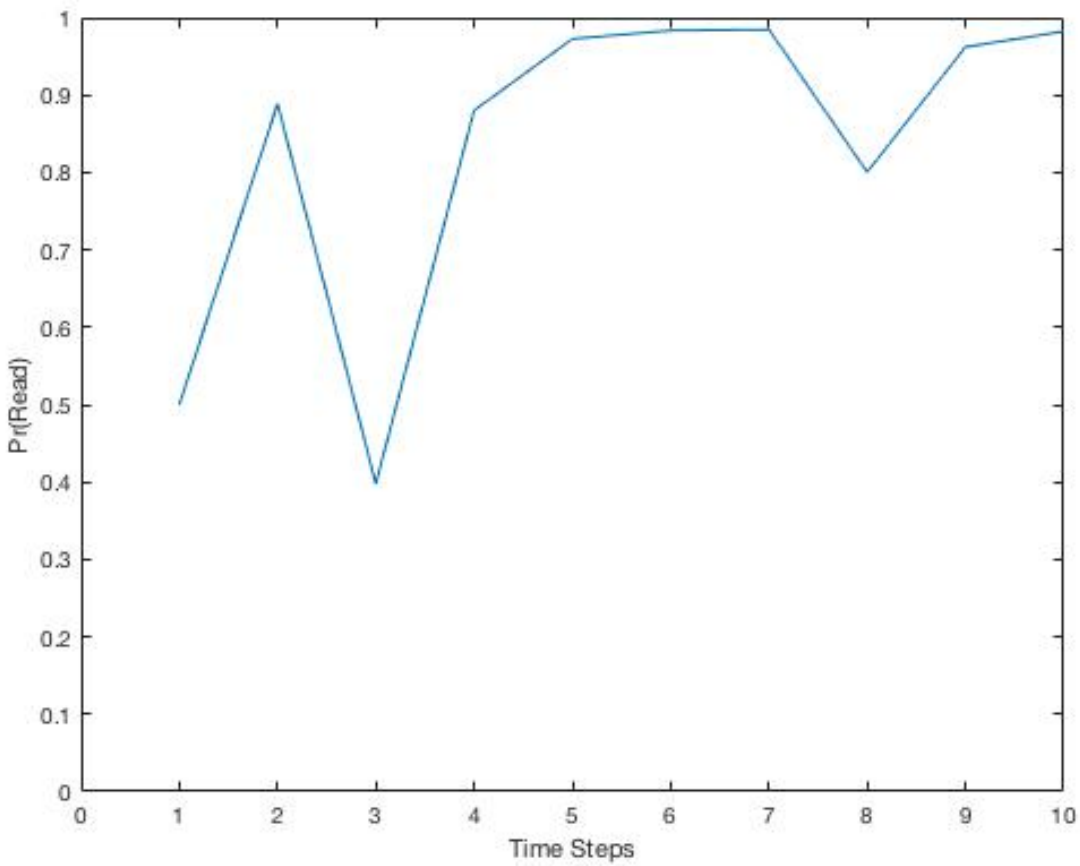
```
for t=2:T,
  % update belief with evidence at current time step
  [engine, ll(t)] = dbn_update_bel(engine, evidence(:,t-1:t));

  % extract marginals of the current belief state
  i = 1;
  marg = dbn_marginal_from_bel(engine, i);
  prRead = marg.T;

  % keep track of results and plot it
  belief = [belief, prRead(2)];
  plot( belief );
  xlabel( 'Time Steps' );
  ylabel( 'Pr(Read)' );
  axis( [ 0 T 0 1] );
  pause(0.25);
end;
```

Update belief

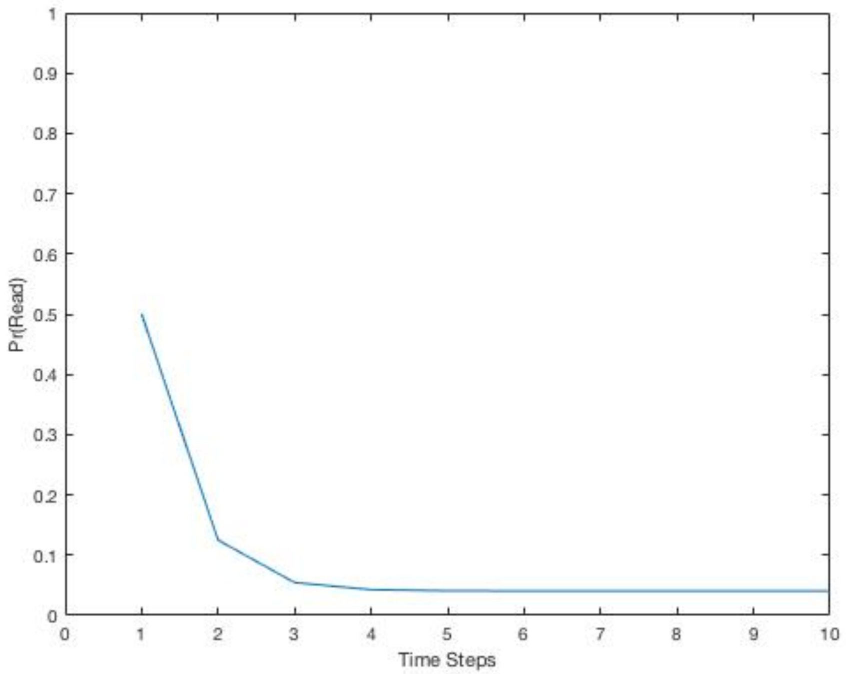
Get prRead from model

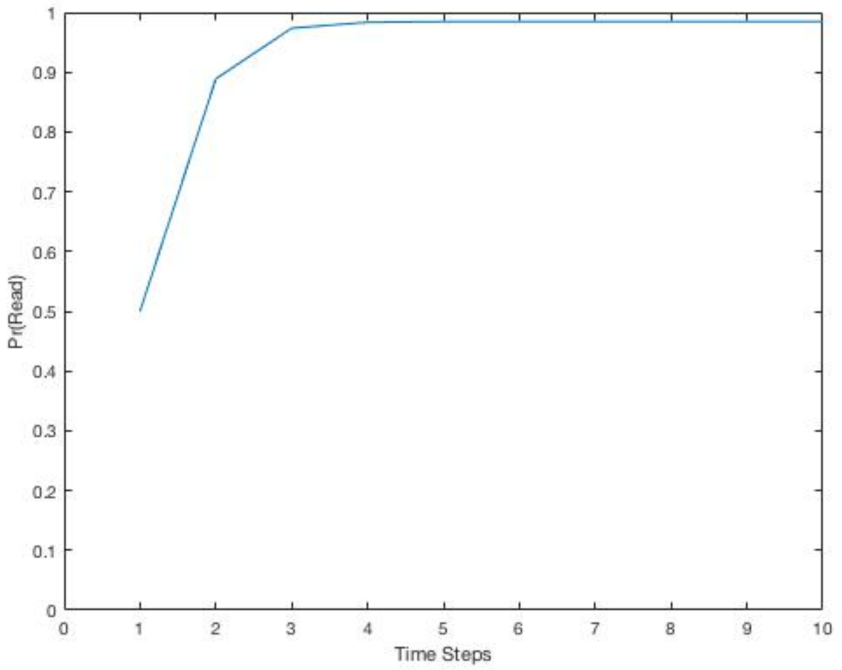
Plot it

Single Plot Results
Case 1: Random Evidence



Single Plot Results
Case 2: Fixed Evidence

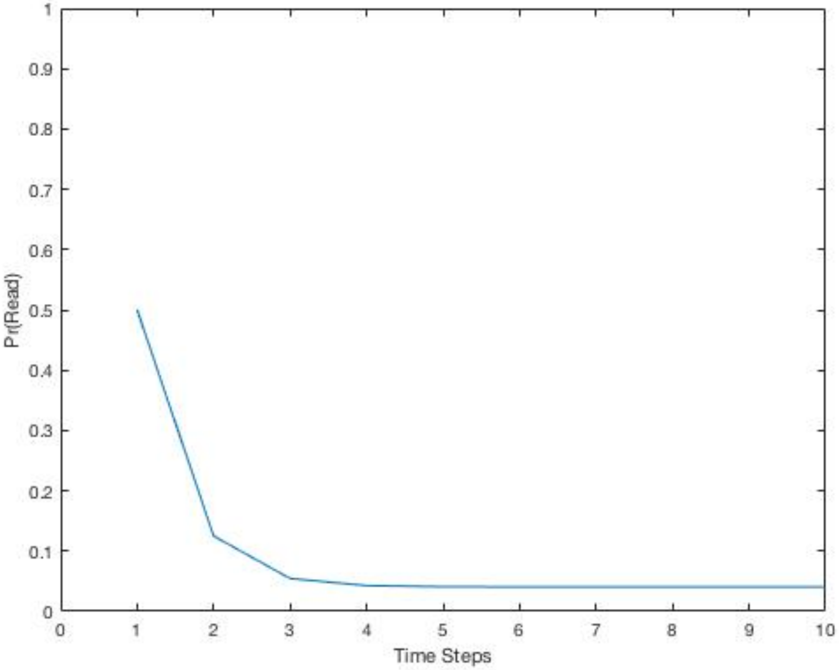




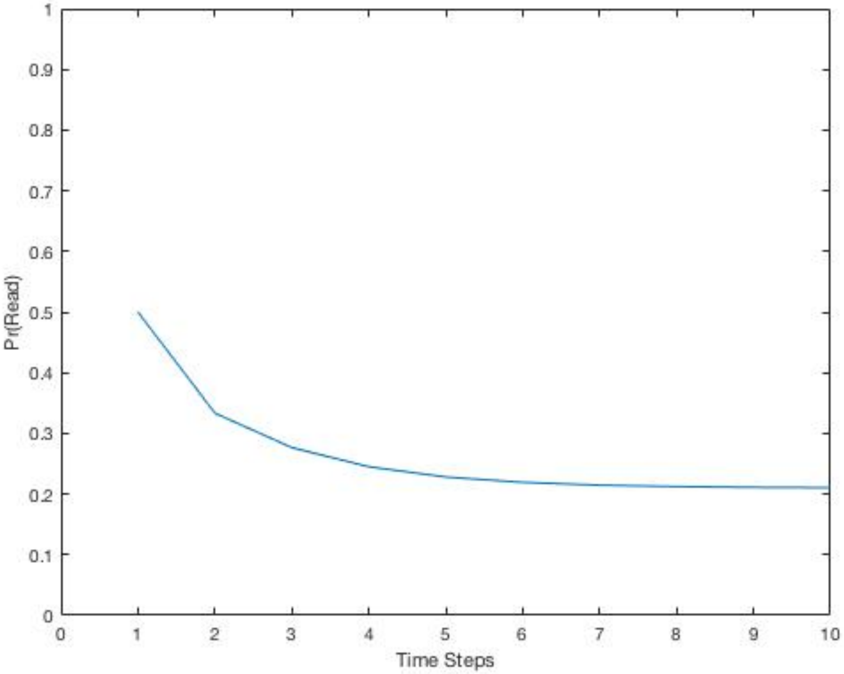
All values = 1 (too short)

All values = 2 (on task)

Single Plot Results
Case 2: Fixed Evidence

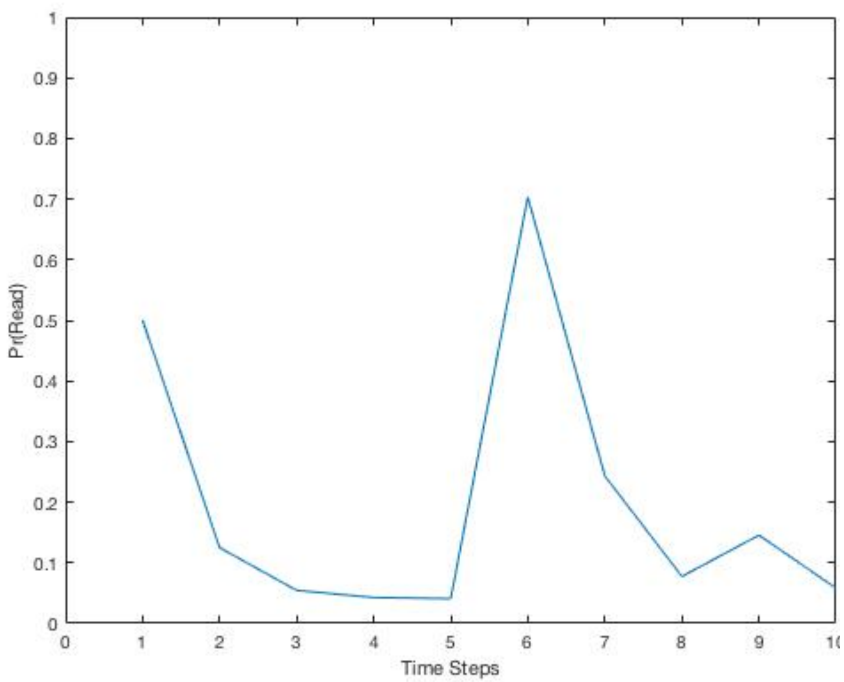


All values = 1 (too short)

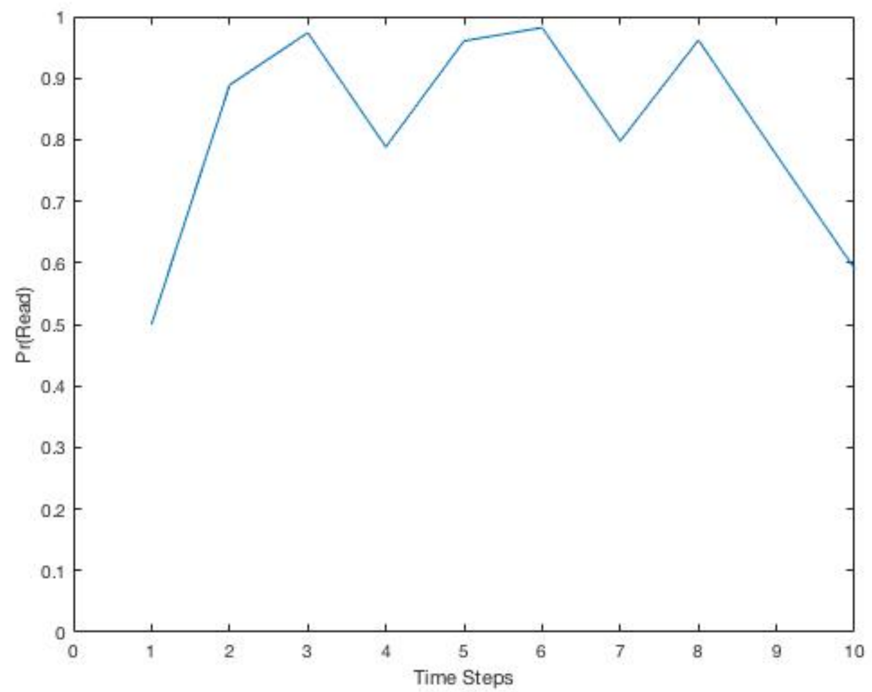


All values = 3 (too long)

Single Plot Results
Case 3: Controlled Randomness
(Sampled Evidence from DBN)

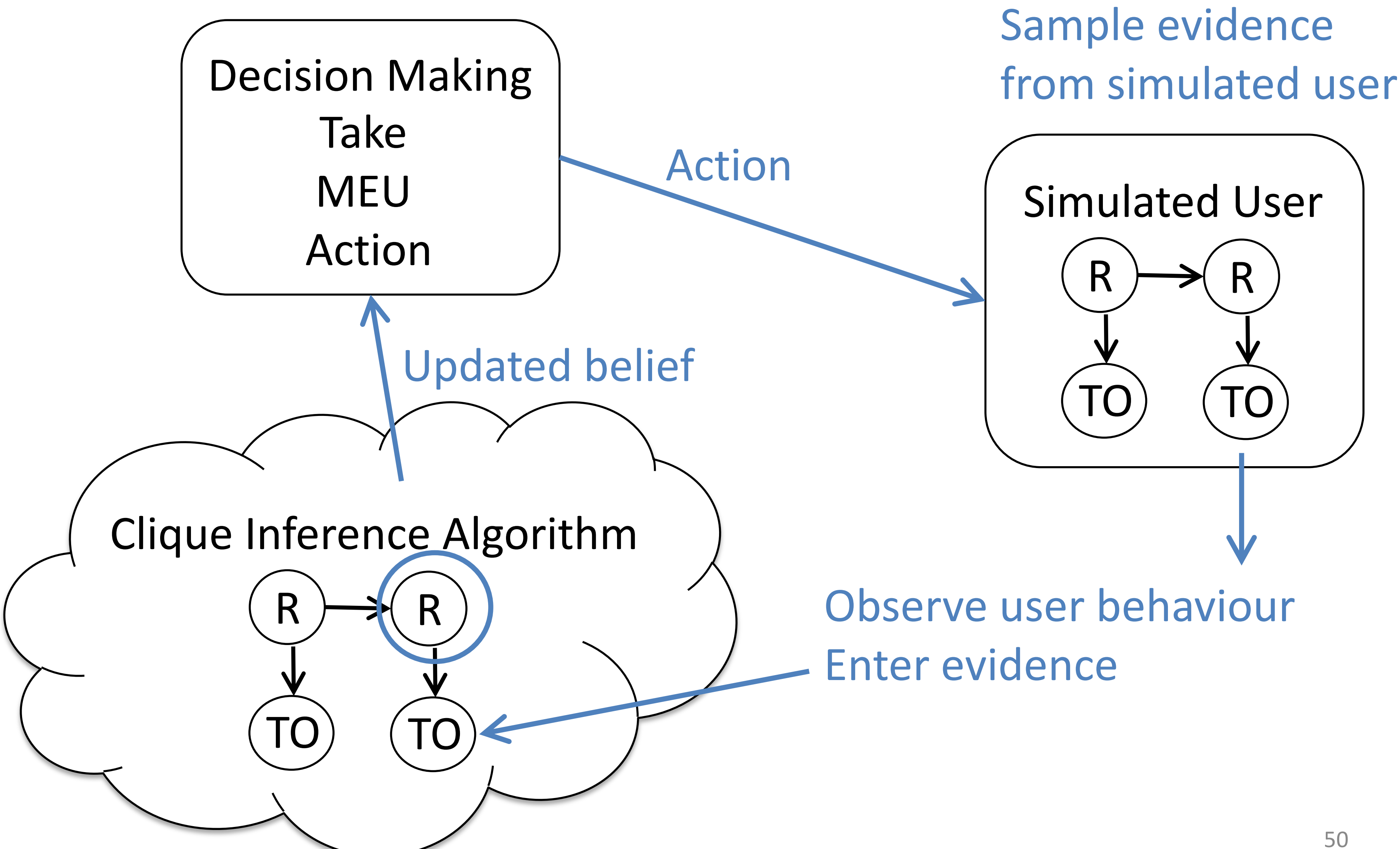


Given Read = 1 (false)



Given Read = 2 (true)

Simulation Interaction



U(Action, Read)

```
function util = util_hints( action, readHints )
% function util = util_hints( action, readHints )
%
% action = hint
%        = do nothing
% readHints = false, true
%
% util \in [-5,+5]
% function util = utility( action, needHhelp, readHints )
%

% doing stuff for the user gets a disruption penalty
util = 0;
if strcmp( action, 'Hint' ), util = util - 1; end;

% help action given will largely depend on whether user reads hints
if readHints == 0,
  if strcmp( action, 'Hint' ), util = util - 4; end;
else
  if strcmp( action, 'Hint' ), util = util + 5; end;
end;
```

Compute Expected Utility Inside get_meu_hints.m

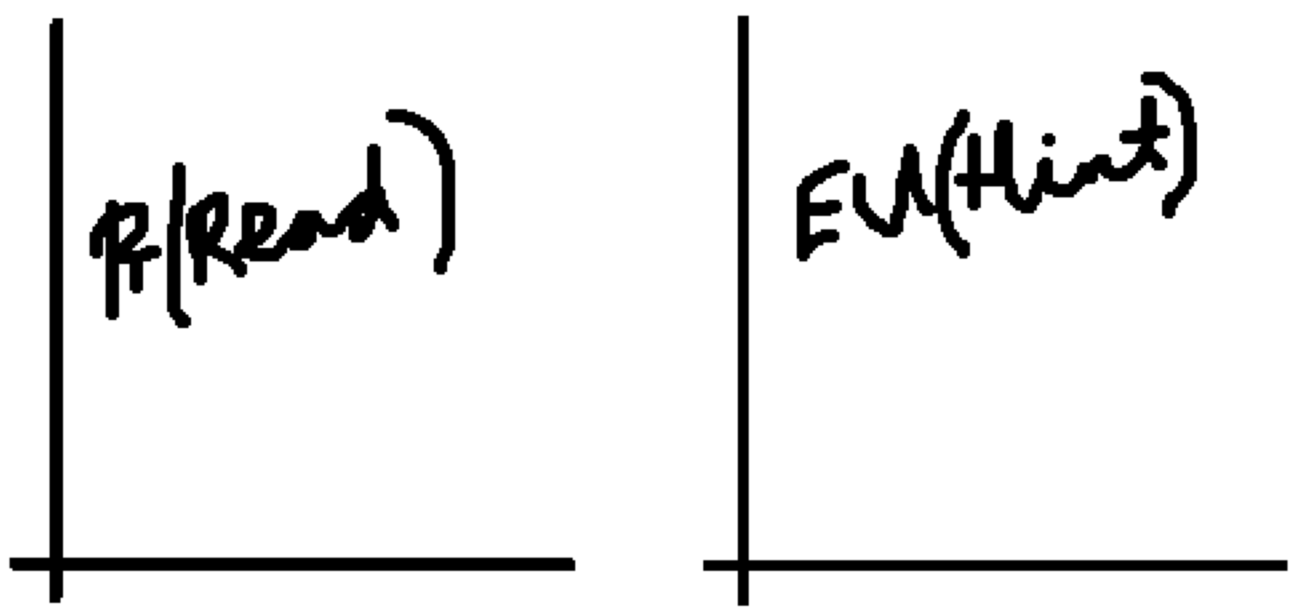
```
function [action, eu_hint] = get_meu_hints( prRead )
% function [action, eu_hint] = get_meu_hints( prRead )
%

% set default
action = 'None';

% compute expected utility of each action
% EU(A) = Pr(Read) x U(A, Read)
%
eu_hint = prRead * util_hints( 'Hint', 1 ) + ...
          (1 - prRead) * util_hints( 'Hint', 0 );

eu_none = prRead * util_hints( 'None', 1 ) + ...
          (1 - prRead) * util_hints( 'None', 0 );

% override default if hinting is a better action
if eu_hint > eu_none,
  action = 'Hint';
end;
```

Modified Simulation: sim_hints_decision.m

```
% setup inference process
%
% sample series of evidence in advance, say 10
%
% t=0:
%      prRead is 0.5 according to our model
%      get best action via expected utility computation
%
% t=1:
%      enter first piece of evidence
%      update belief by computing marginal prRead
%      get best action via expected utility computation
%
% for t=2 to t=T:
%      enter evidence at t
%      update belief by computing marginal prRead
%      get best action via expected utility computation
```

Inside sim_hints_decision.m

```
% setup results to be stored
belief = [];
exputil = [];
subplot( 1, 2, 1 );    % setup plot for graph
```



Inside sim_hints_decision.m

```
% setup results to be stored
belief = [];
exputil = [];
subplot( 1, 2, 1 );     % setup plot for graph

% at t=0, no evidence has been entered, so the probability is same as the
% prior encoded in the DBN itself
%
prRead = get_field( dbn.CPD{ dbn.names('Read') }, 'cpt' );
belief = [belief, prRead(2)];
subplot( 1, 2, 1 );
hold on;
plot( belief, 'o-' );
hold off;
```

% inference step

% plot belief

Inside sim_hints_decision.m

```
% setup results to be stored
belief = [];
exputil = [];
subplot( 1, 2, 1 );    % setup plot for graph

% at t=0, no evidence has been entered, so the probability is same as the
% prior encoded in the DBN itself
%
prRead = get_field( dbn.CPD{ dbn.names('Read') }, 'cpt' );
belief = [belief, prRead(2)];
subplot( 1, 2, 1 );
hold on;
plot( belief, 'o-' );
hold off;

% log best decision
[bestA, euHint] = get_meu_hints( prRead(2) );
exputil = [exputil, euHint];
disp(sprintf('t=%d: best action = %s, euHint = %f', 0, bestA, euHint));
subplot( 1, 2, 2 );
hold on;
plot( exputil, '*-' );
hold off;
```

% inference step

% plot belief

% plot EU

Inside sim_hints_decision.m (cont.)

```
% at t=1: initialize the belief state
%
[engine, ll(1)] = dbn_update_bel1(engine, evidence(:,1));

marg = dbn_marginal_from_bel(engine, 1);
prRead = marg.T;
belief = [belief, prRead(2)];
subplot( 1, 2, 1 );
hold on;
plot( belief, 'o-' );
hold off;
```

% inference step

% plot belief

Inside sim_hints_decision.m (cont.)

```
% at t=1: initialize the belief state
%
[engine, ll(1)] = dbn_update_bel1(engine, evidence(:,1));

marg = dbn_marginal_from_bel(engine, 1);
prRead = marg.T;
belief = [belief, prRead(2)];
subplot( 1, 2, 1 );
hold on;
plot( belief, 'o-' );
hold off;

% log best decision
[bestA, euHint] = get_meu_hints( prRead(2) );
exputil = [exputil, euHint];
disp(sprintf('t=%d: best action = %s, euHint = %f', 0, bestA, euHint));
subplot( 1, 2, 2 );
hold on;
plot( exputil, '*-' );
hold off;
```

% inference step

% plot belief

% plot EU

Inside sim_hints_decision.m (cont.)

```
% Repeat inference steps for each time step
%
for t=2:T,
  % update belief with evidence at current time step
  [engine, ll(t)] = dbn_update_bel(engine, evidence(:,t-1:t));

  % extract marginals of the current belief state
  i = 1;
  marg = dbn_marginal_from_bel(engine, i);
  prRead = marg.T;
```

% inference step

```
end;
```

Inside sim_hints_decision.m (cont.)

```
% Repeat inference steps for each time step
%
for t=2:T,
  % update belief with evidence at current time step
  [engine, ll(t)] = dbn_update_bel(engine, evidence(:,t-1:t));

  % extract marginals of the current belief state
  i = 1;
  marg = dbn_marginal_from_bel(engine, i);
  prRead = marg.T;

  % log best decision
  [bestA, euHint] = get_meu_hints( prRead(2) );
  exputil = [exputil, euHint];
  disp(sprintf('t=%d: best action = %s, euHint = %f', 0, bestA, euHint));
  subplot( 1, 2, 2 );
  hold on;
  plot( exputil, '*-' );
  xlabel( 'Time Steps' );
  ylabel( 'EU(Hint)' );
  axis( [ 0 T -5 5] );
  hold off;

end;
```

% inference step

% plot EU

Inside sim_hints_decision.m (cont.)

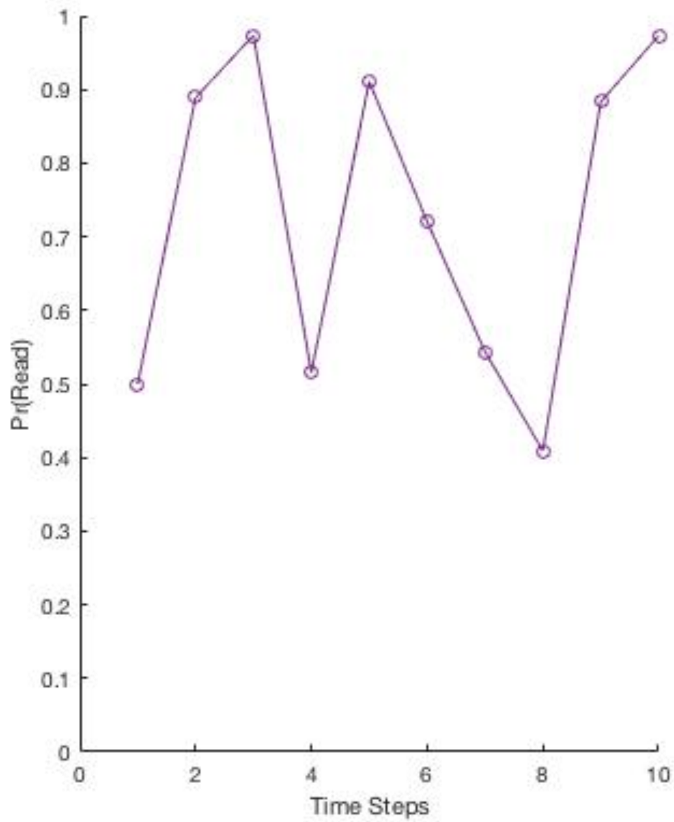
```
% Repeat inference steps for each time step
%
for t=2:T,
  % update belief with evidence at current time step
  [engine, ll(t)] = dbn_update_bel(engine, evidence(:,t-1:t));

  % extract marginals of the current belief state
  i = 1;
  marg = dbn_marginal_from_bel(engine, i);
  prRead = marg.T;

  % log best decision
  [bestA, euHint] = get_meu_hints( prRead(2) );
  exputil = [exputil, euHint];
  disp(sprintf('t=%d: best action = %s, euHint = %f', 0, bestA, euHint));
  subplot( 1, 2, 2 );
  hold on;
  plot( exputil, '*-' );
  xlabel( 'Time Steps' );
  ylabel( 'EU(Hint)' );
  axis( [ 0 T -5 5] );
  hold off;

  % keep track of results and plot it
  belief = [belief, prRead(2)];
  subplot( 1, 2, 1 );
  hold on;
  plot( belief, 'o-' );
  xlabel( 'Time Steps' );
  ylabel( 'Pr(Read)' );
  axis( [ 0 T 0 1] );
  pause(0.25);
  hold off;
end;
```

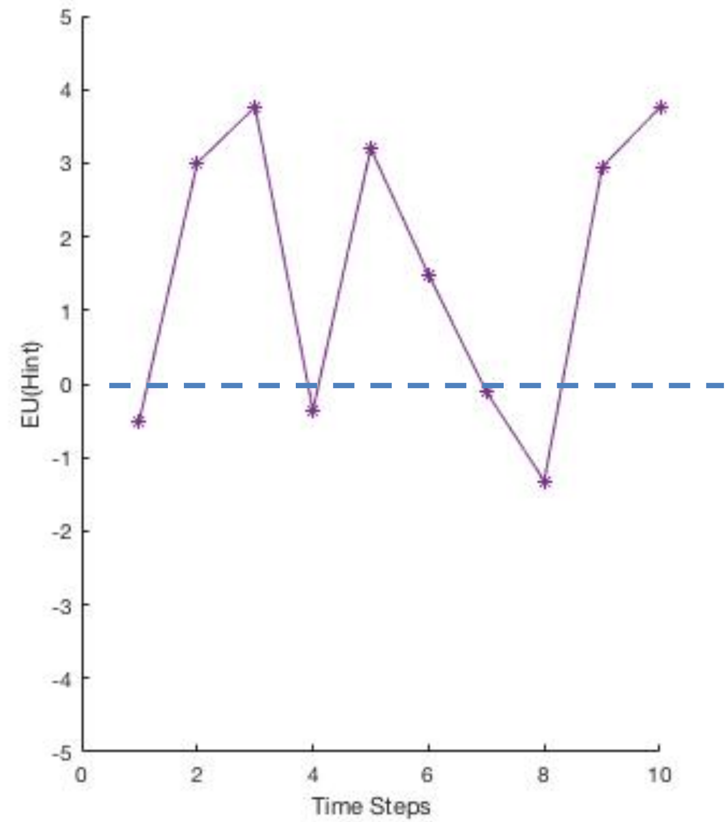
% inference step

% plot EU

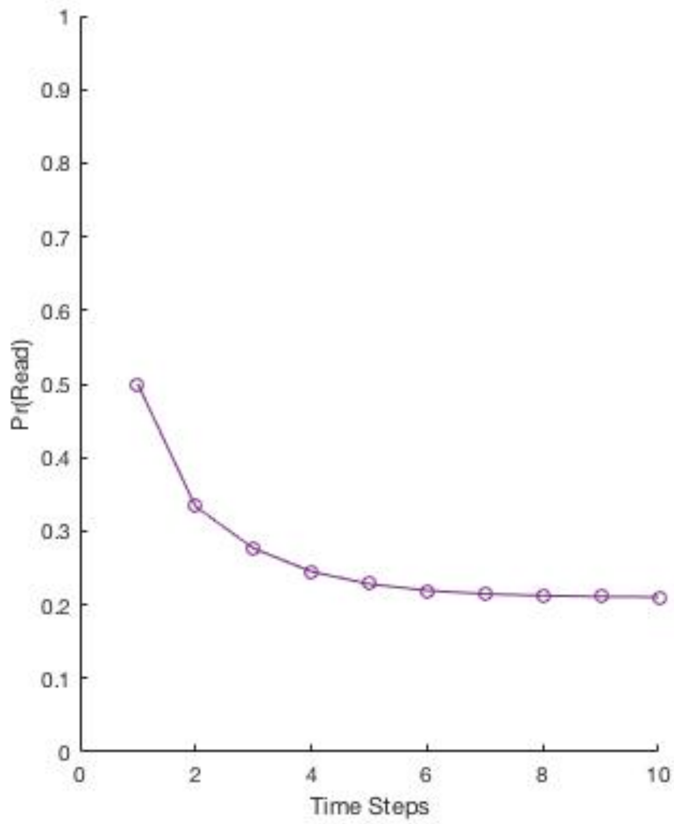
% plot belief

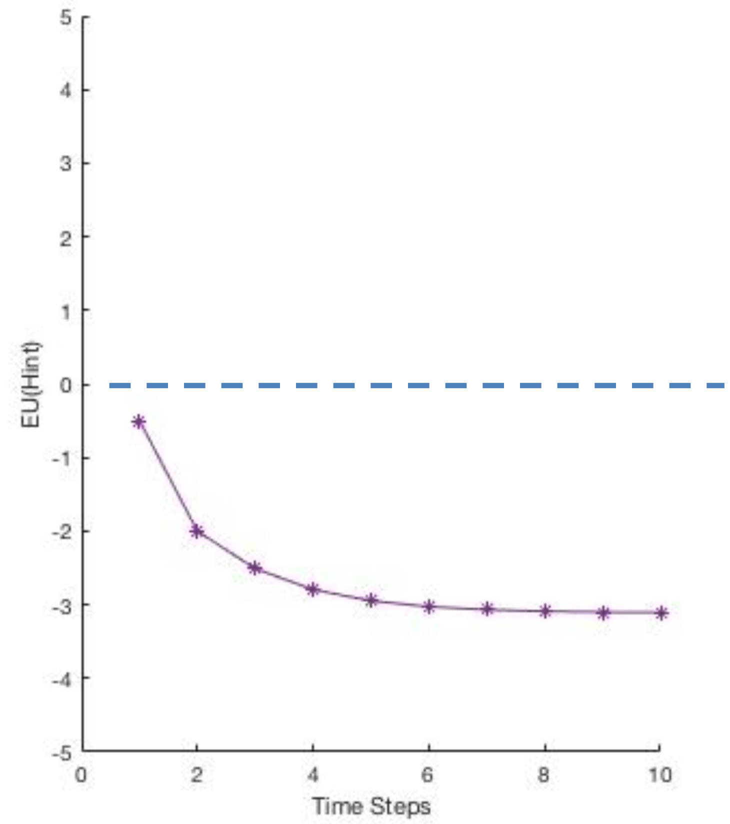
Single Plot Results
Case 1: Random Evidence





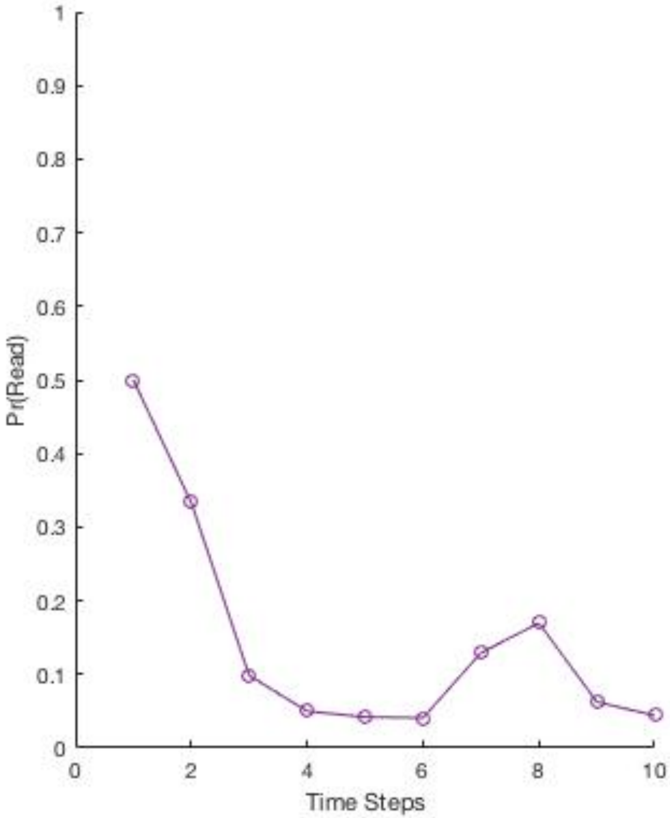
Single Plot Results
Case 2: Fixed Evidence

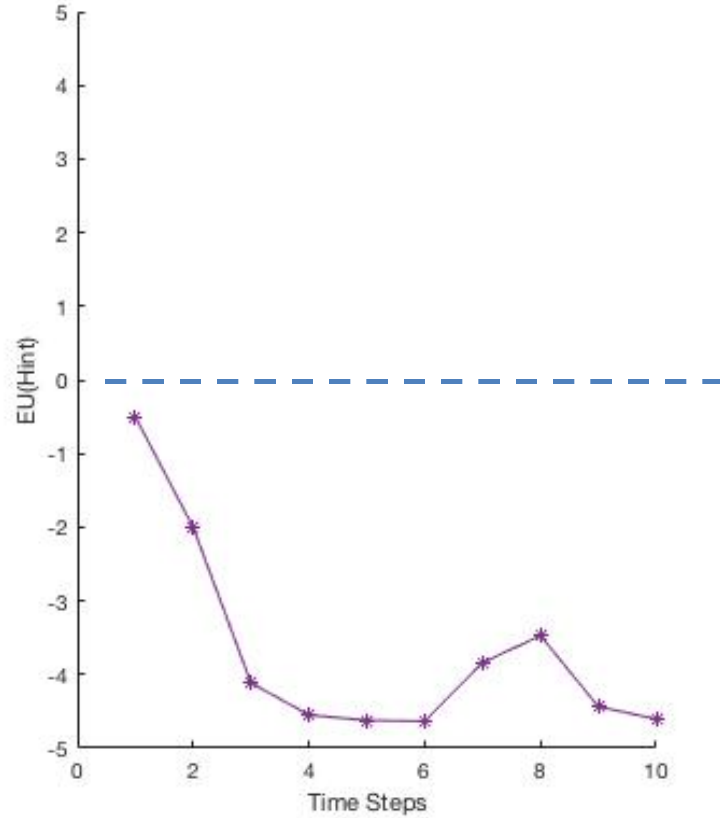




All values = 3 (too long)

Single Plot Results
Case 3: Controlled Randomness
(Sampled Evidence from DBN)





Given Read = 1 (false)

Overview of A5

- Step 1: download files to reproduce previous slides
- Step 2-3: adapt to new problem
  - Instead of mk_hints.m, create your own file for the specified DBN
  - Then adapt sim_hints.m (and associated files) to get it to work on your new DBN
  - Then adapt sim_hints_decision.m (and associated files) to get it to work on your new model

Key Ideas

- Simulation setup
  - System:
    - Encode inference model
    - Algorithm to compute marginal distribution
    - Decision making (compute expected utility)
  - Simulated user
    - Encode model – sample evidence, respond to system action
- Average out results over many trials
  - Properly understand general behaviour
  - Typically: hundreds or thousands of trials