Recognizing Plan/Goal Abandonment

Christopher W. Geib

November 15th, 2002

Honeywell Laboratories

Talk Outline

ï Contribution: A model of plan/goal abandonment, and an implementation of the theory within the the probabilistic hostile agent task tracker (PHATT).

ï Talk Outline:

ñ Foundations of PHATT

ñ Recognizing abandoned PHATT goals

ñ Applications of PHATT

ñ PHATT in the future

Motivation for studying.

ï Real world agentís do it all the time.

ñ Elders

(distracted from taking meds by the phone.)

ñ Hackers

(gives up on hacking and decideds to dos you.)

ñ Terrorists

(choosing a target on the basis of ease.)

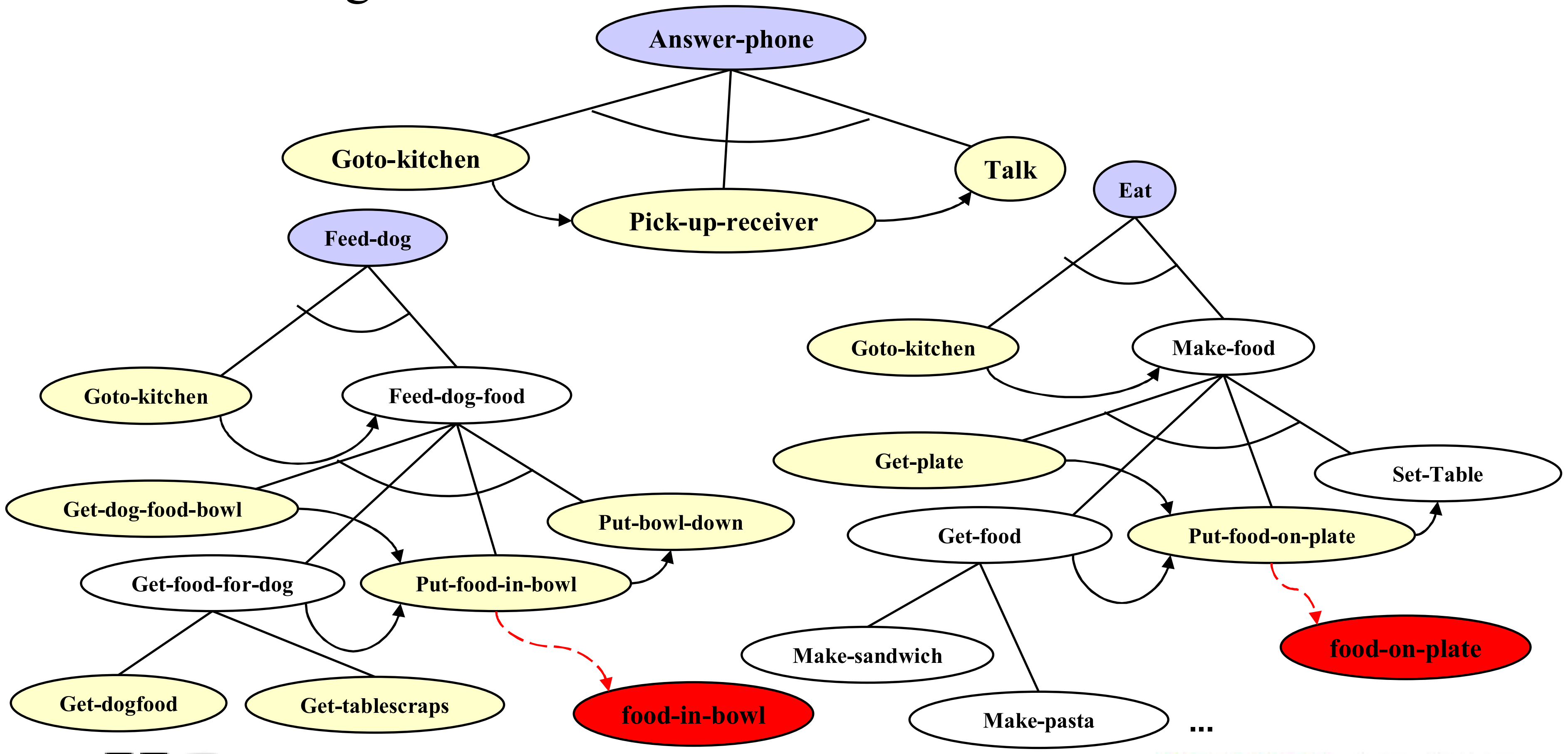
ï If you want to remind people you have to know what they are forgetting.

ï If you donít want to be confused about what they are doing you have to know what they have given up on.

Foundations of PHATT

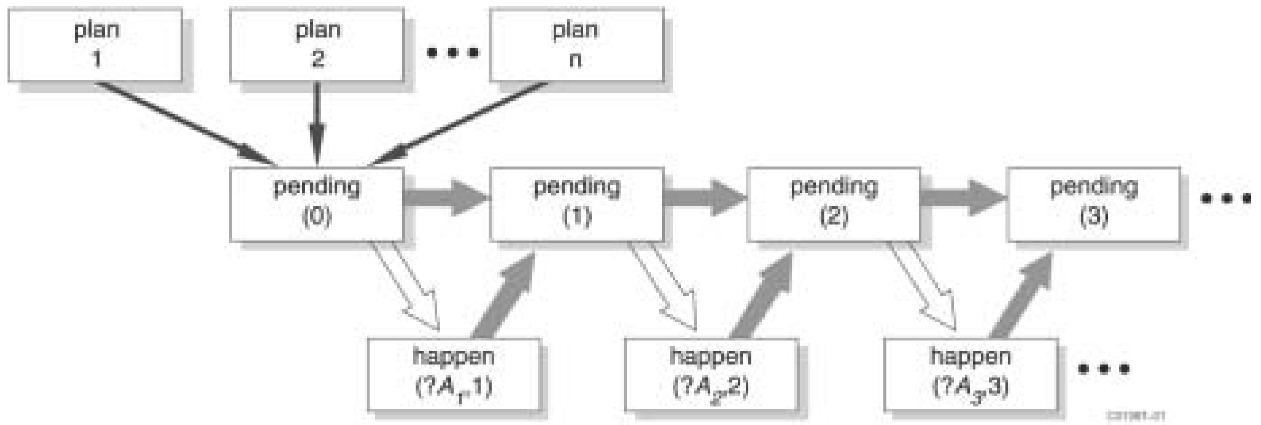
Background: Plan Library

ïAnd/Or tree representation of the set of possible plans
ïDistinguished goal nodes of the trees (most often the roots)
ïPartial ordering constraints



Background: Plan Execution Model

- ï Central Insight: the agents only execute actions that are consistent with their goals and are enabled by the previous actions they have performed.
- ï We define a ***pending set*** as the set of actions that are currently enabled by the agentís hypothesized goals and the actions the agent has taken.
- ï If we know the goals of an agent, we can perform a probabilistic simulation of the actions of a goal directed agent building ***explanations*** (execution traces that include the pending sets and plan structures).



Background: Probabilistic Simulation

ï What probabilities do we need to know?

ñ Probability of choosing a method (when you have a choice of plans.)

ñ Probability of choosing a given action from the pending set.

$$\Pr(\exp) = \prod_{j=0}^{J} \Pr(C_j) \prod_{k=0}^{K} \Pr(A_k)$$

Background: Recognition Algorithm Intuition

ï Flip probabilistic simulation model upside down

ñ Given the set of observations we can build the complete, exclusive and exhaustive set of the explanations for the observed actions. Inferring pending sets and resulting goals along the way.

ñ Establish the probability of each of the explanations (Pr(exp | obs))

- Note: POMDP

ï The conditional probability of the goal given the observations is just the sum of those explanations that have the goal divided by the probability of the observations. (Pr(g | obs))

$$\Pr(g \mid obs) = \frac{\sum_{e}^{Exp_g} \Pr(e \mid obs)}{\sum_{e}^{Exp} \Pr(e \mid obs)}$$

Honeywell Laboratories

Background: Computing Pr(exp)

ï What probabilities do we need to know? Same as before plus one.

ñ Prior probability of a given root intention (NEW)

ñ Probability of choosing a method (when you have a choice of plans.)

ñ Probability of choosing a given action from the pending set.

(Uniform distribution assumption)

$$\Pr(\exp) = \prod_{i=o}^{I} \Pr(G_i) \prod_{j=0}^{J} \left(1 \Big/ \left|\text{Choice}_j\right| \right) \prod_{k=0}^{K} \left(1 \Big/ \left|\text{PS}_k\right| \right)$$

Background: Algorithm

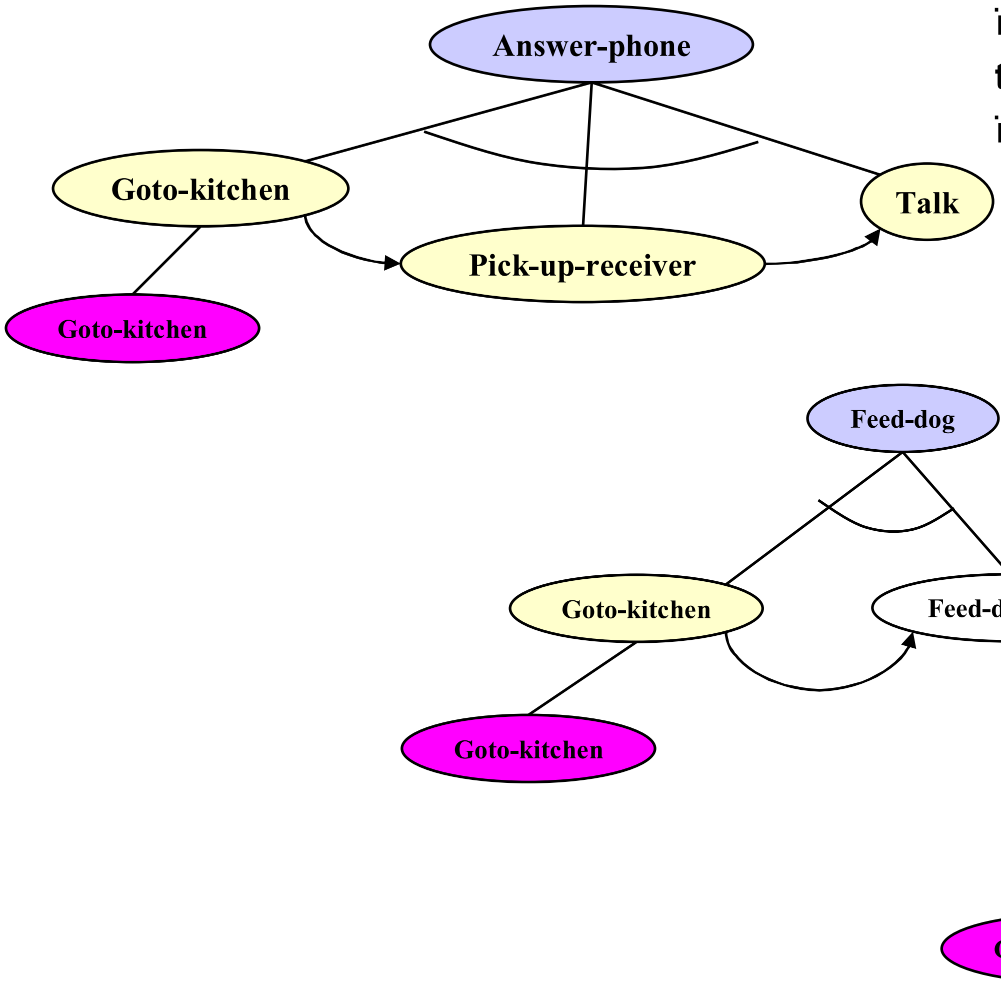
ï Inputs: a sequence of action observations, and a plan library

ï Output: the conditional probability for each root intention

ï Code:

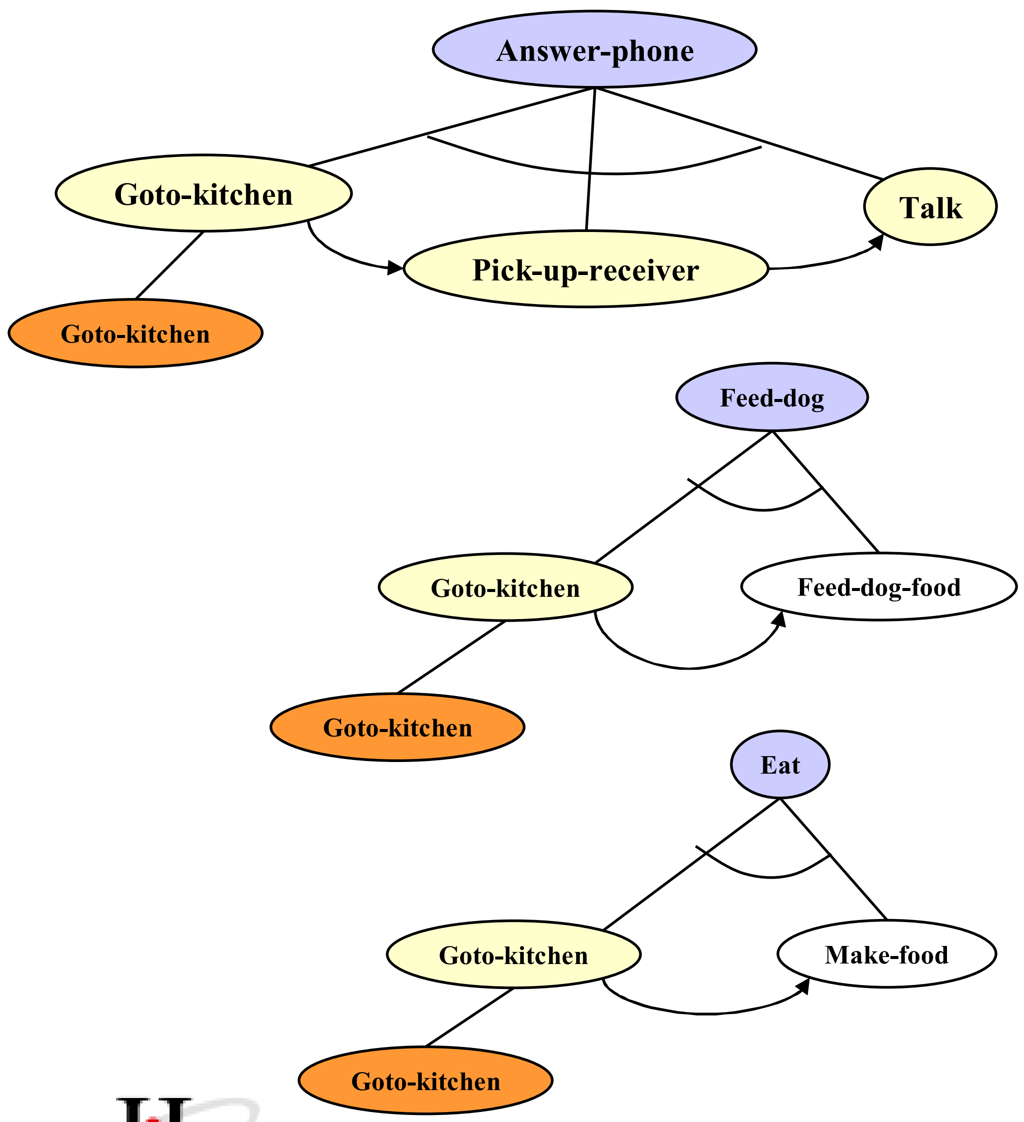
```
For each observation
     Progress pending set
          . remove executed action
          . add newly enabled actions
     Add to possible hypotheses set any new explanations indicated by the actions
     Remove from possible hypothesis set any explanations inconsistent with the executed
     action.
     Loop
For each explanation compute the explanations probability
For each root compute the conditional probability
```

Example: Pending Set at T0



ïThere are 3 elements in the pending set.
ïWe are assuming that the agent is only going to have one goal in this example.
ïPurple denotes a possible observation

Example: Execution Traces at T0



ïWe observe a Goto-kitchen event

ïOur belief about their goal depends mostly on the probability distribution of the roots.

ïAssume

Pr(Answer-phone) = 0.2

Pr(Feed-dog) = 0.3

Pr(Eat) = 0.4

ïCompute

Pr(E1 | Goto-kitchen) = 0.2 * 0.3 = 0.06

Pr(E2 | Goto-kitchen) = 0.3 * 0.3 = 0.09

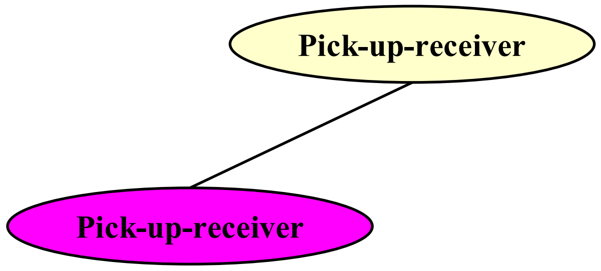
Pr(E3 | Goto-kitchen) = 0.4 * 0.3 = 0.12

ïPr(Answer-phone | obs) = 0.06 / 0.27 = .22

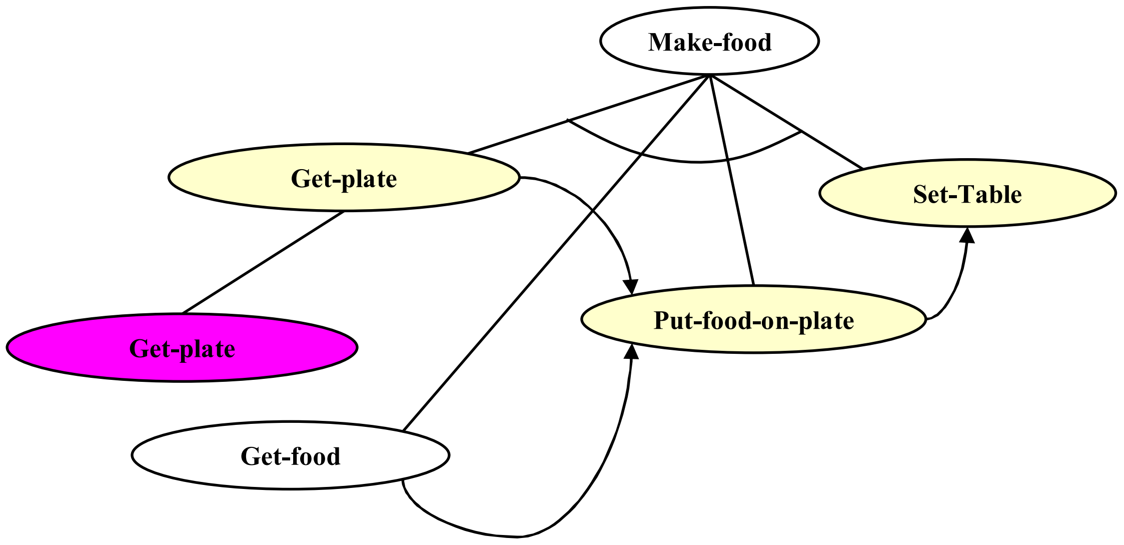
ïPr(Feed-dog | obs) = 0.09 /0.27 = .33

ïPr(Eat | obs) = 0.12 / 0.27 = .44

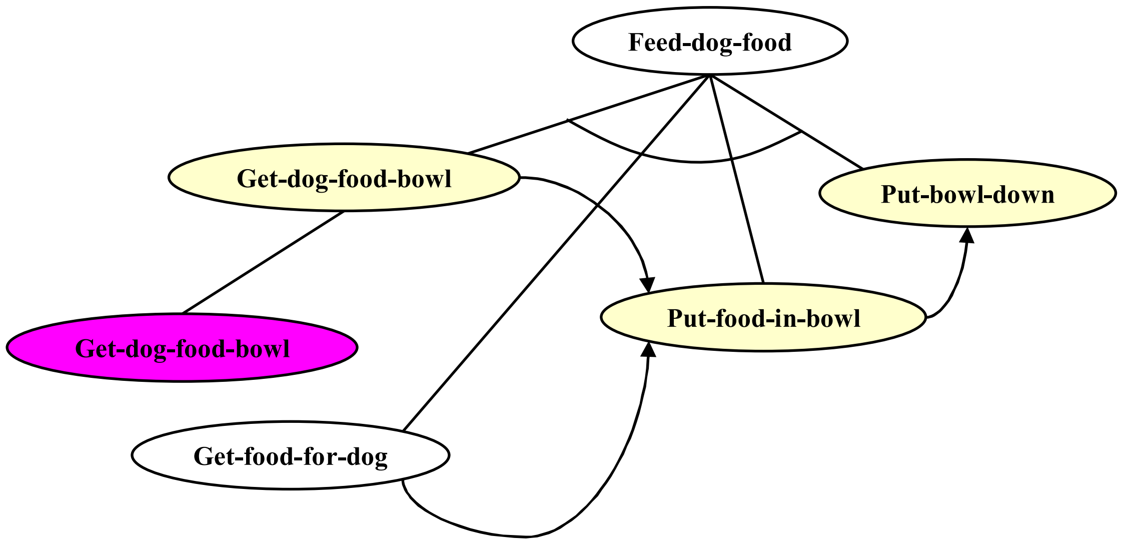
Example: Pending Set at T1(for Answer Phone)



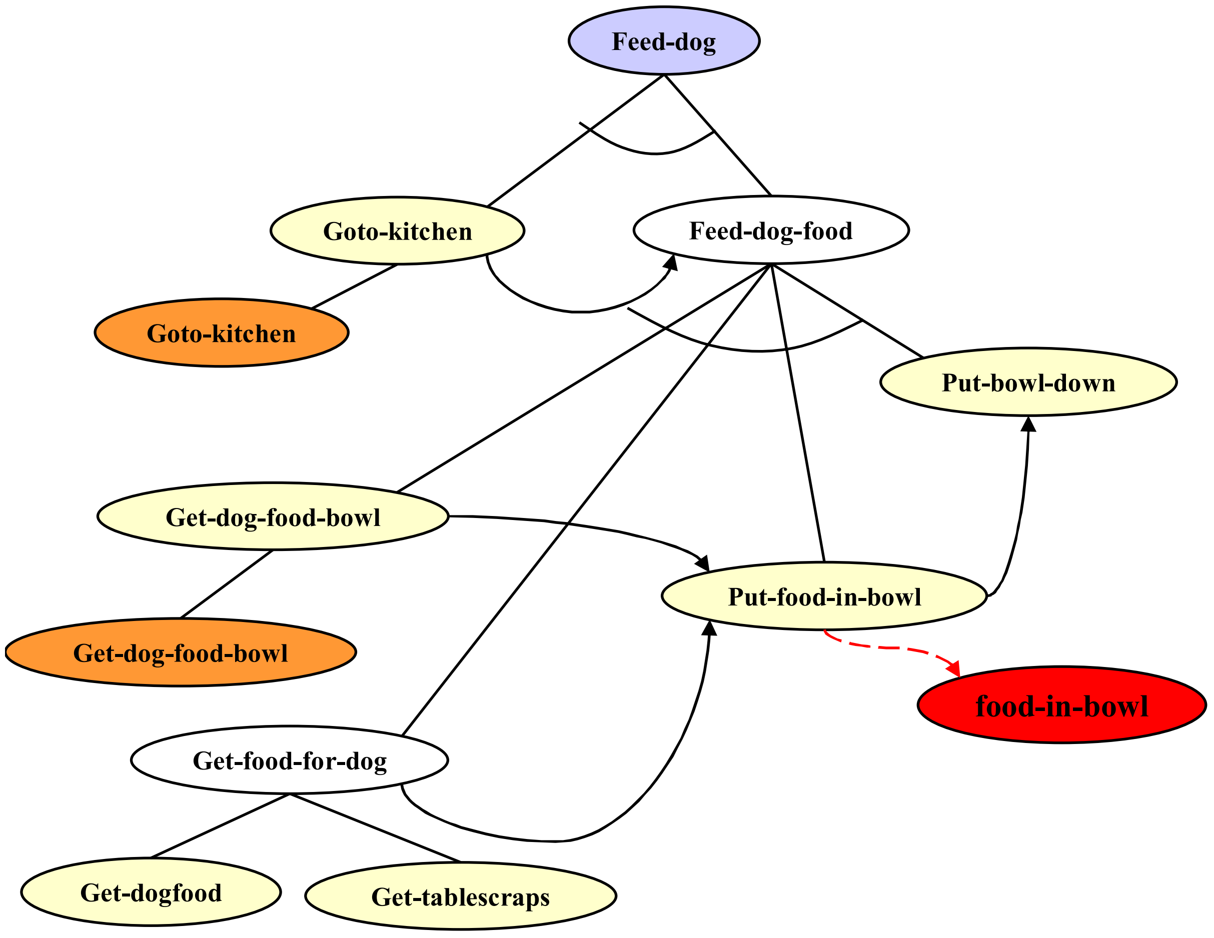
Example: Pending Set at T1 (for Eat execution trace)



Example: Pending Set at T1 (for Feed-dog trace)



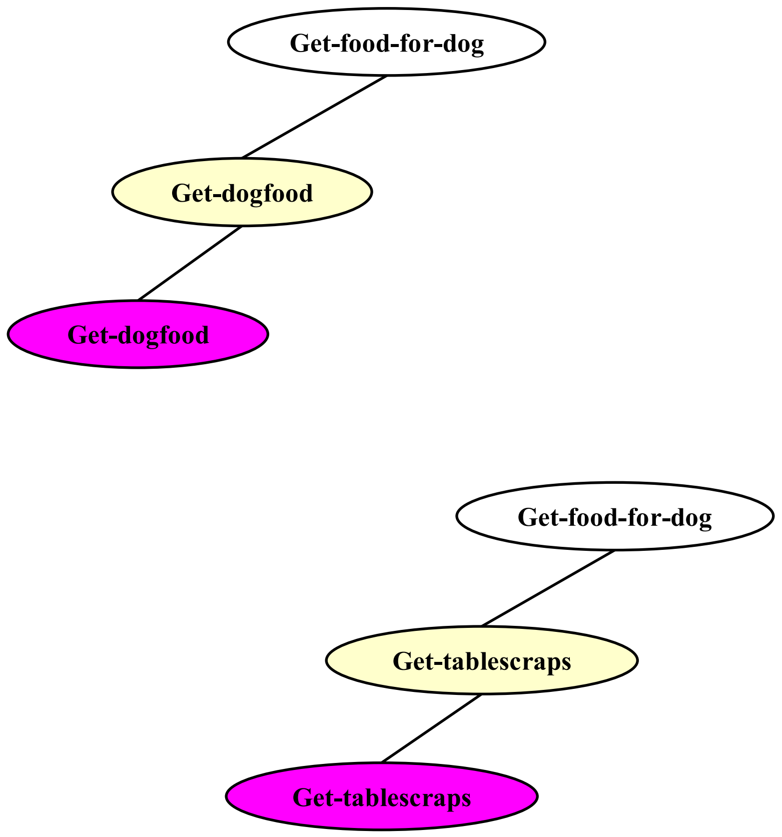
Example: Execution Trace at T1



ïAssume that the action chosen for execution at time T1 was Get-dog-food-bowl

ïNote that this results in only 1 possible execution traces. Since only the one plan is consistent with all the observations

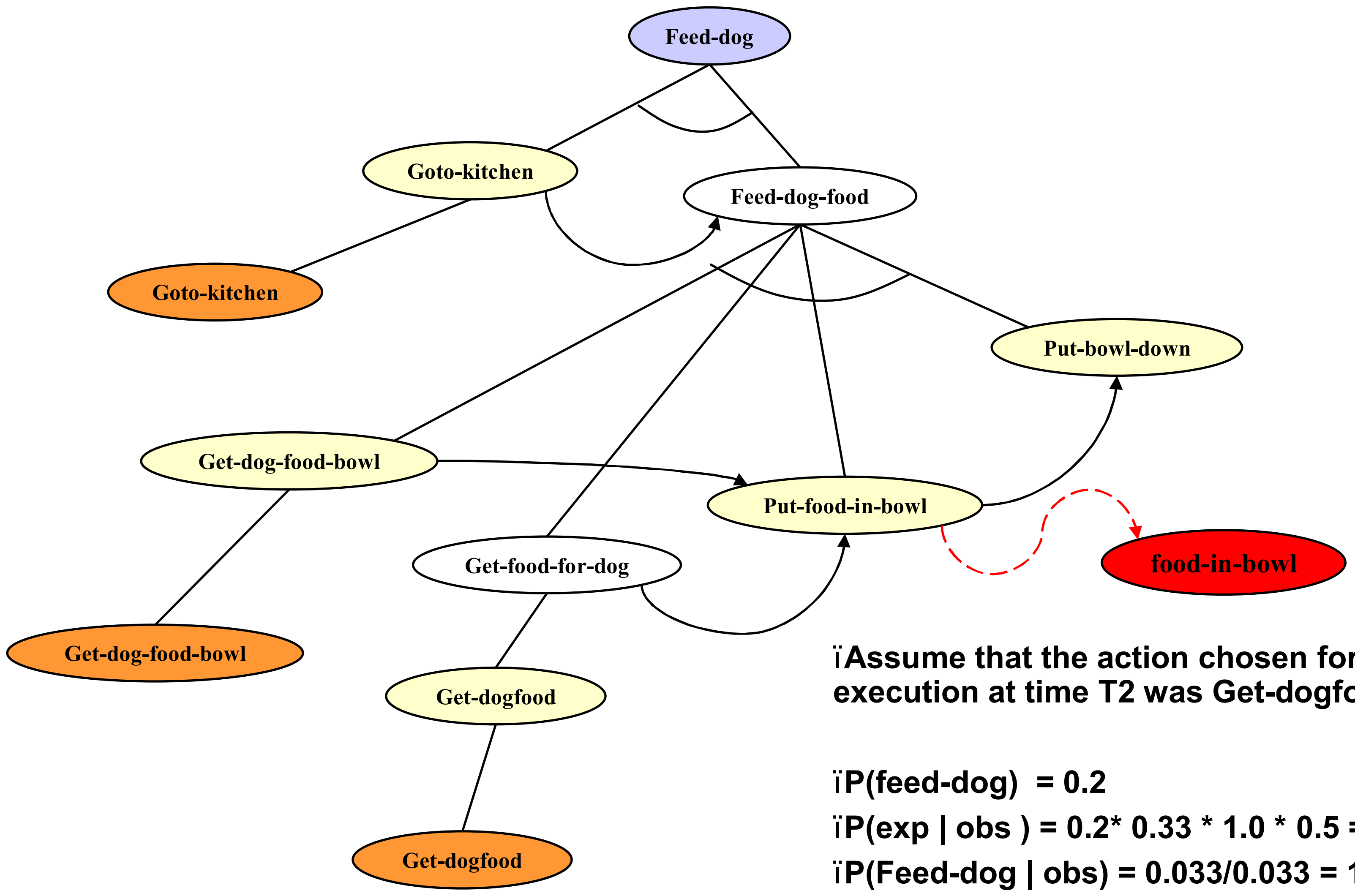
Example: Pending Set at T2



ïThere is only one viable execution trace.

ïSince the plan for getting the food was an
ì or nodeî there is a choice about how to
expand the plan and therefore there are 2 elements
in the pending set.   (they have the same
attachment point namely: Get-food-for-dog)

Example: Execution Trace at T2



ïAssume that the action chosen for execution at time T2 was Get-dogfood

ïP(feed-dog) = 0.2

ïP(exp | obs) = 0.2* 0.33 * 1.0 * 0.5 = 0.033

ïP(Feed-dog | obs) = 0.033/0.033 = 1.0

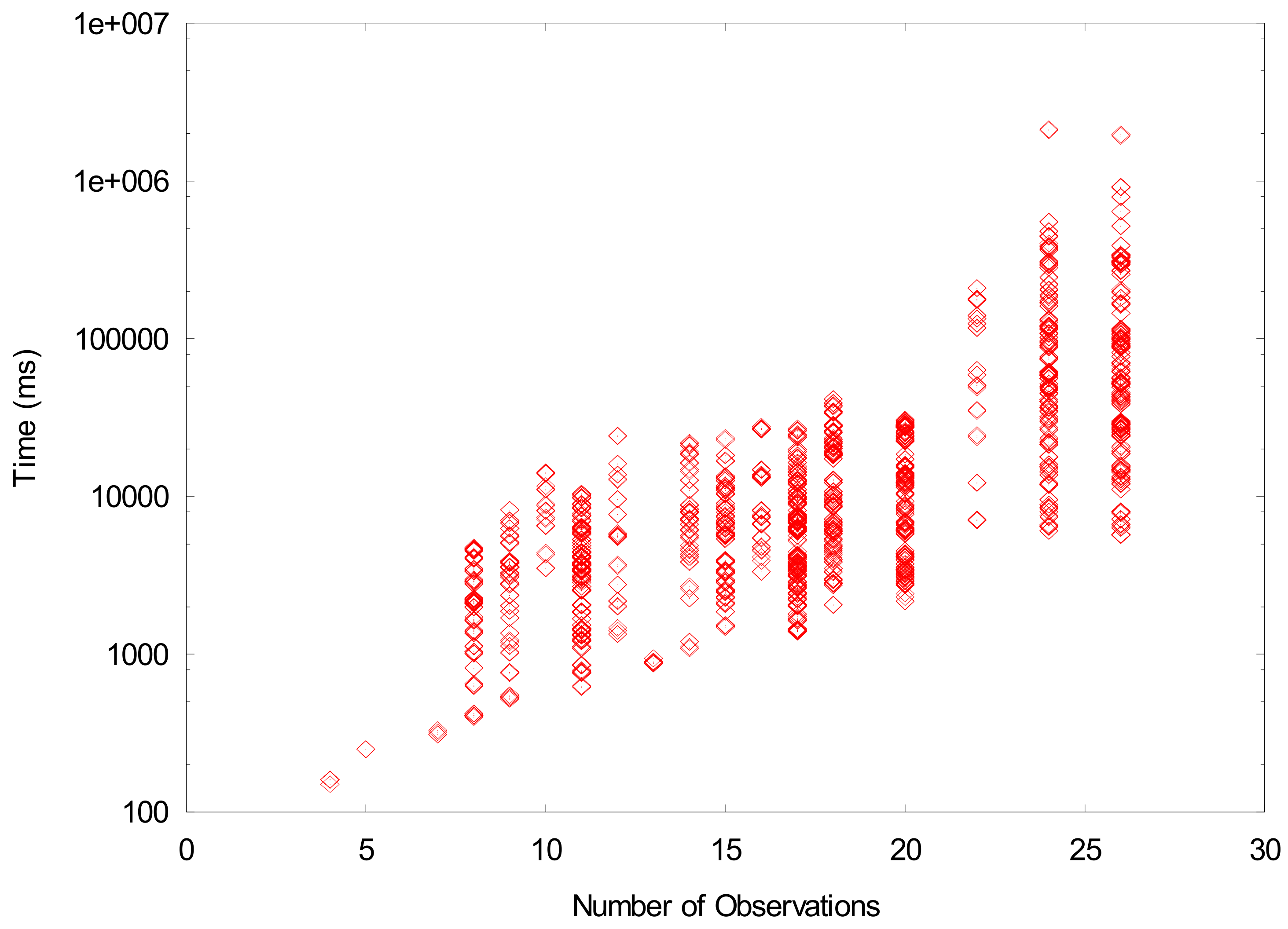
Background: Implementation

ï Implemented in ACL 6.2

ï Efficiency though significant pre-compilation and indexing

ï Some clever bookkeeping
  ñ keeping track of the size of the pending sets
  ñ variable bindings

Background: Runtimes



Background: Conclusions

ï This approach will handle:

ñ Partially observable action streams

ñ partial order plans

ñ effect of world state on goals

ñ overloading of actions

ñ multiple goal Vs. single goal explanations

ñ cumulative effect of not seeing something

ï But we can also extend it to handle

ñ Abandoned goals

Recognizing abandoned PHATT goals

Previous work defines away the problem

ï Only worry about the current goal: Horvitz ë98
  ñ Just trying to assist with the single current goal

ï Assume that the agent only has one goal: Conati ë97
  ñ Fine in tutorial systems where their goal is to learn

ï Assume the agent will come back: Kautz and Allen ë86
  ñ Assume that no goals are abandoned

ï Rely on a cooperative agent for disambiguation: Lesh ë01
  ñ Fine if you can assume cooperative agents

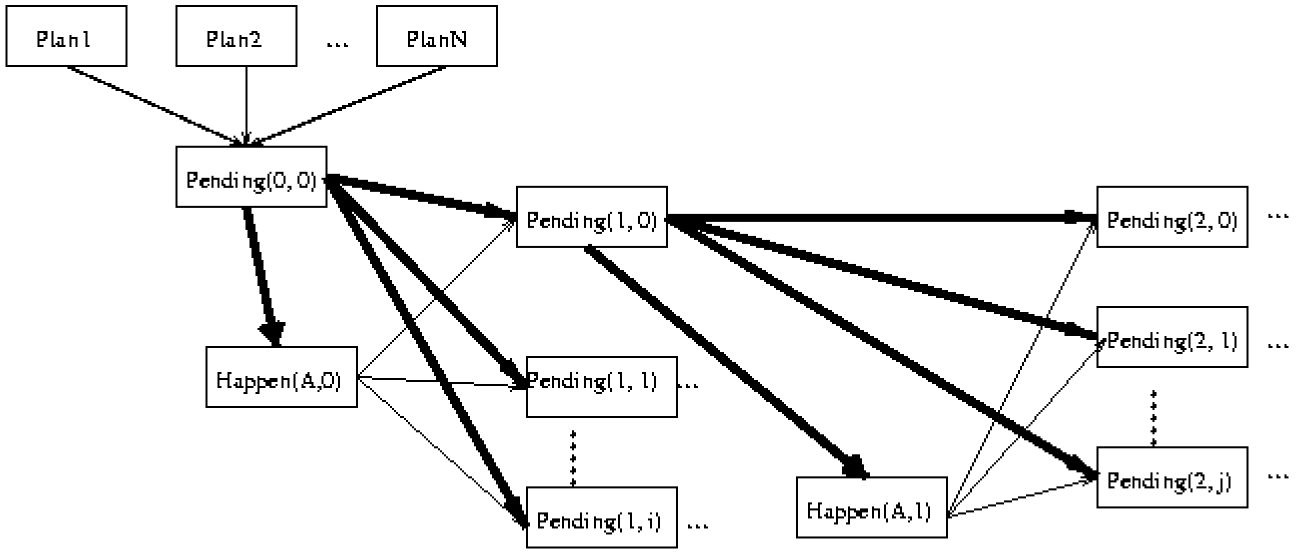
Goal Abandonment: exact solution

ï Abandoning goals moves from a unique pending set to a tree of them.

ï If we want to compute the exact probability that a given goal is abandoned the search space expands by 2^n at each time step.

  ñ Where n is the number of root goals the agent has

  ñ This reflects the fact that at any time step the agent can abandon any subset of their goals.



Exact Solution: Computing Pr(exp)

ï We must add to the explanation probability equation a term that reflects the possibility of goal abandonment.

ï Without abandonment:

$$\Pr(\exp) = \prod_{i=o}^{I} \Pr(G_i) \prod_{j=0}^{J} \left(1 \Big/ |\text{Choice}_j| \right) \prod_{k=0}^{K} \left(1 \Big/ |\text{PS}_k| \right)$$

ï With abandonment:

$$\Pr(\exp) = \prod_{i=o}^{I} \Pr(G_i) \prod_{j=0}^{J} \left(1 \Big/ |\text{Choice}_j| \right) \prod_{k=0}^{K} \left(1 \Big/ |\text{PS}_k| \cdot \Pr(\text{PS}_{k+1} \mid \text{PS}_k, \text{obs}_k) \right)$$

Exact Solution: Why we canít do this

ï The expansion of the search space is a problem.

ï Even if we didnít mind the search space we need a probability distribution over the possible abandoned goals

ñ Philosophical issues

ñ canít assume that goal abandonment is independent of other goals

ñ canít assume that goal abandonment is independent of situations

Approximating: Model mismatch

ï Laskey ë91, Jensen ë90, Haberman ë76, and others have suggested that exceptionally low values for:

$$\Pr(\text{observations} \mid \text{model})$$

are an indication of a probabilistic model mismatch.

ï In our case, the mismatch we are looking for is the abandonment of the goal.

ï We can look for something more specific than the probability of the whole observation stream to indicate the mismatch. Look for the probability that an action has contributed to the goal.

Approx algorithm (one explanation)

ï Compute the probability that nothing has contributed to a specific goal in this explanation. (ie. The probability that you still have the goal but have just naturally failed to execute any of the actions in the plan for it.)

$$\Pr(\mathrm{notcontrib}(q,s,t) \mid \mathrm{model}, \mathrm{obs}) = \prod_{i=s}^{t} 1 - \left(m_{q,i} \big/ |\mathrm{PS}_i| \right)$$

Where $m_{q,i}$ is the number of actions that contribute to goal q at time i

ï If this drops below a user defined threshold assume the goal has been abandoned.   Modify the explanation to abandon the goal and continue as before.

Estimating the probability

ï So having the used the model mismatch to find (and compute the probability of) explanations that assume that a goal has been abandoned we can now estimate the probability of the goal being abandoned across all the explanations.

ï Divide the probability mass of those explanations where the goal is abandoned by the probability mass of all explanations of the observations.

$$\Pr(abandoned(g) \mid obs) \approx \frac{\sum_{e}^{Exp_{A(g)}} \Pr(e \mid obs)}{\sum_{e}^{Exp} \Pr(e \mid obs)}$$

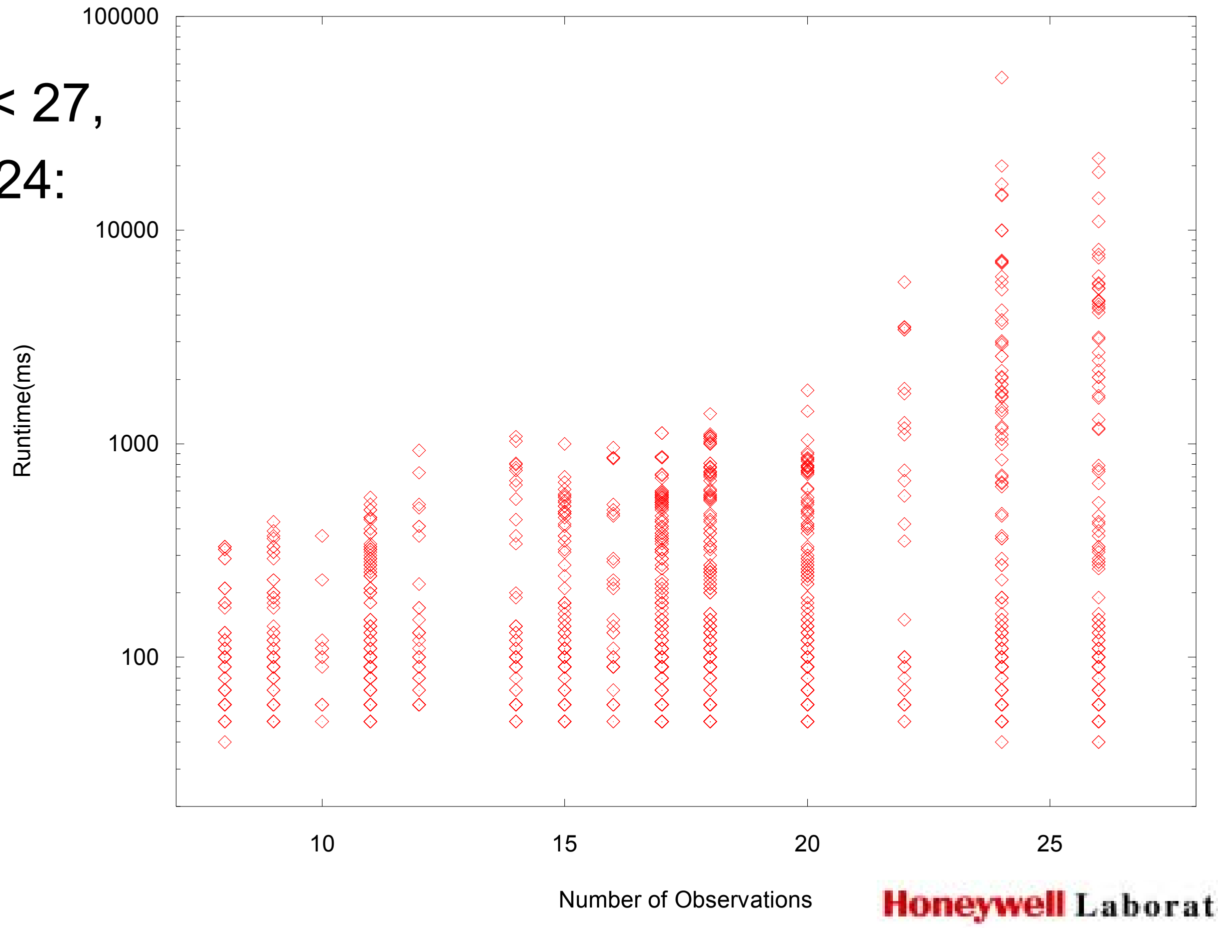
Implementation of approximate algorithm

ï Reasonable runtimes achieved even with large numbers of observations and a reasonably high threshold (to encourage abandoning goals)

ñ 2 goals,

ñ observations < 27,

ñ threshold = 0.24:



Evaluation of abandonment

ï Looking at trying to bound the approximation

ï Looking at trying to identify the correct threshold value.

ï Correctly identified all plans that didnít abandon a goal

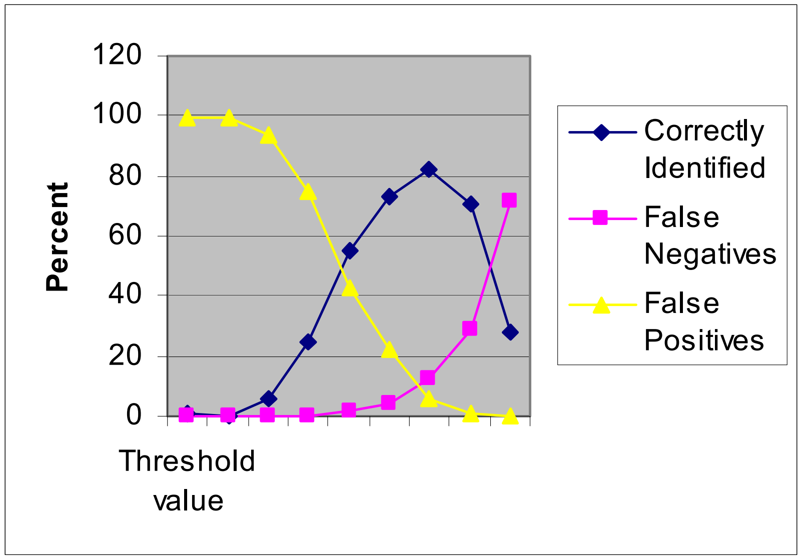
ï Problems with evaluating:

  ñ Not always enough evidence for the system to find

  ñ Abandoning can lead to plans that canít be explained

  ñ Structural issues

ï Threshold .30 => Accuracy .81



Applications of PHATT

Applications

ï NIST Abnormal Situation Management ë99: study project for assistant systems for oil refineries. Watch plan operatorís actions and infer their goals. When possible suggest actions that might assist them in their task.

ï DARPA CyberPanel project ë00-í01: Use reports from existing computer network intrusion detection systems as observations of attacker behavior

ï NIST Independent Lifestyle Assistant ë00-í03: assistant systems for at home elders. Task tracking used to recognize abandoned goals and remind elders.

ï 2 DARPA seedling projects ë02: looking at insider threat, distributed network defense.

ï Educational or mentoring systems

ï Intelligence analysis applications (business or governmental)

Analysis

ï HEY WAIT A MINUTE! Those runtimes look exponential!

ï Preliminary analysis: the good news -

  ñ The length of the observations is NOT the problem

ï Preliminary analysis: the bad news -

  ñ The number of possible non-differentiated plans IS the problem

ï Factors related to the number of non-differentiated plans

  ñ Unobserved probability threshold (too low and too high both bad)

  ñ Branching factor of the plan library

  ñ Longest non differentiating prefix

PHATT in the future

Future work

ï More complete analysis of the features affecting the runtime.

ï Bounding approximation errors for abandoned goals.

ï Industrial grade implementation.

  ñ precompilation of sequences and use of string matching algorithms.

  ñ application of Tree-Adjoining Grammars and O(n^6) parsing??

ï Relationship to existing data mining research

ï Misdirection (Execution of actions simply to mislead the observer.)

ï Dissambiguating multiple agents

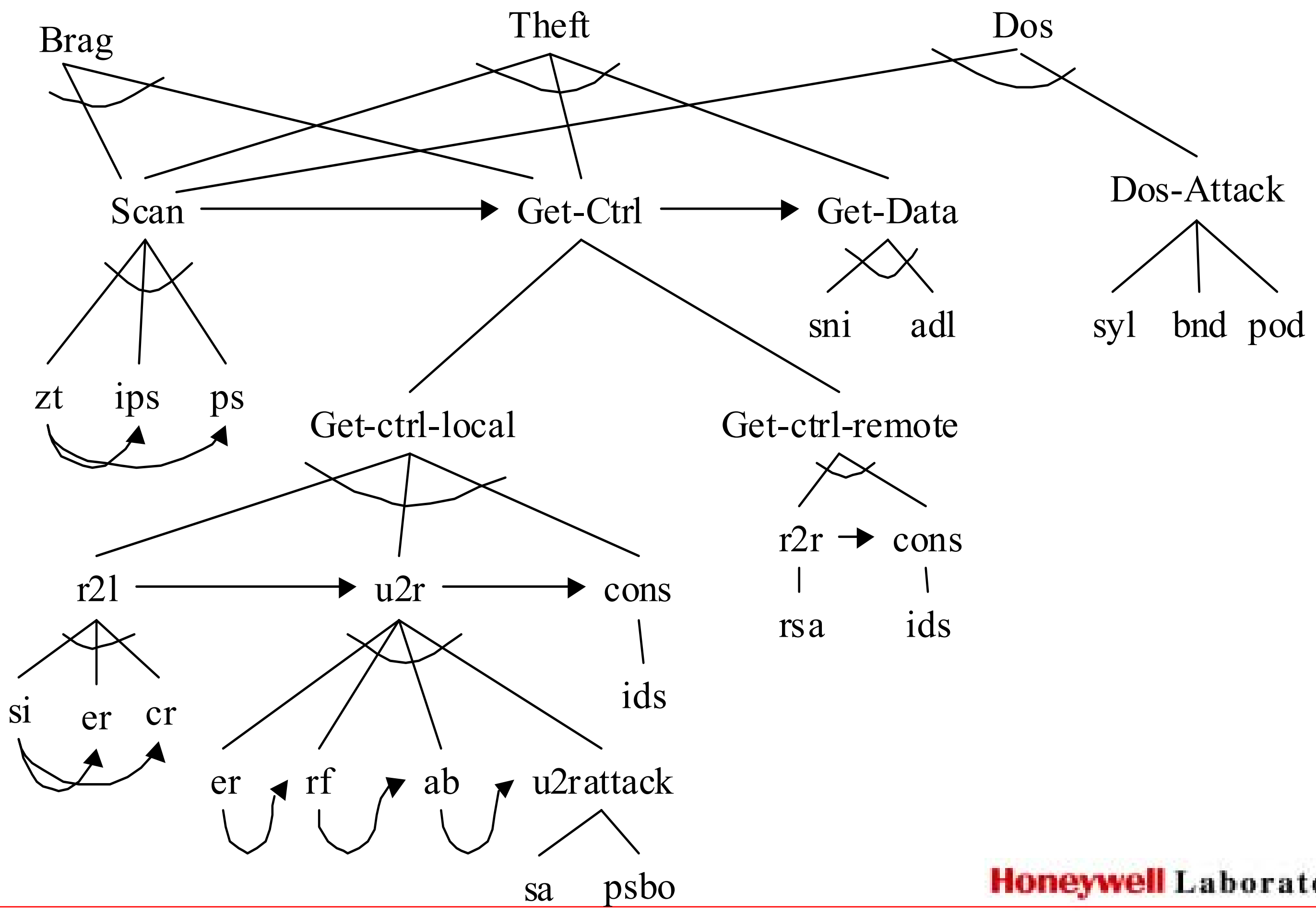
Conclusions

Contribution: A new formal theory and implementation of recognizing plan/goal abandonment.

Thank you

Background: Plan Library

ïAnd/Or tree representation of the set of possible plans
ïDistinguished goal nodes of the trees (most often the roots)
ïPartial ordering constraints



Background: Previous Approaches

ï Set covering (Kautz and Allen ë86)

ï Probabilistic abduction (Charniak and Goldman ë93)

ï Grammar formalisms (Sidner ë85, Vilain ë90, Wellman and Pynadath ë97)

ï Reactive systems (Huber ë94, Rao ë94)

ï Game theory, Mini-max search (Carmel and Markovitch ë96)

Summary

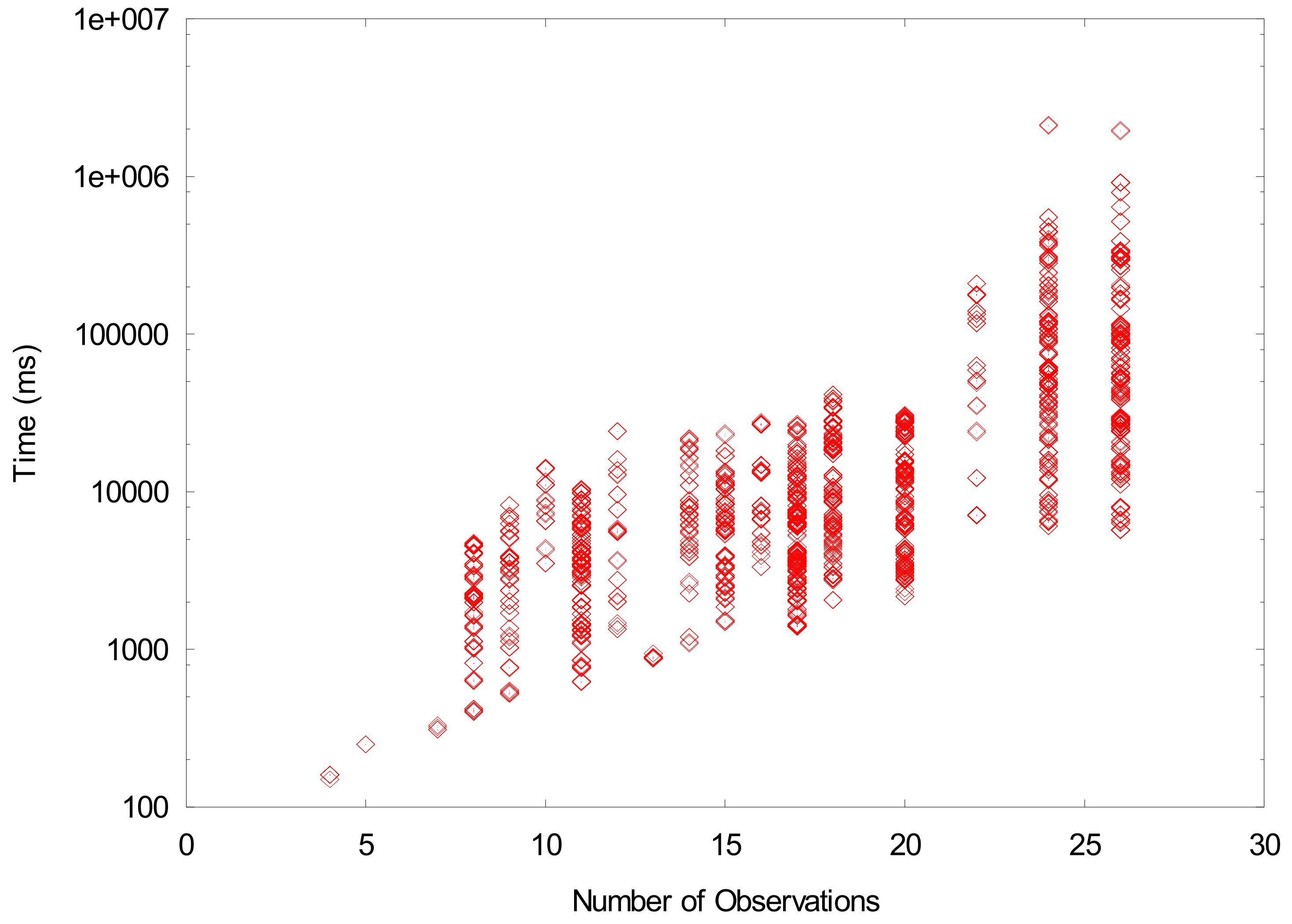
ï We need to be able to do high level inference of the goals/plans of an attacker in order to be able to take active and effective countermeasures.

ï Plan Recognition addresses this need and is a well studied area of research in AI with a large body of results that we are not making sufficient use of.

ï There are issues that still need to be addressed in the plan recognition literature that are requirements for effective use in the network security domain.

ï We present a formal model of recognizing plan abandonment in plan recognition systems for network security systems.

Implementation with no unobserveds



Network Security Needs Plan Recognition

- ï Taking effective preventative measures to counteract an attackerís actions requires understanding the attackerís goals.
- ï Example a denial of service attack from an external source
  - - attacker goal: DOS of single host:
    - ñ reject/block packets for the host
    - ñ limit number of connections to the host
  - - attacker goal: IP spoofing attack
    - ñ all other hosts should refuse connections purporting to come from the attacked host.
  - - Responses for different intentions are not compatible
- ï We are talking about inferring goals at the ìnext level upî of abstraction from the systems we have now.

Implementation

ï Based on Pooleís Probabilistic Horn Clause Abduction

ñ Push all probability out to axioms of a proof

ñ Use first order logic to do your reasoning

ñ Requires an exclusive and exhaustive set of explanations

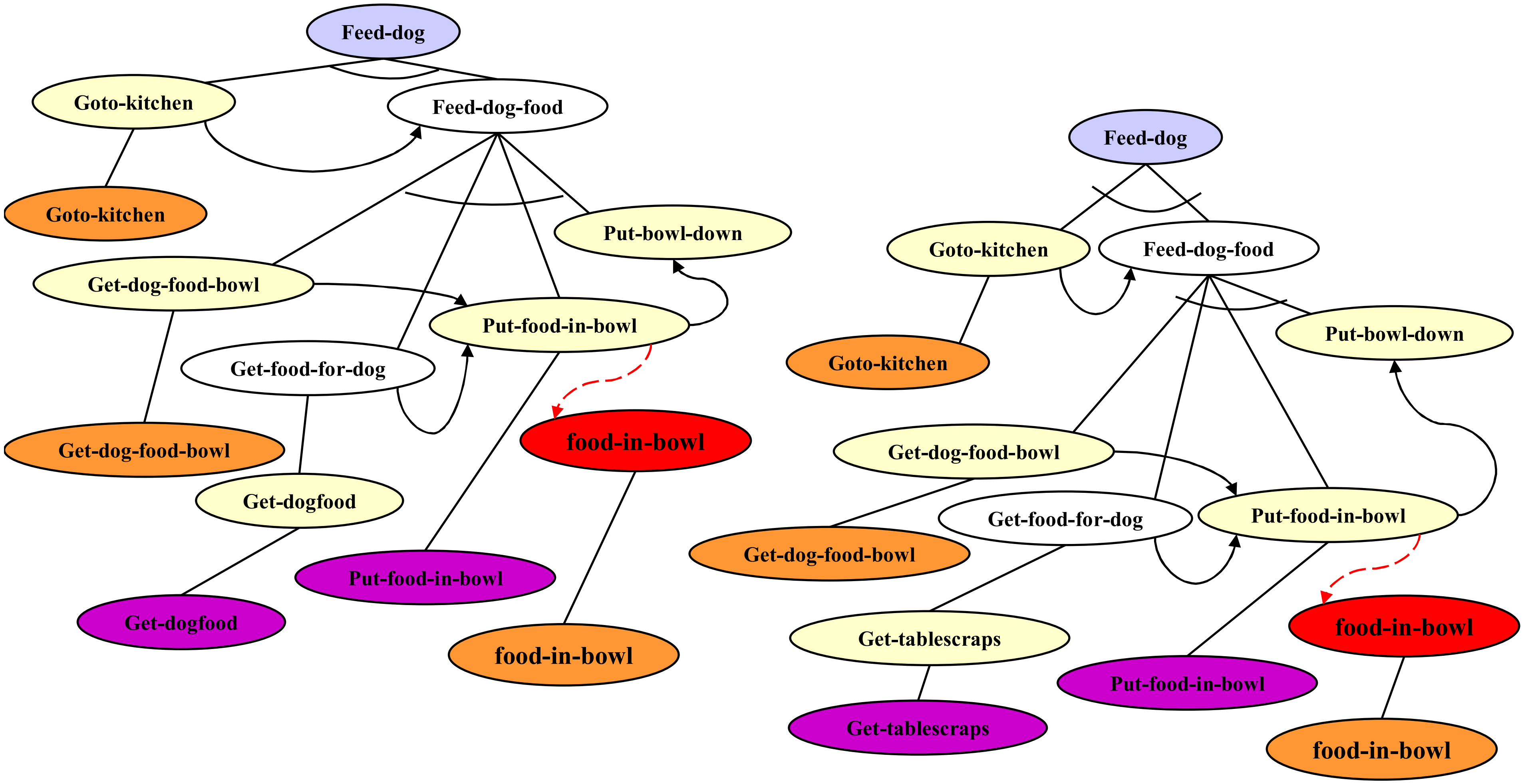
ñ Compute the probability of a single explanation by considering the ìaxiomsî of the explanation and their likelihood.

ï Our current implementation

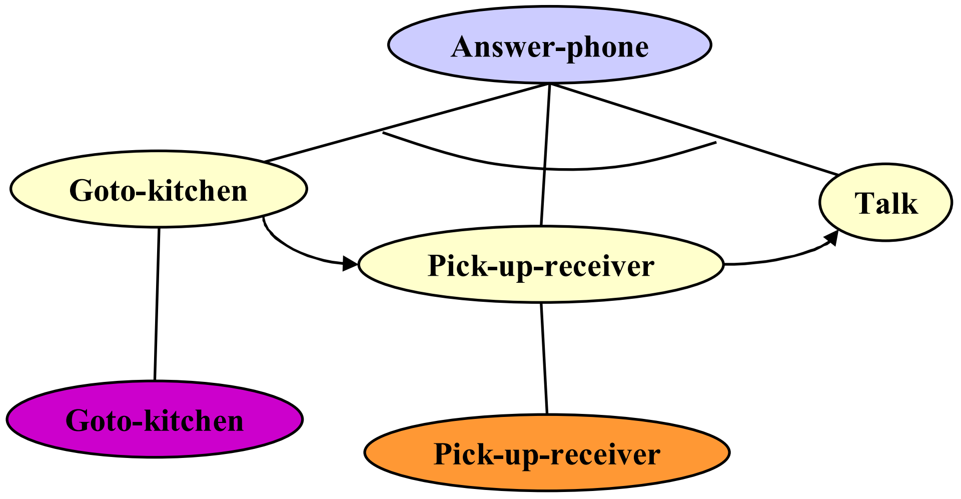
ñ in prolog

ñ not as efficient as it could be.

The traces supported by the observations



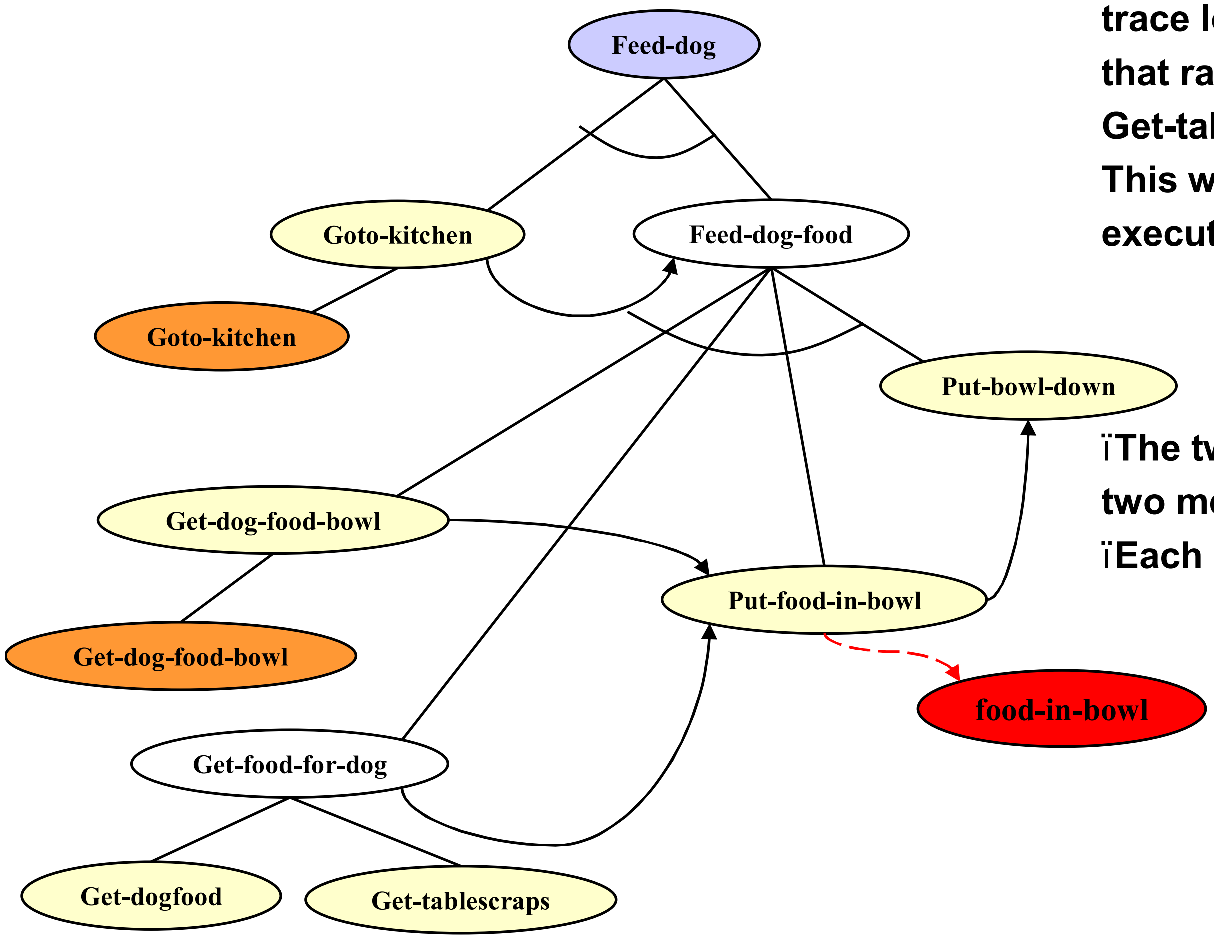
Unenabled actions



ïSuppose all we see is Pick-up-receiver

ïIf we assume that our set of plans is covering then the only way for this to be a reasonable plan is for us to hypothesize an unobserved Goto-kitchen event (shown in dark purple)

Observed Effects



ïRemember that at time t1 our execution trace looked like that at left. Now suppose that rather than seeing Get-dogfood or Get-tablescraps we observed food-in-bowl. This would provide evidence for the two execution traces shown on the next page.

ïThe two execution traces correspond to the two methods of expanding Get-food-for-dog

ïEach requires 2 unobserved actions.

Partial: Filtering continued

ï Definition: an ***unenabled action*** is an executed action that is not enabled by the agents previous actions. The observation of an unenabled action implies the execution of enabling actions that were performed unobserved.

ï Example: all you observe is a **clean** action you can infer the execution of a **recon** and a **break-in**.

ï Definition: an ***unexplained state change*** is an observed state change that does not have a causal explanation within the observed actions.

ï Example: All you observe is **deleted-logs** you can infer the execution of a **clean** action.

Partial: intuition state change

ï Definition: an ***unexplained state change*** is an observed state change that does not have a causal explanation within the observed actions.

ï Example: All you observe is **deleted-logs** you can infer the execution of a **clean** action.

ï The addition of reports of state change to intent recognition is not common in the AI literature. This kind of information isnít important when you have a complete set of observations.

ï Interestingly reports of state changes are often the most common in real domains. Hiding the effects of actions can be a great deal more difficult than hiding the actions. (think of nuclear weapons testing)

Solution: algorithm intuition

ï Insert unobserved actions that are consistent with the

ñ unobserved actions

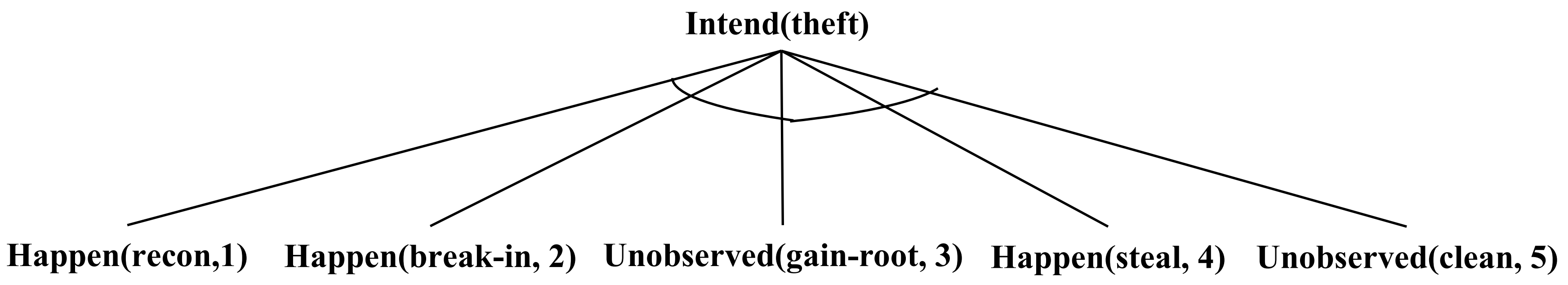
ñ unexplained state changes

ï Produces a complete execution trace

ï User the algorithm as before to compute the pending sets and explanations on the basis of the new ìcompleteî set of observations.

ï Example: observations: **recon, break-in, steal, s(deleted-logs)**

cp(theft | recon, break-in, steal, s(deleted-logs)) = 1.0



Solution: hostile agent problems

ï This algorithm woními work for hostile agents.

ï Hostile agentís like to hide their actions so that you donít know what they are doing. (Think of you dog or cat or small child (or even big child))

ï The given algorithm relies on the fact that we have a complete record of the actions performed by the agent to compute the pending sets.

ï This assumption doesnít hold anymore.

ï In the parlance of Markov Decision Processes (MDP) we have moved from a fully observable case to the partially observable case (POMDP).

Abstract

ï To be effective, both computer network security and assistive technology systems require a new and more powerful kind of plan recognition algorithm. This talk presents the Probabilistic Hostile Agent Task Tracker (PHATT) that performs plan recognition based on a model of plan execution rather than plans as formal models. As a result PHATT is able to handle partially observable domains and domains where the agent can abandon goals. This makes it uniquely suited for the computer network security and assistive technology domains.

Talk Outline

ï Background

ï Problem

ï Solution

  ñ General approach

  ñ Extension to hostile agents

ï Implementation

  ñ Implementation details

  ñ Happy graphs

  ñ Analysis

ï Conclusions

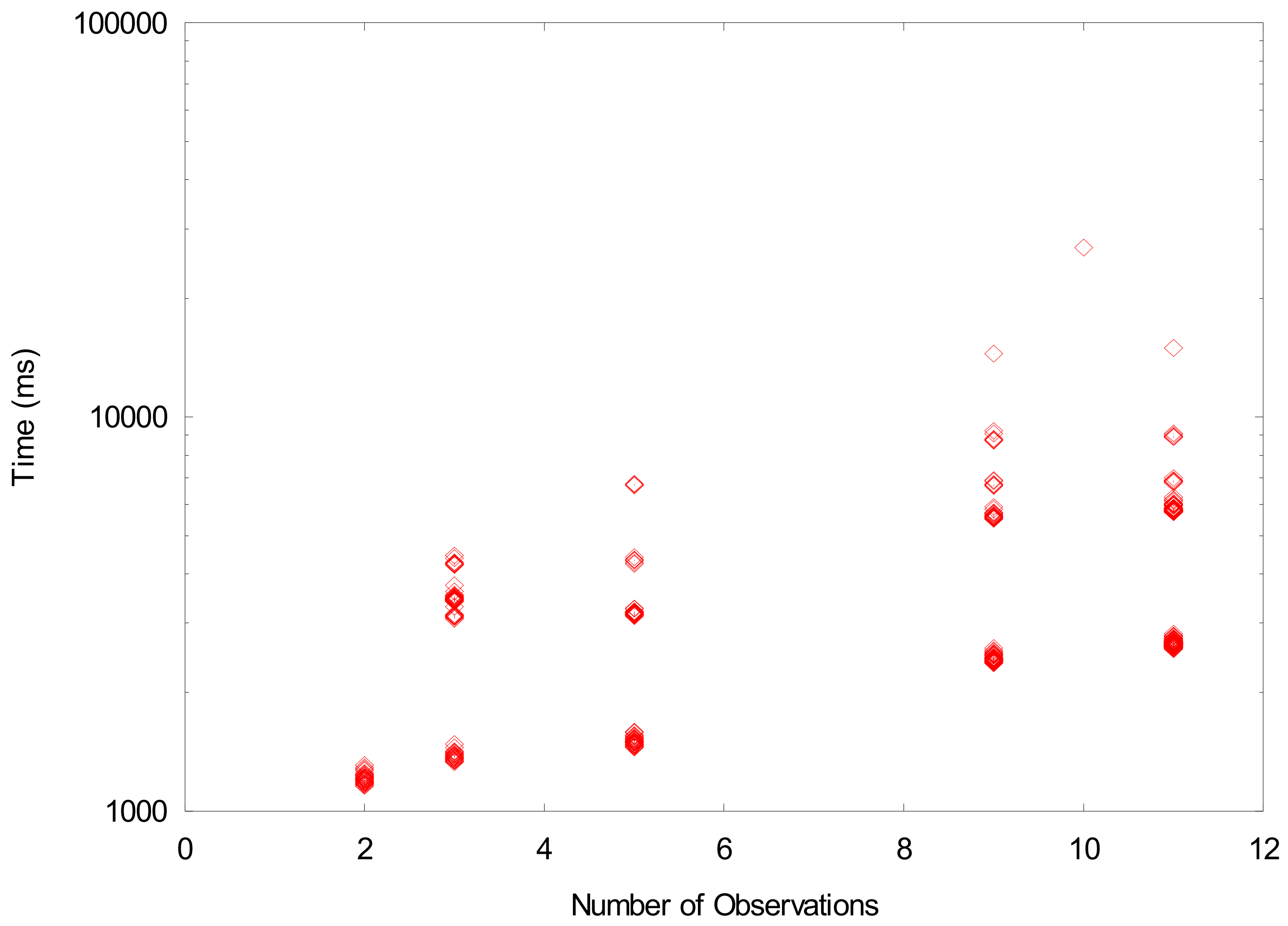
ï Joint work with Robert Goldman

Probabilistic Plan Recognition for Hostile Agents

Christopher W. Geib

May 21st, 2001

Implementation with unobserved actions


Time (ms)
Number of Observations