Lexical Ambiguity and its Impact on Plan Recognition for Intrusion Detection.

Christopher W Geib

University of Edinburgh
School of Informatics
10 Crichton St.
Edinburgh EH3 6ED, United Kingdom
cgeib@inf.ed.ac.uk

Abstract

Viewing intrusion detection as a problem of plan recognition presents unique problems. Real world security domains are highly ambiguous and this creates significant problems for plan recognition. This paper distinguishes three sources of ambiguity: *action ambiguity*, *syntactic ambiguity* and *attachment ambiguity*. Previous work in plan recognition has often conflated these different sources of ambiguity. This paper clarifies this distinction and explicitly studies the effect of syntactic ambiguity on the runtime of a particular plan recognition algorithm. It also argues for new method for controlling plan level ambiguity in probabilistic plan recognition based on the idea of plan "heads" and parsing lexicalized grammars.

Introduction

Given a plan library and a set of observations, the problem of identifying an agent's plans and goals on the basis of their observed actions is called *plan recognition* (PR), and is a well studied problem in AI. Previous work(Harp and Geib 2003; Geib and Goldman 2002) has suggested using PR both for 1) recognizing the high level plans of someone that is attacking or misusing a computer system as well as well as 2) lower level intrusion detection (to identify exploits). However, there are two major issues that make a straight forward application of most prior PR research infeasible.

1. Multiple concurrent goals: Much of the prior work in PR has assumed that an agent is engaged in a single goal at any given time.[1] However this assumption is simply not supported in computer security domains. Any given networked system will more often than not be under attack from multiple, possibly cooperating, possibly competitive sources. Each source may have a single or multiple goals. We can easily imagine a collection of "hackers" that are all using a set of scanning tools to attempt to identify machines to host their software or just for bragging rights. They may very well be scanning, attempting exploits, and other activities at the same time in an effort to achieve different instances of the same goals. Thus, not only can we not assume that a PR system's observations all form a single goal, but worse yet we can imagine multiple instances of exactly the same goal (for different attackers or from different sources.)
2. High ambiguity of individual action observations: Many, if not all, of the actions that are part of compromising a computer's security have a large number of legitimate uses as well as hostile ones. In fact, often almost all of the actions taken to compromise a computer system are individually completely legal and acceptable. It is only within the context of the collection of actions that they become problematic. This means that individual actions are not highly diagnostic of malicious intent. It is only collections of actions within specific contexts that are diagnostic. Unfortunatly, this kind of ambiguity is problematic for much of the prior research in PR (Bui, Venkatesh, and West 2002; Avrahami-Zilberbrand and Kaminka 2005; Geib 2006; Kautz 1991).

To address these problem, we have formulated PR based on Combinatory Categorial Grammars (CCGs)(Steedman 2000), a grammatical formalism developed for use in natural language parsing(NLP). This has been implemented in the ELEXIR(Geib 2009) system and uses CCGs to represent plan libraries. This formulation requires us to introduce the new idea of *plan heads*. We will show that making the correct choices about plan heads enables will allow us to control runtimes even in highly ambiguous domains like computer network security. The rest of this paper will be organized in the following way. First, we will discuss related work in plan recognition, then we will provide an overview of ELEXIR. We will continue with some synthetic studies that demonstrate ELEXIR's ability to control runtimes in highly ambiguous domains, and then discuss limitations of the algorithm.

Background

PR has seen a significant increase in interest due to the availability of large quantities of real world sensor data. Following (Carberry 1990) and (Pynadath and Wellman 2000), we are interested in viewing the problem as one of *parsing* a sequence of observations to produce plans.

Starting from real world data and viewing PR as a parsing task, we can see the problem as made up of three major

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] Some exceptions to this are the early work by (Kautz and Allen 1986) and much of the probabilistic work of (Charniak and Goldman 1990)

tasks: 1) recognizing *actions*, 2) assigning syntactic *categories* to each of the actions, and 3) combining the actions based on their categories into *plans*. Each of these tasks must address ambiguity that we will refer to as: *action ambiguity*, *syntactic ambiguity*, and *attachment ambiguity* respectively. This paper will focus on syntactic ambiguity, however, to clearly disambiguate this work, we will briefly discuss related research on the other two forms.

Action ambiguity is typically a result of sensor noise. The observation of a single real world action is usually made up of multiple, temporally extended, noisy sensor reports. These reports must be converted into a usable sequence of *observations of actions*. For example, labeling video frames showing an agent reaching for a coffee mug as part of a `grasp-mug` action. This problem is typically called *activity* or *behavior recognition*.

Starting from real world noisy data (video, sonar, passive RF, GPS data and others), successful activity recognition research has used Hidden Markov Models (HMMs) (Bui, Venkatesh, and West 2002), Conditional Random Fields (CRFs) (Liao, Fox, and Kautz 2007; Vail and Veloso 2008), and other forms of Bayesian reasoning (Avrahami-Zilberbrand and Kaminka 2005; Hoogs and Perera 2008; Liao, Fox, and Kautz 2005). In many cases, researchers have shown impressive results with significant variation in the sensor noise. However, even a perfect activity recognizer can not eliminate all ambiguity from the problem.

Consider unambiguously recognizing a `grasp-mug` action. The agent's goal is still unclear. Is the agent going to drink out of the mug? place it on the table? clean it? It is only by considering the larger plans created by sequences of observed actions that we can recognize the goals of the agent.

Previous work in PR has not distinguished between syntactic and attachment ambiguity, however in their work on natural language parsing (Sarkar, Xia, and Joshi 2000)(SXJ) clearly lays out the differences between choosing a syntactic category for a given observation (*syntactic ambiguity*) and finding the correct attachments between the categories (*attachment ambiguity*) that will result in a sentence (in our case a plan).

SXJ show that, in the case of natural language parsing, the computational cost of attachment ambiguity is far less than that of syntactic ambiguity. Their results do not directly transfer to the PR domain due to differences between action grammars and natural language grammars. However, this result suggests exploring the cost of syntactic ambiguity in PR. We leave the study of attachment ambiguity in PR as an area for future work.

We can also see this difference in domains that have effectively no sensor noise. Let us return to computer network security. In a well instrumented network, it is possible to observe all of the packets and all of the commands run on the system for any given interval. In this case there is no action level ambiguity; an action is observed in the system or it is not. Again, this doesn't mean there is no ambiguity about the plans being followed by a user or a hacker. Suppose we have a hacker that has gotten super user access to a computer. This will not tell us what his/her end goal is data theft, denial of service, or simply bragging about having broken system security.

We know of no systematic analysis of the effect of varying syntactic ambiguity on PR or specific ways to control it. In this paper, we will first review the PR algorithm (ELEXIR)(Geib 2009) based on parsing plans represented as Combinatory Categorial Grammars (CCGs) (Steedman 2000). We will then discuss how to compute plan level ambiguity. We will then discuss how to use *plan heads* in the CCG representation to control the effects of of plan level ambiguity on ELEXIR's runtime.

ELEXIR Overview

The ELEXIR system(Geib 2009) performs probabilistic PR using a weighted model counting algorithm given a set of *observations* and a *CCG* specification of the plans to be recognized in a *plan lexicon*. To perform plan hypothesis construction, ELEXIR *parses* the observations, based on the plan lexicon, into the complete and covering set of *explanations* each of which contains one or more plan structures. ELEXIR then establishes a probability distribution over the explanations to reason about the most likely goals and plans of the agent. The first step is to encode the plans in CCGs. We refer the interested read to (Geib 2009) for complete details of the formalization and algorithm behind ELEXIR. However, an understanding of how plans are represented in CCGs will be important for our discussion.

Representing Plans in CCG

Consider the simple abstract hierarchical plan drawn as a partially ordered AND-TREE shown in Figure 1. In this

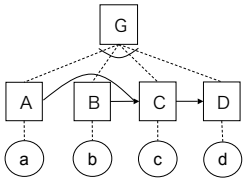

Figure 1: An abstract plan with partial order causal structure

plan, to execute action **G** the agent must perform actions **A**, **B**, **C**, and **D**. **A** and **B** must be executed before **C** but are unordered with respect to each other, and finally **D** must be performed after **C**. To represent this plan in a CCG, each observable action is associated with a set of syntactic *categories*. A set of possible categories, C, is defined recursively by:

Atomic categories : A finite set of basic action categories. $C = \{A, B, ...\}$.

Complex categories : $\forall Z \in C$, $Z \in C$ and non empty set $\{W, X, ...\} \subset C$ then $Z\backslash\{W, X, ...\} \in C$ and $Z/\{W, X, ...\} \in C$.

Complex categories represent functions that take a set of *arguments* ($\{W, X, ...\}$) and produce a *result* (Z). The direction of the slash indicates where the function looks for its arguments. Therefore, an action with category $A\backslash\{B\}$ is a function that results in performing action A when an action with

category B has already been performed. Likewise, $A/\{B\}$ is a function that results in performing A if an action with category B is executed later.

Definition 1.1 *We define a **plan lexicon** as a tuple $PL = \langle \Sigma, C, f \rangle$ where, Σ is a finite set of observable action types, C is a set of possible CCG categories, and f is a function such that $\forall \sigma \in \Sigma, f(\sigma) \rightarrow C_\sigma \subseteq C$.*

where C_σ is the set of categories an observation of type σ can be assigned. We may provide just the function that maps observable action types to categories to define a plan lexicon. For example:

$$a := A, \quad b := B, \quad c := (G/\{D\})\backslash\{A, B\}, \quad d := D. \quad (1)$$

defines one plan lexicon for our example plan.

Definition 1.2 *We define a category R as being the **root** or **root-result** of a category G if it is the leftmost atomic result category in G. For a category C we denote this root(C)*

Thus, G is the root-result of $(G/\{D\})\backslash\{A, B\}$. Further,

Definition 1.3 *we say that observable action type a is a possible **head** of a plan for C just in the case that the lexicon assigns to a at least one category whose root-result is C.*

In our lexicon c is a possible head for G.

In general, a lexicon will allow multiple categories to be associated with an observed action type. This is the source of syntactic ambiguity s the parser must choose between them.

In CCGs *combinators* (Curry 1977) are used to combine the categories of the individual observations. We will only use three combinators defined on pairs of categories:

$$\begin{array}{ll} \textit{rightward application:} & X/\alpha \cup \{Y\}, \;\; Y \;\; \Rightarrow \;\; X/\alpha \\ \textit{leftward application:} & Y, \;\; X\backslash\alpha \cup \{Y\} \;\; \Rightarrow \;\; X\backslash\alpha \\ \textit{rightward composition:} & X/\alpha \cup \{Y\}, \;\; Y/\beta \;\; \Rightarrow \;\; X/\alpha \cup \beta \end{array}$$

where X and Y are categories, and α and β are possibly empty sets of categories.

To see how a lexicon and combinators parse observations into high level plans, consider the derivation in Figure 2 that parses the sequence of observations: a, b, c. Notice, all the

$$\dfrac{\dfrac{\begin{array}{cc} a & b \\ A & B \end{array} \quad \overline{(G/\{D\})\backslash\{A,B\}}}{(G/\{D\})\backslash\{A\}}<}{G/\{D\}}<$$

Figure 2: Parsing Observations with CCGs

hierarchical structure from the original plan for achieving G is included in c's category. Thus, once c is observed and assigned its category, we can use leftward application twice to combine both the A and B categories with c's initial category to produce $G/\{D\}$.

Empirical Studies of Ambiguity in ELEXIR

Using synthetic grammars and observation streams we can test the impact on ELEXIR's runtime of varying the syntactic ambiguity of the grammar. Constructing plan lexicons and keeping the underlying plan structure fixed while varying the number of observable actions in the grammar provides a simple way to control the number of categories associated with each observable action and the associated syntactic ambiguity.

To see if syntactic ambiguity has a measurable effect even on simple problems, we use totally ordered plans with a tree height of two and a branching factor of five. Thus, each plan has twenty-five steps. For our lexicon we generated sixty-one such unambiguous plans. For each of these plans, the leftmost depth first action of each sub-tree was chosen as the head for the sub-tree. Thus the CCG categories can be thought of as encoded the plan as a series of leftmost depth first sub-trees.

With these CCG categories in hand, we generate multiple lexicons with differing levels of ambiguity by controlling the number of observable actions in the grammar. We measure the ambiguity, A, as a real value between zero and one where the number of observable actions, $|C|$, is given by:

$$|C| = (1 - A) * |l|, \quad (2)$$

where $|l|$ is the number of leaf actions in the plans represented by the lexicon. Given a set of plans encoded in CCGs we can then systematically vary the ambiguity of the resulting lexicon. We, use formula 2 to compute the number of observable actions for the lexicon, given the desired ambiguity, and then randomly assign each category to an observable action while guaranteeing that each observable action gets at least one category. Using this method we generated lexicons for the same set of underlying plans with ambiguities of 0.0, 0.1, 0.2, 0.3, 0.4, and 0.5.

Given these CCG plan lexicons we generated observations to test the system by randomly selecting a root-result categories and producing a plan instance for it based on the plan lexicon. ELEXIR is then timed computing the conditional probability of all the root results found by the algorithm given CCG plan lexicon and the sequence of observations. All of our experiments measuring the runtime for our C++ implementation of ELEXIR were conducted on a MacBook with 4Gb of main memory and 2 2.2-GHz CPUs. Figure 3 shows the average, minimum, and maximum runtimes, testing fifty plan instances each for ambiguity values of 0, 0.1, 0.2, 0.3, and 0.4 with a runtime bound of one minute.

All three statistics show significant increases for even very limited amounts of ambiguity. The reason the maximum and average statistics decrease after 0.2 is that the vast majority of the experiments did not return in under a minute. The number of experiments that did return in under a minute is given along the X-axis in the figure. Once the ambiguity exceeds 0.2 more than half of the test cases jumped from runtimes of under ten seconds to over a minute. As the ambiguity increases the number of successful sub-minute tests drops until none of the tests returned in under a minute when the ambiguity reached 0.5 (and average of two categories per observation).

Choosing Heads in Plan Lexicons

The critical choice made by during lexicon construction is which action types will be the plan heads. Different choices

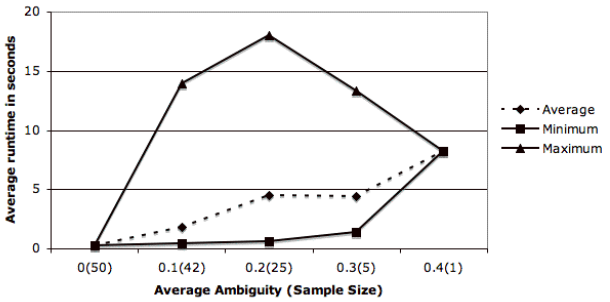

Figure 3: Ambiguity increases min, max and average runtime. Notice the significant ceiling effect above A=0.2

for heads result in different lexicons. For example, the following is an alternative lexicon for our G plan.

$$a := A, \quad b := B, \quad c := C, \quad d := (G\backslash\{A, B\})\backslash\{C\}. \quad (3)$$

We can also represent the plan for G with the following lexicon which has two possible categories for action a:

$$\begin{aligned} a := \{\ & ((G/\{D\})/\{C\})/\{B\}, \quad (4)\\ & ((G/\{D\})/\{C\})\backslash\{B\}\ \},\\ b := B,\ c := C,\ & d := D. \end{aligned}$$

There are also a number of still more complex lexicons where other choices are made for the plans heads. (Geib 2009) has pointed out that correctly choosing plan heads can have significant impact on the runtime of ELEXIR. We hypothesize that correctly choosing plan heads can help in addressing syntactic ambiguity.

It will be helpful to have a value, h, to quantify where the head occurs within a plan. We will establish a canonical order of actions for the plan, that obey the plan's ordering constraints, [2] and define the headedness for a particular plan as the rank of the plan's head action in the ordering divided by the length of the plan. Thus, grammar (3) would have a headedness value of one for the plan for G, grammar (4) would have a headedness value of 0.25 for the plan for G, and our original grammar (1) would have a headedness value of 0.75 for the plan for G.

Reducing Runtimes by Choosing Plan Heads

Our previous experiment held the headedness of plans constant at 0.0. In order to explore the impact that varying headedness might have on controlling ambiguity, we ran experiments systematically varying the headedness of the plans with five values: 0.0 (the same as our previous experiment), 0.25, 0.5, 0.75, and 1.0. Our hypothesis in this experiment is that larger headedness values will delay commitment to high level goals and thereby reduce the runtime of the algorithm.

To create these different lexicons, we used the same set of sixty-one totally ordered plan trees. These plans were then converted to a CCG lexicon by starting at the root of the plan and recursively descending the tree following the actions with the indices given by $\lceil$ (h * *plan-branching-factor*) $\rceil$ collecting siblings that are to the left and the right of the action. When a leaf is reached a CCG category is built maintaining the ordering constraints of the original plan. This process is repeated for all sub plans not covered by the initial category. This results in five grammars where the head of each plan moves from left to right over each of the actions of the plan as the value of h is increased.

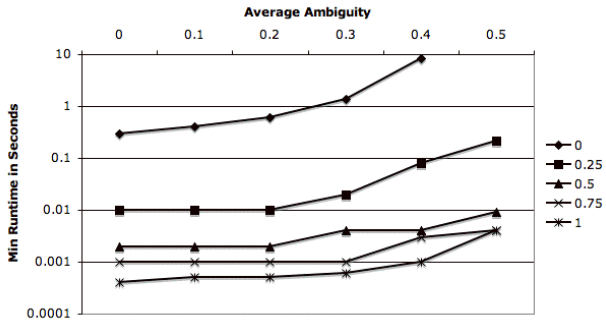

Figure 4: Increasing headedness (moving the head to the right) helps control the cost of ambiguity.

Varying both headedness and ambiguity resulted in thirty distinct grammars. For each grammar, we ran fifty tests recognizing a single plan. The minimum runtimes for each of the test conditions is graphed in Figure 4. As we have already seen, placing a one minute bound on the runtime is sufficient to prevent some of the test cases from being completed. Therefore, rather than an average we have graphed the minimum runtimes and remind the reader that these figures represent a lower bound on runtimes for these problems. However keep in mind that in the first experiment that none of the test cases with an ambiguity of 0.5 returned in under a minute.

Figure 4 provides convincing evidence for our hypothesis. We note that each of the lines for the higher headedness values starts with a faster minimum runtime (sometimes two orders of magnitude) and remain below the 0.0 line and even enables many of the test cases for ambiguity 0.5 to return in under one second.

Further evidence of the ability of headedness to aid in controlling ambiguity in plans is seen in Figure 5. This table presents the number of test cases that returned within the one minute bound. It shows that moving the head to the right in a plan increases the number of test cases with a runtime under one minute, relative to plans with the same ambiguity. Thus, even though ambiguity is being increased as we move to the right, increasing headedness in the plans is allowing ELEXIR to run fast enough to return an increasing number of results within the one minute bound.

For example, a headedness value of 0.75 enables half of the tests to return in under a minute where the ambiguity of the plan lexicon had prevented any of the test cases returning when the lexicon had a headedness value of 0.0. Thus we

[2] For the purposes of our experiments, it will not be significant that actions that are actually unordered with respect to either other can have differing values for headedness. The fact that we can systematically move through the plan's actions is more important.

h	A=0.0	A=0.1	A=0.2	A=0.3	A= 0.4	A=0.5
0.01	50	42	25	5	1	0
0.25	50	45	41	39	18	10
0.5	50	45	43	40	36	17
0.75	50	49	37	40	25	25
1.0	50	49	40	39	29	26

Figure 5: Number of test cases with runtimes under one minute.

can conclude that not only is the minimum runtime for the algorithm being kept low by moving the head of the plan away from the first actions of the plan, but the number of cases that can be brought within a reasonable runtime is also increasing.

Discussion and Limitations

These experiments show that a PR algorithm based on CCGs and headedness is viable and provides a principled way to control ambiguity. However, we have not provided an answer for how to choose plan heads during lexicon design. These decisions have to be made by considering three key factors:

1. Criticality of early recognition: In cases where early recognition is critical, choosing a head that is early in the plan is better. Earlier heads allow earlier recognition and must be weighed against the runtime. We can certainly imagine domains where the need for early recognition outweighs the runtime costs.
2. Runtime: In general, as we have shown, to minimize runtime, choosing actions that fall later in the plan as heads is better.
3. Causal structure: We can see in these experiments aligning choices of plan heads with the causal structure produces the greatest computational wins.

Thus, all three of these features must be considered by the system builder when encoding a PR domain.

While we have spent considerable time describing the way in which ELEXIRaddresses the issue of delaying commitment to root goal hypothesis, we have spent comparatively little time talking about its handling of multiple root goals. This actually falls naturally out of the algorithm that we have outlined here.

Since nothing about the algorithm or the probability model requires that an explanation only contain a single category, it is perfectly acceptable for any or all of the hypotheses to have multiple root goals. Since we are producing the complete set of such explanations, hypotheses with multiple root goals naturally fall out of the explanation generation algorithm given here. However, the probability model does have a bias against unnecessarily complex explanations by considering the root priors.

Since each root goal's prior is included within the probability of the explanation, an explanation that has multiple root goals will (depending on the specific priors) usually be less likely than an explanation that uses fewer root goals. Creating the natural bias for probabilistically "simpler" explanations of the observed actions. However, when required, the algorithm does correctly consider the less likely explanations.

Conclusions

This paper has discussed different sources of ambiguity in the plan recognition. We have provided a systematic study of syntactic ambiguity for a particular PR algorithm. We have shown that even relatively low levels of syntactic ambiguity can be crippling to the runtime of PR algorithms. Finally, we have shown that introducing the idea of heads in plans and moving the heads of plans away from the initial actions of a plan can be a powerful tool to help control the runtime of PR even in the face of significant syntactic ambiguity a critical problem for computer security domains.

Acknowledgments

The work described in this paper was conducted within the EU Cognitive Systems project PACO-PLUS (FP6-2004-IST-4-027657) funded by the European Commission.

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Bui, H. H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research* 17:451–499.

Carberry, S. 1990. Incorporating default inferences into plan recognition. 471–478. MIT Press.

Charniak, E., and Goldman, R. P. 1990. Plan recognition in stories and in life. In Henrion, M.; Schachter, R.; and Lemmer, J., eds., *Uncertainty in Artificial Intelligence 5*. 343–351.

Curry, H. 1977. *Foundations of Mathematical Logic*. Dover Publications Inc.

Geib, C. W., and Goldman, R. P. 2002. Recient advances in intrusion detection (raid) conference, 2002.

Geib, C. 2006. Plan recognition. In Kott, A., and McEneaney, W., eds., *Adversarial Reasoning*. Chapman and Hall/CRC. 77–100.

Geib, C. W. 2009. Delaying commitment in probabilistic plan recognition using combinatory categorial grammars. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1702–1707.

Harp, S. A., and Geib, C. W. 2003. Principles of skeptical systems. In *Proceedings of the AAAI 2003 Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments*.

Hoogs, A., and Perera, A. A. 2008. Video activity recognition in the real world. In *Proceedings of the Conference of the American Association of Artifical Intelligence (2008)*, 1551–1554.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI-86)*, 32–38.

Kautz, H. A. 1991. A formal theory of plan recognition and its implementation. In Allen, J. F.; Kautz, H. A.; Pelavin, R. N.; and Tenenberg, J. D., eds., *Reasoning About Plans*. Morgan Kaufmann. chapter 2.

Liao, L.; Fox, D.; and Kautz, H. A. 2005. Location-based activity recognition using relational Markov networks. 773–778.

Liao, L.; Fox, D.; and Kautz, H. 2007. Extracting places and activities from gps traces using hierarchical conditional random fields. In *International Journal of Robotics Research*, volume 26, 119 – 134.

Pynadath, D., and Wellman, M. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the 2000 Conference on Uncertainty in Artificial Intelligence*, 507–514.

Sarkar, A.; Xia, F.; and Joshi, A. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proceeding of the COLING 2000 Workshop on Efficiency in Large-Scale Parsing Systems*.

Steedman, M. 2000. *The Syntactic Process*. MIT Press.

Vail, D. L., and Veloso, M. M. 2008. Feature selection for activity recognition in multi-robot domains. In *Proceedings of the Conference of the American Association of Artifical Intelligence (2008)*, 1415–1420.