Automated Assistants for Analyzing Team Behaviors

Ranjit Nair (nair@usc.edu) and Milind Tambe (tambe@usc.edu)
Computer Science Department
University of Southern California
941 W. 37th Place, Los Angeles, CA 90089

Stacy Marsella (marsella@isi.edu) and Taylor Raines (raines@isi.edu)
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292

Abstract. Multi-agent teamwork is critical in a large number of agent applications, including training, education, virtual enterprises and collective robotics. The complex interactions of agents in a team as well as with other agents make it extremely difficult for human developers to understand and analyze agent-team behavior. It has thus become increasingly important to develop tools that can help humans analyze, evaluate, and understand team behaviors. However, the problem of automated team analysis is largely unaddressed in previous work. In this article, we identify several key constraints faced by team analysts. Most fundamentally, multiple types of models of team behavior are necessary to analyze different granularities of team events, including agent actions, interactions, and global performance. In addition, effective ways of presenting the analysis to humans is critical and the presentation techniques depend on the model being presented. Finally, analysis should be independent of underlying team architecture and implementation.

We also demonstrate an approach to addressing these constraints by building an automated team analyst called ISAAC for post-hoc, off-line agent-team analysis. ISAAC acquires multiple, heterogeneous team models via machine learning over teams' external behavior traces, where the specific learning techniques are tailored to the particular model learned. Additionally, ISAAC employs multiple presentation techniques that can aid human understanding of the analyses. ISAAC also provides feedback on team improvement in two novel ways: (i) It supports principled "what-if" reasoning about possible agent improvements; (ii) It allows the user to compare different teams based on their patterns of interactions. This paper presents ISAAC's general conceptual framework, motivating its design, as well as its concrete application in two domains: (i) RoboCup Soccer; (ii) software agent teams participating in a simulated evacuation scenario. In the RoboCup domain, ISAAC was used prior to and during the RoboCup'99 tournament, and was awarded the RoboCup Scientific Challenge Award. In the evacuation domain, ISAAC was used to analyze patterns of message exchanges among software agents, illustrating the generality of ISAAC's techniques. We present detailed algorithms and experimental results from ISAAC's application.[1]

[1] This article significantly extends our previous conference paper [25] and our extended abstract [26].

1. Introduction

Teamwork has been a growing area of agent research and development in recent years, seen in a large number of multi-agent applications, including autonomous multi-robotic space missions [10], virtual environments for training [35] and education [17], distributed resource allocation [19] and software agents on the Internet [34]. With the growing importance of teamwork, there is now a critical need for tools to help humans analyze, evaluate, and understand team behaviors. Indeed, in multi-agent domains with tens or even hundreds of agents in teams, agent interactions are often highly complex and dynamic, making it difficult for human developers to analyze agent-team behaviors. The problem is further exacerbated in environments where agents are developed by different developers, where even the intended interactions are unpredictable.

Unfortunately, the problem of analyzing team behavior to aid human developers in understanding and improving team performance has been largely unaddressed. Previous work in agent teamwork has largely focused on guiding autonomous agents in their teamwork [13, 36], but not on its analysis for humans. Agent explanation systems, such as Debrief [15], allow individual agents to explain their actions based on internal state, but do not have the means for a team analysis. Recent work on multi-agent visualization systems, such as [21], has been motivated by multi-agent understandability concerns (similar to ours), but it still leaves analysis of agent actions and interactions to humans.

This article focuses on tools that assist humans to analyze, understand and improve multi-agent team behaviors by:

1. Locating key aspects of team behaviors that are critical in team success or failures;

2. Diagnosing such team behaviors, particularly, problematic behaviors;

3. Suggesting alternative courses of action; and

4. Presenting the relevant information to the user comprehensibly.

Based on our initial efforts at building a team analyst, we arrived at several important design constraints that should be addressed in building such team analysts. First, unlike systems that focus on explaining individual agent behaviors [15, 29], team analysts need to have multiple perspectives at multiple levels of granularity. It is sometimes beneficial to analyze the critical actions of single individuals because failures in these critical actions may have been extremely costly and hence correcting these failures can significantly improve performance. In other circumstances it is the collaborative agent interaction within a small sub-team that is key in team success or failure. Here, analysis may determine which sequences of agent interactions within the sub-team contribute to the team's success or failure. And in some other circumstances, an analysis of the global behavior trends of the entire team is important.

The second key constraint is that as assistants to human users and developers, team analysts must not only be experts in analysis, they must also be experts in conveying this information to humans. The constraint of multiple models has strong implications for the type of presentation as well. Analysis of an agent action can show the action and highlight features of that action that played a prominent role in its success or failure, but a similar presentation would be incongruous for a global analysis, since no single action would suffice. Global analysis requires a more comprehensive explanation that ties together seemingly unconnected aspects and trends of team behavior.

The third constraint is that team analysts should ideally be independent of the underlying team architecture and implementation, to ensure generality of the analysis across teams and even across domains. In particular, by exploiting external behavior traces of the team, team analysts can understand team behaviors without necessarily requiring information on team internals. This bottom-up, data-intensive approach is especially desirable in complex domains where a causal model is weak or unknown. Furthermore, acquiring such causal information can be a significant bottleneck, if agents and teams are developed by different developers. Finally, such a bottom-up approach can reveal unexpected patterns of interaction that are often of interest to developers. This constraint does not imply ignoring "top-down" information where easily available, but rather it stresses the criticality of bottom-up analysis.

These constraints on team analysts have shaped the analyst we have developed, called ISAAC. To address the first constraint, ISAAC relies on multiple models of team behavior, each covering a different level of granularity of team behavior. More specifically, ISAAC relies on three different models that analyze events at three separate levels of granularity: an individual agent action, agent interactions, and overall team behavior. To address the second constraint, multiple modes of presentations are used, each suited to the model being presented. For instance, ISAAC uses a multimedia viewer to highlight individual agent actions, but uses a natural language summary to explain the overall team performance. The content for the summary is determined by ISAAC's automated analysis of key factors determining team performance. To address the third constraint, ISAAC's three models are automatically acquired via machine learning techniques like inductive decision tree learning and learning of probabilistic finite automata (PFA) based on external data traces of team behaviors. With multiple models, the method of acquisition can be tailored to the model being acquired.

An additional novelty in ISAAC is the two techniques it uses to suggest improvements to a team. First, ISAAC presents alternative courses of actions using a technique called 'perturbation analysis'. A key feature of perturbation analysis is that it suggests improvements using actions within the agent's skill set, since this analysis mines data from actions that the team has already performed. Second, ISAAC also aids in comparing patterns of behaviors of different teams. Such a comparison also provides suggestions for team improvement.

ISAAC was first extensively applied in the domain of RoboCup soccer simulation [17, 22] (See Figure 1). RoboCup is a dynamic, multi-agent environment developed to explore multi-agent research issues, with agent teams participating in national and international annual competitions. There are 11 agents in each team that act without any centralized control and act in a complex, dynamic, noisy environment managed by a soccer server [22]. Agent-team analysis is posed as a fundamental challenge in RoboCup since team developers wish to understand the strengths and weaknesses of teams and understand how to improve such teams. There are at least 50 such development groups around the world.

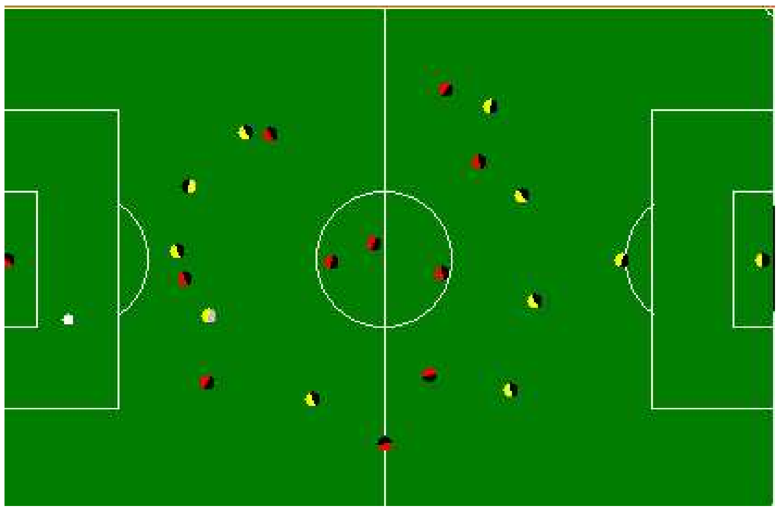


Figure 1. 2-D snapshot of a RoboCup soccer game.

ISAAC has attacked the team analysis challenge in RoboCup: it has been applied to all of the teams from several RoboCup tournaments in a fully automated fashion. This analysis has revealed many interesting results including surprising weaknesses of the leading teams in both the RoboCup-97 and RoboCup-98 tournaments. ISAAC also provided team analysis and natural language summaries at RoboCup-99. At RoboCup-99, ISAAC was also awarded the “Scientific Challenge Award” for outstanding research at a RoboCup tournament. ISAAC is available on the web at *http://coach.isi.edu* and is used remotely by teams preparing for these competitions.

Although ISAAC was initially applied in RoboCup, ISAAC’s techniques are intended to apply in other team domains such as agent-teams in foraging and exploration [4], distributed resource allocation [19] and battlefield simulations [35]. For instance, in this article, we demonstrate the generality of ISAAC’s techniques by applying it to the analysis of communication actions of a team of software agents [38]. The team is engaged in the task of the simulated evacuation of civilians trapped in a hostile territory. Here ISAAC can compare different scenarios based on the sets of sequences of message exchanged between agents. ISAAC could potentially be applied to several other domains, which we discuss in Section 7.

Finally, given the state of the art of team analysis tools, it is important at this juncture to actually build working team analysts, to begin to understand the key principles underlying such systems. This is what we have embarked upon with this investigation. ISAAC is a concrete implemented team analyst that was not only used by its developers but by an entire user community (in this case, the RoboCup simulation community), which provided valuable feedback, and thus guided its development. This investigation has already revealed important principles for building future team analysts, such as multiple granularities of analysis, multiple techniques of presentation, techniques to suggest improvements to agent teams, as discussed in the rest of this paper. These principles appear to be applicable to other domains, as discussed in section 7. We expect that principles revealed via such investigations will eventually serve as a foundation to build general theories for analysis of agent teams and multi-agent systems in general.

The rest of this paper is organized as follows: Section 2 presents an overview of ISAAC. Section 3-5 describes the three models used in ISAAC for analysis. Section 6 presents the evaluation and results. Section 7 demonstrates the generality of ISAAC’s techniques. Section 8 describes related work. Section 9 presents a summary of the work and identifies some directions for future work.

2. Designing a team analyst : An Overview

In order to address the three constraints on the design of team analysts described in the previous section, we use a two-staged approach to the team analysis problem. The first stage is acquiring models that will compactly describe team behavior, providing a basis for analyzing the behavior of the team. As mentioned earlier, this involves using multiple models at different levels of granularity to capture various aspects of team performance. The second stage is to make efficient use of these models in analyzing the team and presenting this analysis to the user. Later sections delve into more specifics of these models. An overview of the entire process is shown in Figure 2.

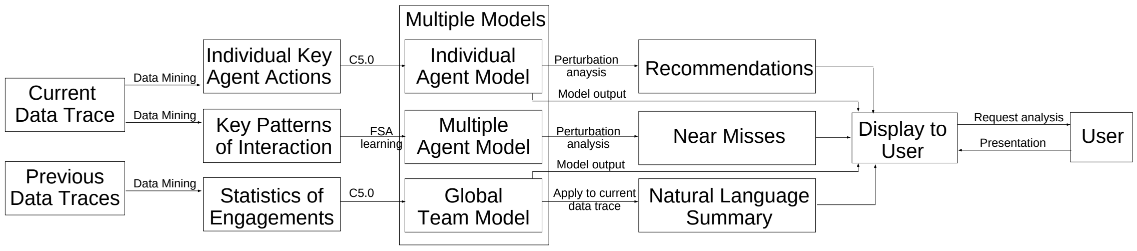


Figure 2. Flow Chart for ISAAC Model Generation and Analysis.

For acquiring team models, ISAAC relies on a bottom-up approach using large data traces. These data traces would be extremely difficult for a human to read and understand much less analyze. There are significant advantages to relying on a bottom-up approach:

1. The analysis of team behavior will be independent of understanding the source code of the team, thus increasing ISAAC's generality by ensuring that the techniques for obtaining the team models are not dependent on the specific team being analyzed.

2. Requiring users to enter a causal model that explains agents' individual and group behavior would have been extremely tedious and difficult; indeed, given complex agent interactions in dynamic environments, the users themselves may not have such a causal model available.

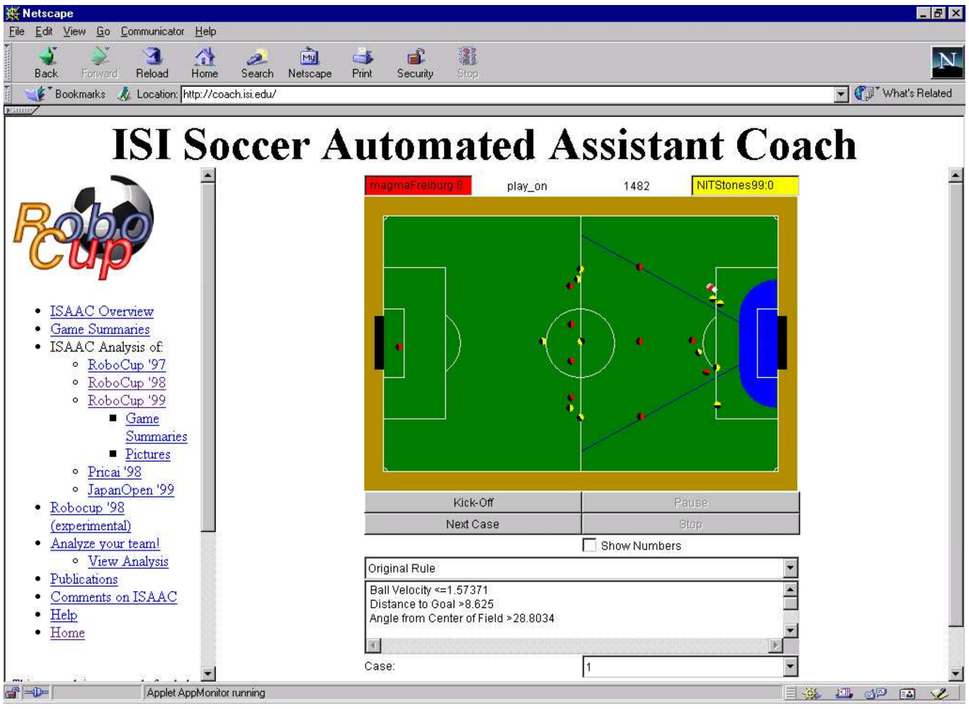
3. We are particularly interested in discovering novel and surprising regularities in the data where the developers of the various teams operating in the domain might be unaware of.

While this bottom-up approach thus offers several advantages, its one potential drawback is the absence of a causal model. This drawback, in some ways, is also a strong point, particularly in domains where causal models may be impoverished or missing. ISAAC can use a data intensive approach even when it is unclear what factors cause success. Thus, by design, ISAAC's analysis and advice relies on strong correlational patterns seen in the team's data traces. Although correlational, these patterns may nonetheless enable a user to draw useful conclusions about different team behaviors. Indeed, the patterns may uncover interactions with other agents, teams and the world that come as a surprise to the designers. This is especially relevant in complex domains with numerous interacting agents.

Input to all models comes in the form of data traces of agent behaviors. In the current implementation of ISAAC, these traces have been uploaded from users around the world through the Internet. As shown in Figure 2, acquiring the models involves a mix of data mining and inductive learning but is specific to the granularity of analysis being modeled. It is important to note that this learning is mainly in service of explaining observed team behavior to the users rather than predicting unseen data. Thus, for analysis of an individual agent action (*individual agent model*) we use rules obtained from decision trees to explain agent behavior and also to suggest improvements. For analysis of agent interactions (*multiple agent model*), critical patterns are learned and their frequency of occurrence is obtained. To develop rules of team successes or failures (*global team model*), game level statistics are mined from all available previous games and again inductive learning is used to determine factors that correlate with success and failure.

In order to maximize human comprehension, the presentation of the models needs to be catered to the granularity of the analysis. ISAAC uses different presentation techniques for the different levels of analysis. For the individual agent model, the features that compose a rule provide implicit advice for improving the team (i.e., some specific features were seen to correlate with failure). To further elucidate, a multimedia viewer is used to show cases matching the rule, allowing the user to better understand the situation and to validate the rules. Figure 3 shows ISAAC's multimedia viewer which displays the soccer field and plays from the game. The viewer can highlight key features specific to ISAAC's analysis. In particular, the viewer highlights features emphasized in a rule. A *perturbation analysis* is then performed to recommend changes to the team by changing the rule condition by condition and mining cases of success and failure for this perturbed rule. These cases from actual games are also displayed in the multimedia viewer, enabling the user to verify or refute the analysis.

For the multiple agent model, a finite automaton is learned in order to characterize the patterns of agent interactions. A perturbation analysis is also performed here to find patterns that are similar to successful



Ball Velocity <=1.57371
Distance to Goal >8.625
Angle from Center of Field >28.8034

Case: 1

Figure 3. ISAAC's multimedia viewer: On the right hand side of the field, ISAAC has highlighted the distance to goal and the angle from the center of field.

patterns but were unsuccessful. Both successful patterns and these "near misses" are displayed to the user as implicit advice. In addition, two teams can be compared based on the probability distributions of their critical patterns of agent interactions obtained from the learned finite automaton. The differences in these distributions may explain the differences in the team's performance. For example, if the comparison between a team that performs poorly and a team that performs well shows that the distributions of their patterns are significantly different, then this may suggest to the user techniques to improve the poorly performing team.

Finally, the global team model also uses decision tree learning, but it requires a different method of presentation. For the analysis of overall team performance, the current engagement is matched against previous rules, and if there are matches, ISAAC presents the conditions of the matched rule as the likely determining factors in the result of the engagement. A natural language summary of the engagement is generated using this rule for content selection and sentence planning. ISAAC makes use of the multimedia display here as well, linking text in the summary to corresponding selected highlights.

Of the three models that we have proposed, the multiple agent model appears to be the most well suited for the comparison of different teams of agents. In contrast, the individual agent model considers only a critical action by a single agent. Thus, it concentrates on events at too fine a granularity to consider it to appropriately characterize a team. The global team model finds the features that contribute to the success of teams in general, and is not team specific. The multiple agent model, on the other hand, considers sequences of collaborative and non-collaborative actions of different agents in a particular team that result in the team achieving a goal. These patterns of actions characterize a team's behavior better than a single action performed by individual agents and hence the multiple agent model was selected as a means of comparing different teams.

We have empirically settled on the three levels of analysis in ISAAC described above. They appear to cover the spectrum of activities that are useful to analyze a team, starting with individual actions, to sub-team interactions, to global team behaviors; and indeed, these levels appear useful in the RoboCup domain. However, it is not necessary that all three levels be used in all applications of ISAAC. Indeed, as discussed in Section 7, we have so far applied one of the three ISAAC models (the multiple agent model) in analyzing communications within a team of software agents. A second model (the global team model) is also potentially applicable.

3. Individual Agent Model

This section examines the first of ISAAC's three models, which focuses on key actions taken by individual agents and is specific to each team. In this and the following two sections, we first provide a conceptual overview of the model being analyzed and then discuss its instantiation in RoboCup.

3.1. Conceptual Overview of the Individual Agent Model

The eventual success or failure of a team is often determined by intermediate successes or failures during critical events – events that may be widely separated in time and/or loosely connected to each other. For instance, in a battlefield simulation, there may be many distinct attacks on enemy units, which are critical to the team's eventual success or failure, that are embedded in a larger history of maneuvering. The individual agent model focuses on such intermediate successes or failure, analyzing actions of individual agents that play a critical role in such intermediate results. The goal is to learn a model of critical individual actions that is useful for developers and other analysts. Thus, this model should be both compact and easily understandable for the human users. Furthermore, since we want to support improvement to teams, it is desirable to be able to engage in "what-if" analysis to determine how small changes in the team could affect performance. Thus, the model should lend itself easily to such modifications. This could be very useful to a developer who is looking for hints on how to improve the performance of his/her team.

To construct such a model, symbolic approaches for machine learning seemed more promising than non-symbolic machine learning approaches like neural networks since the latter could result in a model that is difficult for a human to comprehend. Amongst symbolic approaches, decision trees have often been used for agents' learning about own decisions [32] (or for modeling others [8]) in the presence of large amounts of data. However, unlike these approaches that use decision trees as a model of prediction of agent behavior in unseen cases, we use decision trees as a model to explain observed agent behavior. The key insight here is that we wish to extract key features that discriminate between success and failure of critical actions — these features are likely to be of most interest to agent developers. Decision trees enable us to extract such discriminatory features. We also use rules extracted from decision trees in "what-if" analysis, as discussed later in this section.

The algorithm for the individual agent model is described in Figure 4. The user submits logs of the team's behavior along with what he/she considers to be critical events and his/her choice of features. The MineLogs() function of the individual agent model then mines the logs for positive and negative examples of the critical events. The individual agent model then uses C5.0 to come up with rules that explain these examples. When a user selects a particular rule, all those cases of examples that satisfy this rule are obtained and user can choose to display any of these using the multimedia viewer.

This technique appears to result in compact and easily understandable rules given a suitable choice of features. These rules and the cases they represent can be displayed to the user as implicit advice on how individual agents operate in critical situations. Currently, C5.0 is used to form the rules of success and failure. (Note that C5.0 is used instead of C4.5, since its ability to assign different costs of misclassifications enables a user to tailor the analysis, as explained in the next subsection.)

At present, we assume that the identification of the intermediate success and failure points is part of the domain specific knowledge available to the individual agent analysis model. The other domain

```
IndividualAgentModel(team, logs, choiceOfCriticalEvents, features){
   team: team for which the Model is being constructed;
   logs: Logs of team's behavior;
   examples <- MineLogs (logs, choiceOfCriticalEvents, features);
   rules <- ApplyC5.0 (examples);
   original_rule <- null; conditions <- {}; cases <- {};
   do{
   choice <- GetUserChoice ();
   switch (choice) {
      case selectRule:
         original_rule <- SelectRule (choice, rules);
         cases <- GetCasesFromExamples (original_rule, examples);
         conditions <- GetConditions (original_rule);
         break;
      case selectCondition:
         condition <- SelectCondition (choice, conditions);
         perturbed_rule <- PerturbRule (condition_to_perturb, original_rule);
         cases <-GetCasesFromExamples (perturbed_rule, logs);
         break;
      case selectCase:
         selected_case <- SelectCase (choice, cases);
         DisplayCase (multiMediaViewer, selected_case);
         break;
      }
   } while (choice != exitModel);
}

MineLogs (logs, choiceOfCriticalEvents, features){
   examples <- extract positive and negative examples from logs based on
               choiceOfCriticalEvents and features;
   return examples;
}

GetCasesFromExamples (rule, examples){
   examples: positive and negative examples of team's critical event behavior;
   cases <- obtain snippets of logs corresponding to each example in examples;
   return cases;
}

PerturbRule (condn_to_perturb, original_rule){
   perturbed_rule <- rule obtained by modifying the original_rule by inverting
                     the test of condn_to_perturb;
   return perturbed_rule;
}
```

Figure 4. General Algorithm for Individual Agent Model.

specific knowledge that is provided to the individual agent model is what features to use. The selection of features must be such that they have the breadth to cover all information necessary for the analysis, but should also be as independent of each other if possible. In addition, it is important that the feature be an attribute that can be easily understood by the team developer. In the future, a semi-automated attribute selection may be used [7]. These features, along with the decision on what constitutes intermediate success or failure, are the only background information or bias given to the individual agent analysis technique.

In addition to the implicit advice mentioned above, we developed an automated approach to "what-if" analysis based on the perturbation of the learned rules, to provide explicit advice for team improvement. After ISAAC has produced rules determining which circumstances govern success and failure classifications, it uses a perturbation analysis to determine which changes would produce the most benefit. Since each learned rule consists of a number of conditions, several possible approaches to perturbations could be considered. One approach would be to add or drop conditions in the rule. Another approach would be to incrementally modify the tests in each condition, for instance, by increasing or decreasing the numeric value being tested in a condition. A third approach is to invert the attribute test in the conditions, i.e., an attribute test T_i in the original rule would be modified to $\neg T_i$ to create a perturbed rule. We use this third approach, since it ensures that the set of cases satisfying the perturbed rule will have no overlap with the set of cases satisfying the original rule. Thus, if we invert a condition in a rule that classifies failure examples, the resulting rule will no longer satisfy any of these failure examples — it could thus form the basis of good advice in improving the team. For instance, it is possible that the resulting rule covers a significant number of success cases.

Our approach to rule perturbations could lead to a large number of perturbations however. In particular, a rule R can be expressed as $T_1 \wedge T_2 \wedge \ldots \wedge T_N$, a conjunction of tests T_i where N is the number of conditions in rule R. There are $\sum_{k=1}^{N} \binom{N}{k}$ ($= 2^N - 1$) different non-empty subsets of tests that can be inverted. To obtain a perturbed rule, all the tests in any one such subset should be inverted and hence $2^N - 1$ perturbed rules can be obtained. However, we restrict the set of possible perturbations to include only rules that result from reversing exactly one condition. While changing more than one condition at a time is not necessarily undesirable, the space of perturbation that can be done is now exponential. In addition, we are more interested in showing the effect of making a *small* change in the present behavior, for example, how a change to just a single condition of a failure rule results in transforming it into a success rule. This will tell us how an improvement in performance can be made with minimal effort. Thus, a rule with N conditions will result in exactly N perturbed rules. The successes and failures governed by the perturbations of a rule are mined from the data and examined to determine which condition has the most effect in changing the outcome of the original rule, *turning a failure into a success*. This perturbed condition provides explicit advice for team improvement. Since these cases are mined from the original data using this perturbation technique, the recommended changes must already be within the agent's skill set. Furthermore, perturbations are guaranteed to provide a non-empty set of examples. This is because, if there were a condition that when reversed led to an empty set of examples, then C5.0 would not have included that condition in the rule. In particular, that condition would not discriminate between success and failure cases and thus would not be part of the rule.

As shown in Figure 4, in order to do "what-if" analysis the user can choose any one of the conditions of a rule to invert and thus obtains a perturbed rule. ISAAC then mines through all the positive and negative examples of the team's critical event behavior and returns those cases of examples that satisfy the conditions of this perturbed rule. The user can use the multi-media viewer to display any of these cases. An example of applying perturbation analysis is presented in section 3.3.

3.2. Application of Individual Agent Model to RoboCup

The first step in applying the approach to RoboCup is identifying the domain specific information that would be used by ISAAC as bias in its analysis (in this article we assume that the reader is somewhat familiar with the game of soccer). In particular, in the RoboCup domain, success means outscoring the opponent. Shots on goal are therefore key points of intermediate success or failure as these are situations that can directly affect the outcome of the game. Thus, the focus of ISAAC's individual agent analysis in RoboCup is shots on a team's goal as well as shots by the team on an opponent's goal.

Having defined shots on goal as key events, we need to determine which domain dependent features might be useful in classifying the success or failure of a shot on goal. After an initial set of experiments with a relatively large feature set, ISAAC currently relies on a set of only 8 features such as velocity of the ball, distance to the goal, etc. to characterize successes and failures in order to improve understandability. Besides criteria like coverage and independence, an important criterion for selection of the features to use was the comprehensibility of the feature to human users. A complete list of features is shown in Appendix A.

Having determined which features to use in the analysis and the key events (the cases) to examine, the task is transformed to mining the raw data and feeding it to the C5.0 decision tree learning algorithm. From the resulting decision tree, C5.0 forms rules representing distinct paths in the tree from the root of the tree to a leaf classification of success (goal-score) or failure (goal not scored). Each rule describes a class of similar successes or failures. The resulting rules were found to be compact and few in number. An evaluation of the rules generated and their compactness is presented in section 6.1.

Figure 5 shows an example success rule, describing a rule where shots taken on the Windmill Wanderer team will fail to score (Successful Defense). This rule states that when the closest defender is sufficiently far away ($> 13.6m$) and sufficiently close to the shooters path to the center of the goal ($< 8.98^o$), and the shooter is towards the edges of the field ($> 40.77^o$), Windmill Wanderer will successfully defend against this shot. When viewed using ISAAC, the user can see that the defender is far enough away to have sufficient time to adjust and intercept the ball in most of these cases. Thus the user is able to validate ISAAC's analysis. This rule provides implicit advice to this team to keep a defender sufficiently distant from the ball, or to try to keep the ball out of the center of the field.

Distance of Closest Defender $>$ 13.6 m
Angle of Closest Defender wrt Goal $\leq$ 8.981711
Angle from Center of Field $>$ 40.77474
$\rightarrow$ class Successful Defense

Figure 5. Sample Rule from shots on Windmill Wanderer team of RoboCup-98.

The application of a decision tree induction algorithm to this analysis problem must address some special concerns. The goal-shot data has many more failure cases (failed goal shots) than success cases (goals scored). However, analyzing such data using a traditional decision tree induction algorithm such as C4.5 gives equal weight to the cost of misclassifying successes and failures. This usually yields more misclassified success cases than misclassified failure cases. For example, in our analysis of shots by the Andhill team from the RoboCup'97 tournament, our original analysis misclassified 3 of 306 failure cases (less than 1%), but misclassified 18 of 68 success cases (26%). Since a much larger portion of the success cases is incorrectly classified, this produces overly specific rules that govern success cases. To compensate for this lopsided data set, the ability of C5.0 to weight the cost of misclassification is used.

Specifically, the cost of misclassifying a success case is set to be greater than the cost of misclassifying a failure case [40]. ISAAC uses a 3 to 1 ratio by default, but this is adjustable.

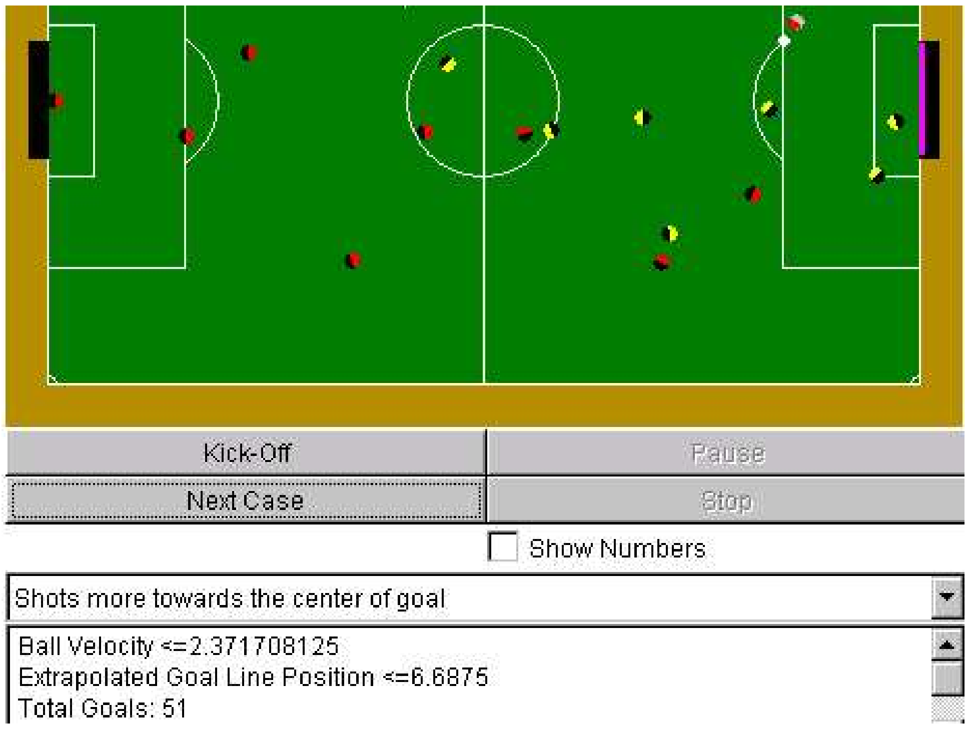
More generally, differential weighting of misclassification cost provides a mechanism for tailoring the level of aggressiveness or defensiveness of ISAAC's analysis. Consider shots on goal against a team. If a very high cost is assigned to misclassifying a successful shot on goal, the rules produced will likely cover all successful shots, and quite a few misclassified failure cases. In this case, the rule conditions are implicitly advising to make the team very defensive. On the other hand, if a low cost is assigned, the rules may not cover all of the successful cases. Therefore, ISAAC would only give "advice" relevant to stopping the majority of shots on goal. This may not be appropriate if we consider any goal to be a serious failure. Therefore, we allow the user to adjust the weight on success case misclassifications.

3.3. Perturbation Analysis in RoboCup Soccer

As explained in section 3.1, we consider perturbations that result from inverting a single condition in the rule. Analysis of the perturbed rules and the cases that satisfy these rules, may be useful in identifying reasons for unsuccessful behavior and also to improve successful behavior further.

Perturbations of a failure rule enable users to see what minimal modifications could be made to agent behaviors to convert the failures into success. Mining instances of perturbed failure rules, the developer determines steps that could be taken to move the agent from failure to successful behavior.For example, one of ISAAC's learned rules states that when taking shots on goal, the Andhill97 team often fails to score when (i) ball velocity is less than 2.37 meters per time step and (ii) the shot is aimed at greater than 6.7 meters from the center of goal (which is barely inside the goal). ISAAC reveals that shots governed by this rule fail to score 66 times without a successful attempt.

Now consider the perturbations of this rule. In cases where the rule is perturbed such that ball velocity is greater than 2.37 meters per time step and the shot aim is still greater than 6.7 meters, Andhill scores twice and fails to score 7 times. In another perturbation, where ball velocity is again less than 2.37 meters per time step but now shot aim is equal to or less than 6.7 meters (i.e. shots more towards the center of the goal), Andhill is now scoring 51 times and failing to score 96 times (See Figure 6). These perturbations suggest that improving Andhill97's shot aiming capabilities can significantly improve performance, while trying to improve agents' shot velocity may not result in a drastic performance increase.



Shots more towards the center of goal

Ball Velocity <=2.371708125
Extrapolated Goal Line Position <=6.6875
Total Goals: 51

Figure 6. Perturbation analysis showing the Andhill '97 team scoring using a perturbed shooting behavior. The rule was perturbed to cover shots taken more towards the center of the goal.

Perturbations of success rules are also useful. There are two reasons for this. First, it allows ranking of conditions contributing to success. In particular, some changes to the rule will take a team further from success than another. For example, a team may succeed 95% of the time when all the conditions are met. The percentage of success may drop to 50% if the first condition is changed and down to 5% if the second condition is changed. In this case, the developer may decide that even if the first condition is not met, this is still the correct course of action while doing so if the second condition is not met is a bad decision. The second reason why perturbing success rules is useful is that by allowing the user to see how the team succeeds or fails when conditions in the success rule are modified, more insight can be drawn as to why these conditions are important. At this juncture if it important for the human user to determine if the factors that ISAAC comes up with are truly the reasons the team is succeeding or failing.

4. Multiple Agent Model

While the previous section focuses on critical actions of individual agents, it is also important to analyze the sequence of actions (possibly including actions of multiple agents) that lead up to these critical actions. Such analysis may for instance reveal that some sequences are more prone to success than others. Therefore, we also need to model the patterns of collaborative or non-collaborative interactions of several agents of the team that lead up to intermediate successes (or failures). Section 4.1 provides a conceptual overview of our approach to learning these patterns, while Section 4.2 discusses its application in RoboCup.

4.1. Conceptual Overview of the Agent Interaction Model

The multiple agent model is a characterization of key aspects of a team's interactions where we once again use intermediate goals to focus our analysis of key interactions. The characterization is in the form of a set of patterns, each with a probability of occurrence, where the patterns consist of sequences of abstracted actions of different agents that result in intermediate successes (or failures). The probability distributions over patterns can be used in two ways. First, the probability distributions of these patterns enable developers to check for surprising, unanticipated patterns. For instance, patterns they had specifically engineered may have very low likelihood of occurrence and conversely, unplanned for patterns may have an unexpected high rate of occurrence. Second, ISAAC can compare the probability distributions of these patterns across two different teams to determine whether they are similar. This comparison reveals if the two teams are significantly dissimilar in their approach to achieving their goals. If the comparison of a less successful team with a highly successful team shows them to be dissimilar, then this suggests that the less successful team's performance could potentially improve by better emulating the more successful team's strategy for agent interaction.

There are some important concerns that must be addressed in acquiring this model. First, the patterns need to support human comprehension as well as being able to differentiate teams. In particular, some differences in agent interactions may be insignificant with respect to goal achievement. Such differences should not be allowed to complicate the representation of the patterns or obscure the comparison of teams. For example, actions that immediately precede (sub)goals are often far more significant than actions far removed. Further, as we look farther away from the goal, we might expect increasing variance in agent actions. Thus we may want at times to use a learning approach that considers only a limited window of actions prior to (sub)goals. We refer to the length of this window as *window size*.

Second, some form of generalization will typically be required that provides a compact representation which suppresses irrelevant details, again in support of comprehension and comparative analysis. For example, in soccer, if a teammate and scorer pass the ball back and forth to each other repeatedly, the fact that it may happen 2 times or 3 times is not as significant as the fact that it happened repeatedly. More generally, it is often important that the pattern or ordering of causal interactions between actions not be violated whereas the number of repetitions of the pattern is often less significant. Thus, we would like the learned patterns to generalize over repetitions, but not over other (potentially causal) structure. We refer to such generalization as *bounded structural generalization*.

A final concern that must be addressed in learning sequences of interactions is to maintain some information about the frequency distribution of different patterns of agent interactions, without over-generalization. In particular, it is important to ensure that generalization does not surprise the developers by creating non-existent patterns of interactions (modulo bounded structural generalization). We refer to this constraint as *frequency distribution accuracy constraint*. Among other implications, one key implication of this discussion is that we need an approach to learning that can be tailored. Tailoring allows us to err on the side of conservative learning and then tailor the learning when the patterns learned are too large or numerous for comprehension or effective contrast.

Thus, to learn the sequences comprising the agent interaction model, ISAAC must be able to learn the probability distribution of the patterns of team interactions based on observation of the team's actions while allowing tailoring of window size and structural generalization. One proposed approach to learning such patterns is to use decision trees as in section 3.1. However, in our preliminary experiments this approach failed for two reason. (i) It was difficult to capture temporal sequencing information in a flexible way; (ii) the goal is to find classes (patterns) and their frequencies as opposed to identifying features that discriminate between classes. A natural way to learn these patterns and their distributions is by learning a probabilistic finite automaton (PFA). A PFA is a good mechanism for representing a probability distribution since it can be used to determine the probability of the occurrence of a sequence of symbols. Several algorithms for learning deterministic probabilistic finite automata (DPFA) and non-deterministic probabilistic finite automata (NPFA) have been proposed in the literature, e.g., the APFA algorithm [28] and Crutchfield algorithm [9] for DPFA and Bayesian state merging method [31] and Augmented Markov Model [12] for NPFAs. To arrive at a model of the underlying distribution they are trying to learn, these algorithms have to perform some generalization by either merging probabilistically similar states [28, 9, 31] or by merging all states and then splitting those states that do not accurately represent the data [12].

Although our approach to learning the agent interaction model was inspired by the above general work on learning finite automata, our initial efforts have led us away from general approaches to learning arbitrary automata. Instead, our initial focus has been on developing a learning approach that allows the learning to be modulated in ways consistent with the constraints discussed earlier, specifically window size, bounded structural generalization and frequency distribution accuracy. To this end, we have found that a representation similar to the prefix tree automaton representation [1] to be appropriate. However, we reverse this representation. In particular, we reverse the direction of all the edges, with the goal state as the root of the tree and add traversals from a unique start state to all the leaf nodes (Note: This violates the tree structure of the automaton but only at this unique start state). In addition, we maintain information about frequency counts of each edge. We discuss our approach more formally below, and discuss how this approach could be extended to take into account the factors of window size and bounded structural generalization.

The finite automaton ISAAC attempts to learn is defined as G = < S, A, L, a_0, g > where,

1. S, a set of symbols $s_1, s_2, \ldots, s_M$ encountered;

2. A, a set of states (or nodes) $a_1, a_2, \ldots, a_N$ where each state a_i consists of the following:

   – a_i^s, the symbol recognized by state a_i. $a_i^s \in$ S;
   – i, the state number which uniquely identifies the state.

3. a_0, the Start State where state number = 0 and $a_0^s = \emptyset$.

4. g, the Goal Symbol where g $\in$ S. The state that recognizes g is referred to as the Goal State;

5. L, a set of directed links $l_1, l_2, \ldots, l_P$ where each link l_i consists of the following:

   – l_i^f, state number of source of the link;
   – l_i^t, state number of destination of the link;

- l_i^{Σ}, number of times the link was traversed while adding new sequences of symbols(frequency of link).

No two links l_i and l_j are such that $l_i^f = l_j^f \neq 0$.

The frequencies of the patterns learned by the finite automaton are the frequencies of the links that begin at the start state, i.e., values of l_i^{Σ} for each link, l_i where $l_I^f = a_0^n = 0$. It is possible to obtain the probability distribution of the patterns of interactions from these frequencies.

Window size is defined as follows:

Def: *Window size* of k indicates that for a string of symbols $\sigma = s_1 s_2 \ldots s_i g$, where g is a goal symbol, the sequence added to the finite state automata will be $s_{i-k+1} s_{i-k+2} \ldots s_i g$ where $s_1 \neq g$, $s_2 \neq g$, $\ldots$, and $s_i \neq g$.

Bounded structural generalization is defined as follows:

Def: *Bounded structural generalization* of k implies that, for a sequence of symbols $s_1 s_2 \ldots s_n$ being added to the finite state automaton, there does not exist any subsequence $s_i s_{i+1} \ldots s_{i+m}$, such that $s_i s_{i+1} \ldots s_{i+m}$ is identical to $s_{i+m+1} s_{i+m+2} \ldots s_{i+2m}$ and $m \leq k$ and $1 \leq i \leq (n - 2m)$.

The algorithm for obtaining the multiple agent model is shown in Figure 7. Suppose the window size is set to k1 and the bounded structural generalization to k2. First, the function ExtractSequences() extracts a sequence of symbols of length equal to k1 that end with (intermediate) goals from the data traces. Next, to factor in bounded structural generalization, the function DoStructuralGeneralization() eliminates repeating subsequences of symbols from this sequence, progressively increasing the length of the subsequence being searched for from 1 to k2. The function AddPattern() then adds the resulting sequences to the finite automaton as follows: ISAAC traverses the finite automaton backwards from the goal state searching for symbols in the sequence one at a time and increasing the frequency of each traversed link. Thus, ISAAC determines the state, a_i up to which the sequence is already learned. ISAAC then proceeds to add the portion of the pattern that is not yet learned to the finite automata at a_i. While adding each symbol of this portion of the sequence ISAAC creates a new state, a_j that recognizes the current symbol and adds a link connecting a_j to a_I. The output of this algorithm is a finite state automaton that stores the frequencies of all the patterns learned. For obtaining the frequency of a specific pattern, we first find the path from the start state to the goal state that matches this pattern. The frequency of the edge from the start state to the next state on the path is the frequency of the pattern. Window size determines the size of the pattern or in other words the number of abstracted actions that are considered as being responsible for the team achieving its (intermediate) goal. The choice of window size is application dependent. There are tradeoffs when choosing window size:

- In many domains, actions that occurred closer to an intermediate goal are likely to have been more responsible for the outcome than actions that happened further away from the intermediate goal. Therefore a large window size may not be beneficial.

- Choosing a very small window size might cause sequences that are actually very different to appear to be the same while choosing a very large window size results in sequences that are essentially similar appearing to be different. Therefore, at very small window sizes, teams may appear superficially similar, while at very large window sizes, teams may appear superficially different.

- Increase in window size can make comprehensibility difficult for the human user.

Bounded structural generalization is used to capture repetitive subsequences within larger sequences. This parameter appears related to window size. For small window sizes, bounded structural generalization greater than zero may not be useful, as there are unlikely to be repetitive subsequences at small window sizes. However, for larger window sizes, bounded structural generalization could be used to capture such repetitive subsequences.

In addition to presenting the results of the learned finite automaton to the user, ISAAC uses the agent interaction model to suggest improvements in teams using two different techniques. The first technique is to compare two teams. To this end, ISAAC first obtains frequencies of patterns of agent interactions that resulted in success, for different teams. Now, if the distributions of the successful patterns for two teams can be shown to be different we can conclude that the two teams are different in the success strategies. In order to determine if the two distributions being similar, ISAAC treats the distribution with greater variance as the underlying distribution and the distribution with lesser variance as the observed distribution. ISAAC then determines whether the null hypothesis that the observed distribution is drawn from the underlying distribution, holds. ISAAC uses the standard Chi-Square goodness of fit method (Figure 8) to determine if this null hypothesis is true. The observed frequencies O_i are frequencies from the distribution

```
CreateModel (team, logs, choiceOfKeyInteractions, winsize, structgen, goalSym){
   team: team for which the Model is being constructed;
   logs: Logs of team's behavior;
   choiceofKeyInteractions: symbols to consider as key interactions;
   winsize: value for window size;
   structgen: value for bounded structural generalization;
   goalSym: symbol that indicates intermediate success or failure;
   patterns <- DoStructuralGeneralization (ExtractSequences (logs, \
               choiceOfKeyInteractions, winsize, goalSym), structgen);
   SuffixTreeAutomaton <- CreateSuffixTreeAutomaton(patterns, goalSym);
}

ExtractSequences (logs, choiceOfKeyInteractions, winsize, goalSym){
   sequences <- Mine logs for strings of key interactions of length \
                winsize starting from goalSym backwards;
   return sequences;
}

DoStructuralGeneralization (sequences, structgen){
   sequences: strings of symbols extracted from logs;
   for each sequence in sequences{
      pattern <- sequence;
      for i <- 1 to structgen{
        pattern <- replace multiple adjacent instances of identical substrings
                   of length i in pattern by a single instance of the substring;
      }
      patterns <- Add(pattern);
   }
   return patterns;
}

CreateSuffixTreeAutomaton (patterns, goalSym){
   patterns: strings of symbols after bounded structural generalization;
   for each pattern in patterns
        suffixTreeAutomaton <- AddPattern(suffixTreeAutomaton, pattern);
   return suffixTreeAutomaton;
}

AddPattern (suffixTreeAutomaton, pattern){
   suffixTreeAutomaton <- add pattern to suffixTreeAutomaton;
   /*(i) determine the substring of pattern that is already present by
     traversing backwards from goalstate. increment frequencies of all links
     traversed.
     (ii) add a new branch consisting of states that recognize that substring
      of pattern not yet present in the automaton.*/
   return suffixTreeAutomaton;
}
```

Figure 7. Algorithm for Building Multiple Agent Model.

with less variance and expected E_i is calculated by considering the underlying distribution to be the one with greater variance. If the χ^2 valued obtained is greater than the χ^2 value for 95% certainty limit, ISAAC discards the null hypothesis.

$$\chi^2 = \sum_{i=1}^{k}(O_i - E_i)^2/E_i$$

where O_i is the observed frequency for pattern i and
E_i is the expected frequency for pattern i.

Figure 8. Chi-Square Goodness of Fit Test

The Chi-Square Goodness of Fit Test is a valid choice for comparing teams. Clearly, parametric tests like the t-test and analysis of variance would be difficult to apply because they assume the underlying distribution is normal. The Chi-Square Goodness of Fit Test is a non-parametric alternative.Furthermore, unlike other non-parametric methods like the Kolmogorov-Smirnov and Anderson-Darling tests, which are restricted to continuous distributions, the chi-square test can be applied to discrete distributions.

Using this method it is possible to determine if two teams are dissimilar under the assumption that the environment they operated in was the same. (Alternatively, it is also possible to determine if the environments are dissimilar given that the same team is being compared under different settings.) If a team that performs poorly is found to be dissimilar to a team that performs well, the probability distributions of the patterns of these two teams may suggest techniques to improve the poorly performing team. The following subsection will provide examples of applications of this technique.

The second technique that ISAAC uses to suggest improvement in teams is perturbation analysis. The goal of perturbation analysis in the agent interaction model is to find unsuccessful patterns of subteam interaction from the data traces that are similar to the successful patterns learnt by the finite automaton. These unsuccessful patterns, which we refer to as "near misses", help the user scrutinize the difference between successful and unsuccessful patterns of interaction. In order to do this perturbation analysis, ISAAC begins with the patterns learned as part of the finite automata (which lead up to success). It then mines new patterns from the behavior traces that are very similar, and yet end in failure. For instance given an n-step pattern learned as part of the finite automaton, ISAAC may mine new n-step patterns, where the first n -1 steps are "similar" to the pattern from the finite state automata but where the last step is dissimilar. By "similar" we mean that the values of the attributes of the step from the new pattern lie in the same range as the attributes of the step of the successful patterns in the finite automata. The perturbations are restricted to the last step where ISAAC explicitly makes sure that one of the attributes of the state are not in the range of attribute values for that step in successful patterns. For instance, if "angle" is one attribute of the state, then the perturbation ensures that the "angle" range is kept the same for all the steps, except for the last step where the angle range is not enforced, thus leading to a perturbation.

Since these "near misses" are almost the same as the successful patterns but for the last step, team developers can determine what attributes are useful in the last step to make an unsuccessful pattern into a successful pattern.

4.2. Application of Method to RoboCup Soccer

In RoboCup soccer a goal occurs as a result of a sequence of kicks by one or more players. It is not only the last kick that is responsible for the goal but the actions of the different agents who interacted just prior to the goal. Thus, in order to analyze the performance of a team, the multiple agent model is important.

The first step necessary to apply the agent interaction model to the RoboCup domain is the determination of the patterns to examine and a notion of success or failure of these patterns. We again use a soccer goal as a notion of success. We consider the sequence of actions (kicks) before the ball enters the goal to be the pattern. The feature of the action that is considered important is the player who performed the action. The player can be one of shooter (the last player, from the team which scored the goal, to kick the ball), teammate (any other player, from the team which scored the goal, who kicked the ball), opponent (any player from the team against whom the goal was scored who kicked the ball). This feature selection is one important aspect of generalization. We do not consider self goals (when the player kicks the ball into his/her own goal) because such goals are presumably unintentional.

Before applying the learning algorithm we have to determine values for both window size and bounded structural generalization. We can then use the learning algorithm described in Section 4.1 to learn a finite state automaton that recognizes all the successful patterns along with their frequencies of occurrence. Looking at the probability distributions of scoring patterns of teams reveals more about the team's scoring strategy. This can be beneficial to developers of this team and as well as for opposition teams. It is also possible to compare the scoring strategy of different teams based on the probability distribution of these patterns.

Window size in the case of RoboCup is the number of kicks we consider to be in the pattern. It would appear that in a sequence of kicks that result in a goal, the kicks that happened earlier are less responsible for the eventual goal taking place. This suggests that increasing window size too much will not be useful.

The user can choose any value of window size that satisfies his/her needs better. We have selected window size to be 3 based on the number of patterns generated, the comprehensibility to a human user and the fact that the effect that a kick had on a goal lessens with the temporal distance of that kick from the resulting goal. Figure 9 shows the number of different patterns obtained (Y-axis) as the window size is increased (X-axis), for the top eight teams at RoboCup-98. A window size of two gives only a maximum of 3 different patterns — *opponent* $\rightarrow$ *shooter* $\rightarrow$ *goal* (opponent kicks the ball to the shooter, who scores the goal), *teammate* $\rightarrow$ *shooter* $\rightarrow$ *goal* (teammate passes the ball to the shooter, who scores the goal) and *shooter* $\rightarrow$ *shooter* $\rightarrow$ *goal* (shooter kicks the ball to himself before scoring the goal). This is clearly too little to analyze teams based on their interactions. Yet as can be seen in the figure, for window size equal to 5, the teams have an average of 16.75 different sequences while one of the teams has as many as 29 different patterns. This may make the analysis difficult to comprehend for a human. With an average of 6.63 different patterns per team, the window size of 3 appears to be a reasonable choice.

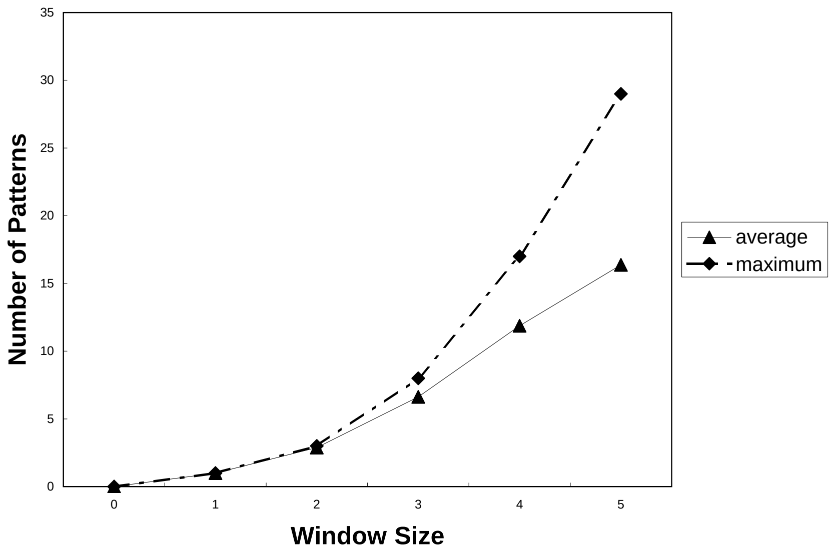


Figure 9. Number of Patterns vs. Window Size for the top 8 teams of RoboCup-98.

In the case of RoboCup Soccer we choose bounded structural generalization to be 0 because it seems unlikely that subsequences will be encountered within sequences when window size is equal to 3. Also, if bounded structural generalization were greater than 0 we might have a situation where it becomes difficult to differentiate between two very different patterns.

From the finite automaton of each team we can obtain all the learned scoring patterns along with their frequencies. With this information, we can first examine teams by themselves. Consider the example of the Windmill Wanderer team from RoboCup-98. Windmill Wanderer scored 17 goals where the shooter dribbled in before the shot ($shooter \rightarrow shooter \rightarrow shooter \rightarrow goal$) and another 9 goals where a teammate controlled the ball before passing to the shooter ($teammate \rightarrow teammate \rightarrow shooter \rightarrow goal$), out of a total 37 goals. Thus, this team scores more often when they control the ball all the way in to the goal. Unfortunately, for Windmill Wanderer, 27 near misses were found similar to the 17 goals from the $shooter \rightarrow shooter \rightarrow shooter \rightarrow goal$, suggesting this pattern was well defended or the team was making some mistakes. Contrasting this to 4 near misses similar to the 9 goals from the pattern $teammate \rightarrow teammate \rightarrow shooter \rightarrow goal$ suggests that this pattern sets the agent up for a better or easier shot. Windmill Wanderer placed third in the tournament, and the 27 near misses may have been a crucial factor in its third place finish.

In addition, we can use the method described in Section 4.1 to compare teams two at a time. For example, comparing AT_Humboldt97 and AT_Humboldt98 — two entries from Humboldt University at RoboCup-98 — we obtain a χ^2 value of 81.34 for a window size of 3, while the threshold is 15.51. This suggests that the two teams are very dissimilar. Looking at the frequencies of the scoring patterns shows that AT_Humboldt97 scored only 1 goal when it controlled the ball all the way into the goal (opponent team's player did not kick the ball before it was shot) while AT_Humboldt98 scored 51 of its 88 goals this way. This suggests that AT_Humboldt97 was extremely opportunistic in its approach and relied on the opponent making a mistake near the goal, while AT_Humboldt98 was good at passing and dribbling the ball into the goal. The results of RoboCup-98 show that AT_Humboldt98 finished in second place in RoboCup-98 while AT_Humboldt97 was ranked lower than 16th. This seems to suggest that the changes in the scoring patterns of AT_Humboldt98 were a key factor in its improved performance. While this analysis does not conclusively establish the reasons for the improved performance of AT_Humboldt98, it at least points us in a fruitful direction for understanding differences in team performance.

5. Automated Engagement Summary – Team Model

The third ISAAC model attempts to address success or failure as a team. In designing this model we had two options possible. One was to tailor the analysis to specific teams. In particular, by analyzing data traces of past behaviors of a specific team, it would be possible to explain why this team tends to succeed or fail. This approach would be similar to the one followed in Section 3, which explained why agents' critical actions tend to succeed or fail in a team specific manner. A second option was to analyze teams in terms of why teams succeed or fail in the domain in general, in a non-team-specific manner (which does not require data traces from the particular team being analyzed, but from other teams in this domain). Despite the advantages of option 1 (team specific explanations), this option was rejected due to the lack of large amounts of team specific engagement (team activity) data, and option 2 was used. In particular, unlike the individual agent model in Section 3, which can obtain lots of individual action data points even from a single engagement, a single engagement is just one data point for the global team model. For instance, even a single RoboCup game provides a large number of shots on the goal to begin learning the individual agent model; yet this single game provides just a single instance and is not enough to begin learning a global team model. Thus unlike previous models, the team model is not specific to a particular team. We use all available data of every engagement in the domain, such that the model provides general explanations in terms of why engagements in a particular domain result in success or failure.

5.1. Conceptual Overview of Team Model

The global team model focuses on the analysis of why teams succeed or fail over the course of an engagement (overall team activity). The assumption of this model is that there can be many different factors that impact a team's overall success or failure. In a complex environment, a team may have to perform many actions well in order to be successful. These actions may involve entirely different sub-teams, and very different kinds of events, perhaps widely separated in time, may be implicated in success or failure. Nevertheless, there may be patterns of these factors that, although not strongly related in the behavioral trace, do in fact correlate with whether a team succeeds in a domain. The global team analysis attempts to find these patterns.

Acquiring the team model involves techniques similar to the previous models as discussed above, except that rather than searching for points of intermediate success or failure, overall features that lead to the final outcome over an entire engagement are more useful. These outcomes may be classified as success, failure, tie, etc. It is again up to the domain expert to choose these features and to provide the classes of the final outcomes. The C5.0 induction algorithm is used on these features, classifying the engagement for each team, and learning the dominant features that lead to that particular outcome. Just as in section 3.1, our use of decision trees is for explaining the outcome of an encounter rather than for prediction of the outcome.

A different approach from section 3.1 is taken for using these rules. When analyzing a specific engagement, we mine the features from the engagement and determine which learned rule the current engagement most closely matches. This rule then provides key features that correlate with team success or failure [37, 42]. ISAAC uses natural language generation to create a summary to ease human understanding of the engagement as a whole. While the rule is a model of the engagement, further explanation of this rule to the human comes from generating the summary.

Thus, with the current method, ISAAC generates a natural language summary of each engagement employing the rule that matched the engagement as the basis for content selection and sentence planning in accordance with Reiter's architecture of natural language generation [27]. Reiter's proposal of an emerging consensus architecture is widely accepted in the NL community. Reiter proposed that natural language generation systems use modules in *content determination*, *sentence planning*, and *surface generation*. ISAAC's NL generation can be easily explained in terms of these modules.

The algorithm for building the global team model is presented in Figure 10. Each engagement (team activity) is considered to be a single data item.The function BuildTeamModel() mines the logs to obtain training data and using C5.0 with this data it obtains rules that describe global team behavior. The CreateTemplate() function creates templates for generating summaries using these rules. ISAAC uses the rules for *content determination* since each rule contains the feature values that were pertinent to the result of the engagement it classifies. Furthermore, the conditions of each rule also have some ordering constraints, since the rules come from a decision tree learning algorithm, and we use this to form our *sentence planning.* We consider branches closer to the root of the tree to have more weight than lower branches, and as such should be stated first. Each condition is associated with a single sentence, and ordered accordingly. *Surface generation* of the text is done by a template instantiation process. Every rule has one (or more) templates that obey the content determination and sentence planning specified. The creation of these templates, specifically the use of connectives, pronouns, etc. is currently done manually.

In order to generate a summary for a particular engagement, the function GenerateSummary() starts with the raw data of the engagement and mines the features it needs, and matches it to a pre-existing rule with the function ObtainBestMatchingRule(). This function finds those rules whose conditions are satisfied by the statistics of the engagement. In the event that there are multiple rules that match the given engagement, the rule that classifies the most training examples is used. If none of the rules that are satisfied by the engagement have the same outcome as the engagement, the rule whose outcome matches the outcome of the engagement most closely, is selected. The template, corresponding to this matched rule, is instantiated with specific feature values (statistics) from the engagement. Finally hyper-links to examples of those features are added, for display in the multimedia viewer.

5.2. Application of Team Model to RoboCup

To learn rules of why teams succeeded or failed in previous engagements, ISAAC reviews statistics of previous games. The domain expert must provide the domain knowledge of what statistics to collect, such as possession time and number of times called off-side. A complete list of statistics collected is presented in Appendix B. ISAAC uses this information to create a base of rules for use in analysis of future games.

ISAAC learns and uses seven classes of rules covering the concepts of big win (a victory by 5 goals or more), moderate win (a victory of 2-4 goals difference), close win (1 goal victory), tie, close loss (by 1 goal), moderate loss (2-4 goals), and big loss (5 or more goal loss). The motivation for such subdivision is that factors leading to a big win (e.g., causing a team to out-score the opponent by 10 goals) would appear to be different from ones leading to a close win (e.g., causing a one goal victory) and should be learned about separately. While this fine subdivision thus has some advantages, it also has a disadvantage, particularly when the outcome of the game is at the border of two of the concepts above. For instance a 2-0 game (moderate win) could very well have been a 1-0 game (close win). Thus, we anticipate that the learned rules may not be very precise, and indeed as discussed below, we allow for a “close match” in rule usage.

To use these rules, ISAAC first matches the statistics of a new (yet to be analyzed) game with the learned rules. If there is a successful match, ISAAC checks the score of the game against that predicted by the matching rule before writing the summary. If the match is exact or close (e.g. the actual game statistic matched a close win rule, although

```
BuildTeamModel (logs, classes, features){
   logs: Logs of team’s behavior;
   classes: Classes for classification of engagements (team activity);
   features: Features used in the classification;
   for each engagement in Logs{
      modelData <- Add (ExtractDataPoint (engagement, features, classes));
   }
   teamModelRules <- ApplyC5.0(modelData, features, classes);
   summaryTemplates <- CreateTemplates (teamModelRules);
}

CreateTemplates (teamModelRules){/*currently with user assistance*/
   teamModelRules: Rules of Global Team Activity obtained by applying C5.0;
   for each rule in teamModelRules{
      for each condition in rule in order of appearance{/*sentence planning*/
         generate sentence corresponding to condition;
         template <- Add(summaryTemplate, sentence)
      }
      summaryTemplates <- Add (summaryTemplates, template);
   }
   return summaryTemplates;
}

GenerateSummary(engagement, teamModelRules, summaryTemplates, classes, features){
   engagement: Data trace of a single team activity;
   teamModelRules: rules of global team model obtained by running BuildTeamModel(
   summaryTemplates: templates for summary generation obtained by BuildTeamModel(
   dataPoint <- ExtractDataPoint (engagement, features, classes);
   matchingRule <- ObtainingBestMatchingRule (teamModelRules, dataPoint);
   summary <- fillTemplate (summaryTemplates[matchingRule], dataPoint);
   return summary;
}

ObtainBestMatchingRule(teamModelRules, dataPoint){
   dataPoint: the feature values and outcome corresponding to an engagement;
   satisfiedRules <- rules from teamModelRules that satisfy dataPoint;
   matchedRules <- rules from satisfiedRules that have same outcome as dataPoint;
   if (matchedRules is empty)
      matchedRules <- rules from satisfiedRules whose outcomes are closest to
                  outcome of dataPoint;
   bestMatchingRule <- rule from matchedRules covering most training examples;
   return bestMatchingRule;
}
```

Figure 10. Algorithm for Global Team Model.

the game had an outcome of 2-0), the template is used as is. If there are multiple matches, the matching frequency is used to select the best rule. However, if no match is close to the actual score, ISAAC still uses the rule, but changes the template to reflect surprise that the score did not more closely match the rule.

Once a rule is matched, ISAAC now has an associated template to shape the game summary. The template orders components of the rule according to their depth in the original decision tree, in accordance with our sentence planning technique. ISAAC then fills in the template, mining the features of this particular game to create a summary based on the rule. An example rule is shown in Figure 11.

Ball in Opposition Half > 69%
Average Distance of Opponent Defender > 15 m
Bypass Opponent Last Defender > 0
Possession time > 52%
Distance from Sideline to Opponents Kicks > 19 m
→ class Big Win

Figure 11. Example team rule for big wins.

To see how this rule is used in creating a natural language summary, we examine one summary generated using this rule as a template, shown in Figure 12. In this case, ISAAC is presenting a summary explaining the reasons for which 11Monkeys was able to defeat the HAARLEM team. The underlined sentences above correspond directly to the rule, with some augmentation by actual statistics from the game. By using the rule for content determination and sentence planning, ISAAC is able to present the user the reasons for the outcome of the engagement, and avoid presenting irrelevant data consisting of irrelevant features. [1]

6. Evaluation and Results

To evaluate ISAAC, we evaluate each of its models in isolation and then the effectiveness of the integrated ISAAC system. Section 6.1 presents the evaluation of the individual agent model, Section 6.2 describes the evaluation of the multiple agent key interaction model, Section

[1] The headline and the first sentence of the summaries were created based on headlines and first sentences of the press reports for the World Cup'98 (Human) Soccer games in Paris. An appropriate headline was chosen at random from a selected set to avoid repetitions.

HAARLEM Offense Collapses in Stunning Defeat at the hands of 11Monkeys!

11Monkeys displayed the offensive and defensive prowess, shutting out their opponents 7-0. 11Monkeys pressed the attack very hard against the HAARLEM defense, keeping the ball in their half of the field for 84% of the game and allowing ample scoring opportunities. HAARLEM pulled their defenders back to stop the onslaught, but to no avail. To that effect, 11Monkeys was was able to get past HAARLEM's last defender, creating 2 situations where only the goalie was left to defend the net. 11Monkeys also handled the ball better, keeping control of the ball for 86% of the game. HAARLEM had a tendency to keep the ball towards the center of the field as well, which may have helped lead them to ruin given teh ferocity of the 11Monkeys attack .

Figure 12. Game Summary for HAARLEM-vs-11Monkeys. The lines underlined correspond to conditions in the rule used to generate the summary. These link to cases of that condition that can be viewed in the multimedia viewer.

6.3 presents the evaluation of the global team model, and Section 6.4 describes the evaluation of the overall ISAAC system.

6.1. Evaluation of the Individual Agent Model

We evaluate along several dimensions: (i) bottom-up discovery of novel patterns; (ii) ability to perform global analysis; (iii) compactness and understandability of model; and (iv) pattern accuracy.

A key measure of ISAAC's individual agent model is the effectiveness of the analysis, specifically the capability to discover novel patterns. Section 3.3 highlighted a rule learned about the Andhill97 team concerning their aiming behavior. This rule was one instance of ISAAC's surprising revelation to the human observers; in this case, the surprise was that Andhill97, the 2nd place winner of RoboCup-97, had so many goal-shot failures, and that poor aim was at least a factor. Not only was this intriguing to other observers, this was also intriguing to the developer of the team, Tomohito Andou. After hearing of this result, and witnessing it through ISAAC's multimedia interface, he told[2] us that he *"was surprised that Andhill's goal shooting behavior was so poor ..."* and *" ... this result would help improve Andhill team in the future."* Some other teams that used ISAAC in preparation for RoboCup included CMUnited99, Headless Chickens, Gongeroos and others, all of which provided positive feedback about ISAAC's capabilities. Indeed, Peter Stone of CMUnited99 pointed out[33]: "I particularly like the way that you can change the rules and see how that affects the cases that are covered. Being able to cycle through the covered cases is also a great feature."

Another interesting result from the individual agent analysis model comes from the number of rules governing shooting behavior and defensive prowess. Figure 13 shows that in each year, the number of rules for defense decreased for the top 4 teams, perhaps indicating more refined defensive structures as the teams progress. Also, the number of rules necessary to capture the behavior of a team's offense is consistently more than that necessary for defense, possibly due to the fact that no single offensive rule could be effective against all opponent defenses. The key here is that such global analysis of team behaviors is now within reach with team analyst tools like ISAAC.

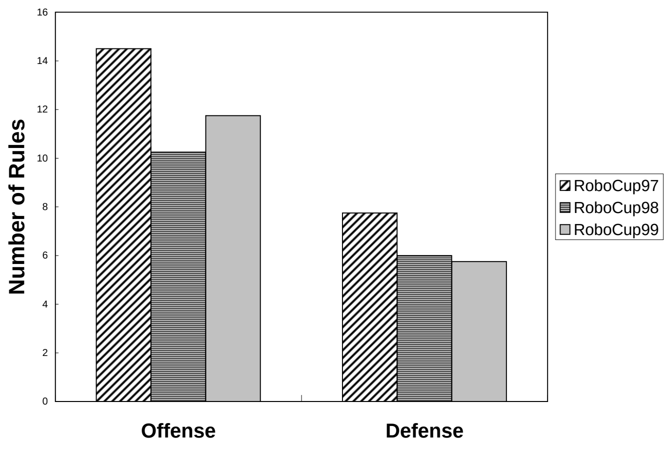


Figure 13. Number of Rules by Year for Top 4 Teams

In order to evaluate if the individual agent model meets it's goals of understandability and compactness we consider the number of rules generated per team and the number of conditions per rule for RoboCup-98 teams. The Figure 14 shows the number of rules of offense for the top 8 teams at RoboCup-98. As this figure clearly indicates, the number of rules that characterize a team's offense are few enough to be comprehensible to a human user. The average number of of rules of offense is 10.875. In addition, these rules are very compact as can be seen in Figure 15, which shows the number of conditions in a rule for the top 8 teams at RoboCup-98. All of these teams had fewer than 3 conditions in a rule on average. Although Figures 14 and 15 focus on the top 8 teams at RoboCup-98, the observations from these figures are true for all RoboCup teams. Owing to the manageable number of rules per team, the comprehensibility of the features selected and the compact nature of the rules generated, the goals of understandability and compactness of the individual model are met. The data used to generate Figures 14 and 15 is presented in Appendix C.

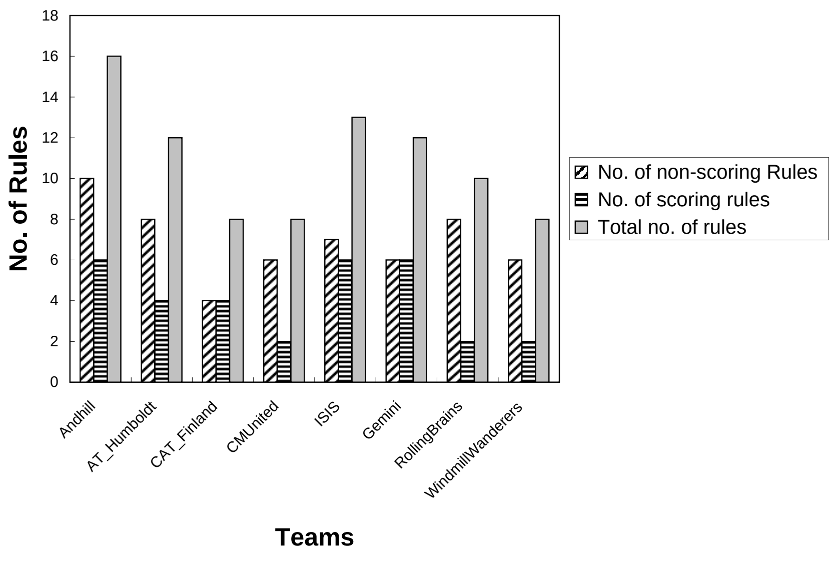


Figure 14. Number of Rules in Individual Model. Scoring rules refer to rules that describe success in shots on goal; Non-scoring rules describe failed goal shots

Another point of evaluation is understanding how well the model captures the shooting behaviors. To this end, ISAAC models were applied to predict game scores at RoboCup-99, a rather difficult problem even for humans. ISAAC used rules describing a team's defense and matched them with the raw averaged data of the shots taken by the other team to produce an estimate of how many goals would be scored against that team in the upcoming game. Performing this analysis for both teams produced a predictive score for the outcome of the game. This prediction obviously ignores many critical factors, including the fact that some early games were unrepresentative and that teams were changed by hand during the competition. Yet in practice, ISAAC's

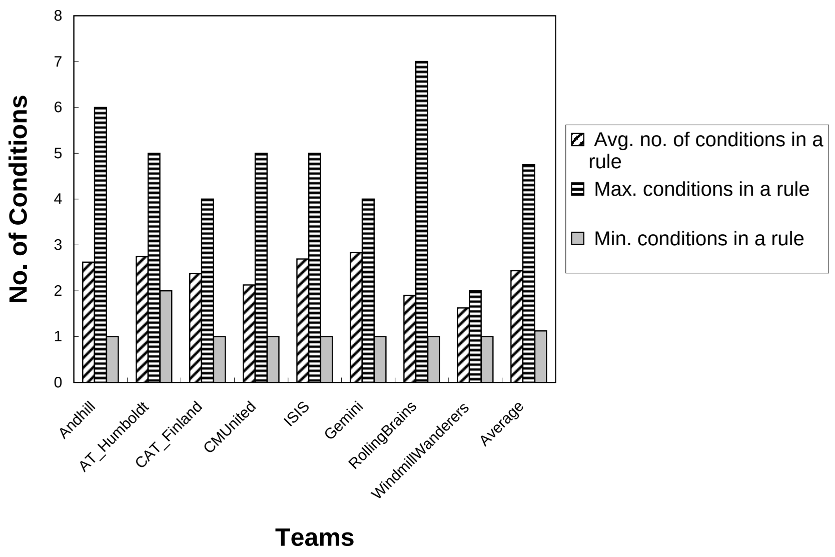


Figure 15. Number of Conditions in a Rule in Individual Model.

predictive accuracy was 70% with respect to wins and losses, indicating it had managed to capture the teams' defenses quite well in its model.

Finally, in addition to the analysis, the multi-media viewer was also quite helpful to developers as seen from comments by another developer, Helmut Myritz of AT_Humboldt who said *"I have to say that we know some weaknesses of the AT_Humboldt98, like a not optimal kick, and a sometimes strange goal kick behavior or a static defense. All these things could be realized well by watching your analysis of our games."* [20]. Note that ISAAC's users — for instance, Andou in Japan and Myritz in Germany — were able to view ISAAC's analysis overseas on the web.

6.2. Evaluation of the Multiple Agent Model

We have already evaluated the multiple agent model to some extent in Section 4.2. There, we showed how the probability distribution obtained for the Windmill Wanderer team of RoboCup-98 could be analyzed and also how the teams, AT_Humboldt97 and AT_Humboldt98 were compared. It is possible to perform such analysis on any RoboCup team and also to compare any two RoboCup teams. The comparison of teams suggests methods for improving the performance of teams that did not perform well.

To illustrate how this comparison is useful, we use the method of comparison from in Section 4.1 to compare the top 8 teams of RoboCup-98 with each other. In Figure 16, the X-axis shows the top 8 teams at RoboCup-98 and the Y-axis shows the χ^2 values obtained by comparing the probability distributions of the scoring patterns of these teams to each other. Values that lie above the threshold line indicate that we can state with a confidence limit of 95% that the two distributions are dissimilar. Some of the observations that can be made from this figure are:

1. The teams AT_Humboldt, CMUnited, Gemini and WindmillWanderer are significantly dissimilar from ISIS while Andhill98, RollingBrains and CAT_Finland are not as dissimilar.

2. All the remaining seven teams are significantly different from CMUnited, the winner of RoboCup-99!

3. Andhill and CAT_Finland are not significantly different from each other.

Observation 1 can be explained by looking at the frequencies of scoring patterns. This revealed that indeed, CMUnited, AT_Humboldt and WindmillWanderer laid more emphasis on ball control and dribbling while ISIS was more opportunistic in its strategy and tried to capitalize on its opponents' mistakes. The results of RoboCup-98 also show that the teams CMUnited, AT_Humboldt and WindmillWanderer finished in the top 3 positions. Observation 2 suggests that CMUnited scoring patterns were very different from the other teams. Looking at the shooting patterns and their frequencies reveal that unlike other teams, CMUnited relied heavily on good dribbling skills and good ball control. This probably accounted for why CMUnited finished in first place in RoboCup-98. Observation 3 is not that surprising given that both teams relied on opportunistic patterns for most of their goals. It is interesting to note that Gemini finished in 5th place and CAT_Finland in 7th place. The data used to generate this graph is shown in Appendix C.

Thus, the multiple agent team interaction model is able to appropriately discriminate between teams and suggest methods of improvements, e.g., ISIS's performance might be improved by giving the team the necessary skills and strategies to use scoring patterns similar to these top three teams.

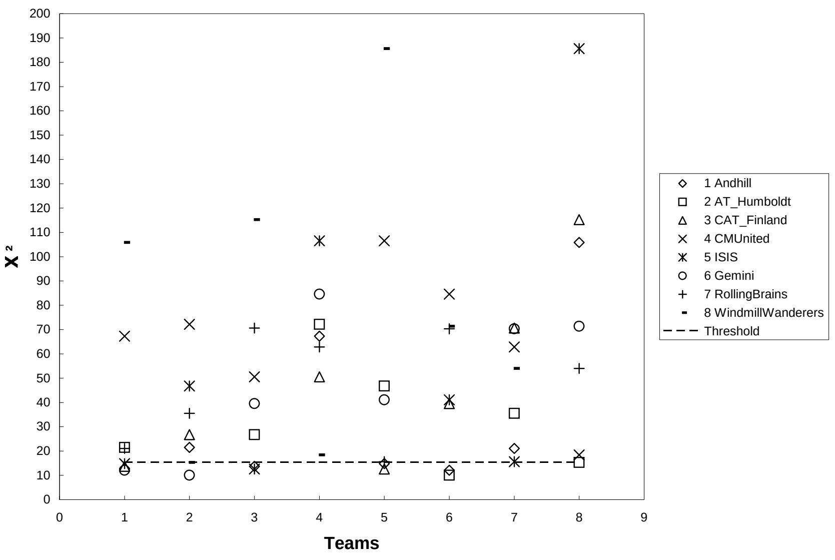


Figure 16. χ^2 values for the top 8 RoboCup-98 teams when compared to each other for window size=3. Threshold = 15.51 represents the χ^2 value for 95% certainty limit.

6.3. Evaluation of the Global Team Model

For the global team model, three different evaluations were performed. In the first, we distributed a survey to twenty of the participants at the RoboCup-99 tournament, who were witnessing game summaries just after watching the games. Figure 17 shows the breakdown of the survey, showing that 75% of the participants thought the summaries were very good.

In the second evaluation, we compared natural language summaries generated without a global team model with summaries generated directly from the original feature set of 12 features, without using C5.0 to pinpoint the justifications of a team's success or failure. Comparison between the two summaries revealed the following. On average, ISAAC uses only about 4 features from its set of 12 statistics in the summaries, resulting in a 66% reduction from a natural language generator not based on ISAAC's machine learning based analysis. Thus, ISAAC's approach was highly selective in terms of content. Furthermore, summaries generated without ISAAC were much longer, lacked variety, and failed to emphasize the key aspects of the game.

Finally, we measured ISAAC's use of the team model for natural language generation by viewing the error rates from the machine learn-

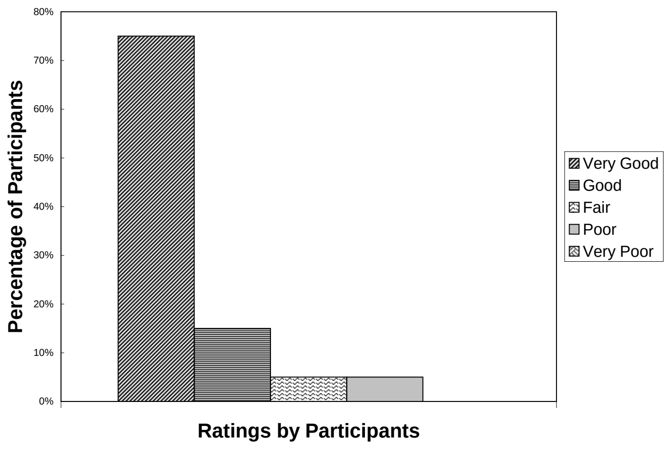


Figure 17. Automated Game Summary Survey Results.

ing algorithm used. These error rates tell us how accurately ISAAC's learned rules reflected the game. On the original set of games for which ISAAC's rules were learned, 87% of the games were classified correctly (70% exact match, 17% close match), resulting in an error rate of 13%. Our test set of (unseen) RoboCup '99 games produced 72% classified correctly (39% exact match, 33% close match), for an error rate of 28%. If an error does occur, ISAAC still produces a summary, but it reflects its surprise at the outcome, thus explaining the error. The error rate on our test data could indicate that a better feature set is possible or that the data may be noisy.

6.4. Evaluation of the overall ISAAC system

Evaluating ISAAC as an integrated system is more difficult. However, some observations can still be made. ISAAC was awarded the "Scientific Challenge Award" for outstanding research by the RoboCup Federation. ISAAC was used extensively at the RoboCup-99 tournament in Stockholm, Sweden (held in conjunction with IJCAI'99), where received a great deal of praise and other feedback. While visualization tools like the 3-D visualization tools [14] by Bernard Jung (see figure 19) were available, what was missing was the kind of analysis provided by ISAAC. ISAAC was continuously running throughout RoboCup-99 with its analysis continually projected on the screen. Developers used ISAAC to analyze opponent teams after the early round matches to get a feel for the skill of upcoming opponents. Spectators and developers alike were able to view ISAAC's game summaries just minutes after a game, and there was also a great deal of speculation concerning ISAAC's predictions on future games. As mentioned earlier, ISAAC was also used in preparation for RoboCup-99; Figure 18 shows ISAAC in use at RoboCup-99.

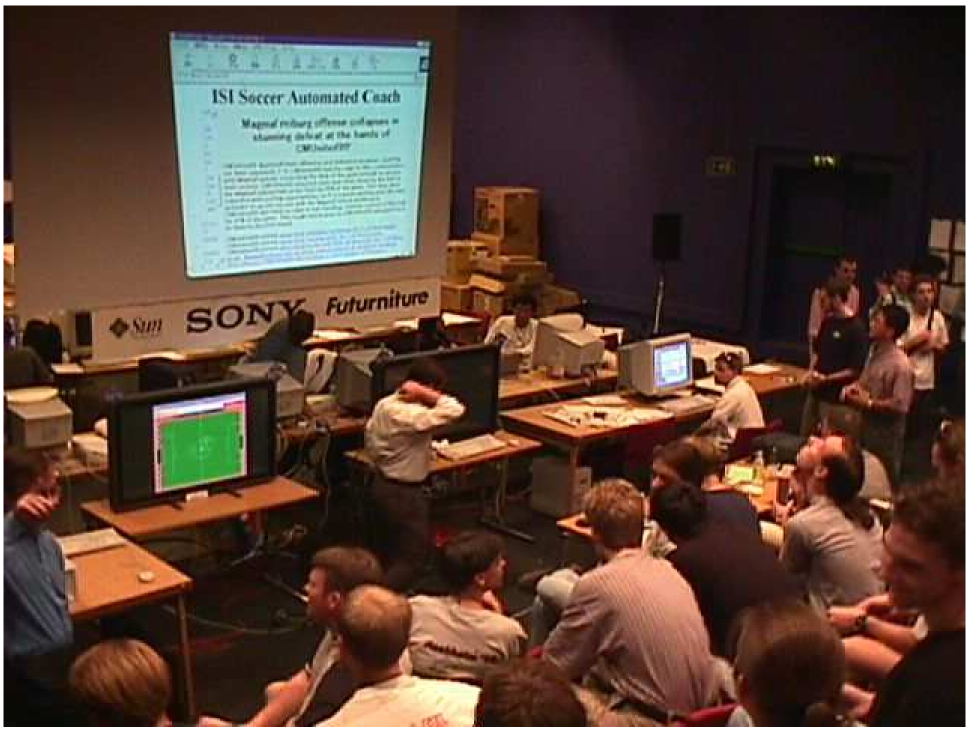


Figure 18. ISAAC in use – RoboCup-99, Stockholm. ISAAC was continuously running throughout the RoboCup-99 tournament with its analysis continually projected on the screen as shown in the figure. This tournament lasted several days and ISAAC analyzed dozens of games of teams. Here, ISAAC's predictions are shown on the right of the screen.

7. Generality of ISAAC

To illustrate the generality of ISAAC's team analysis techniques, this section discusses their applicability in very different domains. We begin by applying ISAAC's analysis techniques to an agent team performing mission rehearsal simulations to evacuate civilians stranded in a hostile location [38]. The team comprises of 11 different heterogeneous agents, viz. a multi-modal user interface agent, a route-planner, a web-querying information-gathering agent and 8 synthetic helicopter pilots.

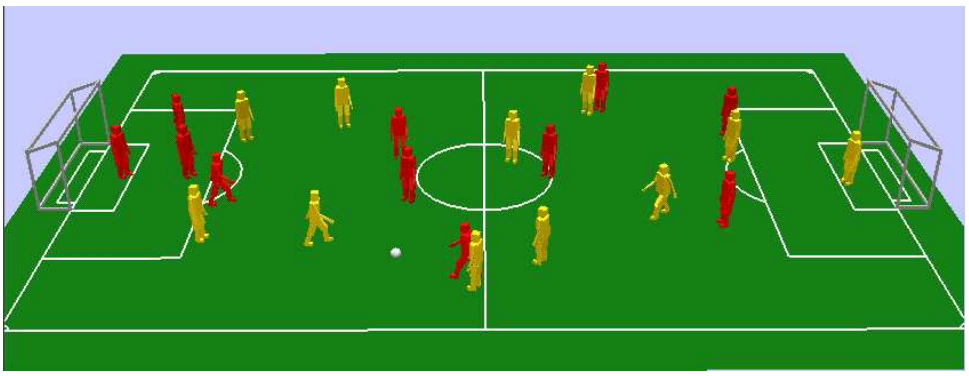


Figure 19. 3-D snapshot of a RoboCup soccer game.

The system must dynamically plan routes avoiding obstacles and enemy threats. The helicopters have to fly a coordinated mission to evacuate the civilians.

Tacticians and agents developers of these mission rehearsal simulations are often interested in understanding the impact of changes in the environment on agent-team performance. For instance, changes in evacuation tactics, threat profiles, and position of civilians may have different impact on team performance; and mission rehearsal simulations could help tacticians select the tactics leading to improved team performance. Here, to analyze team performance, we focus on the sequences of messages exchanged by the agents during a single evacuation rehearsal. In particular, given that these agents are highly heterogeneous, it is difficult to extract their actions as in RoboCup (e.g., user-interface agents' actions are to communicate with a user, while helicopters' actions are to fly, while a web-querying agent issues queries). However, the agents communicate in a common language — indicating commitment to plans or sub-plans and completion of plans and sub-plans — so that sequences of messages between agents can be more easily used for team analysis. Here, changes in the environment that impact team performance are often reflected as changes in the sequences of messages exchanged. For instance, some evacuation tactics may result in increased frequency of message sequences dealing with ground-based missile threats; while other tactics may result in few such sequences. Thus, these other tactics may be determined to be more successful in preempting missile threats.

ISAAC's multiple agent model seems well suited for this analysis. Interactions are considered to be the messages exchanged and are represented by the content of the message. A message, which says that the task has been completed, indicates that the goal of evacuating the civilians has been achieved. APPENDIX D describes two examples of messages exchanged in this domain. Using the learning algorithm described in Section 4.1, we obtain probability distributions of the different patterns of message exchanges that result in the goal being achieved. It is possible to compare different environments based on the probability distributions of these patterns.

In this domain, all messages exchanged are important as they relate to the execution of critical team plans and sub plans, which has a direct bearing on the outcome of the mission. It is, therefore, not desirable to specify a window size of interactions that are responsible for the goal. For the purpose of the algorithm we specify window size to be infinity so as to include all messages in the sequence of symbols learned by the finite automaton. Several messages can be repeated at multiple times during the message logs, which corresponds to the team repeating the execution of a sub plan. If bounded structural generalization is set to zero the resulting finite automaton is extremely large. Figure 20 shows the number of states in the finite automata on the Y-axis and the value of bounded structural generalization on the X-axis. Each data point in this graph corresponds to the size of the finite automata after learning all the logs that we have in this domain. As can be seen from the figure, increasing structural generalization beyond 2 does not decrease the number of states. This suggests that no repeating subsequence of messages of length greater than two was seen in the message logs.

To test ISAAC's application in this domain, we compared two sets of runs. The first set was reported by Gal Kaminka[16]. He had obtained a set of 10 logs of message exchanges among agents. We recently created a new set of runs. We wished to understand if ISAAC's analysis would reveal any similarities or dissimilarities in the patterns seen in the two runs, and possibly the causes for such dissimilarities. These dissimilarities could be due to changes in the composition of the evacuation agent team, changes in our computational environment, etc. It is difficult to answer the question purely by examining the message logs, because each log contains approximately 1000 messages. ISAAC can abstract out the detail and is critical for understanding the activities of the system.

We applied the learning algorithm described in Section 4.1 (See Figure 7) to learn the probability distribution of the patterns of each environment for bounded structural generalization equal to 2 and window size equal to infinity. The pattern of symbols needs to be abstracted abstracted out of the message logs to get rid of irrelevant messages and unnecessary details in the message. One immediate surprise here was the number of different sequences/patterns of execution learned. With ISAAC's abstraction, it is possible to see that the different patterns arise because: (i) some agents may fail during execution, requiring

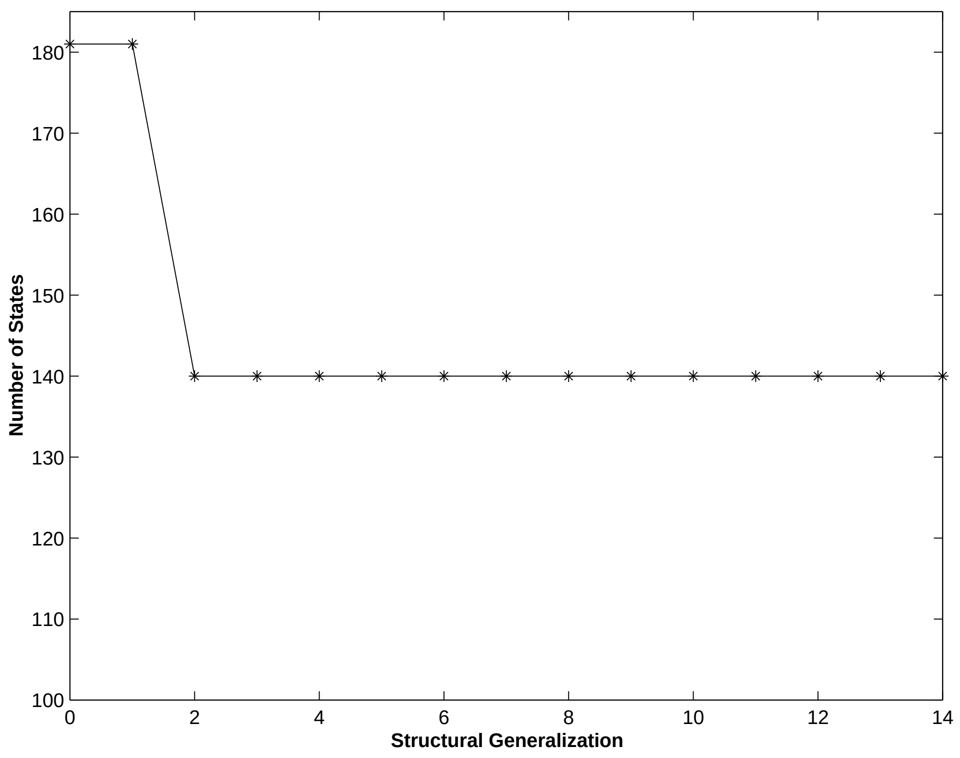


Figure 20. Structural Generalization vs. Number of States in finite automaton.

other agents to work around such failures; (ii) enemy reaction may vary requiring agents to sometimes engage in failure recovery. The combinations of which agents fail and how enemy reactions vary, lead to different patterns.

Is the new set of runs very different from Kaminka's set of runs? Comparing the probability distributions of the two sets of runs, we obtain a χ^2 value of 150.0 while the threshold for 95% confidence limit is 21.3. This indicates that the two teams are very dissimilar, a somewhat surprising result, since we did not anticipate a significant change in this team.

However, observation of the patterns learned reveals the following:

– In Kaminka's set of runs, helicopter pilot agents explicitly synchronize before landing to evacuate civilians. In the new runs, the helicopters do not synchronize explicitly.

– One of the agents in our set of runs, which does route planning appeared to fail more frequently in the new set of runs, compared with Kaminka's set of runs.

Thus by applying ISAAC's technique, differences between two teams can be detected more easily and the reasons for these differences can be found more easily. Tacticians can potentially use tools such as ISAAC to change evacuation simulation parameters and gauge their impact on team performance.

There are several other similar domains where ISAAC could be useful. For example, in team domains like agent teams in foraging and exploration [4], ISAAC could be used for exploring actions, interactions, and global trends such as target hit rate and friendly fire damage, ISAAC could produce a similar analysis of military tactics in the battlefield simulation domain [35] (see Figure 23), and use similar presentation techniques as well. Indeed, ISAAC's techniques could apply to analysis of many types of teams and their actions.

Team analysis would also be valuable in problems requiring distributed, dynamic resource allocation [19], e.g., Figure 21 shows a domain where a set of sensor agents must collaborate to track multiple targets. Here, each sensor agent controls a hardware node (as shown in Figure 22), and must make decisions on which other agent to assist in tracking targets, when to turn on/off to conserve power, which of its multiple radar heads to turn on, what messages to send other agents and when to send (despite the limited bandwidth) etc. As we scale up the number of sensor nodes, analysis of performance of individual nodes, teams and subteams is crucial. The analysis is complicated by problems like noise and sensor failures. We may wish to understand for instance when a sensor team succeeds or fails in tracking particular types of targets. Given the data traces of agent's actions, ISAAC could be used to determine patterns that result in success. It could help in contrasting different strategies for tracking particular types of targets. Furthermore, the global team model could be used to explain to a user why tracking succeeded or failed over an entire set of targets using NL.

8. Related Work

The research presented in this article concerns areas of multi-agent team analysis and comprehensible presentation techniques. We have already discussed some related work in previous sections. In this section, we compare some specific highly related research. Section 8.2 discusses research outside the RoboCup domain, while Section 8.1 focuses on related research within the RoboCup domain itself.

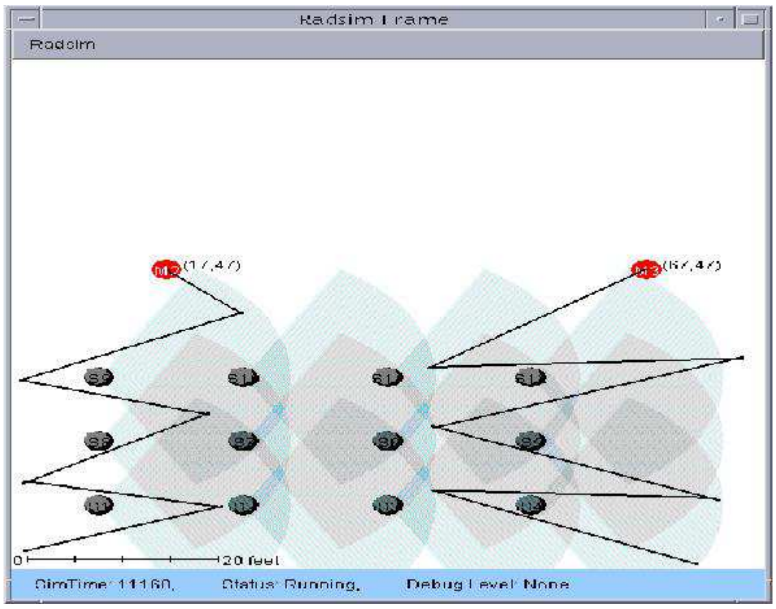


Figure 21. Sensor Agents collaborating to track moving targets, shown in red.

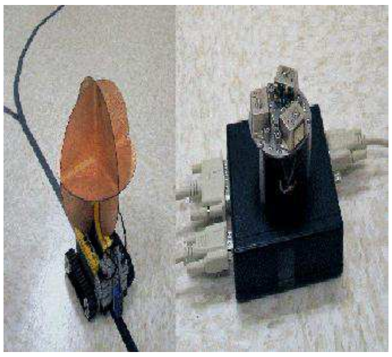


Figure 22. Left: Moving target being sensed; Right: Hardware Sensor

8.1. RoboCup-specific Related Work

In this section, we discuss some of the related work conducted within the RoboCup community. A novelty of ISAAC here is that it is not confined to RoboCup, but is intended to be more general-purpose, and indeed it has been applied to other domains, such as the evacuation rehearsal domain discussed in Section 7.

André et al have developed an automatic commentator system for RoboCup games, called ROCCO, to generate TV-style live reports for matches of the simulator league [3]. ROCCO attempts to recognize events occurring in the domain in real time, and generates correspond-

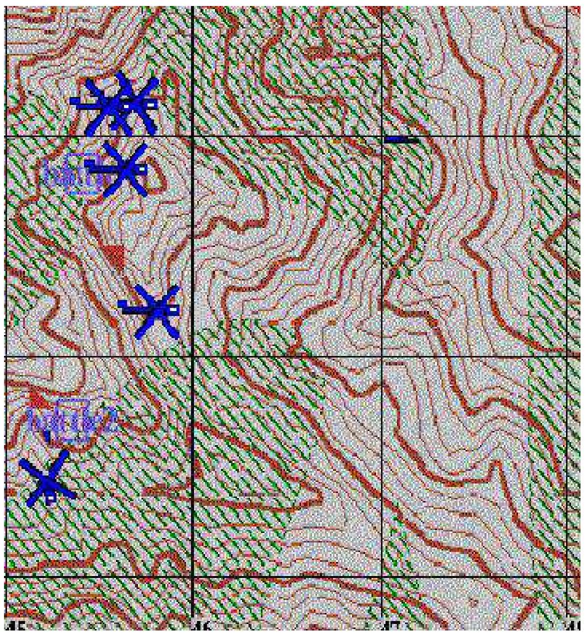


Figure 23. Helicopter agents in a battlefield simulation.

ing speech output. While both ROCCO and ISAAC use multimedia presentations, ROCCO attempts to analyze events quickly to produce live reports. However, the ROCCO analysis does not use multiple models of behavior for multi-perspective analysis as in ISAAC, and its analysis is not designed to help users and developers understand teams' abilities. ROCCO also has no capability to perform perturbation analysis.

Similarly, Tanaka-Ishii et al developed a commentator system, MIKE [39] that presents real-time analysis of a RoboCup soccer game, highlighting the statistics relevant to the game as they happen. ISAAC's global team model provides a summary of the game to explain the key aspects of the game responsible for the outcome, while MIKE looks at games as they happen. Unlike ISAAC's global model, MIKE's focus is on determining what features to tell the human audience during the course of the game. ISAAC also analyzes individual actions and subteam interactions.

Among other related work, Tomoichi Takahashi attempted to evaluate teams based on the collaboration among agents and not just from scores [41]. Some of the key differences between ISAAC and Takahashi's work are: (i) ISAAC's presentation of its analysis, for instance, using techniques such as natural language generation; (ii) ISAAC's perturbation analysis; (iii) ISAAC's capability to compare teams.

The designers of the KULRot RoboCup team used inductive logic programming to validate the programming of their multi-agent team [11]. Thus, they knew what they were looking for in their agents and could incorporate all the background knowledge that the agents were using. In contrast, ISAAC is knowledge-lean, but has more different types of analysis and presentation techniques and is capable of analyzing any RoboCup team.

Stone and Veloso have also used a decision tree to control some aspects of agents throughout an entire game, also using RoboCup as their domain [32]. However, this work pertains to execution of agents rather than analysis of agent teams, and since it is internal to the agent, their work has no means of presentation.

There is of course great general interest in analysis of (human) Soccer games, e.g., see [30]. Human soccer games are much more complex compared to RoboCup and currently require complex analysis including factors such as personalities of star players, any injuries they may have sustained etc., that are beyond the scope of RoboCup. Nonetheless, we expect that techniques for analysis developed for RoboCup soccer games would eventually apply to analysis of human soccer games (and other games as well). One key hindrance at present is the unavailability of raw data in a machine-readable form for analysis tools such as ISAAC.

8.2. General Related Work

Bhandari et al's Advanced Scout uses data mining techniques on NBA basketball games to help coaches find interesting patterns in their players and opponents' behaviors [5]. Advanced Scout also enables coaches to review the relevant footage of the games. Advanced Scout is able to capture statistical anomalies of which coaches can take advantage. However, Advanced Scout does not have some of ISAAC's extensions including the use of multiple models to analyze different aspects of teams, perturbations to make recommendations, and game summaries for an analysis of overall team performance.

Ndumu et al's system for visualization and debugging multi-agent systems comprises a suite of tools, with each tool providing a different perspective of the application being visualized [21]. However, the tools do not perform any in-depth analysis on the multi-agent system, and the system has no capability for perturbing this analysis. ISAAC also uses a visualization component, but only as an aid to understanding its analysis.

Johnson's Debrief system enables agents to explain and justify their actions [15]. This work focuses on agents' understanding the rationales for the decisions they make and being able to recall the situation. Debrief also has a capability for agent experimentation to determine what alternatives might have been chosen had the situation been slightly different. ISAAC performs something similar in its perturbation analysis however ISAAC focuses on an entire team, not just an individual, necessarily.

Marsella and Johnson's PROBES [18] is a system for assessing both agent and human teams in multi-agent simulated training environments for battle-tank crews. In contrast to ISAAC, PROBES was designed to assist instructors controlling the exercise as opposed to the developers of the agents. Further, it approached the analysis problem top-down, in contrast ISAAC's bottom-up approach. That is, it had pre-built expectations about a trainee team's acceptable behaviors in particular training situations (encoded in what are called "situation-space models"). It matched these expectations against observed behaviors of the trainee crews to locate problematic trainee team behaviors and provide appropriate feedback.

SEEK (and its progeny, SEEK2) is an approach to knowledge base refinement, an important aspect of knowledge acquisition [23]. Knowledge base refinement is characterized by the addition, deletion, and alteration of rule-components in an existing knowledge base, in an attempt to improve an expert system's performance. While the alteration of a rule may seem comparable to ISAAC's perturbation analysis, the goals are varied. The refinement done by systems such as SEEK are used to increase performance of the system to correctly classify future cases. ISAAC's goal is not that of increased performance in terms of cases classified but that of increased understandability of the causes of team failure or success. By looking at the changes to the automatically produced rules, the user gains insight as to the effects of each component of a rule.

9. Summary and Future Work

Multi-agent teamwork is a critical capability in a large number of applications including training, education, entertainment, design, and robotics. The complex interactions of agents in a team with their teammates as well as with other agents make it extremely difficult for human developers to understand and analyze agent-team behavior. It is thus increasingly critical to build automated assistants to aid human developers in analyzing agent team behaviors. However, the problem of automated team analysts is largely unaddressed in previous work. It is thus important to understand the key principles that underlie the development of such team analysts.

In terms of these general principles, one key contribution of this article is an understanding of the the key constraints faced by team analysts:

1. It is necessary to have multiple models of team behavior at different levels of granularity. In particular three different useful granularities appear to be: (i) individual agent's critical actions, (ii) subteam interactions and (iii) global team behavior trends.

2. Multiple presentation techniques are necessary each suited to the model being presented.

3. A bottom-up data-intensive approach is particularly critical to team analyses in complex domains where causal models are difficult to understand, input, etc. Such bottom-up analysis should also be independent of underlying team architecture and implementation to improve generality.

A second key contribution is presentation of general techniques to build team analysts that satisfy these constraints. First, ISAAC, our team analyst, uses multiple models of team behavior to analyze different granularities of agent actions: (i) Individual agent model to analyze critical actions of individual agents, (ii) Multiple agent model to analyze interactions within subteams that result in success or failure, (iii) Global team model to analyze an entire engagement. Second, ISAAC combines multiple presentation techniques to aid humans in understanding the analysis, where presentation techniques are tailored to the model at hand. Third, ISAAC uses learning techniques to build models bottom-up from data traces, enabling the analysis of differing aspects of team behavior. It is thus capable of discovering unexpected team behaviors.

A third key contribution are the techniques to provide feedback about team improvement: (i) ISAAC supports perturbations of models, enabling users to engage in "what-if" reasoning about the agents and providing suggestions for improvements in agents' actions that are already within the agents' skill set; (ii) It allows the user to compare different teams based on the patterns of their interactions.

ISAAC is available on the web for remote use at *http://coach.isi.edu.* It has found surprising results from top teams of previous tournaments and was used extensively at the RoboCup-99 tournament. ISAAC was awarded the "Scientific Challenge Award" at RoboCup-99 where its analysis and natural language game summaries drew a crowd throughout the tournament.

Although ISAAC was initially applied in the context of the RoboCup soccer simulation, it is intended for application in a variety of agent team domains. We demonstrated the generality of ISAAC's techniques by applying them to a team of agents involved in the simulated evacuation of civilians trapped in hostile enemy territory. Given the promising results in applying ISAAC's techniques to domains beyond RoboCup, we hope and expect to apply ISAAC to other domains, mentioned in section 7, more thoroughly.

Another direction of future work is to make the multimedia presentation more intelligent by exploiting standard reference models for intelligent multimedia presentations such as [6]. It would be interesting to use this model to decide on the layout and media-allocation for the data that should be conveyed to the human user.

Acknowledgements

This research was made possible by a generous gift from Intel Corporation. We would like to thank Bill Swartout for interesting discussions related to ISAAC, and support of this work at USC/ISI. We also thank several members of the RoboCup community for their valuable feedback about ISAAC, including Peter Stone, Itsuki Noda and Paul Scerri. Finally, we would like to thank the reviewers for their excellent comments for improving this article.

APPENDIX A: Features used in Individual Agent Model for RoboCup Soccer

The following are the eight features used in the Individual Agent Model for RoboCup Soccer:

1. *Ball Velocity:* Velocity with which the ball was kicked.
2. *Distance to goal:* Distance of kicker to goal.
3. *Number of Defenders:* Number of defenders between kicker and goal.
4. *Extrapolated Goal Line Position:* Aim of kick, i.e. Distance of the center of goal to the point where the ball would have crossed the end-line if not intercepted.
5. *Distance of Closest Defender:* Distance of kicker to the closest defender.

6. *Angle of Closest Defender wrt to Center of Goal:* Angle of closest defender's position with respect to the line connecting the centers of the two goals.

7. *Angle from Center of Field:* Angle of kicker's position with respect to the line connecting the centers of the two goals.

8. *Angle of Defender from shot:* Angle of closest defender's position with respect to the direction of kick.

APPENDIX B: Features used in Global Team Model for RoboCup Soccer

The following are the twelve features used in the Global Team Model for RoboCup Soccer:

1. *Possession time:* Percentage of time that the ball was in the first team's possession.

2. *Ball in Opponent half:* Percentage of time ball was in the opposition team's side of the field.

3. *Successful off-side traps:* Number of successful off-side traps laid by first team.

4. *Caught in off-side trap:* Number of times the opposition's off-side traps succeeded.

5. *Defense was bypassed:* Number of times the first team's last defender was bypassed.

6. *Bypassed opponent defense:* Number of times the opposition team's last defender was bypassed.

7. *Distance from sideline*: Average distance of ball from the sideline when ball was in first team's possession.

8. *Distance from sideline for opponent:* Average distance of ball from the sideline when ball was in opposition team's possession.

9. *Distance kept between own players:* Average distance kept between players of the first team.

10. *Distance kept between opponent players:* Average distance kept between players of the opposition team.

11. *Distance of last defender:* Average distance of the first team's last defender from the goal.

12. *Distance of opponent last defender:* Average distance of the opposition team's last defender from the goal.

APPENDIX C: Data for Figures 14 and 15

Tables I, II and III show the data used to generate Figures 14, 15 and 16.

Table I. Data for figure 14

Team	No. of non-scoring Rules	No. of scoring rules	Total no. of rules
Andhill	10	6	16
AT_Humboldt	8	4	12
CAT_Finland	4	4	8
CMUnited	6	2	8
ISIS	7	6	13
Gemini	6	6	12
RollingBrains	8	2	10
WindmillWanderers	6	2	8

Table II. Data for figure 15

Team	Avg. no. of conditions in a rule	Max. conditions in a rule	Min. conditions in a rule
Andhill	2.625	6	1
AT_Humboldt	2.75	5	2
CAT_Finland	2.375	4	1
CMUnited	2.125	5	1
ISIS	2.692307692	5	1
Gemini	2.833333333	4	1
RollingBrains	1.9	7	1
WindmillWanderers	1.625	2	1
Average	2.436781609	4.75	1.125

Table III. Data for figure 16

Team	Andhill	AT_Humb.	CAT_Fin.	CMUnited	ISIS	Gemini	R_Brains	W_Wanderers
Andhill	0	21.4934	13.6993	67.2912	14.8278	12.1384	21.0962	105.8628
AT_Humb.	21.4934	0	26.7541	72.1392	46.7939	10.1018	35.5333	15.3181
CAT_Fin.	13.6993	26.7541	0	50.5455	12.5778	39.5714	70.6	115.2244
CMUnited	67.2912	72.1392	50.5455	0	106.5482	84.5932	62.8619	18.4048
ISIS	14.8278	46.7939	12.5778	106.5482	0	41.0625	15.6448	185.6138
Gemini	12.1384	10.1018	39.5714	84.5932	41.0625	0	70.2857	71.3586
R_Brains	21.0962	35.5333	70.6	62.8619	15.6448	70.2857	0	54.0238
W_Wanderers	105.8628	15.3181	115.2244	18.4048	185.6138	71.3586	54.0238	0

APPENDIX D: Example Messages Exchanged in Evacuation Domain

Figure 24 shows two examples of messages exchanged between agents in the evacuation domain described in section 7. In example 1, a helicopter

```
1)Log Message Received; Fri Sep 17 13:37:03 1999:
Logging Agent: TEAM_trans2
Message==>
tell
:content TEAM_trans2 establish-commitment evacuate 41 kqml_string
:receiver TEAM-EVAC 9 kqml_word
:reply-with nil 3 kqml_word
:team TEAM-EVAC 9 kqml_word
:sender TEAM_trans2 11 kqml_word
:kqml-msg-id 22151+sangat.isi.edu+7 22 kqml_word

2)Log Message Received; Fri Sep 17 13:37:14 1999:
Logging Agent: teamquickset
Message==>
tell
:content teamquickset terminate-jpg constant determine-number-of-helos
number-of-helos-determined *yes* 4 4 98 kqml_string
:receiver TEAM-EVAC 9 kqml_word
:reply-with nil 3 kqml_word
:team TEAM-EVAC 9 kqml_word
:sender teamquickset 12 kqml_word
:kqml-msg-id 19476+tsevet.isi.edu+7 22 kqml_word
```

Figure 24. Examples of messages exchanged between agents in the evacuation domain

agent, *TEAM_trans2*, sends a message to the team, *TEAM-EVAC*, asking its members to initiate a joint goal to execute the *evacuate* plan. Example 2 is a message sent by another agent *teamquickset* to the entire team *TEAM-EVAC* to let the members know that the joint goal to execute sub-plan *determine-number-of-helos* ended successfully. As can be seen these messages are fairly complex and the relevant features of each message needs to be abstracted out so that analysis can be easily done.

References

1. Aho, A., Hopcroft, J., Ullman, J.: Data Structures and Algorithms. Addison-Wesley, 1983.
2. Andou, T.: Personal Communication. 1998.
3. André, E., Herzog, G., Rist, T.: Generating Multimedia Presentations for RoboCup Soccer Games. RoboCup-97: Robot Soccer World Cup I, 1997.
4. Balch, T.: The Impact of Diversity on Performance in Multi-robot Foraging. Proceedings of the Third Annual Conference on Autonomous Agents, 1999.
5. Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratap, R., Ramanujam, K.: Advanced Scout: Data Mining and Knowledge Discovery in NBA Data. Data Mining and Knowledge Discovery, 1997.
6. Bordegoni, M., Faconti, G., Maybury, M. T., Rist, R., Ruggieri, S., Trahanias, P. and Wilson, M.: A Standard Refernece Model for Intelligent Multimdeia Presentation Systems. Journal of Computer Standards and Interfaces, pp. 477-496, 1997.
7. Caruana, R., Freitag, D.: Greedy Attribute Selection. 11th Proceedings of the 11th International Conference on Machine Learning (ICML), 1994.
8. Chiu, B. and Webb, G.: Using Decision Trees for Agent Modeling. Improving Prediction Performance, User Modeling and User-Adapted Interaction, Kluwer Academic Publisher Group, "Dordrecht, Netherlands", 8, pp. 131-152, 1988.
9. Crutchfield, J. P. and Young, K.: Inferring Statistical Complexity. Physical Review Letters,63(2):105-108, 1989.
10. Dorais, G., Bonasso, R., Kortenkamp, D., Pell, B., Schreckenghost, D.: Adjustable Autonomy for Human-Centered Autonomous Systems. Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems, 1999
11. Driessens, K., Jacobs, N., Cossement, N. Monsieur, P., DeRaedt, L.: Inductive Verification and Validation of the KULRot RoboCup Team. Proceedings of the Second RoboCup Workshop, 1998.
12. Goldberg, D. and Matarić: Coordinating Mobile Robot Group Behavior Using a Model of Interaction Dynamics. Proceedings of The Third International Conference on Autonomous Agents, 1999.
13. Jennings, N.: Controlling Cooperative Problem Solving in Industrial Multi-agent System Using Joint Intentions. Artificial Intelligence, Vol. 75, 1995.
14. 3D images from RoboCup soccer. http://www.techfak.uni-bielefeld.de/techfak/ags/wbski/3Drobocup/3DrobocupPics.html, 1999.

15. Johnson, W. L.: Agents that Learn to Explain Themselves. Proceedings of AAAI-94, 1994.
16. Kaminka, G.: Execution Monitoring in Multi-Agent Environments. Ph.D. Dissertation, University of Southern California, Computer Science Department, 2000.
17. Kitano, H., Tambe, M., Stone, P., Veloso, M., Noda, I., Osawa, E. and Asada, M.: The RoboCup synthetic agent's challenge. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.
18. Marsella, S.C. and Johnson, W.L.: An instructor's assistant for team training in dynamic multi-agent virtual worlds. Goettl, Halff, Redfield and Shute (eds) Intelligent Tutoring Systems, Proceedings of the 4th International Conference on Intelligent Tutoring Systems, Springer, pp 464-473, 1998.
19. Modi, J., Jung, H., Tambe, M., Shen, W., Kulkarni, S.: Dynamic Distributed Resource Allocation: A Distributed Constraint Satisfaction Approach. Intelligent Agents VIII Proceedings of the International workshop on Agents, theories, architectures and languages (ATAL'01)
20. Myritz, H.: Personal Communication. 1999.
21. Ndumu, D., Nwana, H., Lee, L., Haynes, H.: Visualization and debugging of distributed multi-agent systems. Applied Artificial Intelligence Journal, Vol 13 (1), 1999.
22. Noda, I., Matsubara, H., Hiraki, K. and Frank, I.: Soccer Server: A Tool for Research on Multi-agent Systems. Applied Artificial Intelligence, Volume 12, Number 2-3, 1998.
23. Politakis, P. and Weiss, S.: Using Empirical Analysis to Refine Expert System Knowledge Bases. Artificial Intelligence(22), pp 23-48, 1984.
24. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, 1994.
25. Raines, T., Tambe, M., Marsella, M.: Automated Agents that Help Humans Understand Agent Team Behaviors. Proceedings of the International Conference on Autonomous Agents (Agents), 2000.
26. Raines, T., Tambe, M., Marsella, M.: Agent Assistants for Team Analysis. AI Magazine, Vol 21, Num 3, pp 27-31, Fall 2000.
27. Reiter, E.: Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? Proceedings of the Seventh International Workshop on Natural Language Generation, 1994
28. Ron, D., Singer, Y., Tishby, N.: On the Learnability and Usage of Acyclic Probabilistic Finite Automata. Journal of Computer and System Sciences 56(2), 1998.
29. Sengers, P.: Designing Comprehensible Agents. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1999.
30. SoftSport Inc.: http://www.softsport.com.
31. Stolcke, A. and Omohundro, S.M.: Hidden Markov Model Induction by Bayesian Model Merging. Advances in Neural Information Processing Systems, Volume 5, S. J. Hanson, J. D. Cowan and C. L. Giles, editors, Morgan Kaufman, pp. 11-18, 1992.
32. Stone, P., Veloso, M.: Using Decision Tree Confidence Factors for Multi-agent Control. Proceedings of the International Conference on Autonomous Agents, 1998.
33. Stone, P.: Personal Communication. 1999.
34. Sycara, K., Decker, K., Pannu, A., Williamson, M., Zeng, D.: Distributed Intelligent Agents. IEEE Expert, 1996.

35. Tambe, M. Johnson, W. L., Jones, R., Koss, F., Laird, J. E., Rosenbloom, P.S., Schwamb, K.: Intelligent Agents for Interactive Simulation Environments. AI Magazine, 16(1) (Spring), 1995.
36. Tambe, M.: Towards Flexible Teamwork. Journal of Artificial Intelligence Research, Vol. 7, 1997.
37. Tambe, M. and Jung, H.: The benefits of arguing in a team. AI Magazine Vol20, Num 4, Winter, 1999.
38. Tambe, M., Pynadath, D. and Chauvat, N.: Building dynamic agent organizations in cyberspace. IEEE Internet Computing Volume 4, Number 2, 2000.
39. Tanaka-Ishii, K., Noda, I., Frank, I., Nakashima, H., Hasida, K., Matsubara, H.: MIKE: An Automatic Commentary System for Soccer. International Conference on Multi-Agent Systems, 1998.
40. Ting, K.: Inducing Cost-Sensitive Trees via Instance Weighting. Principles of Data Mining and Knowledge Discovery (PKDD 98), 1998.
41. Tomoichi, T.: LogMonitor: from player's action analysis to collaboration analysis and advice on formation. Proceedings of the Third RoboCup Workshop, 1999.
42. Toulmin, S.: The uses of argument. Cambridge University Press, 1958.