Analysis and Cleaning of User Traces Through Comparison of Multiple Traces

Michael W. Floyd and **Babak Esfandiari**
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada

Abstract

Traces of user behaviour can be a valuable source of knowledge that can be used during case-based reasoning. This paper presents an approach for analyzing and cleaning user traces. The analysis looks to identify three properties in traces: reasoning with an internal state, non-deterministic behaviour and error. The existence of any of these properties may influence how a system should reason or store knowledge in cases. Initially, each trace is examined to see areas that might contain one of the three properties. Multiple versions of the trace are then generated in order to determine which specific property is present. The analysis is applied to traces generated by observing both a computer and human controller for an obstacle avoidance robot. The results demonstrate that the analysis is able to successfully identify which properties are present and clean many of the errors that exist in the traces.

1 Introduction

Learning by observation allows a software agent to learn a behaviour, or information about the behaviour, by watching an expert perform the behaviour. As the agent watches the expert, it is able to observe and record the inputs received by the expert and the outputs produced by the expert. The record of the expert's behaviour, called a *trace*, can contain valuable information that allows the agent to gain insight into the task the expert is performing or to perform the task itself.

User traces are used as knowledge sources in a variety of learning by observation systems (Coates, Abbeel, and Ng 2008; Gillespie et al. 2010; Ontañón and Ram 2011; Romdhane and Lamontagne 2008; Rubin and Watson 2010). However, most existing work assumes that the recorded traces accurately capture the behaviour of the expert. The traces could be incorrectly recorded if the learning agent made an observation error or if the expert performed an error of manipulation or reasoning and produced an erroneous output. Analyzing the generated traces in order to verify and validate their quality becomes an important consideration since it can directly influence the agent's ability to learn. In addition to analyzing the traces for errors, they can also be examined to get insight into the behaviour the expert has demonstrated. This can include information about how the expert reasons or how much information from the trace is used during reasoning.

This paper will examine how the run of an expert can be analyzed in order to gain information about the expert and the quality of the observations. In Section 2, we define the format of an expert's trace and how they can be stored as cases. A technique for analyzing and cleaning the traces will be presented in Section 3 and evaluated in Section 4. Related work regarding learning by observation and expert traces will be discussed in Section 5 followed by concluding remarks in Section 6.

2 Expert Traces

As an agent observes an expert performing a behaviour, it is able to see the *input* the expert receives and the *output* the expert produces. For an expert that is situated in an environment, the input would correspond to the sensory information that is received and the output would be the actions that are performed. Each time the expert interacts with the environment, by performing an action A_t in response to a sensory input S_t, the observing agent can record this interaction I_t as a pair ($I_t = \langle S_t, A_t \rangle$).

Over time, the expert will receive many sensory inputs and perform many actions. The entire sequence of sensory inputs and actions is represented by the expert's *run* R (Wooldridge 2002):

$$R : S_0 \overset{A_0}{\rightarrow} S_1 \overset{A_1}{\rightarrow} S_2 \overset{A_2}{\rightarrow} \ldots S_u \overset{A_u}{\rightarrow} S_{u+1}$$

As the agent observes and records the expert's run, it will generate a *trace* T of the expert's run that contains a temporally linked series of observed interactions:

$$T : I_0 \rightarrow I_1 \rightarrow I_2 \rightarrow \cdots \rightarrow I_u$$

When storing the expert's trace in cases, it would be possible for each observed interaction to be a case. However, this would result in the current sensory input being the *problem* portion of the case and the action as the *solution* portion. This would be acceptable if the expert selected an action to perform based solely on its current sensory input, but if the expert uses information from past sensory inputs or actions it performed then that case definition would not model the

problem correctly. In order to allow past sensory inputs and actions to also be contained in the problem portion of a case, the entire run of the expert is contained in the case:

$$C_t = \langle R_t, A_t \rangle$$

Looking at the definition of a run at time t, we can see it is composed of the run from the previous point in time, R_{t-1}, along with the action performed in response to R_{t-1}, A_{t-1}, and the current environment state S_t:

$$R_t : R_{t-1} \xrightarrow{A_{t-1}} S_t$$

Each case, which is composed of a run and the associated action can be rewritten as a tuple containing the current environment state, the action associated with the current run, the previous run and the action of the previous run ($C_t = \langle R_{t-1}, A_{t-1}, S_t, A_t \rangle$). This can be further simplified as:

$$C_t = \langle C_{t-1}, S_t, A_t \rangle$$

This simplification is beneficial because each case no longer needs to store the entire run but can instead store the most recent environment state and a link to the previous case.

3 Trace Analysis

We wish to detect and analyze situations in a trace T where the same sensory input (S_1) results in different actions (A_1 and A_2) at different times ($\exists A_1, A_2 \in \mathcal{A} \wedge \exists S_1 \in \mathcal{S}$ such that $\langle S_1, A_1 \rangle \in T \wedge \langle S_1, A_2 \rangle \in T$ and $A_1 \neq A_2$). The likely causes for these situations are:

- **Reasoning with an Internal State:** If the expert reasons with an internal state, that state will also influence the action it performs. Since the internal state is not directly observable by the learning agent, it can make two interactions seem the same when really they are not. This is because the observable features, the sensory inputs, may be the same but the unobservable features, the internal states, may be different and result in a different action being performed.
- **Non-deterministic Behaviour:** In some situations, the expert might not use any external information or its internal state to select an action to perform but instead just randomly perform an action. The randomness of the expert during reasoning will result in similar sensory inputs resulting in different actions. In this situation, it is necessary to identify that the behaviour is random and not due to an undetectable external input or internal state.
- **Error:** Error can come in two forms: *expert error* and *observation error*. The expert might occasionally make errors during reasoning and perform an incorrect action. We say that an action was incorrect if, given the opportunity, the expert would immediately change the action it performed (without getting any feedback about the success or failure of the action). This type of error might occur when observing human experts due to fatigue, lapses in concentration, or other factors. Observation errors can occur if the observations, either of the expert's inputs or outputs, are noisy or contain errors. This could occur if the observer senses the environment differently than the expert (if they had a different quality of sensors) or made an error recording the correct observations.

For an expert that does not use an internal state, a purely reactive expert, each sensory input will result in a single action being performed:

$$Expert_{reactive} : \mathcal{S} \rightarrow \mathcal{A}$$

However, an expert that is state-based or performs non-deterministic behaviour will not always perform the same action for a sensory input but will instead perform one action from a set of possible actions based on the probability distributions for the actions (where each member of $2^{\mathcal{A}}$ is a set of probabilities, one for each action, that sum to 1.0):

$$\begin{aligned} Expert_{state} &: \mathcal{S} \rightarrow 2^{\mathcal{A}} \\ Expert_{non-deterministic} &: \mathcal{S} \rightarrow 2^{\mathcal{A}} \end{aligned}$$

An expert that uses an internal state will need its entire past run, not just the most recent sensory input, in order to consistently select a single action to perform from the set of possible actions:

$$Expert_{state} : \mathcal{R} \rightarrow \mathcal{A}$$

For an expert that displays non-deterministic behaviour, even when presented with its entire run it will not always select a single action to perform:

$$Expert_{non-deterministic} : \mathcal{R} \rightarrow 2^{\mathcal{A}}$$

The traces of experts with state-based or non-deterministic behaviour will likely contain interactions that have the same sensory input but different actions. Even if the expert is reactive, errors can cause interactions with similar sensory inputs to have different actions.

Differentiating between the three properties might not be obvious when only examining a single trace of the expert's behaviour. For example, it would not be possible to tell with any degree of certainty that the expert performed an erroneous action. What was an expert error could also appear to be evidence of state-based or non-deterministic behaviour. In order to help differentiate between the three properties, we will examine multiple traces that were derived from the original.

Two levels of analysis are performed: *single trace analysis* and *multi trace analysis*. Initially, we perform single trace analysis in order to determine if a trace is a candidate for multi trace analysis. Ideally, we would like to perform multi trace analysis on every user trace that is generated. That may not be possible since, as we will see later, multi trace analysis requires the expert to solve additional problems. If the amount of time the expert is available is limited, we would only want to make use of the expert when it is clearly necessary. Additionally, it may be impossible to recreate the exact conditions of the original trace so, in some circumstances, multi trace analysis might not be feasible. In such a case, we can only get a general idea about if any of the three properties exist in the trace but no precise information about which specific properties exist.

Single Trace Analysis

Single trace analysis is used to identify traces that might have any of the three properties. Since, as we described earlier, each of the properties presents itself through interactions where similar sensory inputs results in different actions, we will look to measure how many of these differences occur in the trace. When looking for differences, each interaction in the trace is compared to all other interactions in the trace as well as all interactions that occur in cases that are already in the case base. Cases that are already in the case base are also used during single trace analysis because they contain knowledge collected during past observation sessions. These cases were likely collected as part of a previous user trace and then converted into cases and stored in the case base. When comparing two interactions, I_i and I_j, they are marked as *noteworthy* if they have sensory inputs that are sufficiently similar ($sim(S_i, S_j) > \tau$, where τ is a threshold used to determine if two sensory inputs are sufficiently similar) and they have different actions ($A_i \neq A_j$). If there are N interactions in the trace and n of those interactions are noteworthy, we calculate the ratio of noteworthy interactions ($\frac{n}{N}$). If this ratio is greater than zero, it indicates that at least one of the three properties seems to exist in the trace. However, even if the ratio is zero there could still be occurrences of the properties but there were not enough other interactions (in both the trace and the case base) to identify them.

Multi Trace Analysis

In multi trace analysis, new traces are generated by having the expert replay the sensory inputs it received during the initial trace. If the initial trace contained N interactions between the expert and the environment, the N sensory inputs will be given to the expert, in their original order of occurrence, for the expert to reason with again. Each time the expert replays the initial trace, the entire run of the expert will be recorded as a new trace. Traces can be different because, while they have the same sensory inputs, the expert may perform different actions in each trace. The use of multi trace analysis makes the assumptions that the expert will be available to generate the new traces and that the agent can present the problems to the expert in a similar manner to how the expert receives inputs from the environment. These would be reasonable assumptions if the expert is willing to help the agent learn (and assumes the role of a tutor or teacher) or can be presented with problems to solve without being aware it is helping the agent (Floyd and Esfandiari 2009). However, if the expert was not willing to complete the replay of a trace and ended the process early, then only part of the traces can be compared (the parts that were complete in all traces).

The primary objective of multi trace analysis is to identify the differences, if any, that exist between the initial trace and all replay traces. If the initial trace was used to generate $M - 1$ additional traces, there will be M total traces each containing N interactions. Each of the n noteworthy interactions from the original trace will be grouped together with the corresponding interactions from the other generated traces, resulting in n groups of interactions each containing M interactions. For example, if the 3rd interaction from the original trace was deemed noteworthy, it will be grouped together with the 3rd interactions from each of the other $M-1$ traces.

For each grouping of noteworthy interactions, we calculate the *agreement ratio* AR_i:

$$AR_i = \frac{s_i}{M}$$

where s_i is the number of interactions in the ith grouping that had the same action as the interaction from the initial trace. The agreement ratio can then be used to label each of the groupings:

$$label_i = \begin{cases} state-based & ,if \quad AR_i \geq \alpha \\ non-deterministic & ,if \quad \beta < AR_i < \alpha \\ error & ,if \quad AR_i \leq \beta \end{cases}$$

Interactions of a state-based expert are identified as noteworthy since a single sensory input can lead to multiple actions. However, during replay of those traces the expert will generally perform a single action since it is reasoning with its entire run. This will result in an agreement ratio that is greater than α for noteworthy interactions that are a result of internal state. If there was an interaction that only occasionally (less than a threshold β) resulted in one type of action being performed then it is labelled as being an error. This assumes that the expert generally performs the correct action in the traces and only rarely performs the erroneous action. A noteworthy interaction will be labelled as non-deterministic if it results in several different frequently occurring actions during replay (an agreement ratio between α and β). This assumes that since there was no clear correct action that the expert selected the action in a non-deterministic manner. One limitation of this labelling approach is when the expert selects an action in a non-deterministic way but the probability of selecting the action is low (less than β) or high (greater than α). In these situations, the interactions will incorrectly be labelled as error or state-based instead of non-deterministic. This problem can be minimized by selecting an appropriate value of α and β but can not be completely eliminated.

For any interactions that are labelled as having an error in the original trace, those interactions can be cleaned by replacing them with the correct versions of the interactions from one of the other traces. The correct version of an interaction is selected by finding the most common action from the interactions in a grouping and using one of the interactions with that action (all interactions with the same action will be identical since they will always have the same sensory input). If the analysis was performed a second time, the number of noteworthy interactions due to error should ideally decrease while the number of noteworthy interactions due to state-based or non-deterministic behaviour should remain (although some may be removed if they were only labelled as noteworthy due to the erroneous interactions).

An example of trace analysis is given in Figure 1. Initially, only the original trace, which contains five interactions, is available to the learning agent (the top row in the figure). Single trace analysis is then performed and four noteworthy

interactions are found (indicated with arrows). The first and fifth interactions are noteworthy because they have the same sensory input (S_a) but different actions (A_a and A_c). Similarly, the second and fourth interactions had the same sensory input (S_b) but different actions (A_b and A_a). In order to perform multi trace analysis, the expert is made to encounter the same sequence of sensory inputs, contained in the original trace, to generate additional traces. In this example, two additional traces are generated (the second and third rows in the figure).

Four groupings, one for each noteworthy interaction, are then compiled. Each grouping contains one interaction from each trace and all interactions in a group occur at the same position in their original trace (and therefore have identical sensory inputs). Using the threshold approach we described previously, each grouping can be labelled. Since the groupings that contain the first, second and fourth interactions all have the same actions (A_a, A_b and A_a respectively) they are labelled as being a result of internal state. However, the grouping of the fifth interactions do not all have the same actions. The interaction in the original trace has one action (A_c) whereas the others have a different action (A_a) so this grouping is labelled as an error. A cleaned version of the original trace can now be generated (the final row in the figure). Since there was only one noteworthy interaction labelled as erroneous, the fifth interaction, only one interaction is cleaned in the final trace. The most common action, from the entire grouping, is used as the correct action for that interaction (A_a). It should be noted that the first interaction in the original trace, which was labelled as internal state, was labelled that way because of the erroneous interaction (the fifth interaction). If single trace analysis was performed again, on the cleaned trace, it would no longer appear to be noteworthy since the error would be fixed.

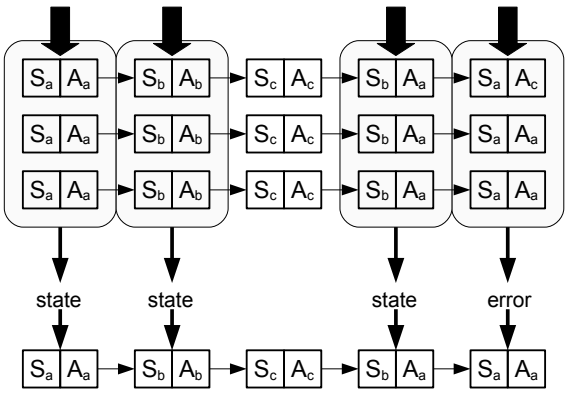

Figure 1: An example of multi trace analysis

After the trace analysis has been completed and any errors have been eliminated, the trace can be converted to cases (as described previously in Section 2) and added to the case base. These cases will now be of a sufficiently high quality that they can be used by the learning agent when it attempts to perform the behaviour it learnt from the expert or they can be used to identify noteworthy interactions in newly recorded traces. If a trace was determined to contain non-deterministic behaviour, the stored cases can be modified to contain probabilities for performing each type of action. Similarly, if the trace analysis showed the expert uses an internal state then a case retrieval algorithm that can infer internal state (Floyd and Esfandiari 2011) can be used.

4 Evaluation

In order to evaluate the trace analysis technique described previously, we will examine traces that have been generated by observing the control program for an obstacle avoidance robot. The control program senses the environment using values from a *touch sensor* and a *sonar sensor* ($S_{robot} = \langle f_{touch}, f_{sonar} \rangle$). The control program can then control the robot using one of five possible actions: move forward, move backward, turn left, turn right and reverse direction by turning 180 degrees ($\mathcal{A}_{robot} = \{Forward, Backward, Left, Right, Reverse\}$).

If the control program's touch sensor indicates it has come in contact with an obstacle (a f_{touch} value of 1), the robot will be moved backward. Otherwise, it will base its action selection on the value of the sonar sensor (f_{sonar}). If the sonar value is less than 2 the robot will reverse direction, if the value is between 2 and 3 the robot will turn, and if the value is greater than 3 it will move forward. The direction the robot turns, when the sonar value is between 2 and 3, will toggle. For example, if it previously turned left it will always turn right next. This requires the control program to have an internal state that represents its previous action. Additionally, the control program artificially adds errors by performing 2% of its actions incorrectly. This behaviour was selected for evaluation because, even though it has a small problem space (approximately 200 states if the sonar values are discretized), trace errors, non-deterministic behaviour or experts with internal states can make it difficult to learn from the control program (Floyd and Esfandiari 2011).

The control program was observed performing its obstacle avoidance behaviour so that a trace of $10,000$ interactions was generated. Additionally, an error free version of the trace was stored in order to evaluate the trace cleaning performance. For the analysis, the threshold for sensory inputs to be considered highly similar is $\tau = 0.9999$ (99.99% similarity) and labelling thresholds of $\alpha = 0.90$ and $\beta = 0.10$ were used. These thresholds were selected so that an erroneous interaction can have at most one other interaction in its grouping with the same action and a state-based interaction can have at most one interaction with a different action. In both situations, these thresholds allow for the possibility that replay traces will also contain some errors. During multi trace analysis, the initial trace had nine additional traces generated. The number of noteworthy interactions (coulmn NW in Table 1) that were identified along with the percentage of noteworthy interactions that were labelled as errors (column Error), non-deterministic behaviour (column ND) and internal state (column State) were measured. Additionally, the number of errors that existed in the original trace (column Initial Errors) and final trace after cleaning (column Final Errors) were also measured.

The results (Row *Error+State-1* in Table 1) show an 85.6% decrease in the number of errors in the trace. We can see that some interactions are correctly labelled as resulting from error and state, but some are labelled as resulting from non-deterministic behaviour. This is because during the generation of aditional traces those traces have errors as well. Some erroneous interactions have other interactions in their grouping that were also erroneous so the agreement ratio indicates that the interaction was a result of non-determinism.

Another area of note is that there appears to be more noteworthy interactions that result from internal state than are actually present in the trace. This is because the erroneous interactions result in a number of correct interactions being identified as noteworthy. The correct interactions seem to be a result of an internal state because the agent continues to respond the same way to them in each of the generated traces. However, if these errors are removed, the correct interactions are no longer identified as noteworthy. This is demonstrated when two additional rounds of trace analysis are performed (Row *Error+State-3* in Table 1). There are still some erroneous interactions in this trace, even after cleaning, but they can not be identified by the analysis because there were no similar interactions in the trace. It should be noted that we have only shown the results from further rounds of analysis for illustrative purposes, in practice the benefit of subsequent analysis is unlikely to outweigh the cost.

Human Expert

The previous evaluation of the trace analysis technique used a computer program as the expert and errors were artificially inserted. We look to extend our evaluation by observing a human expert. The human will likely perform errors by accident due to reasoning errors, fatigue, or pressing an incorrect button. The expert will attempt to control the robot in the same way as the control program and will be observed for 250 interactions. While a trace of the human expert is being generated, the control program will also select actions to perform in order to generate a correct trace of what the human should have done in each interaction.

Upon initial examination of the human trace, it is clear that the human expert made substantially more errors than the control program. The trace had an error rate of 16.8% (42 erroneous interactions out of 250) compared to the 2% error rate that was artificially added to other traces. The two primary sources of error were maintaining the internal state and switching to different actions. The human expert often forgot its internal state (which direction it had previously turned) so numerous errors were a result of turning the incorrect direction. The second source of error occured when the expert had been repeatedly performing one type of action, like moving forward, and continued doing that action even after it should have switched to another action, like turning. This is likely because the expert was quickly clicking one button to repeatedly perform the first action and was not able to react in time to the change in sensory inputs.

In the previous experiment, when identifying noteworthy interactions each interaction was only compared to other interactions in the trace. For this round of analysis, cases from an existing case base were also used to identify noteworthy interactions. This is done to show the value of the previously observed cases during trace analysis, especially when the newly generated trace is relatively short (only 250 interactions). The case base was built from the cleaned trace of the control program (*Error+State-1* trace in Table 1). Other than using the case base during analysis, all other parameters remained the same (9 additional traces generated, τ = 0.9999, α = 0.90 and β = 0.10).

The results show that if only the human expert trace is used to detect noteworthy interactions (row *Human*) then very few noteworthy interactions are detected. The cleaned trace removes only one error (a 2.4% decrease). This is what we might expect since the trace only contains 250 interactions so it is unlikely an erroneous interaction will be highly similar to another interaction in the trace. However, those results are improved when an existing case base is also used to detect noteworthy interactions (row *Human+Casebase*). The number of detected noteworthy interactions increases and, more importantly, the number of errors in the trace decreases (a decrease of 35.7%). While the decrease in errors is not as large as it was in the previous experiments, the results show that cleaning can reduce the number of errors even when the error rate is high (much higher than the artificially added 2% error rate and even higher than the β parameter). Even through the generated replay traces also had high levels of noise, since they too were generated by observing the human expert, the analysis was able to detect that the noteworthy intereactions were largely due to errors and internal state.

5 Related Work

Traces of expert behaviour are routinely used in learning by observation systems (Coates, Abbeel, and Ng 2008; Gillespie et al. 2010; Ontañón and Ram 2011; Romdhane and Lamontagne 2008; Rubin and Watson 2010). However, the majority of existing systems are design to learn from reactive experts so any error reduction does not take into account that what appears to be errors could also be non-determinism or reasoning with an internal state. Previous work has looked to minimize the impact of trace error by collecting multiple traces from a user (Coates, Abbeel, and Ng 2008). However, that work does not look to identify or remove the errors but instead relies on errors being relatively rare so their impact is minimized. Learning by observation has been used to learn non-deterministic behaviour (Gillespie et al. 2010) by using stochastic policies to select actions. This system does not take into account that the expert might have an internal state so stateful behaviour would be learnt incorrectly. An analysis technique, like the one presented in this paper, would be useful for determining when such a system could successfully be used.

The trace cleaning performed by our approach is a form of case-base maintenance that can be classified as a noise reduction algorithm (Cummins and Bridge 2011). Other noise reduction algorithms (Delany and Cunningham 2004; Tomek 1976) are not directly suitable for analyzing traces since they only examine an existing set of cases and would therefore be unable to differentiate error from the other properties. These algorithms would incorrectly classify reason-

	NW	Error	ND	State	Initial Errors	Final Errors
Error+State-1	1238	14.0%	0.6%	85.4%	202	29
Error+State-3	493	0%	0.2%	99.8%	24	24
Human	2	50%	0%	50%	42	41
Human+Casebase	27	55.6%	3.7%	40.7%	42	27

Table 1: The analysis results and quality of each trace after trace analysis

ing with an internal state or non-deterministic behaviour as error and, when cleaning the noise, actually introduce more error into the case base. Additionally, noise reduction is only one aspect of our analysis approach and reducing error is not the only goal of the analysis.

6 Conclusions

This paper has described an approach to analyze and clean traces of an expert's behaviour. The analysis identifies when a single sensory input can, at different times, result in different actions being performed. The expert is made to replay the original trace in order to generate several new versions of the trace and those traces are used to determine if the expert reasoned with an internal state, performed non-deterministic behaviour or performed any errors. The trace can then be cleaned in order to remove any detected errors.

The major assumptions of this approach are that the expert is available to generate new traces and that the agent is able to present the inputs in a realistic manner. If the expert is not available, this approach can not be used since it relies on the generated traces. If the agent does not present inputs to the expert in a way that is similar to how the environment presents them, the expert may behave differently which can compromise the quality of the generated traces. Both of these issues result in a significant limitation of our approach.

Our experiments demonstrated the applicability of the analysis in an obstacle avoidance domain. The results showed that the analysis was able to correctly detect which of the three properties were present in the trace and cleaning was able to remove many of the errors. All of the threshold values used during analysis (τ, α and β) were selected intuitively so future work will look to examine the effects of changing these values. Also, future work will examine alternative approaches for trace analysis that do not require the generation of new traces but are still able to differentiate between errors, non-determinism and stateful behaviour.

References

Coates, A.; Abbeel, P.; and Ng, A. Y. 2008. Learning for control from multiple demonstrations. In *25th International Conference on Machine Learning*, 144–151.

Cummins, L., and Bridge, D. G. 2011. On dataset complexity for case base maintenance. In *19th International Conference on Case-Based Reasoning*, 47–61.

Delany, S. J., and Cunningham, P. 2004. An analysis of case-base editing in a spam filtering system. In *7th European Conference on Case-Based Reasoning*, 128–141.

Floyd, M. W., and Esfandiari, B. 2009. An active approach to automatic case generation. In *8th International Conference on Case-Based Reasoning*, 150–164.

Floyd, M. W., and Esfandiari, B. 2011. Learning state-based behaviour using temporally related cases. In *16th United Kingdom Workshop on Case-Based Reasoning*, 34–45.

Gillespie, K.; Karneeb, J.; Lee-Urban, S.; and Muñoz-Avila, H. 2010. Imitating inscrutable enemies: Learning from stochastic policy observation, retrieval and reuse. In *18th International Conference on Case-Based Reasoning*, 126–140.

Ontañón, S., and Ram, A. 2011. Case-based reasoning and user-generated AI for real-time strategy games. In *Artificial Intelligence for Computer Games*. Springer-Verlag. 103–124.

Romdhane, H., and Lamontagne, L. 2008. Forgetting reinforced cases. In *9th European Conference on Case-Based Reasoning*, 474–486.

Rubin, J., and Watson, I. 2010. Similarity-based retrieval and solution re-use policies in the game of Texas Hold'em. In *18th International Conference on Case-Based Reasoning*, 465–479.

Tomek, I. 1976. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* 6(6):448–452.

Wooldridge, M. 2002. *An introduction to multiagent systems*. John Wiley and Sons.