Department of Computer Science

Integrating Statistical Methods for Characterizing Causal Influences on Planner Behavior over Time

Adele E. Howe, Robert St. Amant
and Paul R. Cohen

Technical Report CS-94-115

June 27, 1994

Colorado State University

Integrating Statistical Methods for Characterizing Causal Influences on Planner Behavior over Time *

Adele E. Howe
Computer Science Dept.
Colorado State University
Ft Collins, CO 80523
howe@cs.colostate.edu
303-491-7589

Robert St. Amant
Paul R. Cohen
Dept. of Computer Science
University of Massachusetts Amherst,
MA 01003
stamant@cs.umass.edu
cohen@cs.umass.edu
415-545-3638

Abstract

Given a complex planner and or environment, it can be difficult to determine why it behaves as it does. Statistical causal modeling techniques allow us to develop models of behavior, but they tend to be limited in what they can model: either continuing, repetitive influences or causal influences without cycles, but not both as appear in most planning environments. This paper describes how two statistical modelling techniques can be combined to suggest specific hypotheses about how the environment and the planner's design causally influence the planner's behavior over many examples of interacting in its environment and to construct models of those influences. One technique, dependency detection, is designed to identify relationships (dependencies) between particular failures, the methods that repair them and the occurrence of failures downstream. Another method, path analysis, builds causal models of correlational data. Dependency detection operates over a series of events, and path analysis models within a temporal snapshot. We explain the integration of the techniques and demonstrate it on data from the Phoenix planner.

*This research was supported by ARPA-AFOSR contract F30602-93-C-0100. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

1 Modelling Planner Behavior

After many years of research, the planning community has developed many algorithms, techniques and strategies for planning. Unfortunately, for those trying to build a planner for a specific environment, we do not know why and how our planning techniques work in particular environments. We do not have models of a planners operating in its environment, but rather, we need tools and techniques to help us construct them.

Because environments and planners do, these models will need to include continuous, numerical factors and discrete, categorical events. They will need to be developed, at least in part, from empirical data, and they will need to use data gathered over time as the planner operates in its environment.

Our models are statistical, that is, they relate behavior to influences over many conditions. With careful consideration of what is likely to influence behavior and how the influences and behaviors can be measured and related, we can construct models that are sufficient to answer specific questions about what makes a planner behave in particular ways. For example, we may ask: What is the relationship between possible decision strategies and the amount of time required to complete a plan? Does the influence, as measured in terms of the resources used by the plan, of the effect of the agent's actions depend on the rate of environment change? Are particular plan failures likely to be preceded by particular planner actions? These are the kinds of questions that we ask routinely when we design and implement planners for specific environments and when we try to generalize planner designs across environments. Thus, these are the kinds of questions that we should routinely be answering with methods of model construction.

This paper describes how two statistical modeling techniques can be combined to answer questions about the influence of numerical and categorical factors over periods of time in which situations and decisions are repeated. Each of the techniques are automated, generating models

from empirical data without intervention. However, the two techniques, like other statistical techniques, are designed to answer questions about either numerical factors without feedback (e.g., from repetition) or categorical factors over many repetitions. Together, they are shown to characterize behavior involving both.

2 Statistical Methods for Identifying Influences on Behavior

Statistical methods are a promising approach to answering questions about influences on behavior. These questions are frequently hypotheses about the interaction of independent factors (influences) on dependent factors. We rarely have recourse to complete, well-defined models of environment or planner and so rely on observations and experiments to build such models. Additionally, environments often are characterized by pseudo-random events (the environment may not be random, but the factors that produce changes may be too difficult or numerous to measure) and indeed sometimes the planners make random decisions, leading to descriptive models based on summaries or classifications.

Statistical methods are designed to answer particular questions. In the course of our explorations with the Phoenix planner, we have developed two methods, one for characterizing trends in a time series and the other for modelling a temporal snapshot. This section explains how the two methods can be combined to answer questions about influences over varying time spans as well as constructing models of increasing detail.

2.1 Combining Two Statistical Methods

It is relatively easy to detect specific, simple effects of one thing on another, but not effects of combinations of factors. Both *Dependency Detection* and *Path Analysis* are techniques for examining combinations.

Dependency Detection (DD) identifies relationships between discrete events in a time series,

execution traces of a planner operating in an environment [12]. DD tests whether the occurrence of one event depends on the occurrence of other events prior to it (i.e., whether one set of events appears to cause another event). For example, we have used DD to search for dependencies between combinations of actions of a planner and plan failure.

Path Analysis (PA) is a technique for building causal models based on multiple linear regression [10,14,17,19,5], which is related to techniques for causal induction [15,9]. Our algorithm for PA builds detailed models of the interaction of various causal factors on the value of an ordinal variable. For example, we have constructed path models of environment and planner factors that directly or indirectly influence the amount of time required to finish a plan.

The primary strengths of dependency detection are capturing transitions through states and the behavior of the system over long periods of time, and discovering hypotheses about behavior. The primary strengths of path analysis are modelling the interaction of multiple continuous influences at a particular point or within a short interval of time, and constructing detailed models of interactions of influences. These strengths complement each other naturally in the framework of building causal models. Three requirements are commonly posited for x to cause y [18]: x must precede y in time; x and y must covary; and causes of y other than x must be controlled. By finding influences between events, DD concentrates on precedence and control. Through application of regression, PA captures covariance and control. In combination, the two approaches cover all three requirements.

We apply dependency detection and path analysis sequentially to answer specific questions about behavior such as "What previous events and environment factors cause a particular failure to occur?". Dependency detection can suggest a hypothesis about a specific interval of interest, the interval between particular precursor events and a particular target event. Path analysis can then characterize in detail how the one event in conjunction with other factors influences the occurrence of the second. Combining the two methods creates a natural progression from

a descriptive to a predictive and possibly explanatory model[1].

2.2 Over a Time Series: Dependency Detection

Dependencies are detected statistically by analyzing execution traces. An execution trace is simply a sequence of interesting events that occurred during some period of observing the planner. Execution traces can be viewed as transitions between events, which can be analyzed for dependencies. Some of the events are designated as influences (called *precursors* in dependency detection) and others as the behaviors of interest (called *target* events). One event, T_t, is said to be *dependent* on another, P_a, if T_t is observed to occur more often following P_a than after any other event. In its simplest form, the observed execution traces include precursor events interleaved with target events, as illustrated in the following example:

$$P_a \rightarrow T_t \rightarrow P_b \rightarrow T_u \rightarrow P_a \rightarrow T_t$$

where $\rightarrow$ indicates temporal order. The statistical analysis requires two steps. Combinations of influencing events are first tested for whether they are more or less likely to be followed by each of the target events. Then the significant combinations are compared to remove overlapping combinations.

To determine whether particular target events are more or less likely after particular precursors, we construct contingency tables of the incidence of the target T_t after the precursor P_a by counting: 1) instances of T_t that follow instances of P_a, 2) instances of T_t that follow instances of all precursors other than P_a (abbreviated $P_{\overline{a}}$), 3) target events other than T_t (abbreviated $T_{\overline{t}}$) that follow P_a and 4) target events $T_{\overline{t}}$ that follow $P_{\overline{a}}$. These four frequencies are arranged in a 2x2 contingency table:

[1] While a PA model can be explanatory as well, in our combination of the methods, the explanatory model is developed by a knowledge based, as opposed to statistical, technique called *Failure Recovery Analysis*[11].

	T_t	$T_{\overline{t}}$
P_a	15	10
$P_{\overline{a}}$	30	50

For this table, we see a strong dependence between the precursor, P_a, and the target event, T_t: 15 cases of T_t following P_a out of 25 cases of P_a. But while P_a leads most frequently to target event T_t, precursors other than P_a lead to T_t relatively infrequently (30 instances in 80). A G-Test on this table will detect a dependency between the target event and its precursor [17]; in this case, $G = 5.174, p < .023$, which means that the two events are unlikely to be independent and conclude that T_t depends on P_a (abbreviated as $[P_a, T_t]$).

The precursors can be single events or combinations of events. For example, we may wish some events to be counted both as influences and as target events (and so ask whether one target event influences the occurrence of others). Given the above notation, we then build contingency tables for three classes of precursors: influences that are not also targets, targets that are also influences, and combinations of targets and influences. A statistical technique based on the G-Test differentiates the three types of precursors by comparing the sum of the effects due to the combinations (e.g., T_uP_a for all possible T_ts) to the effect due to just the grouped effect (e.g., T_u or P_a). The intuition behind the test is that if the combinations do not add much information about the effect then they can be disregarded; conversely, if the grouped effect, T_u or P_a, masks differences between the pairs, then the grouped effect should be disregarded as misleading. For example, by comparing the example dependency, $[P_a, T_t]$ to ones that name the target that preceded P_a as well (e.g., $[T_uP_a, T_t]$, $[T_vP_a, T_t]$, etc), we may find that $[T_uP_a, T_t]$ adds little information over knowing $[P_a, T_t]$, so the precursor that includes the preceding target can be disregarded. Similar comparisons can be done for longer combinations of influences as well.

The output of dependency detection is a list of dependencies found to be significant in the

execution traces and that describe the influences without redundancy. The interpretation of this output depends entirely on what events were being monitored. Dependency detection was designed to detect sources of plan failure in the Phoenix planner [11]. For that application, the events of interest are failures and recovery actions, and it discovers dependencies between particular failures, the recovery methods that repair them and the occurrence of other failures downstream. We designed it to answer the question: "Does a failure depend on some action or event that preceded it?" with the intent that we would use the information to identify bugs in the planner. For our explorations with the Phoenix planner, we have collected almost 1000 execution traces (50-120 traces in each of nine different experiment scenarios) in which we have detected hundreds of dependencies with precursors of many lengths. These dependency sets have been used in isolation to motivate debugging efforts and in comparisons to determine the sensitivity of particular dependencies to the planner's implementation and to the environment's rate of change.

2.3 Snapshots of Influences: Path Analysis

Historically, path analysis was a method for *estimating* the strengths of causal influence, given a causal model and correlational data. It was up to the data analyst to propose a causal model in the first place. Recently, researchers have developed algorithms to propose causal models, given conditional information relationships in data, but these algorithms don't handle estimation, relying instead on statistical packages (notably EQS or LISREL [1,13]) to estimate strengths of causal influence. We have recently developed an algorithm that uses these strengths to guide its search in the space of causal models; in other words, it infers causal models and estimates their parameters at the same time. We have used our algorithm, called FBD, to estimate the causal influence of aspects of the Phoenix environment, such as wind speed and fire perimeter, on variables such as the time until the next plan failure.

FBD is described in detail in [4]; we will just sketch it here. Before analysis begins, we assume that our data are *standardized* to eliminate the effect of differences due to measurement scale. Standardization transforms a distribution of data, such as the distribution of fire perimeters, so its mean is zero and its standard deviation is one. Standardized variables are denoted by uppercase letters and created by the transformation $X_i = (x_i - \overline{x})/s$. Ordinary multiple linear regression builds causal models that predict the value of a performance variable Y given predictors $X_1, X_2, \ldots X_k$. In standardized form, a regression model looks like this:

$$\hat{Y_i} = \beta_{X_1} X_{1_i} + \beta_{X_2} X_{2_i} + \cdots + \beta_{X_k} X_{k_i}$$

If one were to draw a picture of this model, it would have each predictor variable pointing directly to the performance variable and correlated with every other predictor variable.

β coefficients have a causal interpretation. This is possible because they are *partial*: each represents the influence of one predictor variable on Y when the influences of all the other predictor variables are held constant. As such, these coefficients offer a statistical version of the experimental control that we need to assert cause. To show that X_1, say, causes Y we have to show that its correlation with Y is not due to some shared relationship with another variable, say, X_2. Most causal induction algorithms do this by showing that the relationship between X_1 and Y does not disappear when X_2 is held constant. Partial correlation coefficients are used for the purpose, but partial regression coefficients—β coefficients—will serve as well. Thus, if X_1 and Y have a high correlation but β_{X_1} is close to zero, we suspect that X_1's influence on Y is actually due to another variable. Moreover, because β coefficients are in the same units, if $\beta_{X_1} = k\beta_{X_3}$ we can say predictor X_1 has k times the causal influence of X_3.

Unfortunately, multiple regression models are just one "level" deep, which means that all influences are modelled as exerting a *direct* influence on the dependent variable. The FBD algorithm solves this problem as follows: it finds good predictors of the performance variable Y

and then, recursively, treats each predictor as a performance variable and finds good predictors for it. The trick is to decide which predictors to use at a particular level, and which to hold back. FBD applies four tests to select a set of predictors at each level. First, it won't allow a variable to be a predictor if doing so creates a cycle in the resulting causal model (e.g., A causes B and B causes A). Second, it calculates the score

$$\omega_i = \frac{r_{Y,X_i} - \beta_{X_i}}{r_{Y,X_i}}$$

and throws out any predictor for which $\omega_i < T_\omega$, a threshold. ω_i represents the proportion of X_i's *total* influence on Y, measured by the correlation r_{Y,X_i} that is *not* due to its *direct* influence on Y, measured by β_{X_i}. Said differently, ω_i represents the proportion of X_i's influence on Y that is due to its relationships with other predictors. It should be as low as possible. The third test is that $\beta_{X_i} > T_\beta$, because a predictor can pass the second test but still have very little direct influence on Y. The fourth test is more complex and is stated here without elaboration: FBD will select from the predictors that passed the first tests those that form the largest set of predictors such that none is conditionally independent of the performance variable given the others (see [4] for explanation).

At every stage of the development of a model, the candidate predictors for a performance variable are all those but the original performance variable and the current one, but the candidates are quickly whittled down to a smaller set, typically less than five or six. Each of these predictors then becomes a performance variable and the process repeats until we run out of variables that need to be predicted.

FBD has been thoroughly tested and its performance compared with Pearl and Verma's IC algorithm [15]. As expected, the algorithm performed well using measures such as R^2 and differences in estimated correlation. Additionally, it compared favorably using conditional independence measures.

3 An Example with Phoenix

The Phoenix fireboss planner coordinates the efforts of other agents to contain forest fires within a simulation of fires in Yellowstone National Park [6]. Within the simulation, fires burn somewhat unpredictably, and the weather can be controlled during experimentation to change at variable intervals and by variable amounts. The fireboss is forced to deal with challenging conditions: limited resources and a changing environment. As a consequence, the fireboss's plans often fail.

Failure due to environment may be unavoidable; failure due to poor decisions on the part of the planner is unacceptable. In this example, we wish to combine dependency detection with path analysis to identify factors relevant to a major decision the planner must make: which of several possible skeletal plans to use to fight a particular fire.

First, we collected 102 execution traces of the Phoenix fireboss planner in a scenario in which three fires were started at intervals of about 12 hours and the wind speed was allowed to vary by ± 3 kph and the wind direction by ± 15 degrees every 30 minutes. (Wind speed and wind direction strongly influence the spread of the fire.)

The execution traces included failures and the recovery methods applied to repair them. For example, the following is a fragment of a trace from the set: $F_{cfp} \rightarrow R_{rm} \rightarrow F_{ner} \rightarrow R_{spa} \rightarrow F_{ip} \rightarrow R_{aab}$. F_X denotes the occurrence of a plan failure of type X; R_Y denotes the successful use of a recovery method of type Y. The data includes 10 types of plan failure and 8 types of recovery methods. It is impossible to tell much from just this fragment; for example, F_{ner} may depend on R_{rm} but we have only one example of it. We analyzed the traces with dependency detection and identified 23 dependencies: 3 with precursors of FR, 7 with precursors of F, and 13 with precursors of R. The target events were failures.

At the conclusion of the DD phase, the experimenter must select a dependency for further

modeling with PA. The output of DD is a set of dependencies; the starting point for PA should be a single dependency that demarcates an interval. Many criteria can be used to select a dependency for attention. We picked a dependency that was highly significant, was based on a fair amount of data (no cell in the contingency table was less than 5), and included as a target event a failure with costly recovery. The dependency $[R_{rm}, F_{ner}]$ ranked best on these criteria. Its contingency table is:

	F_{ner}	$F_{\overline{ner}}$
R_{rm}	18	62
$R_{\overline{rm}}$	20	314

A G-test on this table yields $G = 23.7, p < .0001$, which is highly significant.

Now, we wish to know *why* R_{rm} seems to cause F_{ner}. R_{rm} is an expensive replanning method that results in recalculation of the entire plan to fight a fire. The F_{ner} failure results when not enough resources are available to complete the plan that was selected. We used dependency detection to identify this relationship; we now apply path analysis to explore the relationship in more detail.

What we need from path analysis is a model of the factors that influence the occurrence of the F_{ner} failure. Once we understand these factors, we can redesign the decision procedure about which plan to use during replanning. Based on some knowledge of how the environment influences fire spread and how other plans influence available resources, we identified a candidate set of variables to be collected during another experiment and considered as part of a path model. It is unrealistic to monitor everything in a planner and environment, and so collecting additional data requires some knowledge on the part of the experimenter as to which factors are likely to be salient. Those influences are collected at the time of replanning and are:

- wind speed,
- fire size,

- number of active bulldozers (working on other fires or waiting for assignment),
- number of fires being fought,
- distance from center of fire to base,
- wind speed changes over last two hours,
- wind direction changes over last two hours.

Because the dependent factor for PA should be a continuous variable, we replaced the event F_{ner} with the variable *Interval*, a measure of the time between the start of the replan and the conclusion of the new plan (either by successful completion or by occurrence of a failure). We seek to maximize the delay between plan start and plan failure because the longer the plan executes, the more likely it is to have passed the critical point of producing a F_{ner} failure (which occurs, if at all, relatively early in a firefighting plan).

We ran 60 trials of the same scenario in which we collected the above information. We used path analysis to generate the model shown in Figure 1. The model accurately identifies change in wind speed and wind direction as independent variables, and captures several expected relationships: as variability in wind direction increases, the size of the fire grows; a greater number of fires causes replanning to happen sooner, reducing *Interval*. The strongest factor affecting *Interval* is *Fires*, and then *Wind Speed* and *Fire Size*. The greater the number of fires, the shorter *Interval* is likely to be, and thus the more likely F_{ner} is to occur.

Using the model we can then approach the task of improving R_{rm}. One benefit of the model is that it helps us distinguish between many possible influences on a variable and concentrate on the ones that have the strongest influence in terms of individual contribution and predictive power. The replanning method can make a better informed decision by first considering the number of fires, then fire size and wind speed. At present, it considers only the number of bulldozers available and the wind speed. Indirect influences, such as change in wind direction, are of secondary interest. The particular plan selected by R_{rm}, of which there are several

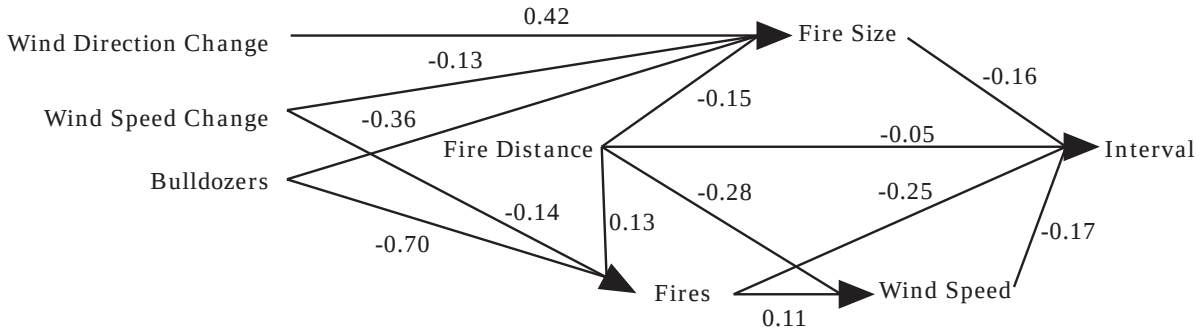

Figure 1: Path analysis model of factors influencing *Interval*

possible, will perform better if it accounts for the more direct influences. The information provided by the model can help us build improved plans as well.

The path analysis model can thus provide detailed solutions to "questions" posed by dependency detection. The reverse is also true. One difficult area for path analysis modeling is the general problem of distinguishing cause from effect based on predictive criteria. As mentioned, the model built by PA is not perfect; for example, if number of fires is a good predictor of the number of bulldozers, then the reverse is likely to hold as well and, in fact, does hold in our model.

To handle this difficulty, we might allow the user to specify variables as independent. This simple approach has limitations. In non-experimental studies, in which variables are not manipulated explicitly, it is not always apparent whether a variable is independent. A potentially better approach is to use dependency detection to find the proper causal direction between two variables. In our model, for example, the relationship *Bulldozers* $\rightarrow$ *Fires* is plausible, given the data, but we believe that the causality could be reversed. We can explore this question by associating the continuous variable *Bulldozers* with the discrete event *BulldozerDeath* and the variable *Fires* with the event *FireSet*. If a dependency is detected from *FireSet* to *BulldozerDeath*, we take this as evidence that the causal influence is from *Fires* to *Bulldozers* in the

path analysis model as well. Not all relationships can be examined in this way, but there are many natural candidates.

4 Conclusion

The results of combining DD and PA are a dependency and a path model of the influences related to that dependency. The next step is interpretation, which requires domain knowledge. We use the path model and the dependency to identify factors that produce early plan failure, and we use domain knowledge to prune out implausible candidates and to determine how those factors might be considered by the planner during replanning.

The model may uncover relationships that are expected but needed to be confirmed, unexpected but plausible, or implausible and likely due to sampling bias. For example, we expected that *Wind Speed* influenced *Interval* but not to what degree; we did not expect *Bulldozers* to indirectly influence *Interval* but upon reflection and some checking, we can construct a domain explanation of why; and we experimentally varied the *Fires* and know it is not possible that the number of bulldozers can have a strong negative effect on it[2].

Together these results of DD and PA answer the question of how a previous event (a recovery action) and environment factors (such as wind speed and number of fires) influence how and when a particular failure occurs. For now, by combining DD and PA, we can answer questions about what events and influences cause subsequent events, as well as how short horizon factors influence performance at the target event. For the future, we are designing methods to answer other questions about planner behavior and to interpret the results of those methods.

[2]We collect data only when R_{rm} occurs. As bulldozers are destroyed by fire, replanning increases. As more fires are set, replanning increases as well. Each instance of replanning makes the occurrence of R_{rm} more probable; thus, our data are biased towards situations with a high correlation between the number of bulldozers and the number of fires.

References

[1] P. Bentler. *Theory and Implementation of EQS: A Strutural Equations Program.* BMDP Statistical Software, Inc., Los Angeles, 1985.

[2] Rodney A. Brooks. Intelligence without reason. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, August 1991.

[3] David Chapman. Penguins can make cake. *AI Magazine*, 10(4):45–50, Winter 1989.

[4] Paul R. Cohen, Lisa Ballesteros, Dawn Gregory, and Robert St. Amant. Experiments with a regression-based causal induction algorithm. 1994.

[5] Paul R. Cohen, Adam Carlson, Lisa Ballesteros, and Robert St. Amant. Automating path analysis for building causal models from data. In *Proceedings of the Tenth International Conference on Machine Learning.* Morgan Kaufmann, 1993.

[6] Paul R. Cohen, Michael Greenberg, David M. Hart, and Adele E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3), Fall 1989.

[7] Oren Etzioni. Intelligence without robots: A reply to Brooks. *AI Magazine*, 14(4):7–13, Winter 1993.

[8] Matthew L. Ginsberg. Universal planning: An (almost) universally bad idea. *AI Magazine*, 10(4):40–44, Winter 1989.

[9] C. Glymour, R. Scheines, P. Spirtes, and K. Kelly. *Discovering Causal Structure.* Academic Press, 1987.

[10] David M. Hart and Paul R. Cohen. Predicting and explaining success and task duration in the Phoenix planner. In J. Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS92)*, San Mateo, CA, 1992. Morgan Kaufmann Publishers, Inc.

[11] Adele E. Howe. *Accepting the Inevitable: The Role of Failure Recovery in the Design of Planners.* PhD thesis, University of Massachusetts, Department of Computer Science, Amherst, MA, February 1993.

[12] Adele E. Howe and Paul R. Cohen. Isolating dependencies on failure by analyzing execution traces. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, College Park, MD, 1992.

[13] K. Joreskog and D. Sorbom. *LISREL VI User's Guide.* Scientific Software, Inc., Mooresville, IN, 1984.

[14] C.C. Li. *Path Analysis – A Primer.* Boxwood Press, 1975.

[15] Judea Pearl and T. S. Verma. A theory of inferred causation. In J.A. Allen, R. Fikes, and E. Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452, San Mateo, CA, April 1991. Morgan Kaufmann.

[16] M. J. Schoppers. In defense of reaction plans as caches. *AI Magazine*, 10(4):51–60, Winter 1989.

[17] Robert R. Sokal and F. James Rohlf. *Biometry: The Principles and Practice of Statistics in Biological Research*. W.H. Freeman and Co., New York, second edition, 1981.

[18] P.C. Suppes. *A Probabilistic Theory of Causality*. North Holland, Amsterdam, 1970.

[19] Sewal Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.