Abstract

Title of dissertation:	Scalable Techniques for Behavioral Analysis and Forecasting
	Amy Sliva, Doctor of Philosophy, 2011
Dissertation directed by:	Professor V.S. Subrahmanian Department of Computer Science

The ability to model, forecast, and analyze the behaviors of other agents has applications in many diverse contexts. For example, behavioral models can be used in multi-player games to forecast an opponent's next move, in economics to forecast a merger decision by a CEO, or in international politics to predict the behavior of a rival state or group. Such models can facilitate formulation of effective mitigating responses and provide a foundation for decision-support technologies.

Behavioral modeling is a computationally challenging problem—real world data sets can contain on the order of $10^{30,000}$ possible behaviors in any given situation. This work presents several scalable frameworks for modeling and forecasting agent behavior, particularly in the realm of international security dynamics. A probabilistic logic formalism for modeling and forecasting behavior is described, as well as distributed algorithms for efficient reasoning in this framework. To further cope with the scale of this problem, forecasting methods are also introduced that operate directly on time series data, rather than an intermediate behavioral model, to forecast actions and situations at some time in the future. Agent behavior can be adaptive, and in rare circumstances can deviate from the statistically "normal" past behavior. A system is also presented that can forecast when and how such behavioral changes will occur. These forecasting techniques, as well as any arbitrary time series forecasting approach, can be classified by a general axiomatic framework for forecasting in temporal databases.

The knowledge gained from behavioral models and forecasts can be employed by decision-makers to develop effective response policies. An efficient framework is provided for identifying the optimal changes to the state of the world to elicit desired behaviors from another agent, balancing cost with likelihood of success. These modeling and analysis tools have also been incorporated into a prototype decision-support system and used in several case studies of real-world international security situations.

SCALABLE TECHNIQUES FOR BEHAVIORAL ANALYSIS AND FORECASTING

by

Amy Lynn Sliva

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Dedication

To my family: Dana, Kathy, Tom, Barbara, and Alex

Acknowledgements

First and foremost, I have to thank my family for supporting me through my graduate education. Specifically, my mother Kathy and her partner Steve, and my father Tom and his wife Barbara have been there for me at every step of this experience. Special thanks are due to my wonderful sister and best friend, Dana, who has always been full of good advice, inspirational pep talks, and excellent proofreading skills. In addition, I want to thank Eric Baker for all of his love and unconditional support during the home stretch of my studies—I could not have done this without him.

Of course, I also owe boundless gratitude to my advisor, Professor V.S. Subrahmanian, for his unwavering research advice and professional mentorship. With the Laboratory for Computational Cultural Dynamics, Professor Subrahmanian has given me a perfect outlet for exploring my interests in computer science and international affairs and the unique opportunity to hone my skills as a truly interdisciplinary researcher. His exciting, big-picture ideas have shown me that academic research can not only help to understand the world, but also help change it, and he will continue to inspire me as I move forward with my career. In addition, I want to thank the rest of my committee, Professors Dana Nau, Samir Khuller, Ashok Agrawala, and Jonathan Wilkenfeld for their time, advice, and research collaborations that have contributed to this thesis. In particular, collaborations with Professor Wilkenfeld and the Center for International Development and Conflict Management have been crucial aspects of my studies and the work presented here.

My other colleagues—graduate students, researchers, and faculty—deserve special thanks as well. I have had a wonderful experience working with fellow students Gerardo Simari, Vanina Martinez, Austin Parker, and Matthias Bröcheler, as well as researchers Francesco Parisi, Cristian Molinaro, Aaron Mannes, and Jana Shakarian, and I want to thank them all not only for their valuable scholarly contributions, but for enriching my time at Maryland. I also want to acknowledge the programming and systems support from LCCD staff members Damon Earp and Dan LaRocque. They have both put in countless hours maintaining the databases and systems necessary for my experiments and designing amazing interfaces to help make my research systems a reality. Recognition should also go to the faculty and students at the Maryland School of Public Policy for the valuable insights I gained while taking classes there. My experience as a policy student has only made me a better computer scientist with a deeper understanding of potential real-world applications. In particular, I want to thank Jonas Segal and Professor I.M. Destler for being excellent research and career advisors.

Finally, I have to thank Amrit Bandyopadhyay, Sorelle Friedler, and Niky Stockhausen for being great friends, roommates, and sounding boards over the last several years, and Megan Parker for her amazing and supportive friendship from the very beginning of this process. I am eternally grateful to you all!

Contents

Chapter 1

Introduction

The field of forecasting is extensive and widely studied, with a vast amount of general techniques [18] as well as specialized forecast models for a variety of domains such as finance [118], epidemiology [57], politics [97, 75, 16, 114], and even product liability claims [111]. This work focuses primarily on procedures to forecast behaviors of an agent at a given time in the future or for a specific situation and ways to use this behavioral knowledge to develop effective countermeasures or strategies. The ability to forecast the behaviors and situations of other agents has applications in many diverse contexts. For example, behavioral models can be used in multi-player games to forecast an opponent's next move, in economics to forecast a merger decision by a CEO, or in international politics to predict the behavior of a rival state or group. Such forecasts can facilitate the formulation of appropriate and effective responses to the behavior of other agents independently of any specific application.

The modeling and forecasting techniques described here are particularly applicable to the realms of international conflict, politics, and development, with clear uses as decision-support technologies in national security, intelligence analysis, and public policy domains. The modern global political environment is growing increasingly complex, characterized by webs of interdependency, interaction, and conflict that are difficult to untangle. Technological expansion has led to an explosion in the information available, as well as the need for more sophisticated analysis methods. Behavioral models and forecasts can be leveraged to produce more actionable intelligence from varied data and formulate the most effective policies—for conflict management, counterterrorism, or international development—that consider the behaviors of various actors in international affairs.

The next section briefly presents some real-world applications of using behavioral modeling, forecasting, and analysis in this international relations context.

1.1 Behavioral Analysis of Security & Conflict Situations

Agent behaviors in the domain of international security and conflict can be understood as the confluence of many dynamic factors—cultural, economic, social, political, and historical—in an extremely complex system. Several applications have recently been developed for reasoning about conflict that incorporate the behavioral modeling and forecasting capabilities discussed in the following chapters as a means of managing the analytic complexity of these situations. For example, the possible economic, temporal, and cultural correlations have been investigated for the behaviors of socio-cultural-economic groups from different parts of the world, including Afghan and Pakistani tribes such as the Afridis, Shinwaris, and Waziris; approximately 50 ethnopolitical organizations from the Middle East and Asia Pacific regions; various stakeholders in the Afghan drug economy; political parties such as the Pakistan People's Party; and even nation states. Two applications that include these behavioral models and forecasts are described below.

The Cultural Adversarial Reasoning Architecture

There is a constant need to reason about diverse cultures from around the globe, facilitating development of effective international policies and agreements for peace and development. Cultural reasoning tools focus on understanding how different cultural groups make decisions and on which factors those decisions are based. The Cultural Adversarial Reasoning Architecture (CARA) [117] is a general architecture for cultural reasoning that consists of several components:

(i) A Semantic Web extraction engine to elicit data about the group.

(ii) An opinion-mining engine that captures the group's opinions.

(iii) An algorithm to correlate environmental variables with actions of the group.

(iv) Algorithms to forecast what groups will do in a given situation (real or hypothetical) and how these actions might impact the situation on the ground.

(v) A simulation or game environment within which analysts and users can see what the group has done and what it might do in future situations.

The behavioral modeling and forecasting procedures presented in the subsequent chapters fulfill item (iv) on the above list.

The SOMA Terror Organization Portal

The SOMA Terror Organization Portal (STOP) [107] is a web-based tool that allows analysts and users to access data on terror organizations throughout the Middle

East and North Africa. The underlying dataset is the Minorities at Risk Organizational Behavior (MAROB) [46, 125] data collected by the Center for International Development and Conflict Mangaement at the University of Maryland, which tracks the behaviors and contextual factors for 118 ethnopolitical organizations between the years of 1980 and 2004. As a decision-support tool, STOP allows users to explore behavioral models of these organizations, as well as access the forecasting tools that are described here. STOP also includes an important social networking component, allowing collaborative work between analysts studying the same topics or groups. The STOP system and case studies of its use in analyzing the behaviors of Hezbollah [73] and Hamas [74] are described in further detail in Chapter 8.

Of course, agent behavior, especially human behavior, is equally complex in non-conflict situations as well, and the behavioral analytics presented in the following chapters are not limited to this context. These modeling and forecasting tools can form the basis of decision-support technologies in other dynamic situations in international relations, such as managing logistics, resource allocation, and policy analysis for disaster recovery situations, post-conflict reconstruction, and social policies and infrastructure creation in developing countries.

1.2 Related Work

Behavioral forecasting and modeling inherently involves a degree of uncertainty, which is often modeled through a probability distribution or error terms in a statistical forecast. Probabilistic logic programs (PLPs) [79] have been proposed as a paradigm for probabilistic logical reasoning with no independence assumptions. PLPs use a possible worlds model based on prior work by [53], [44], and [82] to induce a set of probability distributions on a space of possible worlds. Related approaches to probabilistic reasoning can be found in [122] and [43]. The former generalizes the fixpoint theory for Horn clause rules to a "quantitative" case, where the logical truth values can range between 0 (false) and 1 (true). In the latter, uncertainty in the probability assignments themselves are modeled by allowing certain events to be "unmeasurable." Similar to the representation used in some probabilistic logic semantics, a lower and upper bound can be derived—the inner measure and outer measure, respectively—for an event where the probability is unknown.

Past work on PLPs [80, 79] focuses on the entailment problem of checking if a PLP entails that the probability of a given formula lies in a given probability interval. In Chapter 2, PLPs are used as a stochastic representation of agent behavior in the Stochastic Opponent Modeling Agents (SOMA) [103, 62, 61] framework. Unlike other treatments of PLPs, this formalism is used for solving the problem of finding the most probable model (i.e., the most probable set of actions an agent will take) rather than entailment of a formula.

As mentioned above, PLPs do not make any independence assumptions; in fact, this framework can be utilized to find the dependencies between measurable or changeable variables and the behaviors of an agent. However, this robust reasoning mechanism also presents a challenge for scalability, as the number of possible worlds that must be enumerated, especially in real-world models, can be prohibitively large. In [69], several techniques are described to reduce the number of worlds that must be considered by partitioning the logic program and identifying logical equivalence classes. [70] extends this method to a semantics that uses maximum entropy optimizations for computing the answer to a probabilistic logic entailment query. Previous work on SOMA [62, 61] also addressed this tractability problem by identifying equivalence classes and heuristic algorithms to reduce the number of variables in the linear constraints derived from a PLP. [49] takes a different approach to scalability in probabilistic logic, foregoing the enumeration of possible worlds altogether. Rather, a set of inference rules is developed for deduction in probabilistic logic, giving rise to an anytime algorithm that can return the tightest probability bounds computed so far for a query formula, eventually converging on the tightest possible bounds. In the following chapter, a distributed computation approach is used to enhance the existing reasoning algorithms in the SOMA framework, allowing for a reduction in computation time, the ability to scale to real-world behavioral models, and an improvement in the accuracy of approximate solutions.

Another approach to uncertainty management in predictive modeling is ensemble forecasting—running the same model multiple times with different initialization parameters and inducing a probability distribution over the results based on their frequency among the different inputs. In [13], a targeted Monte Carlo approach is used to sample the most important initial settings in the parameter space and provide a probabilistic forecast in several well-known statistical models.

Statistical methods have also been utilized to model agent behavior—including the context of international conflict—as a stochastic process. Beginning with Richardson's classic work [95], human conflict situations ranging from murder to world wars have been shown to follow a power law distribution, indicating through the property of scale invariance that there is no qualitative distinction between small and large events and that severe conflicts are more frequent than would otherwise be expected. In [32], analysis of terrorism events show a similar power law distribution for the frequency-severity distribution of attacks. [32] posits that scale invariance is a general feature of violent conflict, where the scaling exponent depends on the type of conflict being modeled. Similar models in [31] also account for some of the internal dynamics of terrorist groups that lead to variously sized cells; in a steady state, the number of cells of a particular size, and by assumption the correlated severity of their attacks, also follows a power law distribution. This model is further extended in [29], which shows that as a group increases in size and gains experience over time, the frequency of its attacks also increase, potentially accounting for this rise in overall severity. While these techniques can identify important statistical patterns in the incidence of particular behavioral events, they do not address the issue of forecasting future behaviors of particular agents acting in this dynamical system and provide a longer-term analytical view rather than decision-support models.

Graphical probabilistic and stochastic models—such as hidden Markov models (HMMs) [92], Bayesian networks [84], and conditional random fields [65]—have also been used extensively for making forecasts in a variety of contexts. Abramson and Finizza [2] use a Bayesian network to model both political and economic factors of the oil market, using Monte Carlo simulations over various scenarios and prior probability distributions in these networks to forecast economic events of interest. Schrodt [97] and Bond et al. [16] have developed methods to build hidden Markov models to describe how a conflict might evolve over time. In [97], two distinct HMMs are trained—one with states representing "high" conflict, and another for "low" conflict. Weeks are forecasted as having either high or low conflict depending on which model best fits the leading data. [16] uses the well-known Viterbi algorithm for traversing an HMM to forecast the stage of a conflict indicated by the observed actions of international actors. In [35], Dagum et al. utilize dynamic Bayesian networks to build more realistic models of the dependencies and nonlinearity necessary in many time-series analyses. This system is applied to monitoring of ICU patients, allowing for advanced warning of a crisis situation rather than early recognition.

The main advantage of these graphical models is that they can represent complex dependencies and relationships among the parameters. However, in most of these cases, the graphical models are painstakingly constructed over time using data and expert knowledge. Friedman et al. [48] present an algorithm for learning the structure and the parameters of a dynamic Bayesian network simultaneously using an EM procedure. This method looks at potential structures until the estimated likelihood of the data given the model converges. However, this still requires human input, such as whether hidden states should be included and what the possible states of the network should be. [55] proposes a generalized method for Bayesian network construction. In this case, a general schema is used to describe the high-level structure and relationships of a Bayesian network, but the specific graph is instantiated at runtime using a program of specially designated ground Prolog terms. While the users of this system do not require specific domain knowledge regarding dependencies, the general structure of the network must still be carefully defined by a human expert. A similar approach to generalized graphical models is taken in [71], where a mixture of maximum entropy and Markov models are used to model the behavior of web users. The models are trained with historical data across all users, allowing for general parameters to be computed. However, over time specific data on individual users can be substituted for the general model. As with other similar approaches, the structural features must be specified in advance, while only the model parameters are automatically refined.

The modeling and forecasting work in the following chapters takes a different approach to understanding agent behavior. In the SOMA framework (Chapter 2), an algorithm for automatically extracting probabilistic logic behavioral rules from a relational database has been developed. These rules are then used for forecasting with no assumptions as to their relationships or conditional independence. While graph-based methods tend to be much more efficient, logical models can sometimes be more intuitive and provide possible explanations for behaviors in the model. Prior work in the literature [81, 89] has attempted to bridge the gap between the graphical reasoning approaches and the logical knowledge-representation models such as PLPs. The probabilistic logic framework developed in [81] explicitly includes conditional semantics, providing a direct means for constructing a Bayesian network from a logic program. In [89], Poole describes the independent choice logic as a knowledge representation that is equivalent to Bayesian networks, allowing incorporation of Bayesian probability into predicate logic and first-order features into Bayesian networks.

Graph-based models, such as those mentioned above, are also often predicated on finding *indicators* that have previously been identified as correlated with the events or actions being predicted. In addition to those methods already described, [6] presents a Bayesian method for finding an "alarm" for a particular event of interest. Using historical and current data, the authors find the most likely alarm condition that occurs one time step before the given event, and use this to forecast one-step-ahead whether the event of interest will occur. In the CONVEX [75] and SitCAST [108] methods (Chapter 3 and 4), forecasts are made directly from time series data, skipping the model construction step altogether and avoiding the assumption that predictive indicators of a particular action can be known *a priori* or can only be for one-step-ahead forecasts.

Like all predictive modeling applications, behavioral forecasting has an inherently temporal component. Time series forecasting is a very diverse field that has been applied in a wide range of contexts, from politics to biology to economics. With a large array of possible statistical models, there have been several attempts to better understand the relationship between these different forecasting procedures. In [54], the author proposes a theoretical framework for unifying a diverse set of forecasting methods for univariate time series. Many of these distinct methods can be integrated into a common general mathematical framework using a structural time series representation and the Kalman filter as a means of approximating the parameters and making forecasts. When estimating a model based on historical data, it is often difficult to determine what the "best" model is, both in terms of fit and predictive ability. [128] presents the AFTER algorithm that attempts to produce better forecasts by assigning weights to several different candidate models and combining the results to achieve more accurate forecasts than through a single model. Raftery et al. [93] address the same problem, using two different Bayesian model averaging techniques to handle uncertainty in selecting predictors in a linear regression model. Bayesian model averaging can reduce the set of candidate models and combine the remaining possible models for purposes of prediction. In terms of accurately fitting the data and in making forecasts, the averaged models outperform other standard model selection techniques that choose only a single model.

In Chapter 6, a generalized framework describing time series forecasting is also presented. Rather than a statistical model that can be used to represent different types of forecasters, a general theory is proposed for integrating forecasting procedures directly into database operations and queries. The Change Analysis Predictive Engine (CAPE) [108] system in Chapter 5 also addresses the issue of combining forecasting methods. CAPE uses various forecasts to model different aspects of behavior to determine when a behavioral change should be predicted and when forecasts should be made that are consistent with "normal" behavior.

Understanding and forecasting the behavior of another agent can be used in order to develop effective response policies or counterstrategies. The use of behavioral models for this type of reasoning is prevalent in games and adversarial settings. In many of these methods, the aim is to determine what actions an opponent agent will take at a particular game state in order to ascertain the best response. Kumar and Nau [64] unify many different algorithms—such as Alpha-Beta, AO*, and SSS*—by providing a general branch and bound algorithmic framework that encompasses these well-known search procedures as variations of AND/OR graph search. In [24], the classic minimax algorithm is generalized to account for the fact that behavioral models of the opponent's decision-making strategy can be incorporated into the search. Game tree search algorithms are often based on assumptions about the other player's strategy or equilibrium behavior. In contrast, the behavioral forecasts presented in the following chapters are made based on a particular agent's actual pattern of behavior, which may not adhere to optimality constraints or strategic rationality. In addition, these search algorithms can become very inefficient. [113] provides a more efficient mechanism for finding equilibria in two-player games using a sequence representation of possible strategies and solving a linear optimization problem. This approach can be generalized to maximization of a set of non-linear constraints to find the equilibria strategies in an n-player game.

One popular game for these studies is the iterated prisoner's dilemma [9]. [8] presents an adaptive method for modeling an opponent's behavior in the iterated prisoner's dilemma with noise, updating the model when it is clear that deviant behavior actually represents a new strategy rather than noise. The probabilistic rules derived to model the opponent are similar to the *ap*-programs used by the SOMA framework in Chapter 2. However, SOMA forecasts future behavior based on a linear programming computation over these probabilities, while [8] derives deterministic versions of the rules for decision making.

Behavioral modeling techniques have also been used in game-theoretic multi-agent environments with a human actor component for modeling interactions such as negotiations [63], protests [101], and the emergence of social structures [22]. In [63], Kraus et al. formalize the concept of a crisis and give a strategy for automated computer agents to achieve a subgame perfect equilibrium in a negotiation. These agents can also interact with human negotiators, in which case a heuristic algorithm gives better performance than the theoretical equilibrium strategy. Complex models of human behavior are constructed in [101], incorporating theories of emotion, physiological factors, preferences, and beliefs into agent behavior. These models are used to simulate a crowd-tipping situation where an instigator agent causes a protest to turn violent; humans can play the role of certain agents in this scenario. In [22], a theoretical framework is given for describing all collective social action as a function of individual action on the part of cognitive agents. For cooperative models to emerge from these agents, structures specifying underlying social connections, such as dependencies, acquaintances, and lines of communication, must be specified. [123] provides an approach called behavior bounding to analyze the consistency of agent models with human-player behavior, using a hierarchical taxonomy to automatically classify the degree of agreement between agents and humans based on defined behavioral bounds.

Such work is similar to the probabilistic behavioral models used in the SOMA or CAPE systems (Chapters 2 and 5). However, the modeling and forecasting methods in these two frameworks are based more on data-driven techniques rather than general theories from the social and cognitive science literature, making them adaptable and customizable when applied to real-world situations. [30] uses a similarly data-centric approach for analyzing the strategic calculus of actors in the Israel-Palestine Conflict. Using empirical statistical analyses, conventional theories regarding the use of suicide attacks and strategic substitution as a response to counterterrorism policies have little support in the data, while competition among groups and their internal dynamics seem to play a larger role in this conflict.

There is substantial work in the AI planning and games community on discovering sequences of actions that lead to a given outcome, sometimes specified as a goal condition similar to the Actionable State Change Attempts framework in Chapter 7 (see [78] for an overview). In general, AI planning assumes the effects of actions to be explicitly specified. However, some approaches, such as partially observable Markov decision process (POMDPs) [58], do account for imperfect knowledge about the state of the world resulting from particular actions, similar to the concepts of probabilistic success and effectiveness used in the Actionable State Change Attempts. A similar problem is addressed in the area of reasoning about actions [11, 85]; work in this field generally assumes that descriptions of effects of actions on fluent predicates, causal relationships between such fluents, and conditions that enable actions to be performed are available. In similar work, [10] uses a Bayesian reasoning method to model inferences about an agent's likely behavior based on external observations. As with the other behavioral modeling and forecasting techniques developed here, the approach taken in the Actionable State Change Attempts framework in Chapter 7 attempts to solve this problem in a fundamentally different and data-driven way, assuming (i) actions only change certain parameters in the system, (ii) all attempted changes succeed probabilistically depending on the set of attempted changes, and (iii) the effects of the changed parameters on the state can only be determined by appeal to past data. Finally, research within the machine learning community on the problem of classification [76] is also related to finding ways to influence agent behavior. However, while such methods are useful for classifying situations in past data and identifying similar future situations, they do not address the issue of how to *arrive once again at similar situations*, i.e., they do not help identify effective policies.

1.3 Organization of this Thesis

Chapter 2 begins with a discussion of the *Stochastic Opponent Modeling Agents* (SOMA) [103, 62, 61] framework for modeling agent behavior and forecasting the most probable actions that may be taken in a particular situation. SOMA uses action probabilistic logic programs (*ap*-programs), which are extensions of the PLPs described in [79]. This probabilistic logic-based approach to knowledge representation lends SOMA not only to automated reasoning over such models, but in producing results that can improve the users' understanding of dependencies or relationships among complex sets of variables. The forecasting component of SOMA involves computing the most probable world, or set of worlds, from an *ap*-program; this problem is briefly reviewed as well as several algorithms that can be used to solve it. Because finding the most probable world is extremely computationally expensive—especially in real-world security situations, which can contain on the order of $10^{30,000}$ possible worlds—several distributed algorithms are introduced that can be used to improve the computation time, scalability, and accuracy of the most probable world computation. These explicitly parallel procedures make SOMA amenable as a decision-support technology for modeling and forecasting agent behavior, as the greater efficiency may provide opportunities for multiple iterative forecasts and more in-depth analysis.

Next, several forecasting methods are presented for predicting an agent's behavior and the situations it might encounter directly from time series behavioral data. Chapter 3 provides a detailed description of CONVEX [75], which forecasts the actions an agent might take in a given situation based on how it has responded to similar situations in the past. The SitCAST Situation Forecaster [108] is then described in Chapter 4 as a system that can be applied to time series data to produce a probabilistic forecast of the *possible situations* that might occur at a specific time in the future. The results of a SitCAST forecast can be used as input into the CONVEX forecasting engine, predicting what an agent will do at a specific time, along with a probability distribution over these behaviors.

The combined SitCAST+CONVEX forecasting procedure essentially predicts what an agent will do at a given time point, assuming it will act similarly to the statistically dominant past behaviors; this represents a forecasting framework for the "normal operating procedures" of the agent. However, it is often more important to understand how and when an agent will change its behavior or strategy than when it will adhere to the norm. Chapter 5 presents the *Change Analysis Predictive Engine* (CAPE) [108] for forecasting *changes* in an agent's behavior that cannot be captured by SitCAST and CONVEX. Here, the general architecture of the *Change Analysis Predictive Engine* CAPE system is discussed as well as two algorithms—

the CAPE-Learn algorithm for constructing a model of behavioral changes and the CAPE-Forecast algorithm for forecasting an agent's behavior at a given time in the future taking into account the potential for behavioral change.

While time series forecasting is a very diverse field—indeed, the methods presented here are themselves quite distinct—there are generalized principles of forecasting that can be identified across all of these approaches. Chapter 6 introduces a general theoretical framework for integrating forecasting into temporal databases. The notion of a *forecast operator* that can be used on time series data is defined axiomatically, and several broad classes of such operators are identified and explored.

Once we can model, forecast, and explain the behaviors of other agents, the next step is to use this knowledge to formulate effective counterstrategies or mitigating policies. Chapter 7 presents the *Actionable State Change Attempts* formalism, a novel framework for generating optimal ways to change the state of the world given a goal regarding another agent's behavior. Rather than looking only at what another agent will do, this approach can identify the optimal countermeasures—balancing cost and likelihood of success—for changing the state of the world in such a way to induce desired behaviors or outcomes on the part of the opponent agent. Using time series behavioral knowledgebases, an algorithm is given for finding this optimal state change attempt, as well as a trie-indexed enhancement that improves the running time and scalability of the approach. These methods are tested on real data from the U.S. education system [112] to demonstrate their efficiency when applied to social policy analysis. Further experimental analysis using synthetic data analyzes the algorithms' ability to choose the best policies. Similarities between planning under uncertainty and traditional machine learning are explored, and for this problem the Actionable State Change Attempts framework provides a more intuitive formal representation and empirical success.

As described above, all of these behavioral modeling and forecasting methods can be applied to real-world international security situations. In Chapter 8, prototype applications of the above techniques are described. Specifically, the *SOMA Terror Organization Portal* (STOP) [107] is presented as an online decision-support tool that allows analysts to apply the forecasting procedures of the preceding chapters to behavioral data on terror organizations from the Minorities at Risk Organizational Behavior dataset [46, 125]. STOP also incorporates an important social networking component, allowing for collaboration and cooperation among analysts working in similar fields. In this section several use-cases of behavioral forecasting methods are described, examining how the Stochastic Opponent Modeling Agents framework from Chapter 2 has been used to understand and forecast the behavior of actors in the Afghan drug economy and how STOP has been used to analyze the terrorist organizations Hezbollah and Hamas.

Finally, Chapter 9 contains concluding remarks and a discussion of ongoing and future work for behavioral forecasting and policy analysis.

Chapter 2

Distributed Computation in Stochastic Behavioral Models

The work described in this chapter has appeared in [103], [62], and [61].

Probabilistic logic programs (PLPs) [79] have been proposed as a paradigm for probabilistic logical reasoning with no independence assumptions. PLPs use a possible worlds model based on prior work by [53], [44], and [82] to induce a set of probability distributions on a space of possible worlds. Past work on PLPs [80, 79] focuses on the entailment problem of checking if a PLP entails that the probability of a given formula lies in a given probability interval.

However, PLPs can also be used as a stochastic representation of agent behavior and are the foundation of the Stochastic Opponent Modeling Agents (SOMA) framework. Agent models can provide information on the types of behavior that can be expected in various scenarios, which is relevant in many contexts—from multi-player computer games to studies of economic activity to issues of international affairs and public policy. Several applications are currently being developed for cultural adversarial reasoning [117] where PLPs and their variants are used in the SOMA framework to build a model of the behavior of certain agents. Thus far models have been built of several socio-cultural-economic groups in different parts of the world, including Afghan and Pakistani tribes such as the Afridis, Shinwaris, and Waziris, approximately 50 violent ethnopolitical organizations in the Middle East and Asia Pacific regions, various stakeholders in the Afghan drug economy, political parties such as the Pakistan People's Party, as well as nation states. Of course, all of these models only capture a limited set of possible actions that these entities might take in any given situation. Such PLPs contain rules that state things such as, "There is a 50 to 70% probability that group g will take action(s) A when condition C holds." In such applications, the problem of interest is that of finding the most probable action (or sets of actions) that the group being modeled might take. This corresponds precisely to the problem of finding a "most probable world" (MPW), and the PLP formalism is able to solve this problem without making any independence assumptions.

Several exact and heuristic algorithms have been developed for finding the most probable world [62, 61], allowing for better scalability with a high level of accuracy when compared to the naive solution to this problem. However, while these algorithms are able to reduce the computation time necessary to solve the MPW problem and can be applied to problems with up to $10^{30,000}$ worlds, even better results can be achieved by utilizing the concurrent resources provided by a computing cluster. The focus of this chapter is on distributing the computational load posed by the MPW problem with several explicitly parallel algorithms. These algorithms allow for a reduction in computation time, an increase in the accuracy of heuristic solutions, and the ability to further scale the computations to even larger PLPs, making the SOMA framework more amenable to a host of real-world problems.

Behavioral models using PLPs can be constructed automatically from real data using any number of well-known machine learning approaches, such as decision trees. In this chapter, one possible method, the APEX algorithm, is presented for extracting such rules from a relational database. However, there are many possible algorithms for finding these rules, and APEX is given merely as a basic proof of concept. A prototype implementation of this process is also described and applied to political science data regarding the behavior of violent ethnopolitical organizations in the Middle East. These behavioral models have produced tangible results of use to U.S. military officers and show promise in applications to other datasets and cultural reasoning domains [14, 115]. A more thorough discussion of applications will be presented later in Chapter 8.

In the remainder of this chapter, Section 2.1 first recalls the syntax and semantics of PLPs [80, 79] that are utilized by the SOMA framework. The *most probable world* (MPW) problem is defined by immediately using the linear programming methods of [80, 79]—these methods are exponential because the linear programs are exponential in the number of ground atoms in the language. Then, in Section 2.2.1, several algorithms for solving the MPW problem are reviewed. Distributed algorithms for improving the performance of finding the most probable world are presented in Section 2.3. In Section 2.4 a method for extracting PLPs automatically from real data is given and applications of the entire framework are discussed. Section 2.5 describes a prototype implementation of the parallelized PLP framework and includes a set of experiments to assess these distributed algorithms.

2.1 Action Probabilistic Logic Programs

The Stochastic Opponent Modeling Agents (SOMA) framework is a language that can be used for modeling behavior in diverse situations, including cultural-adversarial situations in international affairs. This framework makes use of action probabilistic logic programs (*ap*-programs), an immediate and obvious variant of the probabilistic logic programs introduced in [80, 79]. Assume the existence of a logical alphabet that consists of a finite set $\mathcal{L}_{cons}$ of constant symbols, a finite set $\mathcal{L}_{pred}$ of predicate symbols (each with an associated arity), and an infinite set $\mathcal{V}$ of variable symbols. Function symbols are not allowed in this language. Terms and atoms are defined in the usual way [68]. Also assume that a subset $\mathcal{L}_{act}$ of $\mathcal{L}_{Pred}$ are designated as *action symbols*—these are symbols that denote some action. Thus, an atom $p(t_1, \ldots, t_n)$, where $p \in \mathcal{L}_{act}$, is an *action atom.* Every (resp. action) atom is a (resp. action) wff. If F, G are (resp. action) wffs, then $(F \wedge G), (F \vee G)$ and $\neg G$ are all wffs (resp. action wffs).

Definition 1 (*p*-annotation/*ap*-annotation)**.** *If F is a wff (resp. action wff) and $\mu = [\alpha, \beta] \subseteq [0, 1]$, then $F : \mu$ is called a p-annotated (resp. ap-annotated—short for "action probabilistic" annotated) wff. μ is called the p-annotation (resp. ap-annotation) of F.*

Without loss of generality, assume that F is in conjunctive normal form (i.e., it is written as a conjunction of disjunctions).

Definition 2 (ap-rule)**.** *If F is an action formula, $A_1, A_2, ..., A_m$ are action atoms, $B_1, \ldots, B_n$ are non-action atoms, and $\mu, \mu_1, ..., \mu_m$ are ap-annotations, then $F : \mu \leftarrow A_1 : \mu_1 \wedge A_2 : \mu_2 \wedge \ldots \wedge A_m : \mu_m \wedge B_1 \wedge \ldots \wedge B_m$ is called an* *ap*-rule. *If this rule is named c, then* $Head(c)$ *denotes* $F : \mu$*;* $Body^{act}(c)$ *denotes* $A_1 : \mu_1 \wedge A_2 : \mu_2 \wedge ... \wedge A_m : \mu_m$ *and* $Body^{state}(c)$ *denotes* $B_1 \wedge \ldots \wedge B_n$*.*

Intuitively, the above ap-rule says that an entity (e.g., a group g, a person p, etc.) *will take action* F *with probability in the range* μ *if* $B_1, \ldots, B_n$ *are true in the current state (a term that will be defined shortly) and if the entity will take each action* A_i *with a probability in the interval* μ_i *for* $1 \leq i \leq n$*.*

Definition 3 (ap-program)**.** *An* action probabilistic logic program *(ap-program for short) is a finite set of ap-rules.*

Figure 2.1 shows a sample rule base consisting of some automatically derived *ap*-rules about Hezbollah using behavioral data from the Minorities at Risk Organizational Behavior (MAROB) dataset [125, 46]. The behavioral data in MAROB has tracked over 118 ethnopolitical organizations across the Middle East and North Africa for about 25 years from 1980 to 2004. For each year, values have been gathered for about 175 measurable variables for each group in the sample [125, 46]. These variables include strategic conditions such as the tendency to commit bombings and armed attacks, as well as background information about the type of leadership, whether the group is involved in cross border violence, etc. The automatic derivation of these rules was based on the straightforward data mining algorithm that will be discussed in Section 2.4. Figure 2.1 contains four of these extracted rules for the group Hezbollah, describing some conditions under which it has used a particular strategy, along with probability range. For example, the third rule indicates that when Hezbollah has a strong, single leader and its popularity is moderate, its propensity to conduct armed attacks is 42 to 53%. However, when it has had a standing military, its propensity to conduct armed attacks is 93 to 100%.

1.	*kidnap:* $[0.35, 0.45]$	$\leftarrow$	*interOrganizationConflicts.*
2.	*kidnap:* $[0.60, 0.68]$	$\leftarrow$	*unDemocratic* $\wedge$ *internalConflicts.*
3.	*armed_attacks:* $[0.42, 0.53]$	$\leftarrow$	*typeLeadership(strongSingle)* $\wedge$ *orgPopularity(moderate).*
4.	*armed_attacks:* $[0.93, 1.0]$	$\leftarrow$	*statusMilitaryWing(standing).*

Figure 2.1: Four simple rules for modeling the behavior of a terrorist organization.

Definition 4 (world/state)**.** *A* world *is any set of ground action atoms. A* state *is any finite set of ground non-action atoms.*

Note that both worlds and states are just ordinary Herbrand interpretations. As such, it is clear what it means for a state to satisfy $Body^{state}$.

Definition 5 (Reduction of an *ap*-program)**.** *Let* Π *be an ap-program and* s *a state. The* reduction of Π w.r.t. s*, denoted by* Π_s *is* $\{F : \mu \leftarrow Body^{act} \,|\, s$ *satisfies* $Body^{state}$ *and* $F : \mu \leftarrow Body^{act} \;\wedge\; Body^{state}$ *is a ground instance of a rule in* $\Pi\}$.

Note that Π_s ***never has any non-action atoms in it.***

A fixpoint operator T_{Π_s} is associated with an *ap*-program Π and a state s and maps sets of ground *ap*-annotated wffs to sets of ground *ap*-annotated wffs.

Definition 6 ($U_{\Pi_s}(X)$)**.** *Suppose* X *is a set of ground action atoms. An intermediate operator* $U_{\Pi_s}(X)$ *is defined as follows.* $U_{\Pi_s}(X) = \{F : \mu \mid F : \mu \leftarrow A_1 : \mu_1 \;\wedge \cdots \wedge\; A_m : \mu_m$ *is a ground instance of a rule in* Π_s*, and for all* $1 \leq j \leq m$*, there is an* $A_j : \eta_j \in X$ *such that* $\eta_j \subseteq \mu_j\}$.

Intuitively, $U_{\Pi_s}(X)$ contains the heads of all rules in Π_s whose bodies are deemed to be "true" if the action wffs in X are true.

In order to assign a probability interval to each ground action atom, the procedure from [80] is used where a linear program $CONS_U(\Pi, s, X)$ is derived from $U_{\Pi_s}(X)$ as follows. For each world w_i, let p_i be a variable denoting the probability of

w_i being the "real world." As each w_i is just a Herbrand interpretation, the notion of satisfaction of an action formula F by a world w, denoted by $w \mapsto F$, is defined in the usual way. The following constraints are in $CONS_U(\Pi, s, X)$:

1. If $F : [\ell, u] \in U_{\Pi_s}(X)$, then $\ell \leq \Sigma_{w_i \mapsto F}\, p_i \leq u$ is in $CONS_U(\Pi, s, X)$.

2. $\Sigma_{w_i} p_i = 1$ is in $CONS_U(\Pi, s, X)$.

These constraints are referred to as type (1) and (2), respectively. The fixpoint operator $T_{\Pi_s}(X)$ can now be defined.

Definition 7 ($T_{\Pi_s}(X)$)**.** *Suppose* Π *is an ap-program, s is a state, and X is a set of ground ap-wffs. The operator $T_{\Pi_s}(X)$ is then defined as* $\{F : [\ell(F), u(F)] \mid (\exists\mu)\, F : \mu \in U_{\Pi_s}(X)\} \cup \{A : [\ell(A), u(A)] \mid A$ *is a ground action atom*$\}$.

Thus, $T_{\Pi_s}(X)$ works in two phases. It first takes each formula $F : \mu$ that occurs in $U_{\Pi_s}(X)$ and finds $F : [\ell(F), u(F)]$ and puts this in the result. Once all such $F : [\ell(F), u(F)]$ have been put in the result, it tries to infer the probability bounds of all ground action atoms A from these *ap*-formulas. The $T_{\Pi_s}(X)$ operator has a least fixpoint, $T^{\omega}_{\Pi_s}$, which contains all of the ground action atoms in X annotated with tight probability intervals.

2.2 Maximally Probable Worlds

As explained through the above Hezbollah example, it may be necessary to know what actions a group might take in a given situation. Solving this behavioral forecasting problem is exactly the problem of finding the most probable world given an *ap*-program and a state.

Definition 8 (lower/upper probability of a world)**.** *Suppose* Π *is an ap-program and* s *is a state. The* lower probability, $\mathsf{low}(w_i)$, *of a world* w_i *is defined as:* $\mathsf{low}(w_i) =$ **minimize** p_i ***subject to*** $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$*. The* upper probability, $up(w_i)$, *of world* w_i *is defined as* $up(w_i) =$ **maximize** p_i ***subject to*** $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$*.*

Thus, the low probability of a world w_i is the lowest probability that world can have in any solution to the linear program. Similarly, the upper probability for the same world represents the highest probability that world can have. It is important to note that for any world w_i, a point probability cannot be *exactly* determined. This observation is true even if all rules in Π have a point probability in the head because this framework does not make any simplifying assumptions (e.g., independence). Checking if the low (resp. *up*) probability of a world exceeds a given bound is in the class EXPTIME [61].

The MPW Problem. The *most probable world* problem (MPW for short) is the problem where, given an *ap*-program Π and a state s as input, we are required to find a world w_i where $\mathsf{low}(w_i)$ is maximal.[1]

A Naive Algorithm. A *naive* algorithm to find the most probable world would be to directly find $\mathsf{low}(w_i)$ for each world w_i as follows:

1. Compute $T^{\omega}_{\Pi_s}$; $Best = NIL$; $Bestval = 0$.

2. For each world w_i do:

   (a) Compute $\mathsf{low}(w_i)$ by minimizing p_i subject to $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$.

[1]A similar **MPW-Up Problem** can also be defined. The *most probable world-up* problem (MPW-Up) is: given an *ap*-program Π and a state s as input, find a world w_i where $\mathsf{up}(w_i)$ is maximal. However, here the MPW problem is only addressed as stated.

(b) If $\mathsf{low}(w_i) > Bestval$ then set $Best = w_i$ and $Bestval = \mathsf{low}(w_i)$.

3. If $Best = NIL$, then return any world whatsoever, else return $Best$.

The naive algorithm does a brute force search after computing $T^{\omega}_{\Pi_s}$, finding the lower bound probability for each world and choosing the best one.

There are two key problems with this algorithm. The first is that in Step (1), computing $T^{\omega}_{\Pi_s}$ is very difficult. When some syntactic restrictions are imposed, this problem can be solved without linear programming at all, such as when Π is a PLP (or *p*-program as defined in [79]) where all heads are atomic.

The second problem is that in Step 2(a), the number of (linear program) variables in $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ is exponential in the number of ground action atoms, as there are $2^{card(\mathcal{L}_{act})}$ possible worlds for $card(\mathcal{L}_{act})$ ground actions. When $card(\mathcal{L}_{act})$ is only 20—which is a rather limited set of actions for modeling agent behavior—the linear program contains over a million variables. However, when the number of actions is increased only slightly to 30 or 40 (or more in many real-world problems), the number of variables will be inordinately large.

2.2.1 Efficient Algorithms for Solving the MPW Problem

While the naive algorithm is guaranteed to find a correct solution to the set of constraints in $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, it quickly becomes computationally intractable as the number of ground atoms increases. [62] presents several additional algorithms that improve the computation time for finding the most probable world by reducing the number of worlds included as variables in the linear program. Two of these algorithms exploit the concept of head-oriented processing, looking at equivalence classes of worlds that satisfy the same rules in the *ap*-program; these methods still produce an exact solution for finding the most probable world. A heuristic approach is also used to further reduce the size of the constraints, randomly sampling the space of worlds to decide which variables to include in the linear program and using a binary search process to adjust the lower-bound constraints accordingly.

In this section, these algorithms are briefly reviewed, as the distributed approaches described in Section 2.3 can be applied to any of these methods for finding the MPW.

Head-Oriented Processing

The head-oriented processing algorithms are based on the recognition that the linear program $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ is often over parameterized, containing a variable for each world even though several worlds together may actually represent a single probability mass. Using this intuition, worlds can be grouped into equivalence classes, reducing the number of variables required to specify the linear constraints.

Given a world w, state s, and *ap*-program Π, let $Sat(w) = \{F \mid c$ is a ground instance of a rule in Π_s and $Head(c) = F : \mu$ and $w \mapsto F\}$. Intuitively, $Sat(w)$ is the set of heads of rules in Π_s that are satisfied by w.

Definition 9 ($\sim$-equivalence of worlds)**.** *Suppose* Π *is an ap-program,* s *is a state, and* w_1, w_2 *are two worlds . Worlds* w_1 *and* w_2 *are* equivalent*, denoted* $w_1 \sim w_2$*, iff* $Sat(w_1) = Sat(w_2)$*.*

In other words, two worlds are equivalent iff the they satisfy the formulas in the heads of exactly the same rules in Π_s. It is easy to see that $\sim$ is an equivalence relation, and $[w_i]$ denotes the $\sim$-equivalence class to which a world w_i belongs.

The Head-Oriented Processing (HOP) algorithm uses this equivalence relation to reduce the number of variables in $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$. The key insight is that for any $\sim$-equivalence class $[w_i]$, the summation $\Sigma_{w_j \in [w_i]} p_j$ either appears *in its entirety* in each constraint of type (1) in $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ or does not appear at all.

Effectively, the number of variables in the linear program has been reduced from $2^{card(\mathcal{L}_{act})}$ to $2^{card(\Pi_s)}$—a savings that can be significant in some cases (though not always!). The number of constraints in the linear program stays the same. This *reduced set of constraints* using equivalence classes is formally defined as follows.

Definition 10 ($\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$)**.** *For each equivalence class $[w_i]$, there is a variable p'_i in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ denoting the summation of the probability of each of the worlds in $[w_i]$. $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ contains the following constraints:*

1. *If $F : [\ell, u]$ is in $T^{\omega}_{\Pi_s}$, then $\ell \leq \Sigma_{[w_i] \mapsto F} p'_i \leq u$ is in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$.*

2. *$\Sigma_{[w_i]} p'_i = 1$ is in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$.*

Here, $[w_i] \mapsto F$ in constraint (1) means that the worlds in $[w_i]$ satisfy F.

Before reviewing the HOP algorithm, some additional notation is required. Let $\mathit{FixedWff}(\Pi, s) = \{F | F : \mu \in T^{\omega}_{\Pi_s}\}$. Given $X \subseteq FixedWff(\Pi, s)$, $Formula(X, \Pi, s)$ is defined as:

$$\bigwedge_{G \in X} G \wedge \bigwedge_{G' \in \mathit{FixedWff}(\Pi, s) - X} \neg G'.$$

$Formula(X, \Pi, s)$ is the formula where X consists of all and only those formulas in $FixedWff(\Pi, s)$ that are true. Given two sets $X_1, X_2 \subseteq FixedWff(\Pi, s)$, $X_1 \approx X_2$ iff $Formula(X_1, \Pi, s)$ and $Formula(X_2, \Pi, s)$ are logically equivalent.

The HOP Algorithm:

1. Compute $T^{\omega}_{\Pi_s}$; $Best = NIL$; $Bestval = 0$.

2. Let $[X_1], \ldots, [X_n]$ be the $\approx$-equivalence classes defined above for Π, s.

3. For each equivalence class $[X_i]$ do:

   (a) If there is exactly one interpretation that satisfies $Formula(X_i, \Pi, s)$, then

      i. **Minimize** p'_i **subject to** $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ where $[w_i]$ is the set of worlds satisfying exactly those heads in X_i. Let Val be the result.

      ii. If $Val > Bestval$, then set $Best = w_i$ and $Bestval = Val$.

4. If $Bestval = 0$ then return any equivalence class whatsoever otherwise return $Best$.

Though the complexity of HOP is also exponential, it may be preferable to the naive algorithm due to the reduced number of variables in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$. However, the required satisfiability checks to compute the equivalence classes can often make HOP intractable in practice.

A variant of the HOP algorithm, called SemiHOP, tries to avoid computing the full equivalence classes. The SemiHOP algorithm finds sub-partitions rather than finding pairs of sets that represent the same equivalence class, therefore omitting the checks for logical equivalence of every possible pair.

Definition 11 (Sub-partition)**.** *A* sub-partition *of the set of worlds of* Π *w.r.t.* s *is a partition* $W_1, \ldots, W_k$ *where:*

1. $\bigcup_{i=1}^{k} W_i$ *is the entire set of worlds.*

2. For each W_i*, there is an equivalence class* $[w_i]$ *such that* $W_i \subseteq [w_i]$*.*

A sub-partition can be generated by looking at all subsets of $FixedWff(\Pi, s)$, i.e., all possible subsets of ap-wffs in $T^{\omega}_{\Pi_s}$.

The SemiHOP algorithm uses subsets of equivalence classes instead of full equivalence classes. The size of the constraints is still reduced (through the elimination of some variables), though the reduction is not necessarily maximal.

Definition 12 ($\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$)**.** *Let $W_1, \ldots, W_k$ be a sub-partition of the set of worlds for Π and s. For each W_i, $\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ uses a variable $p^{\star}_i$ to denote the summation of the probability of each of the worlds in W_i. $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ contains the following constraints:*

1. *If $F : [\ell, u]$ in $T^{\omega}_{\Pi_s}$, then $\ell \leq \Sigma_{W_i \mapsto F} p^{\star}_i \leq u$ is in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$.*
2. *$\Sigma_{W_i} p^{\star}_i = 1$ is in $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$.*

Here, $W_i \mapsto F$ in constraint (1) implies that the worlds in W_i satisfy F.

SemiHOP Algorithm:

```
1. Compute $T^{\omega}_{\Pi_s}$; $Best = NIL$; $Bestval = 0$.
2. For each set $X \subseteq FixedWff(\Pi, s)$ do:
    (a) If there is exactly one interpretation that satisfies $Formula(X, \Pi, s)$ then
        i. **Minimize** $p^{\star}_i$ **subject to** $\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ where $W_i$ is a sub-
           partition of the set of worlds of $\Pi$ w.r.t. $s$. Let $Val$ be the result.
        ii. If $Val > Bestval$ set $Best = w_i$ and $Bestval = Val$.
3. If $Bestval = 0$ then return any sub-partition whatsoever otherwise return
   $Best$.
```

The key advantage of SemiHOP over HOP is that the set $[w_i]$ of worlds does not need to be constructed, i.e., finding the equivalence classes $[w_i]$ is not necessary. This advantage comes with a drawback, though, since the size of the set $\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$ can be a bit bigger than $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$

Randomized Heuristic Approximations

Even though HOP and SemiHOP try to reduce the number of constraint variables when finding the most probable world, the size of the linear program can still be quite large, especially in real-world behavioral modeling applications. A randomized heuristic algorithm can be used in conjunction with any of the exact approaches (naive, HOP, SemiHOP) to reduce the number of variables even further.

Let $\mathcal{C}$ be the set of constraints generated by either naive, HOP, or SemiHOP. In the randomized algorithm, an *a priori* commitment is made to only look at some set S_k of k variables from the linear program, eliminating all variables not in S_k from any summation in $\mathcal{C}$ and deriving a modified set of constraints $\mathcal{C}'$.

It is immediately apparent that, as all the lower bounds are set to ℓ, a solution to $\mathcal{C}'$ may or may not exist. Rather than weakening the lower bound from ℓ to 0 (which would guarantee a solution), a *binary heuristic* is used to *modify the lower bounds* of such constraints as little as possible. If $\mathcal{C}'$ is solvable by itself, then each variable in the random sample S_k is simply minimized subject to $\mathcal{C}'$ and the most probable world is returned. If not, then the lower bound of one or more constraints in $\mathcal{C}'$ is decreased as follows to derive a new set of constraints $\mathcal{C}^{\bullet}$.

Suppose $c^{\star} \in \mathcal{C}'$ is a constraint where $\ell^{\star} \leq \Sigma_{q_i \in S_k} q_i \leq u$. The heuristic tries to replace $\ell^{\star}$ by $\frac{\ell^{\star}}{2}$. If this yields a solvable set of equations, $\frac{\ell^{\star}}{2}$ is replaced by $\frac{3 \times \ell^{\star}}{4}$.

If the resulting system is unsolvable, it is replaced with $\frac{5\times\ell^\star}{8}$, and so forth. This method is called the *binary heuristic* because it resembles a binary search.

Once this process of modifying the lower bounds has completed, each variable in S_k is minimized subject to the new constraints $\mathcal{C}^\bullet$. The variable with the highest minimal value—the approximate MPW w.r.t. the random sample—is returned.

2.3 Parallel Algorithms for Finding a Maximally Probable World

In the previous sections, several algorithms were reviewed that can be used to solve the most probable world problem. However, even with the given constraint simplifications and heuristic approximations, the computation time and resource requirements of these methods will not always permit the desired level of performance. This problem is especially true considering the scale of real-world behavioral modeling applications (which may easily contain 32,000 ground action atoms, or on the order of $2^{32{,}000} \approx 10^{9{,}900}$ possible worlds) and the need for users to produce multiple iterative "what if" scenarios and forecasts.

In this section various parallel algorithms are presented that leverage the sequential naive, HOP, SemiHOP, and binary heuristic algorithms. Parallelism will not only reduce the computation time necessary for finding the most probable world, but will also provide increased scalability, allowing a larger number of behaviors (i.e., ground action atoms) to be modeled and permitting analysis of larger *ap*-programs. In addition, through parallel sampling in the randomized binary heuristic, a larger proportion of this huge space of possible worlds can be covered, improving the accuracy of the approximation result.

```
Algorithm PMPW($\mathsf{CONS}, m, N$)
1.      $Batches = divideVariables(\mathsf{CONS}, m)$
2.      $MaxVals = \emptyset$
3.      for each parallel process $n := 0$ to $N-1$ do
4.          $b_n$ = $batch_n \in Batches$
5.          $MAX_n = 0$
6.          for each variable $p_i \in b_n$ do
7.              $VAL(p_i)$ = minimize $p_i$ subject to $\mathsf{CONS}$
8.              if $VAL(p_i) > MAX_n$ set $MAX_n = VAL(p_i)$
9.          add $MAX_n$ to $MaxVals$
10.     $BestVal = \max(MaxVals)$
11.     $Best = \arg\max(MaxVals)$
12.     return $Best$
```

Figure 2.2: The general PMPW parallel algorithm for computing the most probable world.

Parallelism for Reducing Computation Time

All of the algorithms discussed in Section 2.2 lend themselves to distributed computation in a fairly straightforward, pleasantly parallel way. A new class of algorithms, the Parallel Most Probable World (PMPW) algorithms, operate identically to the serial naive, HOP, SemiHOP, or binary heuristics, except that the computation of $\mathsf{low}(w_i)$ for each world w_i (or equivalence class $[w_i]$) is distributed among N nodes of a computing cluster such that m worlds (resp. equivalence classes) at a time are given to each node. Figure 2.2 contains the basic PMPW algorithm.

PMPW is intentionally designed to be general and applicable to any of the serial MPW algorithms. The input parameter CONS is a set of linear constraints that can be computed as $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, $\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, or the $\mathcal{C}^{\bullet}$ returned by the binary heuristic. The procedure $divideVariables$ performs the division of the variables in CONS into batches of size m to be distributed across the N parallel processes. This number of worlds (resp. equivalence classes) for which to compute the low value on each node can be determined in several ways. The most obvious is to simply divide the problem evenly across all of the N nodes such that $m = \frac{|\mathsf{CONS}|}{N}$ where $|\mathsf{CONS}|$ is the number of variables in the constraints.

Beginning on Line 3, PMPW finds the minimum probability for each variable p_i in $batch_n$ assigned to parallel process n, storing the maximum of these values in $MaxVals$. After completing all of the distributed computation, the overall most probable world is returned. In the best case, this division of labor in the PMPW algorithm can allow for a computation time improvement of up to a factor of N for N parallel processes.

Parallelism for Increasing Computation Capacity

The computation speedup afforded by the PMPW algorithm can help improve the scalability of the MPW problem. For any given amount of time, a greater number of possible worlds or equivalence classes can be processed. However, rather than simply distributing the MPW algorithms and performing the same computations in parallel, another "pleasantly parallel" algorithm can be designed to better address the issues of scale in finding the most probable world of larger *ap*-programs, i.e., programs with a greater number of ground action atoms. In the PMPW-LR algorithm presented below, the structure of an *ap*-program Π is exploited to divide the associated linear constraints into distinct components for concurrent most probable world computations.

Before presenting the PMPW-LR algorithm, a structural representation of the relationships among actions in an *ap*-program must be defined. An *ap*-program Π can be represented as a graph in which the vertices are literals in the program, and the edges indicate co-occurrence of these literals in an *ap*-rule.

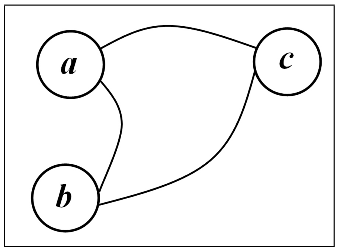

Figure 2.3: The literal-relationship graph G_Π for a simple *ap*-program Π.

Definition 13 (Literal Relationship (LR) Graph)**.** *Let* Π *be an* ap-*program. The* literal relationship graph *(LR-graph)* $G_\Pi = (V, E)$ *is an* undirected *graph defined as follows.*

$V = \{l \mid l$ *is a literal (positive or negative) appearing in a rule in* $\Pi\}$*.*

$E = \{(l_i, l_j) \mid l_i, l_j \in V$ *and* l_i *and* l_j *are either complementary literals or they both appear in a rule in* $\Pi\}$*.*

$G_\Pi = (V, E)$ *denotes the* Literal Relationship Graph *for the program* Π.

For example, consider a simple *ap*-program Π:

$$\begin{array}{lll} (a \vee b) & : [0.7, 1] & \leftarrow . \\ ((a \wedge b) \vee (b \wedge c)) & : [0.2, 0.6] & \leftarrow . \\ (a) & : [0.4, 0.4] & \leftarrow . \end{array}$$

Figure 2.3 shows the LR-graph associated with Π.

The *rank* of an LR-graph is the maximum cardinality of the connected components of the graph.

Definition 14 (Rank of an *LR*-graph)**.** *Let* Π *be an* ap-*program and* $G_\Pi = (V, E)$ *be the LR-graph for* Π*. Graph* G_Π *has* rank k *if* k *is the maximum cardinality of any connected component in* G_Π*.*

For example, when the rank of the LR-graph G_Π is 1, this means that all rules in Π are only literals, and there are no complementary literals. The graph in the example above has rank 3. On the other hand, if the second probabilistic statement was deleted from the program Π for Figure 2.3, the result would be a graph of rank 2. Note that the rank of an LR-graph can be computed in polynomial time (w.r.t. the size of the graph), and the LR-graph itself can also be computed in polynomial time. As a consequence, checking if the LR-graph's rank is below some *a priori* set bound b is a polynomial-time operation.

Each connected component c in an LR-graph G_Π represents a subprogram Π_c of Π that utilizes only the literals in that component. Therefore, each connected component comprises a separate set of linear constraints $\mathsf{CONS}(\Pi_c, s, T^{\omega}_{\Pi_s})$, $\mathsf{RedCONS}_U(\Pi_c, s, T^{\omega}_{\Pi_s})$, $\mathsf{S\_RedCONS}_U(\Pi_c, s, T^{\omega}_{\Pi_s})$, or $\mathcal{C}^{\bullet}$. By finding the maximally probable world in each component, these individual solutions can be compared to find the MPW across all components of the original set of constraints. The PMPW-LR algorithm given in Figure 2.4 uses this methodology to divide a much larger *ap*-program into smaller pieces that can be computed in parallel across N nodes.

The PMPW-LR algorithm first calls the procedure $buildLR(\Pi)$ to construct the LR-graph for the *ap*-program Π. Using this graph, the function $getComponents(G)$ returns the connected components of G. The remainder of this algorithm is similar to the general PMPW algorithm in Figure 2.2. First, the components in C are divided amongst the N parallel processes. Then, on each node the most probable world is found for the assigned components w.r.t. $\mathsf{CONS}(c_i)$, the portion of the linear constraints associated with the component c_i. In Lines 13 and 14, these results are combined to return the overall most probable world.

While the PMPW-LR approach does not provide any explicit savings with regard to the computation time, it does allow the analysis of much larger *ap*-programs that can be divided into computationally feasible parallel components. However,

```
Algorithm PMPW-LR($\Pi$, CONS, $N$)
1.    $G = buildLR(\Pi)$
2.    $C = getComponents(G)$
3.    $MaxVals = \emptyset$
4.    $BatchSize = \lceil C/N \rceil$
5.    for each parallel process $n := 0$ to $N-1$ do
6.       $MAX_n = 0$
7.       for each component $c_i \in C$, $i = (BatchSize * n)$ to
               $[(BatchSize * n) + BatchSize - 1]$ do
8.          if $i \geq |C|$ then END
9.          for each variable $p_j$ in CONS($c_i$) do
10.            $VAL(p_j)$ = minimize $p_j$ subject to CONS($c_i$)
11.            if $VAL(p_j) > MAX_n$ set $MAX_n = VAL(p_j)$
12.      add $MAX_n$ to $MaxVals$
13.   $BestVal = \max(MaxVals)$
14.   $Best = \arg\max(MaxVals)$
15.   return $Best$
```

Figure 2.4: The PMPW-LR algorithm for computing the most probable world using distributed linear programs. CONS(c_i) denotes the portion of the constraints associated with the LR-graph component c_i.

the success of PMPW-LR is highly dependent on the structure of the underlying *ap*-program. For example, if an *ap*-program Π produces a graph of rank $card(\mathcal{L}_{act})$, then there is only a single component (i.e., the original program Π) in which case PMPW-LR will be identical to the serial algorithms.

Parallelism for Improving Solution Accuracy of Heuristics

The randomized binary heuristic was introduced in Section 2.2 as an approximation algorithm for finding the most probable world using a random sample of the set of all possible worlds (resp. equivalence classes). In addition to improving the running time and scalability of solving the MPW problem, distributed algorithms can also be utilized to improve the quality and accuracy of these approximation solutions by examining a greater portion of the sample space. In this section, two distributed approximation algorithms are described. The first, PMPW-BH, looks at concurrent random samples of worlds or equivalence classes using the binary heuristic independently across N parallel processes. The second algorithm, PMPW-IBH, is an anytime iterative sampling approach that will examine multiple successive sets of parallel random samples to find the most probable world, refining the sample set with each iteration of the parallel computation.

Figure 2.5 contains the PMPW-BH (Parallel Most Probable World–Binary Heuristic) algorithm, which is able to examine a greater proportion of the possible worlds through distributed application of the binary heuristic. With this method, each parallel computation investigates a distinct random sample of possible worlds (resp. equivalence classes). The procedure $Binary(\mathsf{CONS}, r)$ on Line 4 returns a set of linear constraints $\mathcal{C}^{\bullet}$, which is the result of applying the binary heuristic to a sample of r worlds (resp. equivalence classes). To avoid sampling bias, the processes are asynchronous, and each takes its own random sample of size r. The resulting most probable worlds from each sample are then compared to find the most probable world overall. Using the PMPW-BH algorithm, a greater proportion of the possible worlds may be sampled, thereby improving the chances of finding a more accurate approximate solution with respect to the solutions returned by the exact naive, HOP or SemiHOP algorithms.

The PMPW-IBH algorithm (Figure 2.6) is a version of PMPW-BH that incorporates iterative sampling to further increase its ability to examine larger samples of possible worlds and achieve greater solution accuracy. PMPW-IBH maintains a set of the k most probable worlds returned by the current iteration on each of N parallel nodes . This set of known probable worlds is then propagated to the ran-

```
Algorithm PMPW-BH(CONS, $r$, $N$)
1.    $MaxVals = \emptyset$
2.    for each parallel process $n := 0$ to $N-1$ do
3.        $MAX_n = 0$
4.        $\mathcal{C}_n^\bullet = Binary(\mathsf{CONS}, r)$
5.        for each variable $p_i$ in $\mathcal{C}_n^\bullet$ do
6.            $VAL(p_i)$ = minimize $p_i$ subject to $\mathcal{C}_n^\bullet$
7.            if $VAL(p_i) > MAX_n$ set $MAX_n = VAL(p_i)$
8.        add $MAX_n$ to $MaxVals$
9.    $BestVal = \max(MaxVals)$
10.   $Best = \arg\max(MaxVals)$
11.   return $Best$
```

Figure 2.5: The PMPW-BH parallel algorithm for computing an approximation of the most probable world using the binary heuristic .

dom sample of r worlds in the following iteration, so only $r - k$ new worlds will be sampled and the previous most probable set of k worlds will be retained.

For example, if $r = 1000$ and $k = 20$, then in the first iteration the binary heuristic will randomly select 1000 worlds to use in the MPW computation. From these results, the 20 most probable worlds will be used as part of the second iteration's sample, choosing only $r - k = 980$ new worlds and generating a new set of constraints $\mathcal{C}^\bullet$ with the binary heuristic. Using these new constraints, the 20 most probable worlds are again found and propagated to the next iteration.

PMPW-IBH continues this progressive refinement process until $maxIter$ iterations have been completed on each parallel process. Finally, in Line 12 the results from all of the MPW computations are compared and the overall most probable world is returned. As an anytime algorithm, PMPW-IBH allows the user to choose the desired level of solution refinement by setting the number of sample iterations.

The functionality of all of the PMPW algorithms can be further generalized to find the k most probable worlds from each parallel process, comparing these to

```
Algorithm PMPW-IBH(CONS, $r, maxIter, k, N$)
1.    $MaxVals = \emptyset$
2.    for each parallel process $n := 0$ to $N-1$ do
3.       for $i := 1$ to $maxIter$ do
4.          $MAX_n = 0$
5.          $prevWorlds_n = \emptyset$
6.          $\mathcal{C}_n^\bullet = modCons(prevWorlds, Binary(\mathsf{CONS}, r))$
7.          for each variable $p_j$ in $\mathcal{C}_n^\bullet$ do
8.             $VAL(p_j)$ = minimize $p_j$ subject to $\mathcal{C}_n^\bullet$
9.             if $VAL(p_j) > MAX_n$ set $MAX_n = VAL(p_j)$
10.         $prevWorlds_n = k \arg\max(MAX_n)$
11.      add $MAX_n$ to $MaxVals$
12.   $BestVal = \max(MaxVals)$
13.   $Best = \arg\max(MaxVals)$
14.   return $Best$
```

Figure 2.6: The PMPW-IBH parallel algorithm for computing the most probable world with iterative random sampling for the binary heuristic.

find the k most probable worlds overall. This additional knowledge may be useful in real-world applications, such as those described in the following section.

2.4 Applications of *ap*-programs

The Stochastic Opponent Modeling Agents (SOMA) framework is a method for stochastic behavioral modeling of agent behavior based on the *ap*-programs and reasoning algorithms described throughout this chapter. SOMA has been used to develop *ap*-programs that model the cultural, economic, and social dynamics correlated with strategic behaviors of groups engaged in various international conflicts, security situations, and political affairs. The groups modeled include tribes in the Afghan-Pakistan border region involved in the drug trade (e.g., the Shinwari, Waziri, and Mohmand tribes), about 50 terror organizations from the Middle East and Asia Pacific regions (e.g., Hezbollah, Fatah Revolutionary Council/Abu Nidal

Organization, the Kurdish group PKK), as well as political parties (e.g., Jamaat-i-Ulema Islami, Pakistan People's Party). Many of these applications and a prototype decision-support system will be described in greater detail in Chapter 8.

For each of these groups, a small set of actions that the group has taken in the past was identified. For each such action, conditions were found that are good predictors of when the group would choose it as a strategy and when they would not. These relationships were encoded as rules in the *ap*-program syntax.

The rules themselves have been developed in two ways: (i) by manually having students (and in the case of about 20 groups, terrorism experts) determine the correlations and construct the rules, and (ii) by automatically extracting them from certain datasets (behavioral time series datasets are discussed further in Chapter 3). Any well-known algorithm for producing association rules can be used to find SOMA rules as well. Here, APEX is presented as one possible algorithm for SOMA rule extraction from standard relational time series data. To extract *ap*-rules automatically, action attributes (i.e., the dependent variables describing agent behavior), which will occur in the heads of the rules, and environmental attributes, which will describe the state conditions in the bodies of the rules, must be identified in the data. The APEX algorithm consists of three main steps:

1. Select an action condition (an action attribute with an instantiated value) to be the head of the rule.

2. Fix one environmental condition as part of the body of the rule.

3. Add varying combinations of the remaining environmental conditions to the body to determine if significant correlations exist between the body conditions and the outcome condition.

The significance of a rule can be measured using the standard definitions of support and confidence from the literature.

Definition 15 (Support)**.** *For an action condition AC, an environmental condition EC, and a database DB, the* support *is defined as:*

$$S_{AC,EC} = \frac{|\{t \ s.t. \ t \in DB \wedge (AC = true \wedge EC = true)\}|}{|DB|}.$$

Definition 16 (Confidence)**.** *For an action condition AC, an environmental condition EC, and a database DB, the* confidence *is defined as:*

$$C_{AC,EC} = \frac{|\{t \ s.t. \ t \in DB \wedge (AC = true \wedge EC = true)\}|}{|\{t \ s.t. \ t \in DB \wedge EC = true\}|}.$$

The APEX algorithm calculates the difference between the confidence value produced by an environmental condition and by its negation. If this difference is above a given threshold, then the *ap*-rule is extracted and added to the behavioral model. To obtain the probability range for the extracted rule, the confidence value initially obtained is used, plus/minus the standard deviation σ of the values involved in its calculation. Note that this algorithm is not a novel one and simply performs calculations to capture interesting correlations in the data to construct rules.

The complete APEX Algorithm for a database DB with a set of action conditions AC, environmental conditions EC, and confidence difference threshold t is summarized in Figure 2.7. The final input into APEX is a special boolean function called *STAT-TESTS*, which takes an *ap*-rule as input and tests whether certain statistical conditions are satisfied by the rule. There are no restrictions whatsoever on how *STAT-TESTS* may be implemented—it could compute p-values and ensure that

```
Algorithm APEX(DB, AC, EC, k, t, STAT-TESTS)
1.      set $Rules = \emptyset$
2.      for each action condition $a_i \in AC$ do
3.         set $Head = a_i$
1.         for each condition $e_j \in EC$ do
4.            set $FixedCond = e_j$
5.            for each combination $VariedCond$ of
                   $1, 2, ..., k$ of remaining conditions
                   $v_1, v_2, ..., v_k \in EC$ do
6.               set $Body = FixedCond \wedge VariedCond$
7.               compute $PosConf = C_{Head,Body}$
8.               set $Neg = \neg(FixedCond \wedge VariedCond)$
9.               compute $NegConf = C_{Head,Neg}$
10.              set $Prob = |PosConf - NegConf|$
11.              if $Prob >= t$ $\wedge$
                   STAT-TESTS($Head : [Prob - \sigma, Prob + \sigma] \leftarrow Body$)
12.                 add $(Head : [Prob - \sigma, Prob + \sigma] \leftarrow Body)$ to
                    $Rules$
13.      Return $Rules$;
```

Figure 2.7: The APEX Algorithm.

they fall within a given bound, or it might involve a t-test, or confidence intervals, etc. The user invoking the algorithm can decide what tests, if any, are necessary.

The APEX algorithm provides a flexible way to automatically extract behavioral models from time series data. It was used to create models of the strategies of violent organizations—many of which are terrorist groups—from the Minorities at Risk Organizational Behavior (MAROB) dataset [125, 46] from the Center for International Development and Conflict Management at the University of Maryland. MAROB contains around 175 parameters to monitor about 118 ethnopolitical organizations in the Middle East and Asia Pacific regions that claim to represent the interests of repressed ethnic minorities, often employing violence and terrorism. These 175 attributes describe various aspects of the groups, such as whether or not they engaged in violent attacks, if financial or military support was received from foreign governments, and the type of leadership in each group. As an explicitly behavioral dataset, it was easy to divide these attributes into outcome conditions—or strategic actions that could be taken by the group (i.e., bombings, kidnappings, armed attacks, etc.)—and environmental conditions (i.e., the type of leadership, the kind and amount of foreign support, whether the group has a military wing, etc.). Values for these parameters are available for up to 25 years per group between 1980 and 2004, though there are fewer time points for some groups (e.g., those that have been around for a shorter duration). For each group, MAROB provides a time series as a relational table where the columns correspond to the 175 parameters and the rows correspond to the years.

Automated extraction using APEX has been applied thus far to about 50 groups in the Middle East from Morocco to Afghanistan, as well as the Asia Pacific region (specifically, Bangladesh and the Philippines). SOMA programs (*ap*-programs) for groups such as Hezbollah, Hamas, FRC-ANO, PKK, Kurdistan Democratic Party of Iran, and Hizb-i-Islami have been extracted from the MAROB data and made available in a prototype decision-support system, the SOMA Terror Organization Portal (STOP), with users throughout the national security community. In addition, in-depth case studies of the behaviors of Hezbollah and Hamas have been conducted using these automatically extracted models. For instance, approximately 14,000 *ap*-rules have been extracted for Hezbollah; some examples of these rules are given in Figure 2.8. In Chapter 8, these applications and case studies will be described in greater detail.

1. $(ARMATTACK = 1) : [0.01, 0.79] \leftarrow (ORGSUCIMPL = 1) \wedge$
   $(STATEVIOLENCE = 1) \wedge (AUTHORG = 0) \wedge (ORGST4 = 1)$
   Armed attacks are carried out with a probability between 0.01 and 0.79 if Lebanon has not come to agreement with Hezbollah, the state is not using lethal violence against Hezbollah, Lebanon is not authoritarian, and Hezbollah solicits external support only as a minor/infrequent strategy.

2. $(DSECGOV = 1) : [0.16, 0.84] \leftarrow (ORGLOC = 1) \wedge$
   $(DIASUP = 0) \wedge (INTERORGCON = 1) \wedge (MILITIAFORM = 2)$
   Domestic government/state lives and security personnel are targets of terrorism with a probability between 0.16 and 0.84 if Hezbollah is in Lebanon, Hezbollah has not received support from the Lebanese diaspora, there is inter-organizational conflict, and Hezbollah has a standing military wing.

3. $(KIDNAP = 1) : [0.34, 1.0] \leftarrow (ORGLOC = 1) \wedge$
   $(ORGDOMGOALS = 2) \wedge (ORGST4 = 1)$
   Hezbollah carries out kidnappings with a probability between 0.34 and 1.0 if they are located in Lebanon, the major goal of Hezbollah is focused on creating or increasing remedial policies, and Hezbollah solicits external support only as a minor/infrequent strategy.

4. $(TLETHCIV = 1) : [0.13, 1.0] \leftarrow (ORGLOC = 1) \wedge (ORGST3 = 1)$
   Transnational targets of terrorism are chosen based on ethnicity with a probability between 0.13 and 1.0 if Hezbollah is in Lebanon and Hezbollah uses electoral politics only as a minor/infrequent strategy.

5. $(TTSECGOV = 1) : [0, 0.68] \leftarrow (ORGCULTGR = 0) \wedge$
   $(INTERORGCON = 1) \wedge (DIASUP = 0)$
   Transnational government/state lives and security personnel are targets of terrorism with a probability between 0 and 0.68 if Hezbollah expresses no cultural grievances, there is inter-organizational conflict, and Hezbollah has not received support from the Lebanese diaspora

Figure 2.8: A sample of the *ap*-rules extracted by APEX from the MAROB dataset about the behavior of Hezbollah. The atoms in the rules are represented as a variable and its value. The English translation of each rule is also provided.

2.5 Implementation and Experiments

In this section, implementations and experimental evaluations of the general PMPW algorithm, which was developed using about 6,700 lines of Java code, are discussed. Implementations of the serial versions of the naive, HOP, $\mathsf{SemiHOP}$, and the binary heuristic algorithms were also developed using approximately 6,000 lines of Java code, and experimental results for these algorithms are discussed in [62, 61].

As described in Section 2.3, the PMPW algorithm is very general and can be applied to any of the serial algorithms (i.e., CONS can be the constraint set generated by any of the serial algorithms). An experimental evaluation was done to test PMPW applied to the naive, HOP, $\mathsf{SemiHOP}$ and binary heuristic constraints (i.e., CONS computed as $\mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, $\mathsf{RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, $\mathsf{S\_RedCONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$, and $\mathcal{C}^{\bullet}$). However, in the serial experiments reported in [62, 61], HOP proved intractable due to the satisfiability checks necessary for generating the equivalence classes; as such, experiments for PMPW applied to the exact version of HOP were not included. As the binary heuristic can be used in conjunction with any of the three exact algorithms, the full set of parallel experiments examined a total of 5 distinct instances of PMPW. These experiments were performed on a Linux computing cluster comprised of 64 8-core nodes with between 10GB and 20GB of RAM. The linear constraints were solved using the QSopt linear programming solver library [7], and the logical formula manipulation code from the COBA belief revision system [38] and SAT4J satisfaction library [1] were used in the implementation of the HOP and $\mathsf{SemiHOP}$ algorithms to identify the equivalence classes or sub-partitions.

For each experiment using PMPW the number of *ap*-rules in the input program was held constant at 10. Each evaluation used the following procedure: (i) the number of worlds was varied from 32 to 1,024, (ii) a new *ap*-program was randomly generated and sent to an instance of PMPW for each of the algorithms (the two exact solvers—naive and SemiHOP—as well as all combinations with the binary heuristic), and (iii) at least 10 runs were executed for each number of possible worlds, and the average time taken by each parallel algorithm relative to the serial experiments from [62, 61] and the other parallel running times was recorded.

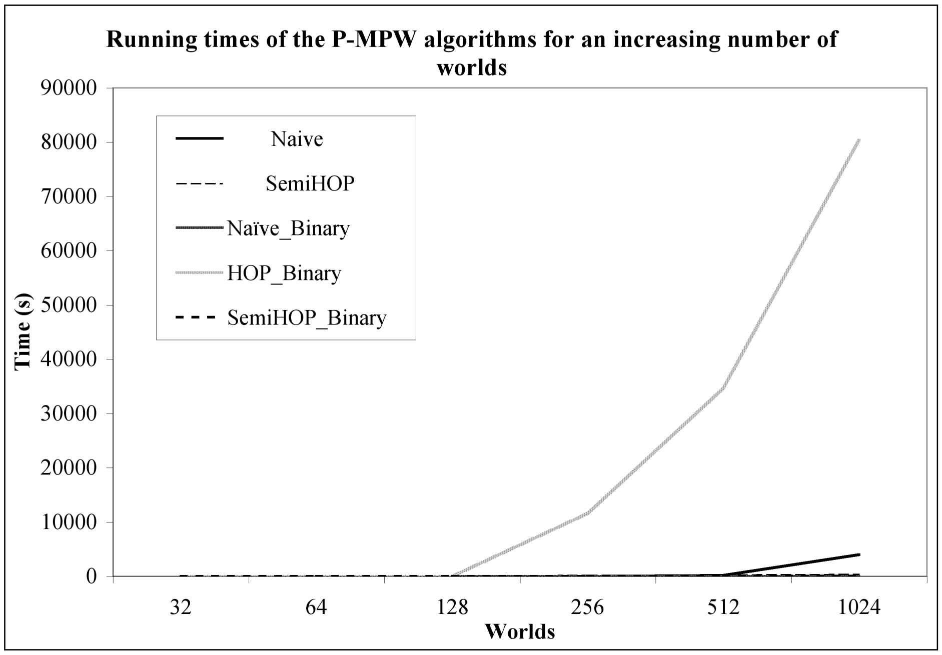

Figure 2.9: Running time of the PMPW versions of the naive, SemiHOP, naive$_{bin}$, HOP$_{bin}$, and SemiHOP$_{bin}$ algorithms for an increasing number of worlds .

Each experimental run utilized 64 processors in parallel on the above mentioned computing cluster to solve a single MPW problem. Only one core per compute node was used because of the high memory requirements of the linear program in the MPW computations. As expected, the pleasantly parallel PMPW algorithms produce a marked speedup in the computation time for finding the most probable world by simply distributing the computations of low(w_i) across the parallel processes (Figure 2.9). Where the basic naive algorithm requires almost 4 hours (13,636.23 seconds) [62, 61] for problems with 1,024 possible worlds, the naive PMPW algorithm (i.e., PMPW where $\mathsf{CONS} = \mathsf{CONS}_U(\Pi, s, T^{\omega}_{\Pi_s})$) completed the same computation in only about an hour (4016.83 seconds). This is a very promising result for situations where an exact solution is necessary, as the same MPW problems can now be solved in a more reasonable amount of time.

A similar speedup can be seen for the SemiHOP and binary heuristic algorithms. The PMPW SemiHOP algorithm uses slightly under 6 minutes (339.65 seconds) as opposed to 33.47 minutes (2,008.1 seconds) [62, 61] to solve for 1,024 worlds, and the PMPW naive algorithm with the binary heuristic shows an improvement from the serial time of 136.08 seconds [62, 61] to 21.78 seconds. In some cases, however, the PMPW version of the SemiHOP algorithm actually performs worse as compared to the serial SemiHOP algorithm from [62, 61]. This anomaly occurs in those instances where there are no sub-partitions with only a single satisfying interpretation; in such cases, it is not actually necessary to solve an MPW computation (as described in Section 2.2.1 for the HOP and SemiHOP algorithms), so the overhead of managing parallel threads is greater than the running time of the serial version. In most instances, though, the PMPW algorithm greatly improves the efficiency of computing the most probable world. Table 2.1 contains the average speedup achieved by using the PMPW algorithms compared to the running time of their serial counterparts given in [62, 61]. PMPW Naive and Naive_{Binary} showed the greatest speedup over their serial versions at 75.35% and 103.84% respectively. These algorithms had the greatest room for improvement since they contained the most variables (as they were using worlds rather than equivalence classes), making the distribution of these MPW computations extremely efficient relative to the original serial algorithms.

Worlds	Naive	SemiHOP	Naive_{Binary}	HOP_{Binary}	SemiHOP_{Binary}
32	50.48	02.73	111.21	94.40	76.64
64	66.15	40.88	123.72	97.78	80.26
128	83.23	47.97	121.58	97.10	78.06
256	89.47	52.01	86.80	13.61	44.73
512	90.29	55.84	89.95	29.15	9.68
1024	72.45	49.16	89.81	1.86	27.88
Avg	75.35	41.43	103.84	55.65	52.87

Table 2.1: Percent speedup achieved with the PMPW algorithms.

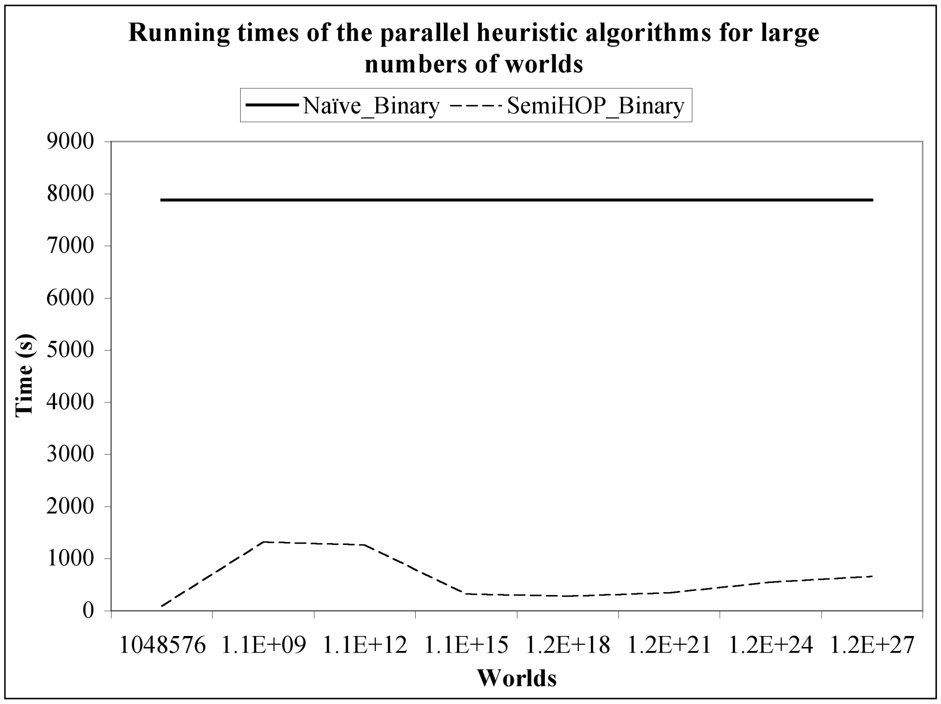

Figure 2.10: Running time of the PMPW versions of the naive_{bin} and SemiHOP_{bin} algorithms for large numbers of worlds.

Similar improvements can be seen when using the PMPW algorithm with the binary heuristic (i.e., $\text{CONS} = \mathcal{C}^{\bullet}$) on large numbers of worlds, providing an average speedup of about 66%. The running times for PMPW with the binary heuristic applied to both the naive algorithm and SemiHOP are shown in Figure 2.10. Due to the parameter reduction of using sub-partitions and the binary heuristic, PMPW with SemiHOP$_{Binary}$ is able to vastly outperform the parallel Naive$_{Binary}$ algorithm, scaling to MPW problems consisting of on the order of 10^{27} worlds in about 30 minutes. The ability to scale to such large numbers of worlds and improve the computation time indicates that the PMPW algorithms can make this framework much more applicable to real-world behavioral modeling problems.

2.6 Conclusions

In this chapter, the theory of *ap*-programs has been reviewed, as well as basic algorithms for reasoning in this framework to find the most probable world, which is analogous to forecasting agent behavior. This work has also been significantly expanded to include distributed algorithms and applications to behavioral modeling and forecasting in international security situations. These methods are the first parallel algorithms for *ap*-programs (the first parallel algorithms for any kind of PLP) and allow for more efficient computation for all of the basic exact and heuristic approaches to solving the MPW problem. It was showed experimentally that the PMPW implementation can cope with the immense scale of real-world behavioral modeling situations, taking only 30 minutes to find the most probable world for applications with about 10^{27} possible worlds. This level of efficiency and scalability is far superior to any past efforts. In addition, the PMPW-LR algorithm can further improve the scalability of this framework to allow users to look at even larger *ap*-programs, and parallel approximation algorithms—PMPW-BH and PMPW-IBH—can improve the quality of the randomized binary heuristic.

The success of these distributed reasoning algorithms for *ap*-programs makes this framework applicable to real-world situations where experts may need to address a broad range of possible actions or run multiple iterative "what if" forecasts. In addition, the APEX algorithm was provided as a method for automatically extracting *ap*-rules from time series data. Applications of this framework have produced tangible results of use to U.S. military officers and national security experts, showing promise for use with other datasets and expanded cultural reasoning domains [14, 115].

There are, of course, many problems in *ap*-programs and behavioral modeling that remain open. First, it would be desirable to develop even more efficient parallel algorithms—the current scaling offered, while much more efficient, is not proportional to the number of CPUs used. Finally, while *ap*-programs provide an effective way to model the behavior of various cultural-adversarial groups, the algorithms for finding the most probable world are not yet efficient enough for real-time computation and behavioral forecasting. Such improvements to this framework would make these models more useful as a tactical decision-support tool.

Chapter 3

Context Vectors as a Similarity-based Paradigm for Forecasting

The work in this chapter has appeared in [75].

In Chapter 2, the Stochastic Opponent Modeling Agents (SOMA) [62, 61] framework was introduced as a robust method for behavioral modeling and forecasting, with applications to groups (i.e., terror organizations, drug traffickers, political parties) involved in global security situations. SOMA uses *ap*-programs and scalable distributed reasoning algorithms to compute the most probable actions an agent will take in a given situation. This and other similar approaches are based on *indicators* that have previously been identified—and possibly encoded as probabilistic logic rules—as being correlated with specific agent behaviors being predicted. However, there are many cases in which these indicators are not known *a priori* or it may be necessary to make a timely forecast or "what if" analysis without constructing an intermediate behavioral model to learn these indicators in advance.

For example, consider again the the Minorities at Risk Organizational Behavior (MAROB) [46, 125] dataset used in the previous chapter. Recall that this data tracks the behavior for 118 ethnopolitical organizations in the Middle East and Asia Pacific on a yearly basis from 1980 to 2004. Only a handful—around 43—of the approximately 175 attributes in the data represent strategic actions taken by the group, while the others represent variables relating to the environment or context in which the group functions. This context includes variables about the degree of military and financial support the group gets from foreign nations or the ethnic diaspora, the degree of state government repression and persecution against the group, and so forth. It also includes variables about the structure of the group and how factionalized it may or may not be, the level of violence and protests in which the group engages, and the amount of participation in the political process.

The ontology generated from this schema is quite shallow in contrast to the deep ontologies seen in semantic web approaches such as RDF or OWL. However, experts still want to use this information on past behavior, particularly data related to the contextual or situational factors, to predict the actions taken by an agent, even without deep ontological or probabilistic knowledge, such as *ap*-programs. More formally, suppose there is historical data about an agent's behavior over many time periods, i.e., there is a set $\mathcal{PB}$ of past behaviors. Each time period in $\mathcal{PB}$ is a pair (c, a) consisting of two vectors: c is a vector containing the values of the context attributes associated with the agent, and a is a vector containing the values of the action variables. Suppose now that a user wants to identify what the agent might do in the current situation or in a hypothetical scenario. In either of these cases, the situation can also be represented by a vector q describing the context or the environment in which the agent is hypothesized to be, or actually is, functioning. The user is interested in determining what the associated action vector will be for this given situation, providing a forecast of the likely strategy an agent will employ in certain contexts. For example, the action vector might indicate that a terrorist group will engage in bombings with a "high" degree of intensity, while simultaneously showing that they will not resort to kidnappings.

In this chapter, a computational theory and algorithms for making such behavioral forecasts are developed that have the following features:

(i) The proposed algorithms are very fast.

(ii) The algorithms are highly accurate for forecasting agent behavior, as demonstrated through application to the MAROB data [46, 125].

(iii) The results produced by the algorithms are easily explainable.

The goal here is to develop computational methods to better tame the complexity involved in modeling, forecasting, and analyzing agent behavior, especially in international security or conflict situations. In this regard, there has been relatively little work on automated prediction of what an agent (ethnopolitical group, terrorist organization, etc.) might do in the future in a specific situation. All previous efforts to build such predictive models in the socio-cultural-political domain—including those in Chapter 2—have focused on three phases: a data collection phase, a model construction phase, and a forecasting phase using the model.

Efforts by Schrodt [97] and Bond et al. [16] and previous work on *ap*-programs [62, 61, 116, 115] have all tried to address the data collection phase in an automated manner. However, data collection is not addressed in this chapter as it has been covered in these other three architectures.

When it comes to the model building phase, these previous approaches vary. Schrodt [97] and Bond et al. [16] have developed methods to build hidden Markov models (HMMs) to describe how a particular conflict might evolve over time. However, these HMMs are painstakingly constructed in a very time-consuming process, which must be repeated for each country or group involved and requires a degree of subjectivity. Nevertheless, HMMs and their variants (e.g., stochastic automata, stochastic Petri nets, etc.) can form very valuable modeling tools and mechanisms for behavioral forecasting and deserve continued study.

Another method to forecast agent behavior in security or conflict situations is the SOMA framework described in the previous chapter (Chapter 2) and in [116, 115, 107, 62, 61]. The APEX algorithm (Figure 2.7) for automated extraction of stochastic behavioral rules—*ap*-rules—has an advantage over the HMM models of [97, 16] in that it is fully automated and fast. These behavioral rules are then used to forecast agent behavior using efficient linear programming and scalable distributed algorithms (Sections 2.2.1 and 2.3). However, while this method is very robust and the behavioral rules can aid in understanding the forecasted actions, computational efficiency is still a major challenge of this logic-based approach, especially for real-time tactical analysis.

In contrast to all of these methods, the CONVEX framework described in this chapter does *not* build an intermediate model of an agent's behavior. Rather, time series data is used directly to assess the similarity between a given context (expressed as the query vector) and information about the agent's behavior in the past when confronted with similar situations. These past situations that the agent encountered are then used for predictions—the "model building phase" present in related works is therefore skipped completely. CONVEX predictions can be made solely by examining the data without an agent-specific model that takes time to build and may require subjective methods.

In Sections 3.1 and 3.2, this behavioral forecasting problem is formalized. Section 3.3 then describes two classes of algorithms to address this problem. Strictly speaking, these are two general algorithms; however, their accuracy depends upon various parameters. Section 3.4 describes a prototype implementation showing that these algorithms are scalable, extremely general, and capable of producing highly accurate predictions from time series behavioral data.

3.1 Behavioral Time Series Data

To forecast agent behaviors directly from data, a formal representation of this problem and the underlying time series is required. Such behavioral time series datasets consist of a relational database describing the behavior of an agent g. Assume the existence of some arbitrary universe $\mathcal{A}$ whose elements are called *attribute names* (attributes for short). Each attribute V_i has an associated domain $dom(V_i)$.

In these datasets the attributes fall broadly into two categories—*environmental or contextual (independent) attributes* describing the context in which an agent functioned during a given time frame, and *action (dependent) attributes* describing actions taken by the agent during a given time frame. Note that environmental attributes can also include actions taken by other agents or external actors that may impact the agent or group in question. Using this intuition, assume that any agent g has an associated *context schema* $CS(g) = (C_1, \ldots, C_n)$ of context attributes and *action schema* $AS(g) = (A_1, \ldots, A_m)$ of action attributes where each $C_i, A_j \in \mathcal{A}$ is an attribute and $\{C_1, \ldots, C_n\} \cap \{A_1, \ldots, A_m\} = \emptyset$. The full behavioral time series for a particular agent or group g is denoted by T_g.

Year	LEAD	ORGPOP	DEMORG	FORSTFINSUP	ARMATTACK	HOSTAGE
1993	4	2	1	0	1	0
1994	3	2	1	0	1	0
1995	3	2	1	0	0	0
1996	3	2	1	0	1	0
1997	3	2	1	0	0	0

Figure 3.1: Small subset of actual data for a group in the MAROB dataset. LEAD, ORGPOP, DEMORG, and FORSTFINSUP are context attributes, while ARMATTACK and HOSTAGE are action attributes.

Example 1. *Figure 3.1 shows a small example of the behavioral time series data T_g for a terrorist organization g from the MAROB data [46, 125]. The table shown represents actual data for a subset of the approximately 175 MAROB attributes. Here, $CS(g) = \{LEAD, ORGPOP, DEMORG, FORSTFINSUP\}$, where the context in which the group is operating is described by categorical attributes denoting the type of group leadership, the amount of popular support, whether the group is internally democratic, and whether they receive financial support from a foreign state, respectively. The action schema $AS(g) = \{ARMATTACK, HOSTAGE\}$ contains two action attributes indicating whether the group uses armed attacks or takes hostages as a strategy at a particular time point.*

3.2 A Formal Vector Model of Agent Behaviors

A behavioral time series T_g can also be represented as a collection of pairs of vectors assigning values to the context attributes and action attributes, respectively.

Definition 17 (g-behavior). *Suppose g is an agent with context schema $CS(g) = (C_1, \ldots, C_n)$ and action schema $AS(g) = (A_1, \ldots, A_m)$. A g-*behavior *is a pair*

$$\langle (c_1, \ldots, c_n), (a_1, \ldots, a_m) \rangle$$

where $c_i \in dom(C_i)$ and $a_j \in dom(A_j)$.

Definition 18 (Past Behavior)**.** *A* past behavior *for agent g is a finite set, $\mathcal{PB}(g)$, of g-behaviors.*

Note that given the entire behavioral dataset T_g for an agent g, any $T'_g \subseteq T_g$ can be expressed as a past behavior $\mathcal{PB}(g)$.

Example 2. *Consider again the behavioral time series for terror organization g given in Figure 3.1. The corresponding past behavior in vector form consists of 5 g-behaviors:*

$$\begin{aligned}\mathcal{PB}(g) = \{ \quad & \langle (4,2,1,0),(1,0)\rangle,\\ & \langle (3,2,1,0),(1,0)\rangle,\\ & \langle (3,2,1,0),(0,0)\rangle,\\ & \langle (3,2,1,0),(1,0)\rangle,\\ & \langle (3,2,1,0),(0,0)\rangle \quad \}\end{aligned}$$

Definition 19 (Context/Action Vector)**.** *Suppose g is an agent with context schema $CS(g) = (C_1, \ldots, C_n)$ and action schema $AS(g) = (A_1, \ldots, A_m)$. A* context vector *w.r.t. agent g is an expression of the form $(c_1, \ldots, c_n)$ where each $c_i \in dom(C_i)$.*

An action vector *w.r.t. agent g is an expression of the form $(a_1, \ldots, a_m)$ where each $a_j \in dom(A_j)$.*

Intuitively, a context vector specifies a value for each context attribute, and thus can be viewed as either a real or hypothetical context within which the agent functions (or is hypothesized to function). An action vector assigns a value to every action attribute and represents the actions taken by an agent.

Given a past behavior $\mathcal{PB}(g)$ and a query context vector q, the goal is to find an appropriate action vector $(a_1, \ldots, a_m)$ describing the agent's behavior given q.

Clearly, the past behavior constitutes a set of historical behavioral data from which to forecast what actions the agent g might take when in a situation or context characterized by the query vector q.

3.3 Algorithms for Forecasting Agent Behavior

Using the vector data representation, this section presents two general algorithms to predict an action vector $(a_1, \ldots, a_m)$ from a given past behavior $\mathcal{PB}(g)$ of an agent and a query context vector $(c_1, \ldots, c_n)$. Both algorithms use distance functions in metric spaces to compare the query vector to context vectors in the past behavior.

3.3.1 Distance functions

It is clear that each context vector (and the query vector) is a point in the n-dimensional vector space $dom(C_1) \times \cdots \times dom(C_n)$. A distance function d can be defined on this vector space. As is commonly the case with distance functions, d must satisfy the following three axioms:

A1. $d(x, x) = 0$

A2. $d(x, y) = d(y, x)$

A3. $d(x, z) \leq d(x, y) + d(y, z)$

In the above formulas, x, y, z are all context vectors, i.e., members of $dom(C_1) \times \cdots \times dom(C_n)$. There are any number of well-known distance functions in the literature that meet these requirements. Six such distance functions are studied in this chapter and experimentally evaluated, but the forecasting algorithms given below also work with other distance functions, as well as weighted distance functions.

Suppose context vectors $(c_1, \ldots, c_n), (c'_1, \ldots, c'_n)$ are members of $dom(C_1) \times \cdots \times dom(C_n)$.

1. **Euclidean distance.** $d_{EUC}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \sqrt{(c_1 - c'_1)^2 + \cdots + (c_n - c'_n)^2}$.

2. **Canberra distance.** $d_{CAN}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \sum_{i=1}^{n} \frac{|c_i - c'_i|}{|c| + |c'|}$ — when divisions by zero occur, $d_{CAN}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = 0$.

3. **Chebyshev distance.** $d_{CHEB}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \max_i(|c_i - c'_i|)$.

4. **Cosine distance.** $d_{COS}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \frac{\sum_{i=1}^{n} |c_i * c'_i|}{|c| * |c'|}$.

5. **Hamming distance.** $d_{HAM}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \sum_{i=1}^{n} isDiff(c_i, c'_i)$ where $isDiff(c_i, c'_i)$ is 1 if c_i and c'_i are different, and 0 if they are equal.

6. **Manhattan distance.** $d_{MAN}((c_1, \ldots, c_n), (c'_1, \ldots, c'_n)) = \sum_{i=1}^{n} |c_i - c'_i|$.

Example 3. *Recall the past behavior $\mathcal{PB}(g)$ from Example 2 for terror organization g. Using the Manhattan distance function the distance between the context vectors in the first two g-behaviors can be computed as:*

$$d_{MAN}((4,2,1,0),(3,2,1,0)) = |4-3| + |2-2| + |1-1| + |0-0| = 1.$$

3.3.2 The $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ Algorithm

In order to forecast what an agent will do in a given situation, either real or hypothetical, a query vector can be compared to similar contexts that the agent experienced in the past by utilizing a distance function over $dom(C_1) \times \cdots \times dom(C_n)$.

In the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ algorithm given in Figure 3.2, the k contexts from an agent's past data closest to the query vector are identified, and the average over the associated action vectors is computed to forecast actions in the query context.

Given a past behavior $\mathcal{PB}(g)$ for an agent of interest g, a query vector q, a distance function d, a number $k \geq 1$, and an action schema $AS(g)$, $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ begins by initializing a sorted list $NearK$ of k elements. In the loop on Lines 3–6, each g-behavior in the past behavior is examined to find the k past context vectors closest to the query vector according to the distance function d. The function *insert_sort* on Line 6 is a procedure that will insert a new element into a list of fixed size, maintaining ascending order w.r.t. distance from the query vector. Then, in the loop beginning on Line 7, for each action attribute A_j, the values $a_{j_1}, \ldots, a_{j_k}$ are retrieved from the action vectors associated with the k nearest context vectors in $NearK$. V is then computed to be the average of these values; if V is an integer, then it is simply added to the behavioral forecast in Line 9. Otherwise, the interval $[\lfloor V \rfloor, \lceil V \rceil]$ is added to the forecast. This interval indicates that $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ cannot forecast the value of A_j exactly, but it is expected to either be $\lfloor V \rfloor$ or $\lceil V \rceil$ (assuming $dom(A_j)$ consists of integers—otherwise, if $dom(A_j)$ contains of real values, the forecast can be anywhere in the closed interval in question or the value V can be returned, depending on the semantics of the domain).

Example 4. *Consider again the behavioral time series table T_g shown in Figure 3.1 for a terrorist organization g with the context schema $CS(g) = \{LEAD, ORGPOP, DEMORG, FORSTFINSUP\}$ and the action schema $AS(g) = \{ARMATTACK, HOSTAGE\}$. Suppose we want to forecast the behavior of group g in the scenario described by query vector $q = (4, 1, 1, 0)$ using $\mathsf{CONVEX}^{k\_\mathrm{NN}}$, where d is the Euclidian distance function and $k = 1$ (i.e., only the single closest context vector from the past*

```
Algorithm CONVEX$^{k\text{-NN}}(\mathcal{PB}(g), q, d, AS(g), k)$
1.    $NearK =$ list of $k$ elements initialized to $+\infty$
2.    $Forecast = \emptyset$
3.    for each $(cv, av) \in \mathcal{PB}(g)$ do
4.        $Dist = d(q, cv)$
5.        if $Dist < d(q, NearK[k-1])$ then
6.            $insert\_sort((cv, av), NearK)$
7.    for each $A_j \in AS(g)$ do
8.        $V =$ average of $NearK[0].A_j, \ldots, NearK[k-1].A_j$
9.        if $V$ is an integer then add $V$ to $Forecast$
10.           else add $[\lfloor V \rfloor, \lceil V \rceil]$ to $Forecast$
12.   return $Forecast$
```

Figure 3.2: The CONVEX$^{k\text{-NN}}$ algorithm for forecasting agent behavior.

behavior will be considered). Of the five years of data shown in the table, the year that is closest to the query vector is 1993 where $d(q, (4, 2, 1, 0)) = 1$*, so the g-behavior* $\langle(4, 2, 1, 0), (1, 0)\rangle$ *is added to* $NearK$*. The next step is averaging over the action vectors associated with the k closest context vectors to obtain a forecast. In this case* *CONVEX*$^{k\text{-NN}}$ *would forecast the action vector* $(1, 0)$*, where* $ARMATTACK = 1$ *and that* $HOSTAGE = 0$*, by simply looking at the single year 1993.*

Now, consider the query vector $q = (3, 2, 1, 1)$ *and* $k = 4$ *with the same distance function. In this case, years 1994, 1995, 1996, 1997 are all the nearest neighbors with* $d(q, (3, 2, 1, 0)) = 1$*. To forecast the behaviors for the query context,* *CONVEX*$^{k\text{-NN}}$ *will average the values of the two action attributes over these 4 g-behaviors stored in* $NearK$*. For* $ARMATTACK$*,* $V = avg(1, 0, 1, 0) = 0.5$*, so the forecast for* $ARMATTACK$ *will be the interval* $[0, 1]$*, meaning that the value of* $ARMATTACK$ *will be either* 0 *or* 1*. For* $HOSTAGE$*,* $V = avg(0, 0, 0, 0) = 0$*, so the prediction for* $HOSTAGE$ *is* 0*.* *CONVEX*$^{k\text{-NN}}$ *then returns the action vector* $([0, 1], 0)$ *as the forecasted behavior of group g given the query context.*

Note that if $k = 3$ in the above case, the years 1994, 1995, 1996, 1997 would all still be nearest neighbors of the query vector, but the algorithm would select only the first three and return the same answer as above based on the values of the action attributes in these three years.

The following result shows that $\mathsf{CONVEX}^{k\text{-NN}}$ runs very fast. In practice k is usually small (as will be shown later in Section 3.4), and this proposition states that the algorithm is linear in the size of the past behaviors $\mathcal{PB}(g)$ of the agent g.

Proposition 1. *Suppose $\mathcal{PB}(g)$ is a past behavior for agent g, d is a distance function over $dom(C_1) \times \cdots \times dom(C_n)$, $AS(g)$ is an action schema for g, and $k \geq 1$. If d is computable in constant time, then the $\mathsf{CONVEX}^{k\text{-NN}}$ algorithm runs in time $\mathbf{O}(k \cdot |\mathcal{PB}(g)| + k \cdot |AS(g)|)$.*

Proof. The loop in Lines 3–6 of the $\mathsf{CONVEX}^{k\text{-NN}}$ algorithm is executed at most $|\mathcal{PB}(g)|$ times. In each iteration of the loop, an insertion into a sorted list is necessary, which can be done in $\mathbf{O}(k)$ time. The second loop beginning on Line 7 executes at most $|AS(g)|$ times, with each iteration performing a linear retrieval operation on the list, again taking $\mathbf{O}(k)$ time.

3.3.3 The CONVEXMerge Algorithm

The basic $\mathsf{CONVEX}^{k\text{-NN}}$ algorithm assigns an equal weight to each of the k nearest neighbors regardless of the variances in their distance to the query vector. The CONVEXMerge algorithm addresses this issue by assuming that the importance of the k nearest neighbors is inversely proportional to the distance between those context vectors and the query vector. In other words, suppose the two nearest neighbors in $\mathcal{PB}(g)$ of the query vector q are under consideration. The first neighbor may be at distance 1 away, while the second nearest neighbor may be at distance 10 away. In this case, the value assigned to action attribute A_j in the nearest g-behavior must have greater priority than the value assigned to action attribute A_j by the second nearest g-behavior. This intuition is captured by the following definition, where the set of k nearest neighbors w.r.t. a query vector q and a past behavior $\mathcal{PB}(g)$ is denoted $kNN_{q,\mathcal{PB}(g)} = \{(cv_1, av_1), \ldots, (cv_k, av_k)\}$.

Definition 20 (Conditional Probability of an Action)**.** *Let k be a fixed integer s.t. $k \geq 1$, q be a query vector, $\mathcal{PB}(g)$ be a past behavior for agent g, and $AS(g)$ be an action schema. The* probability $\mathbf{P}(A_j = a|q, \mathcal{PB}(g))$ *of action attribute $A_j \in AS(g)$ having value a in the context described by q with past behavior $\mathcal{PB}(g)$ is:*

(i) $\mathbf{P}(A_j = a|q, \mathcal{PB}(g)) = \frac{|\{(cv_i,av_i) \mid (cv_i,av_i) \in kNN_{q,\mathcal{PB}(g)} \wedge A_j = a \in \ av_i\}|}{k}$ *if* $\Sigma_{(cv_i,av_i) \in kNN_{q,\mathcal{PB}(g)}} d(q, cv_i) = 0$;

(ii) $\mathbf{P}(A_j = a|q, \mathcal{PB}(g)) = 1$ *if* $\{(cv_i, av_i) \mid (cv_i, av_i) \in kNN_{q,\mathcal{PB}(g)} \wedge A_j = a \ in \ av_i\} = kNN_{q,\mathcal{PB}(g)}$;

(iii) $\mathbf{P}(A_j = a|q, \mathcal{PB}(g)) = 0$ *if* $\{(cv_i, av_i) \mid (cv_i, av_i) \in kNN_{q,\mathcal{PB}(g)} \wedge A_j = a \ in \ av_i\} = \emptyset$; *otherwise*

(iv) $\mathbf{P}(A_j = a|q, \mathcal{PB}(g)) = 1 - \frac{\Sigma_{(cv_i,av_i) \in kNN_{q,\mathcal{PB}(g)} \wedge A_j = a \ \ in \ \ av_i} d(cv_i,q)}{\Sigma_{i=1}^{k} d(cv_i,q)}$.

The first case of Definition 20 occurs when all k nearest neighbors are at distance zero from the query vector. The following definitions are the cases when all of the k nearest neighbors agree on the value a for attribute A_j, none of the neighbors have value a for A_j, and the general case when none of these special conditions is true. The denominator of the term $\frac{\Sigma_{(cv_i,av_i) \in kNN_{q,\mathcal{PB}(g)} \wedge A_j = a \ \ in \ \ av_i} d(cv_i,q)}{\Sigma_{i=1}^{k} d(cv_i,q)}$ in the last definition is the sum of the distances to q from each of its k nearest neighbors. The numerator is the sum of the distances between q and those nearest neighbors whose attribute A_j has a given value a. Thus, the smaller the numerator is, the "closer" (or more similar) the attribute value $A_j = a$ is to the g-behaviors that are most similar to q. This ratio is subtracted from 1 because a small distance must indicate a high probability.

Example 5. *Suppose we want to forecast the behavior of agent g given the past behavior $\mathcal{PB}(g)$, the action schema $AS(g) = A_1$, and the query vector $q = (0, 0)$ where $k = 3$. The k nearest neighbors to q are the set*

$$\begin{array}{rl} kNN_{q,\mathcal{PB}(g)} = \{ & (c_1 = (0,1), a_1 = (0)), \\ & (c_2 = (0,2), a_2 = (0)), \\ & (c_3 = (1,1), a_3 = (1) \ \} \end{array}$$

In this case,

$$d_{EUC}(q, c_1) = 1,$$

$$d_{EUC}(q, c_2) = 2, \text{ and}$$

$$d_{EUC}(q, c_3) = \sqrt{2}$$

For each possible value in $dom(A_1)$ the probability of that action occurring in the context of query vector q can be computed:

$$\begin{aligned} \mathbf{P}(A_j = 0 | q, \mathcal{PB}(g)) &= 1 - \frac{1+2}{1+2+\sqrt{2}}. \\ &= 0.32 \\ \mathbf{P}(A_j = 1 | q, \mathcal{PB}(g)) &= 1 - \frac{\sqrt{2}}{1+2+\sqrt{2}}. \\ &= 0.68. \end{aligned}$$

```
Algorithm CONVEXMerge($\mathcal{PB}(g), q, d, AS(g), k$)
1.     $NearK =$ list of $k$ elements initialized to $+\infty$
2.     $Forecast = \emptyset$
3.     for each $(cv, av) \in \mathcal{PB}(g)$ do
4.         $Dist = d(q, cv)$
5.         if $Dist < d(q, NearK[k-1])$ then
6.             $insert\_sort((cv, av), NearK)$
7.     for each $A_j \in AS(g)$ do
8.         $V = \arg\max_{a \in dom(A_j)}(\mathbf{P}(A_j = a | q, \mathcal{PB}(g)))$
9.         add $V$ to $Forecast$
9.     return $Forecast$
```

Figure 3.3: The CONVEXMerge algorithm.

The reason that the probability of $A_1 = 1$ is higher than that of $A_1 = 0$ is intuitively because two of the three nearest neighbors of the query vector q have $A_1 = 0$, and these two nearest neighbors have a distance of 1 and 2 from the query vector. In contrast, while there is only one nearest neighbor having $A_1 = 1$, its distance from the query vector is $\sqrt{2}$. Because it is closer to the query vector, context vector c_3 is more important for predictive purposes than context vector c_2 of distance 2 away, which leads to a reduced probability for A_1 having the value 0.

Figure 3.3 presents the CONVEXMerge algorithm, which finds the probability that $A_j = a$ for each $a \in dom(A_j)$. The values with the highest probability are returned in the forecasted action vector. With slight modifications, this algorithm can return the entire probability distribution over all action values. The CONVEXMerge algorithm is slightly less efficient than the CONVEX$^{k\text{-NN}}$ algorithm; its complexity includes an additional multiplicative factor, $|dom(A_j)|$, because the probability of each $A_j = a$ possibility must be computed.

Proposition 2. *Suppose $\mathcal{PB}(g)$ is a past behavior for agent g, d is a distance function over $dom(C_1) \times \cdots \times dom(C_n)$, $AS(g)$ is an action schema for g, and $k \geq 1$.*

If d is computable in constant time, then $\mathsf{CONVEXMerge}(\mathcal{PB}(g), q, d, AS(g), k)$ *runs in time* $\mathbf{O}(k \cdot |\mathcal{PB}(g)| + k \cdot |AS(g)| \cdot |dom(A_j)|)$.

3.4 Implementation and Experiments

Both the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ algorithms were implemented and experimentally evaluated to test this approach for behavioral forecasting. The implementations required about 1,200 lines of Java code, and all experiments described in this section were run on a computer with a dual-core processor at 2GHz, with 2GB of RAM, running the Windows Vista operating system.

The Minorities at Risk Organizational Behavior (MAROB) dataset [46, 125] was used as the behavioral time series for these experiments, consisting of yearly behavioral information about violent ethnopolitical groups (many of them engaged in terrorism) in the Middle East from 1980 to 2004. All of the attributes in MAROB are categorical variables with integer domains that were populated by humans using large volumes of trusted news reports. For each group and each year, the human coders tried to arrive at a judgment of which categorical value should be specified for each attribute during that year. For example, for the variable DOMORGPROT measuring the level of domestic protest by a group in a given year, the coders try to determine from press reports the number of protests staged (and protesters involved) by the group for the year in question and arrive at a reasoned judgment of the appropriate value for this attribute. As described in Example 2, for each group g, the g-behaviors for MAROB contain the coded data for a given year. Thus, for each group g, $\mathcal{PB}(g)$ has at most 25 g-behaviors.

Distance	CONVEX^1	CONVEX^2	CONVEX^3
Canberra	0.857	0.938	0.942
Chebyshev	0.857	0.949	0.945
Cosine	0.857	0.950	0.947
Euclidean	0.857	0.950	0.948
Hamming	0.857	0.951	0.948
Manhattan	0.857	0.951	0.947

Figure 3.4: Average accuracy of the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ algorithm.

Only groups with more than 10 years of data were considered in the experiments. If a group had y years of data for $y > 10$, training sets of size $t = 10, \ldots, y-1$ were constructed. Every single year was used as a query vector, and $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ were used to try and forecast the actual action vector in that year from random training sets of size t chosen from the remaining years of data. This process was repeated for all 118 groups in the MAROB dataset.

The *correctness ratio* of the forecasting algorithms was used as a metric of accuracy and is defined as the number of correct predictions, divided by the total number of predictions. A prediction was considered correct if, for a given query year, the actual value of a given action variable coincided with the *single* value predicted by the algorithm. However, recall that the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ algorithm can yield *intervals* rather than single values. To compute the correctness ratio in these cases, a single value was obtained by rounding V off to an integer value.

Figures 3.4 and 3.5 show the accuracy results for the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ algorithms for a variety of distance functions and $k = 1, 2, 3$. Each entry in the tables shows the accuracy of the algorithm, averaged as t (the size of the training data) was varied from 10 to $y - 1$ as indicated above. There are several important observations that can be made from these results:

Distance	CONVEXMerge^1	CONVEXMerge^2	CONVEXMerge^3
Canberra	0.857	0.851	0.849
Chebyshev	0.857	0.860	0.859
Cosine	0.857	0.859	0.857
Euclidean	0.857	0.861	0.860
Hamming	0.857	0.861	0.861
Manhattan	0.857	0.861	0.861

Figure 3.5: Average accuracy of the CONVEXMerge algorithm.

(i) **Always check 2 nearest neighbors.** Choosing $k = 2$ seems to give the best results, irrespective of which algorithm or distance function was used. *Thus, it is best to look at the two nearest neighbors, not one or three.*

(ii) **CONVEX$^{k\_\mathrm{NN}}$ consistently outperforms CONVEXMerge.** Regardless of which distance function was used or what value of k was selected, the CONVEX$^{k\_\mathrm{NN}}$ algorithm was more accurate than CONVEXMerge. Of course, when $k = 1$, both algorithms had identical performance.

(iii) **The accuracy of k varies with the distance function.** When $k = 2$ or $k = 3$, CONVEX$^{k\_\mathrm{NN}}$ seems to perform almost equally well. However, for three distance functions (Hamming, Manhattan, and Canberra), the accuracy of CONVEX$^{k\_\mathrm{NN}}$ with $k = 2$ is better than that with $k = 3$, while the situation is reversed for the other three distance functions.

(iv) **Hamming and Manhattan are the best distance functions.** In terms of the best accuracy, CONVEX$^{k\_\mathrm{NN}}$ with $k = 2$ using either the Hamming or the Manhattan distance seems to yield the best performance—95.1% accuracy. Actually, Hamming wins by a marginal amount (0.951355 vs. 0.951263).

The fact that using Hamming and Manhattan distance yielded the best result is not entirely surprising. Intuitively, Hamming distance is simply a count of the number of vector positions that are different, while Manhattan distance is the sum of the differences in each position. For the type of data being used here, this approach to distance seems more applicable than others, such as Euclidean or Cosine distance.

To drill down a bit deeper and see how the accuracy of the algorithms varies with the amount of training data, the performance of the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ algorithms was compared as t was varied for the Hamming and Manhattan distance functions. Figures 3.6 and 3.7 show how the accuracy of the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ algorithms vary for $k = 1, 2, 3$ using the Hamming distance and a training set size varied from 10 to 23. Both algorithms show virtually no change with t, indicating that $t = 10$ is adequate for making a highly accurate behavioral forecast, at least for the MAROB dataset. Figures 3.8 and 3.9 show the corresponding results when the Manhattan distance is used instead. In both cases (as well as for the other four distance functions) the results are similar—choosing a training set greater than 10 offers virtually no improvement in accuracy.

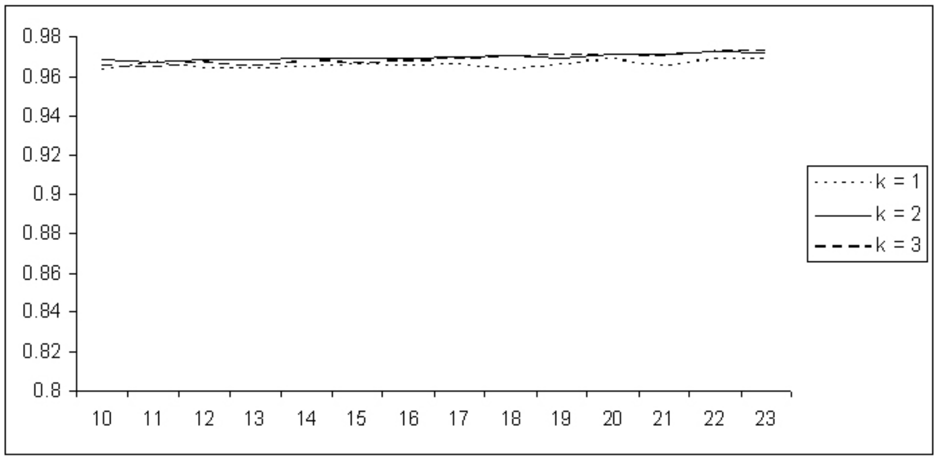

Figure 3.6: Hamming distance, $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ Algorithm

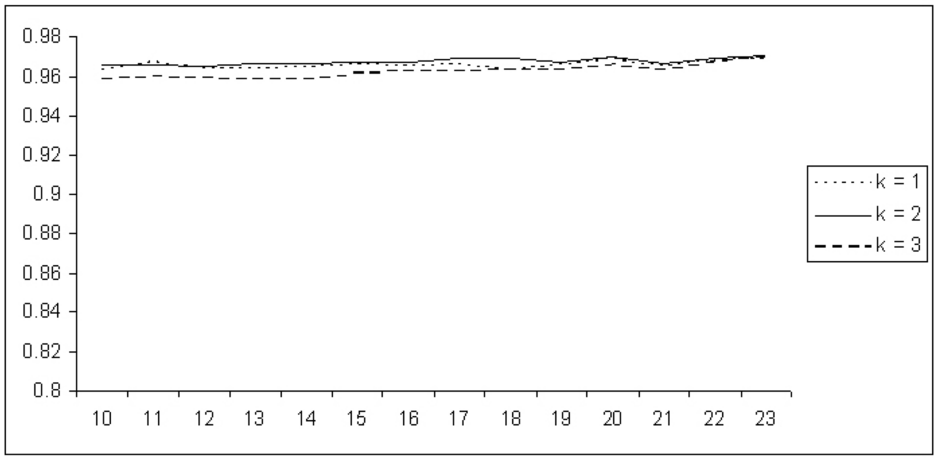

Figure 3.7: Hamming distance, CONVEXMerge Algorithm

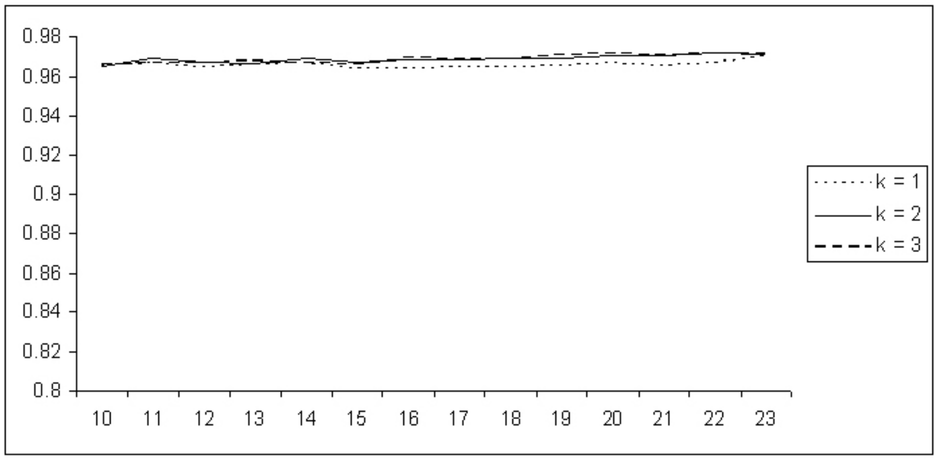

Figure 3.8: Manhattan distance, CONVEX$^{k\_\mathrm{NN}}$ Algorithm

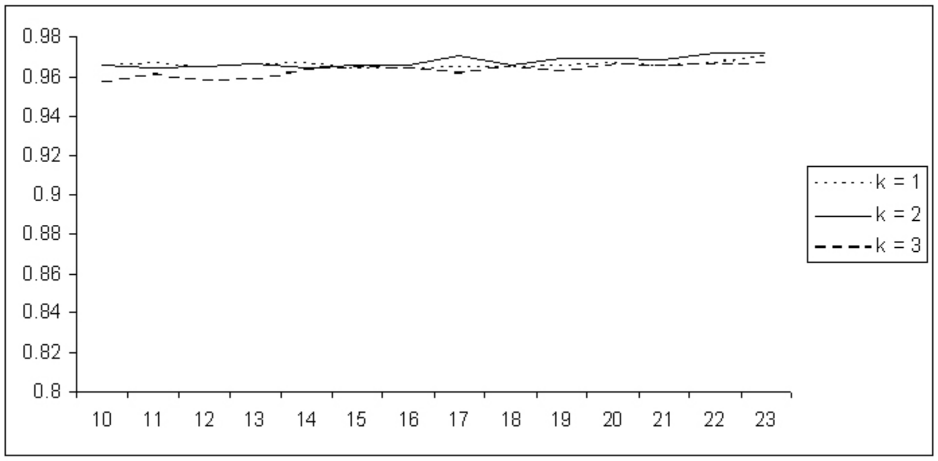

Figure 3.9: Manhattan distance, CONVEXMerge Algorithm

3.5 Conclusions

In this chapter, two general algorithms, $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$, were presented based on viewing behavioral data as points in a high dimensional metric space and assessing the distance between a query context vector and historical data. By examining the k nearest neighbors with respect to different distance functions, forecasts can be made about agent behavior in a real or hypothetical context. The main distinction between these two approaches is their weighting of the k nearest neighbors—one regards all the selected k nearest neighbors the same, while the other looks at the actual distances between the query vector and the k nearest neighbors and tries to normalize the answer based on these distances.

The initial expectation was that the second method, $\mathsf{CONVEXMerge}$, would outperform the first by taking the specific distances into account. Surprisingly, empirical tests indicate that the first algorithm, $\mathsf{CONVEX}^{k\_\mathrm{NN}}$, is consistently more accurate, independent of the amount of training data used, the choice of k, and the choice of distance function. Moreover, for the $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ algorithm, $k = 2$ provides consistently better results than $k = 1$, and the best results are obtained with either Hamming or Manhattan distance (though there is not much difference in terms of the distance metric used). Last, but not least, 10 years of training data seems to be more than enough for accurate behavioral forecasts using the MAROB dataset. Experiments on approximately 25 years of real data for 118 ethnopolitical groups in the Middle East show that the best configurations of $\mathsf{CONVEX}^{k\_\mathrm{NN}}$ and $\mathsf{CONVEXMerge}$ are able to consistently produce forecasts with over 95% accuracy.

It is important to qualify these comments with several caveats. It is impossible to assert that the algorithms in this chapter will accurately predict any kind of action vector from any kind of context vector. The MAROB data with which these algorithms were evaluated has characteristics that are symptomatic of a class of problems in the social sciences, but are by no means representative of all problems where forecasts may be desired. For example, the domains of all attributes are small (in MAROB, no variable has more than 7 possible values and most have between 2 and 4), which is also true of many other datasets in the social sciences. The accuracy of these methods when the domains of attributes are large is still an open question. Moreover, there are issues related to numerical data that need to be addressed (MAROB only contains categorical data). For example, to study variables such as GDP, percentage of arable land, etc., it is necessary to have a good approach for handling numeric data. This raises interesting questions of scale; if some attributes are on a small scale such as 0 or 1, while others are on a wider scale (e.g., 1000 to $100,000$), then the wider scale may have a disproportionate impact on the distances computed. Fortunately, there seem to be straightforward ways to normalize them.

Another important problem relates to checking whether certain subsets of context attributes (rather than all) will elicit better results. Because the accuracy of the current algorithms is already over 95%, this possibility has not yet been investigated, but it is certainly worth examining. It is also important to explore whether placing weights on attributes would lead to improved accuracy of predictions, or whether a temporal weighting of the nearest neighbors in the CONVEXMerge algorithm (i.e., giving greater weight to more recent contexts) may prove more effective.

In $\mathsf{CONVEX}^{k\text{-NN}}$ and CONVEXMerge, the user specifies a query vector representing a real or hypothetical situation, and the algorithms predict what an agent might do in that query context, allowing for highly accurate "what if" analyses. In the next chapter, a method is presented to *forecast* the query vector itself, providing more robust temporal forecasts from behavioral time series.

Chapter 4

Forecasting Future Situations from Time Series Data

The work presented in this chapter has appeared in [108].

Both of the previous chapters have presented methods for forecasting the actions that an agent might take, either through the use of stochastic behavioral models (Chapter 2) or by using context vector comparisons on behavioral time series data (Chapter 3). However, there are also many applications where it may be desirable to forecast the environmental situation (i.e., the independent variables in the previous behavioral forecasts) that an agent might experience at a particular time in the future. For example, given behavioral data about an agent for the years 1980–2010, the possible context in, say, the first half of 2011 could be predicted, and this insight can then be used to forecast the actions the agent might take during that time period. Such capability can be very important for making temporal behavioral forecasts, rather than simply "what if" analyses of user-supplied scenarios. Predictive knowledge about what behaviors an agent will take at a specific time point can be used in decision-support technologies to help users determine the best counterstrategy or mitigating policy for the future.

In this chapter the SitCAST Situation Forecaster [108] is presented for finding a probabilistic forecast over a behavioral time series of the possible future situations that could occur at some time t in the future. The SitCAST method is flexible and allows users to insert the appropriate statistical model for the data. Similar types of reasoning are also used in realms such as planning and games; however, the aim here is not to devise a plan for achieving a particular goal, or a strategy to best oppose an adversary (some of these issues will be addressed in Chapter 7). SitCAST assumes no prior strategy or goal on the part of the user and is designed to provide a forecast of what the probable contexts might be at some future time point.

The SitCAST forecaster uses the same type of behavioral time series dataset described in Section 3.1, where the attributes $\mathcal{A}$ of a relational time series T_g for an agent g can be divided into *environmental* variables, which describe the context in which g functioned, and *action* variables representing the actions taken by g in a given time frame. Each agent g has an associated context schema $CS(g) = \{C_1, \ldots, C_n\}$ of environmental variables and an action schema $AS(g) = \{A_1, \ldots, A_m\}$ of action variables.

Here, assume that each attribute V in the behavioral time series has a domain $dom(V)$ consisting only of *integers*—real-valued variables must be discretized for use with the SitCAST forecasting method. $dom(CS(g))$ denotes the domain of all the environmental variables in the set $CS(g)$, and $dom(AS(g))$ is the domain of all variables in the action schema. In addition, assume that each environmental variable can be represented as an independent time series $C_i[y]_{y=y_1}^{y_k}$, where $C_i \in CS(g)$ and

$y_1 < y_2 < \cdots < y_k$ are time periods in the table T_G with k rows corresponding to *equally spaced* time periods. $C_i[y]$ denotes the value of attribute C_i at time y.

4.1 The SitCAST Algorithm

Using behavioral time series data of the form described above, a user may want to forecast what the environmental context might be like for an agent g during time period y_{k+s} for $s \geq 1$. That is, what value will $C_i[y_{k+s}]$ have for all environmental variables $C_i \in CS(g)$ s time periods in the future? In this section, the SitCAST forecasting algorithm is proposed for answering this question, building upon any standard statistical time series forecasting algorithm [18] (e.g., linear regression, quadratic regression, or logistic regression)—denoted ts—to forecast the possible situation. Let $C_i[y_{k+s}]$ denote the value of C_i predicted by a time series algorithm ts for time period y_{k+s}.

Definition 21 (Possible Situation)**.** *If ts is a time-series forecasting algorithm and T_g a behavioral time series for agent g, a* possible situation *w.r.t. ts and time y_{k+s} is the vector $(ps(C_1), \ldots, ps(C_n))$ where ps is a mapping:*

$$ps : C_i \in CS(g) \rightarrow dom(C_i),\ for\ all\ C_1, \ldots, C_n \in CS(g)$$

such that, for all $1 \leq i \leq n$, $ps(C_i) = \lfloor C_i[y_{k+s}] \rfloor$ or $ps(C_i) = \lceil C_i[y_{k+s}] \rceil$.

In other words, in a possible situation, the value of each environmental attribute at time y_{k+s} must be set to either $\lfloor C_i[y_{k+s}] \rfloor$ or $\lceil C_i[y_{k+s}] \rceil$. Let $\mathcal{PS}(ts, y_{k+s})$ denote the set of all possible situations at time y_{k+s} using time series predictor ts.

Example 6. *Consider the simple behavioral time series T_g shown below for an agent g. The context schema consists of only four environmental variables, i.e., $CS(g) = \{C_1, C_2, C_3, C_4\}$, and the action schema is $AS(g) = \{A_1\}$.*

Time	C_1	C_2	C_3	C_4	A_1
1	2	4	1	1	a_1
2	2	3	1	2	a_2
3	1	3	1	2	a_3
4	1	3	1	2	a_4
5	1	2	1	2	a_5

Suppose the time series forecasting function ts is simple linear regression. To forecast what the environmental situation will be like at time $y_{k+1} = 6$, ts returns $C_1[6] = 0.5$, $C_2[6] = 1.8$, $C_3[6] = 1$, and $C_4[6] = 2.4$.

As all attributes (action and environment) in $\mathcal{A}$ have integer domains, it follows from Definition 21 that $ps(E_1[6])$ must be either 0 or 1, $ps(E_2[6])$ must be either 1 or 2, $ps(E_3[6]) = 1$ and $ps(E_4[6])$ is either 2 or 3. This leads to eight *possible situations in $\mathcal{PS}(ts, 6) = \{$*

$$\begin{aligned}
S_1 &= (0, 1, 1, 2), \\
S_2 &= (0, 1, 1, 3), \\
S_3 &= (0, 2, 1, 2), \\
S_4 &= (0, 2, 1, 3), \\
S_5 &= (1, 1, 1, 2), \\
S_6 &= (1, 1, 1, 3), \\
S_7 &= (1, 2, 1, 2), \\
S_8 &= (1, 2, 1, 3) \quad \}
\end{aligned}$$

A probability distribution can be induced on this set of possible situations depending on how close each value $C_i[y_{k+s}]$ predicted by ts is to the actual floor and ceiling values from $dom(C_i)$.

Definition 22 (Probability of a Possible Situation)**.** *If ts is a time series forecasting algorithm and T_g a behavioral time series for agent g, the probability of a possible situation S at time y_{k+s} w.r.t. ts and T_g is:*

$$\mathbf{P}(S, y_{k+s} \mid ts, T_g) = \prod_{c_i \in S} \mathbf{P}(c_i \mid C_i[y_{k+s}])$$

where $C_i[y_{k+s}]$ is the value forecasted by ts for attribute C_i and

$$\mathbf{P}(c_i \mid C_i[y_{k+s}]) = \begin{cases} 1 & , \textit{if } C_i[y_{k+s}] \in dom(C_i) \\ (\lceil C_i[y_{k+s}] \rceil - C_i[y_{k+s}]) & , \textit{if } c_i = \lfloor C_i[y_{k+s}] \rfloor \\ (C_i[y_{k+s}] - \lfloor C_i[y_{k+s}] \rfloor) & , \textit{if } c_i = \lceil C_i[y_{k+s}] \rceil \end{cases}$$

In the above definition, $\mathbf{P}(c_i \mid C_i[y_{k+s}])$ is the probability that attribute C_i will have value c_i, given the forecast returned by the time series prediction algorithm ts. If ts returns an integer in $dom(C_i[y_{k+s}])$, then the probability $\mathbf{P}(c_i \mid C_i[y_{k+s}]) = 1$. Otherwise, the probability is computed depending on whether c_i is the floor or ceiling of the value returned by ts. For example, recall the variable C_2 from Example 6, whose predicted value at time 6 according to ts was 1.8. According to Definition 22, this means that the value of C_2 is 2 with a probability 0.8 and the value is 1 with probability 0.2. Since it is assumed that each environmental attribute in the context schema comprises an independent time series, $\mathbf{P}(S, y_{k+s} \mid ts, T_g)$ is the product of the independent probability of each value in the possible situation.

Example 7. *Consider again the possible situations in $\mathcal{PS}(ts, y_{k+s})$ using the behavioral time series T_g from Example 6, where ts is simple linear regression and $y_{k+s} = 6$. Each of these possible situations has the following probability:*

$$\begin{aligned}
\mathbf{P}(S_1, 6 \mid ts, T_g) &= 0.5 \times 0.2 \times 1 \times 0.6 = 0.06\\
\mathbf{P}(S_2, 6 \mid ts, T_g) &= 0.5 \times 0.2 \times 1 \times 0.4 = 0.04\\
\mathbf{P}(S_3, 6 \mid ts, T_g) &= 0.5 \times 0.8 \times 1 \times 0.6 = 0.24\\
\mathbf{P}(S_4, 6 \mid ts, T_g) &= 0.5 \times 0.8 \times 1 \times 0.4 = 0.16\\
\mathbf{P}(S_5, 6 \mid ts, T_g) &= 0.5 \times 0.2 \times 1 \times 0.6 = 0.06\\
\mathbf{P}(S_6, 6 \mid ts, T_g) &= 0.5 \times 0.2 \times 1 \times 0.4 = 0.04\\
\mathbf{P}(S_7, 6 \mid ts, T_g) &= 0.5 \times 0.8 \times 1 \times 0.6 = 0.24\\
\mathbf{P}(S_8, 6 \mid ts, T_g) &= 0.5 \times 0.8 \times 1 \times 0.4 = 0.16
\end{aligned}$$

Thus, the most probable situations at time 6 *according to ts are* S_3 *and* S_7*, each with 24 % probability of occurring.*

```
Algorithm SitCAST($T_g, CS(g), ts, y_{k+s}$)
1.     Situations $= \emptyset$
2.     Poss_vals $= \emptyset$
3.     for each $C_i \in CS(g)$ do
4.         $Pred = ts(T_g, C_i, y_{k+s})$
5.         if $Pred \notin dom(C_i)$ then
6.             $P_{floor} = \lceil Pred \rceil - Pred$
7.             $P_{ceil} = 1 - P_{floor}$
8.             add $(C_i, \lfloor Pred \rfloor, P_{floor})$ to $Poss\_vals$
9.             add $(C_i, \lceil Pred \rceil, P_{ceil})$ to $Poss\_vals$
10.        else add $(C_i, Pred, 1)$ to $Poss\_vals$
11.    $\mathcal{PS}(ts, y_{k+s}) = combinations(Poss\_vals)$
12.    for each $S_j \in \mathcal{PS}(ts, y_{k+s})$ do
13.        add $(S_j, \prod_{v \in S_j} prob_v \; s.t. \; (C_i, v, prob_v) \in Poss\_vals)$ to
           $Situations$
14.    return $Situations$
```

Figure 4.1: The SitCAST algorithm for forecasting all possible situations for time period y_{k+s}.

Figure 4.1 presents the SitCAST forecasting algorithm, which will predict all of the possible situations that could occur at some time in the future, along with their associated probabilities. Given a behavioral time series T_g for an agent g, a context schema $CS(g)$ of environmental attributes, a time series prediction function ts, and a time period y_{k+s}, SitCAST first looks at each environmental variable in

Lines 3–10 to find the possible values according to ts and compute their probability $\mathbf{P}(c_i \mid C_i[y_{k+s}])$. Each of these values and probabilities $(C_i, v, prob_v)$ is stored in $Poss\_vals$. Then, in Line 11, $\mathcal{PS}(ts, y_{k+s})$ is computed with the call to the procedure *combinations*, which finds all possible combinations of the environmental attribute values from $Poss\_vals$. Finally, in Line 13, the probability of each possible situation is computed and stored in *Situations*, which is returned as the probabilistic forecast of all contexts an agent could experience at time y_{k+s}.

4.2 SitCAST and CONVEX

In Chapter 3, two algorithms were introduced, CONVEX$^{k\_\mathrm{NN}}$ (Figure 3.2) and CONVEXMerge (Figure 3.3), that are able to forecast the actions an agent will take in a given context (either real or hypothetical) by examining the similarity between this query context and past situations experienced by the agent [75]. Rather than relying on users to provide a possible situation as input for a "what if" analysis, the future situations forecast by SitCAST can be used as possible query vectors, and CONVEX$^{k\_\mathrm{NN}}$ (resp. CONVEXMerge) can be used to predict the behaviors an agent will take in each of the possible situations. Thus, SitCAST and CONVEX$^{k\_\mathrm{NN}}$ (resp. CONVEXMerge) are able to jointly forecast what actions an agent will take at some time point y_{k+s} in the future, together with a probability. For the remainder of this chapter, the CONVEX$^{k\_\mathrm{NN}}$ and CONVEXMerge algorithms are referred to collectively as CONVEX.

Definition 23. *(Probability of an Action) If T_g is a behavioral time series for agent g, $AS(g)$ is an action schema, and $\mathcal{PS}(ts, y_{k+s})$ is a set of possible situations, then* *the probability that action attribute $A_j \in AS(g)$ has the value a at time y_{k+s} is:*

$$\mathbf{P}(A_j[y_{k+s}] = a) = \sum_{S \in \mathcal{PS}(ts, y_{k+s}) \,\wedge\, A_j = a \in \textsf{\textit{CONVEX}}(T_g, S, d, AS(g), k)} \mathbf{P}(S, y_{k+s} \mid ts, T_g)$$

In other words, the probability of action attribute $A_j[y_{k+s}]$ having the value a at time y_{k+s} is computed as follows.

1. Find all possible situations $S \in \mathcal{PS}(ts, y_{k+s})$ that can arise at time y_{k+s} such that CONVEX forecasts action variable $A_j = a$ in those situations.

2. Sum the probability $\mathbf{P}(S, y_{k+s} \mid ts, T_g)$ of all such situations. This is the probability that $A_i[y_{k+s}] = a$.

Example 8. *Consider the same 8 possible situations and their respective probabilities given by SitCAST in Example 7. Suppose that there are three possible values in the domain $dom(A_1) = \{0, 1, 2\}$, and that CONVEX forecasts the action vectors:*

$(A_1 = 0)$ *in possible situations S_1, S_4, S_6.*

$(A_1 = 1)$ *in possible situations S_2, S_7.*

$(A_1 = 2)$ *in possible situations S_3, S_5, S_8.*

The probability that $A_1[6] = 0$ above is the sum of the probabilities that either situation S_1, S_4, or S_6 will occur. These probabilities are 0.06, 0.16, and 0.04, respectively, yielding a 26 % probability that $A_1 = 0$ at time 6. The probabilities for all possible actions are:

$$\mathbf{P}(A_1[6] = 0) = 0.06 + 0.16 + 0.04 = 0.26$$

$$\mathbf{P}(A_1[6] = 1) = 0.04 + 0.24 = 0.28$$

$$\mathbf{P}(A_1[6] = 2) = 0.24 + 0.06 + 0.16 = 0.46$$

This result indicates that the most likely *outcome based on the possible situations at time* 6 *is that the agent in question will engage in the action denoted by* $A_1 = 2$ *with 46% probability.*

To make a forecast, then, of what actions an agent will take at some point in the future, SitCAST+CONVEX will select the most probable action values for each action attribute A_j and combine these into a single most probable action vector. Note that, although CONVEX itself runs in polynomial time, the combined SitCAST+CONVEX algorithm can take exponential time because the number of possible situations to consider may be exponential in the number of environmental variables (i.e., the size of the context schema $CS(g)$). One way around this problem is to introduce a window size, w, for generating the possible situations. Let w be a real number between 0 and 0.5 inclusive. Whenever an environmental variable is predicted to be a real number r between $\lfloor r \rfloor$ and $\lceil r \rceil$, one of the following options is applied:

- If $\lfloor r \rfloor + 0.5 - w \leq r \leq \lfloor r \rfloor + 0.5 + w$, then generate two possible situations.
- Otherwise, if $r \leq \lfloor r \rfloor + 0.5 - w$, then reset r's value to $\lfloor r \rfloor$.
- Otherwise, if $r \geq \lfloor r \rfloor + 0.5 + w$, then reset r's value to $\lceil r \rceil$.

To see how this works, suppose $w = 0.1$. This means that if the predicted value $C_i[y_{k+s}]$ is in the range $j.4$ to $j.6$ for an integer j, then two possible situations are generated for $C_i[y_{k+s}]$. Otherwise, if the predicted value is between j and $j.4$, then the value is modified to j. If the predicted value is between $j.6$ and $j + 1$, then the value is modified to $j + 1$.

Example 9. *Recall the forecasts made for each environmental attribute by ts in Example 6, and suppose a window size of* $w = 0.1$.

1. $C_1[6] = 0.5$. *In this case, no change is made to* $C_1[6]$ *because* 0.5 *is within the window* $[0.4, 0.6]$, *i.e.,* $\lfloor 0.5 \rfloor + 0.5 - 0.1 = 0.4 \leq 0.5 \leq \lfloor 0.5 \rfloor + 0.5 + 0.1 = 0.6$.
2. $C_2[6] = 1.8$. *In this case,* $C_2[6]$ *is reset to* 2 *because* $1.8 \geq \lfloor 1.8 \rfloor + 0.5 + 0.1 = 1.6$.
3. $C_3[6] = 1$. *In this case,* $C_3[6]$ *stays 1 because* $1 \leq \lfloor 1 \rfloor + 0.5 - 0.1 = 1.4$.
4. $C_4[6] = 2.4$. *In this case, no change is made to* $C_4[6]$ *because* 2.4 *is in the window* $[2.4, 2.6]$, *i.e.,* $\lfloor 2.4 \rfloor + 0.5 - 0.1 = 2.4 \leq 2.4 \leq \lfloor 2.4 \rfloor + 0.5 + 0.1 = 2.6$.

By using this method, SitCAST will now only generate four possible situations, and $\mathcal{PS}(ts, 6) = \{$

$$\begin{aligned} S'_1 &= (0, 2, 1, 2), \\ S'_2 &= (0, 2, 1, 3), \\ S'_3 &= (1, 2, 1, 2), \\ S'_4 &= (1, 2, 1, 3) \quad \} \end{aligned}$$

This result shows a 50 % reduction in the number of possible situations generated by SitCAST, and hence the amount of possible query vectors that must be run through the CONVEX algorithm. If $w = 0.31$ *had been set as the window size, then there would only be two possible situations to consider:* $\mathcal{PS}(ts, 6) = \{$

$$\begin{aligned} S^\star_1 &= (0, 2, 1, 2), \\ S^\star_2 &= (1, 2, 1, 2) \quad \} \end{aligned}$$

4.3 Implementation and Experiments

A prototype implementation of the SitCAST algorithm has been developed using about 6,000 lines of Java code. For the statistical time series models, the Flanagan Java Scientific library [45] was used to implement a polynomial regression and a logistic regression for forecasting the values of the environmental variables. A specialized variant of the Minorities at Risk Organizational Behavior (MAROB) data [46, 125] (introduced in Chapters 2 and 3) was used as the behavioral time series for these experiments. From the MAROB data, 28 environmental variables were selected as the context schema $CS(g)$ for each of 7 terrorist organizations under consideration. These context attributes included such things as sources of funding for the group, its involvement in electoral politics, its internal structure, etc. About 5 action attributes were chosen as $AS(g)$ for each group, including various strategic level behaviors in which the group might engage, such as rebellions, violent clashes with other groups, etc. The time series for this data consisted of 24 time periods (i.e., $k = 24$) representing semi-annual periods from 1995 to 2006.

Group	SitCAST	SitCAST+ CONVEX
1	0.855	0.4
2	0.765384615	0.769230769
3	0.843846154	0.361538462
4	0.857692308	0.523076923
5	0.875384615	0.630769231
6	0.814615385	0.623076923
7	0.875384615	0.875384615
	0.841043956	0.597582418

Table 4.1: Accuracy of the SitCAST and SitCAST+ CONVEX algorithms applied to behavioral time series data for 7 terrorist organizations.

Using this data, experiments were run using SitCAST alone to forecast the possible situations, as well as SitCAST+CONVEX to predict the groups' behaviors. All experiments were performed on a Linux computing cluster comprised of 64 8-core nodes with between 10GB and 20GB of RAM. For the statistical time series forecaster, experiments were run with polynomial regressions of order 1, 2, and 3, as well as logistic regression of order 1 and 100. For purposes of managing the computational complexity, the CONVEX algorithm was only run on the most probable of the situations forecast by SitCAST, rather than combining the action predictions across all possible environmental forecasts. The basic CONVEX$^{k\_\mathrm{NN}}$ algorithm (Figure 3.2) was used with $k = 2$ and the Hamming distance function, which were empirically shown in the last chapter (Section 3.4) to be the most effective parameter settings for behavioral forecasts with the MAROB data.

Table 4.1 shows the results from these experiments. SitCAST has an average accuracy of 84.1% for forecasting the environmental variables alone. When combined with CONVEX to forecast the actions taken by the respective groups, the SitCAST+CONVEX algorithm has an average accuracy of 59.76%. Remember from Chapter 3 that CONVEX is able to achieve a forecast accuracy of 95.1% for a given query context. However, due to the uncertainty in the probabilistic forecasts (and since the most probable situation at time y_{k+s} does not represent the full range of what might occur), SitCAST+CONVEX is less precise in its behavioral predictions.

4.4 Conclusions

In this chapter, the SitCAST method was presented for forecasting contextual situations in a behavioral time series dataset. Such forecasts can prove useful both individually to inform analysts what the possible environment might be at some time in the future, as well as in conjunction with a behavioral forecaster such as CONVEX, allowing temporal extrapolation of not only the environment, but also the actions that an agent might take.

The efficacy of SitCAST as a means of forecasting the context attributes in a behavioral time series has been empirically shown, as the algorithm achieves a very high accuracy in predicting future situations. However, further experimentation is necessary to examine the performance of SitCAST under different conditions and with different datasets. In particular, CONVEX was only used to forecast the actions resulting from the most probable situation produced by SitCAST; the results of the system may improve if the full distribution of action attributes over all possible situations is computed, or if different situation pruning techniques are used.

While SitCAST+CONVEX performs reasonably well under the test conditions in this chapter with the MAROB data, the question arises regarding what accounts for the 40% of the actions that were not accurately predicted. One theory is that, because SitCAST+CONVEX are based on statistical regression and vector similarity methods, they only look at the most frequently used past behaviors—the "normal" operating procedures—of an agent. While this approach will be effective most of the time, these algorithms are unable to predict situations where an agent might change its behavior or evolve a new strategy. This problem of behavioral change is addressed in detail in the following chapter, and a novel system, the Change Analysis Predictive Engine (CAPE) [108], is proposed as a comprehensive method for forecasting agent behavior that includes divergence from the norm.

Chapter 5

Automatically Forecasting Changes in Agent Behavior

The work in this chapter has appeared in [108].

Agent behavior is a continuously evolving, dynamic phenomenon. In the previous chapters, several methods were introduced—SOMA (Chapter 2), CONVEX (Chapter 3), and SitCAST (Chapter 4)—that can be used to forecast an agent's behavior or contextual situation based on the historical likelihood of particular behavioral patterns in a time series dataset. Indeed, most past work [75, 97, 116, 115, 16] on modeling agent behaviors focuses on learning a model of the typical behavior of the agent, and then using that model to predict what the agent might do in the future. In contrast, this chapter discusses algorithms to determine when a given agent will *change* its behaviors or strategy from the expected norm. Though such occurrences may have a lower probability, it is often crucial to understand when an agent's behavior will diverge from the past in order to take mitigating actions or modified policy responses.

For example, it is well-known that terrorist groups are constantly evolving and adapting their behavior. When a group establishes a standard operating procedure over an extended period of time, the problem of predicting what that group will do in a given situation (real or hypothetical) is easier than the problem of determining when, if, and how the group will exhibit a significant change in its strategic actions.

Methods such as CONVEX [75] described in Chapter 3 can produce highly accurate forecasts of what a given agent will do in a particular situation of interest based on its past behaviors in similar contexts. These "what if" forecasts can be generalized to temporal predictions of agent behavior by applying the CONVEX algorithms jointly with the SitCAST Situation Forecaster [108] from Chapter 4, producing a probabilistic forecast of an agent's behavior at some time in the future (see Section 4.2). However, the ability of such systems to predict when an agent will change its behavior has yet to be proven. As explained in Section 4.3, much of the forecasting error incurred by SitCAST+CONVEX arises when the agent in question changes its behavior. Thus, incorporating the ability to predict changes into the behavioral forecasting mechanism will improve the overall accuracy.

In this chapter, an architecture called the Change Analysis Predictive Engine (CAPE) is proposed as a comprehensive system to effectively predict behavior, including when and how an agent will *change* its behaviors. The CAPE algorithms have been tested on about 10 years of real-world data from the MAROB [46, 125] dataset for 5 terrorist organizations in two countries and—in those cases at least—have proven to be highly accurate.

The rest of this chapter details how this forecasting has been accomplished with the CAPE algorithms. In Section 5.1, the general architecture of the CAPE system is described, which incorporates the CONVEX and SitCAST forecasters presented in Chapters 3 and 4, respectively. Section 5.2 focuses on the important problem of forecasting changes in agent behavior. Intuitively, CAPE tries to learn conditions under which an agent has changed its behaviors and use these rules to predict the onset of future behavioral changes. Unlike the focus of systems such as SOMA [116, 115, 107, 62, 61] and CONVEX [75], the major object of study is *change in behavior*, and not the *behavior* itself. The concept of a *change table* is presented for representing changes, and *change analysis algorithms* are proposed to study this table, generating rules that determine the conditions under which agents changed their behaviors. This section concludes with a formal definition of the CAPE algorithm, which builds on the SitCAST+CONVEX method described in Section 4.2. Intuitively, when a behavioral change is predicted, CAPE reports this change, otherwise, it will forecast the typical behavior found by SitCAST+CONVEX.

In Section 5.3, *preliminary* results of experimental evaluations are presented where the CAPE system is applied to behavioral time series data for several terror groups. Depending upon the granularity with which the actions are modeled, these algorithms are either 80 % accurate, or about 69.32 % accurate. This result suggests that the granularity at which predictions are made is a key issue to be explored, especially when trying to understand the full picture of agent behavior and the potential onset of strategic changes.

5.1 CAPE Architecture

Behavioral time series (Section 3.1)—datasets that track the behavioral and contextual information for an agent over time—can be analyzed to understand not only what an agent is most likely to do in the future in the average case, but also

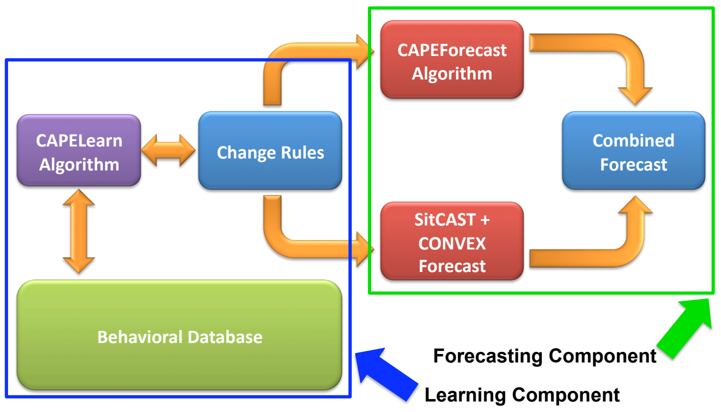

Figure 5.1: Architecture of the CAPE framework.

when an agent is likely to change its behavior. Figure 5.1 shows the general architecture of the Change Analysis Predictive Engine (CAPE), a system for modeling behavioral change. The CAPE method consists of two major components—one focusing on *learning* the conditions under which an agent changed its behavior in the past, and another focusing on *forecasting* what the agent will do at some time in the future, including whether or not it will exhibit a change in strategy.

The learning architecture underlying CAPE processes a relational behavioral time series $T(g)$ to learn a model of behavioral changes for agent g. Recall that these datasets contain attributes $\mathcal{A}$ which can be divided into a context schema $CS(g) = \{C_1, \ldots, C_n\}$ of *environmental (independent) variables* describing the context in which an agent g functioned during a given time frame, and an action schema $AS(g) = \{A_1, \ldots, A_m\}$ of *action (dependent) variables* describing actions taken by the agent. Each attribute $V \in \mathcal{A}$ has an associated domain $dom(V)$, and $dom(CS(g))$ and $dom(AS(g))$ denote the domains of the environmental and action variables, respectively. Assume that each environmental variable can be represented as an independent time series $C_i[y]_{y=y_1}^{y_k}$ where $y_1 < y_2 < \cdots < y_k$ are time periods in the table T_g, which has k rows corresponding to *equally spaced* time periods. $V[y]$ denotes the value of variable $V \in \mathcal{A}$ at time period y.

A behavioral time series that meets the above criteria—the Minorities at Risk Organizational Behavior (MAROB) database [46, 125]—is used throughout this chapter. Examples of environmental variables from MAROB, which tracks information on violent ethnopolitical organizations, include such things as the level of diaspora support, the level of foreign state financial support, the level of repression experienced by a group, and so forth. Action variables represent the strategic or tactical behaviors of a group, such as whether they used suicide bombings as a strategy, whether they used attacks against domestic security organizations, whether they mounted transnational attacks on foreign groups, etc.

From the input data, the CAPE-Learn algorithm in Figure 5.1 constructs a *change table*, which captures that part of a behavioral time series table that changed. A *change analyzer* then analyzes the change table automatically and learns environmental conditions that specify when the behavior of an agent changed. The output of CAPE-Learn is a set of *change rules* that indicate conditions under which an agent is likely to alter its behavior.

The forecasting component of CAPE in Figure 5.1 builds on the CONVEX [75] and SitCAST [108] forecasters described in Chapters 3 and 4. The CONVEX algorithms forecast what an agent might do in a given situation, while SitCAST produces a set of possible situations that might be true at time y_{k+s} in the future and a probability distribution over that set of situations. When these possible situations are used as query context vectors for CONVEX, SitCAST+CONVEX jointly forecasts what actions an agent will take at some time point y_{k+s} in the future, together with a probability. The CAPE-Forecast algorithm applies the change rules generated by the learning component of CAPE to a given situation to determine the onset of behavioral changes. If the change rules indicate a likely change in behavior, then this is the result predicted by CAPE. Otherwise, CAPE-Forecast returns the prediction from the combined SitCAST+CONVEX method.

5.2 The **CAPE** Algorithms

SitCAST+CONVEX does fairly well in predicting what actions an agent will take at a given time (see Section 4.3); however, both the CONVEX algorithms and SitCAST focus on how an agent will behave based on *typical past behaviors.* Agents occasionally change their behaviors, and predicting such *changes* in behavior is often much more important than predicting that the behavior will conform to what is most normal. In this section, the CAPE algorithms are described for learning models of behavioral change and forecasting future behaviors. That is, for any action attribute $A_j \in AS(g)$ for an agent g, what are the environmental conditions under which the value for A_j will change?

5.2.1 The Change Table

When learning the conditions under which an agent changes its behavior, the first step is to construct a *change table*, a specialized relational data structure for storing changes that occurred in a behavioral time series.

Definition 24 (Change Table)**.** *Let T_g be a behavioral time series for agent g with context schema $CS(g)$ and action schema $AS(g)$. The* change table *$CH(g, A_j)$ for agent g w.r.t. $A_j \in AS(g)$ is a table derived from T_g as follows:*

1. *The set of rows in $CH(g, A_j)$ is given by $\{y_t \mid A_j[y_t] \neq A_j[y_{t-1}]\}$.*

2. *The set of columns in $CH(g, A_j)$ consists of the column associated with A_j and the set of all $C_i \in CS(g)$ such that $C_i[y_{t-1}] \neq C_i[y_{t-2}]$.*

3. *If $C_i[y_{t-1}]$ is not eliminated in the previous two steps, then its value is set to the pair $(C_i[y_{t-2}], C_i[y_{t-1}])$.*

4. *If $A_j[y_t]$ is not eliminated in the first two steps above, then its value is set to the pair $(A_j[y_{t-1}], A_j[y_t])$.*

In other words, the change table $CH(g, A_j)$ for agent g w.r.t. action A_j eliminates all rows in the original table T_g where the action A_j did not exhibit a change in value from the previous time period y_{t-1}. In addition, it eliminates all columns except for those environmental variables which changed from $C_i[y_{t-2}]$ to $C_i[y_{t-1}]$. The change table *documents the changes that have occurred in the original table for agent g, based on the assumption that changes in an environmental variable from time period y_{t-2} to y_{t-1} are potentially responsible for a change in the action variable one time period later, i.e., from time period y_{t-1} to y_t.* It is easy to account for other time lags in a similar manner.

Example 10. *Consider the simple behavioral time series T_g shown below for agent g. The context schema consists of only four environmental variables, i.e., $CS(g) = \{C_1, C_2, C_3, C_4\}$, and the action schema is $AS(g) = \{A_1\}$.*

Time	C_1	C_2	C_3	C_4	A_1
1	2	4	1	1	1
2	2	3	1	2	1
3	1	3	1	2	2
4	1	3	1	2	2
5	1	2	1	2	1

To construct a change table from T_g, the changes in the value of action A_1 are first identified. A_1 changed twice in this data—once in time period 3 and once in time period 5. The resulting change table will have only two rows after the execution of step 1 from Definition 24.

Time	C_1	C_2	C_3	C_4	A_1
3	1	3	1	2	2
5	1	2	1	2	1

Next, all context attributes that did not change in the time interval preceding the action changes get eliminated. Thus, the columns corresponding to attributes C_1 and C_3 are removed because they did not change their values in either of the time periods from 1 to 2 or from 3 to 4. After step 2, we now have the following table.

Time	C_2	C_4	A_1
3	3	2	2
5	2	2	1

In step 3, the values of the environmental variables in the above table are replaced by the pairs of values associated with the changes that occurred.

The first row of this table, for example, indicates that the value of C_2 changed from 4 in time 1 to 3 in time 2 and that the value of C_4 changed from 1 in time 1

Time	C_2	C_4	A_1
3	(4,3)	(1,2)	2
5	(3,3)	(2,2)	1

to 2 *in time* 2*. Finally, this same replacement is made for the action* A_1*, resulting in the final change table.*

Time	C_2	C_4	A_1
3	(4,3)	(1,2)	(1,2)
5	(3,3)	(2,2)	(2,1)

The row associated with time 3 *says that there was a change in the action variable* A_1 *from value* 1 *in time* 2 *to value* 2 *in time* 3*. The fact that* C_2 *changed from time* 1 *(value* 4*) to time* 2 *(value* 3*) and* C_4 *changed from time* 1 *(value* 1*) to time* 2 *(value* 2*) mean that these changes in the context are potential causes or indicators of the change in attribute* A_1 *one time period later.*

The final change table, as shown above, encapsulates all past changes of the action A_j. More importantly, the final change table dramatically reduces the number of possible environmental attributes under consideration when modeling the conditions under which these changes occur.

Note that in the form presented above, changes in the environmental variables from time y_{t-2} to y_{t-1} are used to learn changes in the action variables from time period y_{t-1} to y_t. In other words, these change rules can be used to make predictions "one time period ahead." It is easy to learn rules that consider a larger time lag between environmental and action changes via the following simple changes to the change table construction. In the following definition, a time lag of h time periods is considered.

Definition 25 (h-Change Table)**.** *Let T_g be a behavioral time series for agent g with context schema $CS(g)$ and action schema $AS(g)$. For integer $h \geq 1$ the h-*change table *$CH^h(g, A_j)$ for agent g w.r.t. $A_j \in AS(g)$ is a table derived from T_g as follows:*

1. *The set of rows in $CH^h(g, A_j)$ is given by $\{y_t \mid A_j[y_t] \neq A_j[y_{t-1}]\}$.*

2. *The set of columns in $CH^h(g, A_j)$ consists of the column associated with A_j and the set of all $C_i \in CS(g)$ such that $C_i[y_{t-h}] \neq C_i[y_{t-h+1}]$.*

3. *If $C_i[y_{t-h+1}]$ is not eliminated in the previous two steps, then its value is set to the pair $(C_i[y_{t-h}], C_i[y_{t-h+1}])$.*

4. *If $A_j[y_t]$ is not eliminated in the first two steps above, then its value is set to the pair $(A_j[y_{t-1}], A_j[y_t])$.*

This definition merely changes the parameters associated with the potential environmental indicators C_i to look h time periods prior to an action change in A_j, rather than just one time period back.

5.2.2 Learning Change Indicators from the Change Table

By isolating the changes that occurred in the action and context attributes in a behavioral time series, the change table provides a set of potential correlations, that is, changes in environmental attributes that can be indicators of behavioral changes. When learning conditions that forecast a change in an action variable, only *conjunctive conditions* on context attributes are considered. If C_i is an environmental variable in $CS(g)$ for an agent g, and $c_f, c_t \in \mathit{dom}(C_i)$ are values in its domain s.t. (c_f, c_t) is a pair of values in the change table, then $C_i(c_f, c_t)$ is an *environmental change atom*. If $C_{i_1}(c_{f_1}, c_{t_1}), \ldots, C_{i_n}(c_{f_n}, c_{t_n})$ are environmental change atoms, then $EC = (C_{i_1}(c_{f_1}, c_{t_1}) \wedge \ldots \wedge C_{i_n}(c_{f_n}, c_{t_n}))$ is an *environmental change condition of size n.* There is no loss of generality in the assumption that only conjunctions are considered because *sets* of conditions are being learned and, hence, disjuncts can be easily accounted for. Two metrics, $precision(EC)$ and $recall(EC)$ can be defined to indicate the strength of the correlation between an environmental change condition EC and a particular behavioral change. These definitions use the usual concept of satisfaction of a condition by a tuple [120] w.r.t. the semantics of the value pairs stored in the change table. For example, in a change table, $y \models EC$ for a tuple at time period y and an environmental change condition $EC = (C_{i_1}(c_{f_1}, c_{t_1}) \wedge \ldots \wedge C_{i_n}(c_{f_n}, c_{t_n}))$ means that $C_{i_j}[y] = (c_{f_j}, c_{t_j})$ for all C_{i_j} in EC. On the other hand, in a behavioral time series, $y \models EC$ means that $C_{i_j}[y] = c_{t_j}$ and $C_{i_j}[y-1] = c_{f_j}$ for all C_{i_j} in EC.

Definition 26 (Precision)**.** *Let $CH(g, A_j)$ be a change table for agent g w.r.t. an action attribute A_j and EC be an environmental change condition. The* precision *of condition EC w.r.t. a change in A_j from a_f to a_t is defined as*

$$precision(EC) = \frac{card(\{y \mid y \in CH(g, A_j) \wedge y \models EC \wedge A_j[y] = (a_f, a_t)\})}{card(\{y \mid A_j[y] = (a_f, a_t)\})}$$

Intuitively, the numerator of the term $precision(EC)$ is the number of time periods y in the change table which satisfy EC and where action variable A_j changed value from a_f to a_t. The denominator represents the number of time periods y in the change table where action variable A_j changed value from a_f to a_t. Thus, $precision(EC)$ computes the conditional probability of EC being true, given that the action variable A_j changed value from a_f to a_t according to the change table. Clearly, this conditional probability should be high for an EC to be considered a good predictor of change in action A_j. Note that the precision is computed without ever looking at the original behavioral table—only the change table is used.

In contrast, the *recall* of an environmental condition EC connects the change table with the original time series.

Definition 27 (Recall). *Let T_g be a behavioral time series for agent g and EC be an environmental change condition. The* recall *of condition EC w.r.t. a change in A_j from a_f to a_t with time lag $h \geq 1$ is defined as:*

$$recall(EC) = \frac{card(\{y \mid y \in T_g \wedge y \models EC \wedge A_j[y+h-2] = a_f \wedge A_j[y+h-1] = a_t\})}{card(\{y \mid y \in T_g \wedge y \models EC\})}$$

In the above definition, $recall(EC)$ is the ratio of the number of time periods y in the original table T_g which satisfy EC and where action variable A_j changed value from a_f to a_t, over the number of time periods y in T_g that satisfy EC. Intuitively, this is the conditional probability of action attribute A_j changing its value from a_f to a_t, given that environmental condition EC is true according to the *original behavioral table* (not the change table). Obviously, environmental change conditions that are strong predictors will also have a high recall.

```
Algorithm CAPE-Learn($T_g, CH(g, A_j), a_f, a_t, sz, pt, rt,$
 $STAT\text{-}TESTS$)
1.      $Rules = \emptyset$
2.      $\mathcal{C} = \{EC \mid EC$ is an environmental change
        condition of size $\leq sz$ and each atom in
        $EC \in CH(g, A_j)\}$
3.      for each $EC \in \mathcal{C}$ do
4.          $precision(EC) = \frac{card(\{y \mid y \in CH(g,A_j) \wedge y \models EC \wedge A_j[y]=(a_f,a_t)\})}{card(\{y \mid A_j[y]=(a_f,a_t)\})}$
5.          $recall(EC) = \frac{card(\{y \mid y \in T_g \wedge y \models EC \wedge A_j[y+h-2]=a_f \wedge A_j[y+h-1]=a_t\})}{card(\{y \mid y \in T_g \wedge y \models EC\})}$
6.          if $precision(EC) \geq pt \wedge$
                $recall(EC) \geq rt \wedge STAT\text{-}TESTS(EC)$ then
7.              add $A_j(a_f, a_t) : [recall(EC)] \leftarrow EC$ to $Rules$
8.      return $Rules$
```

Figure 5.2: The CAPE-Learn algorithm for learning behavioral change rules from an agent's change table.

The CAPE-Learn algorithm given in Figure 5.2 learns environmental change conditions from the change table for an agent g that are indicators of a particular change in action attribute A_j from value a_f to a_t. CAPE-Learn also incorporates a special boolean function called *STAT-TESTS*, which takes an environmental change condition EC and tests whether it satisfies certain statistical conditions that a user may wish to impose. There are no restrictions whatsoever on how *STAT-TESTS* may be implemented—it could compute p-values and ensure that they fall within a given bound, or it might involve a t-test, or confidence intervals, or no additional tests at all, depending on the needs of the user.

In Line 3, CAPE-Learn begins to cycle through the set of all environmental change conditions of size sz or less composed of environmental change atoms in the change table. If a condition EC has a sufficiently high precision (exceeding a given threshold pt), a sufficiently high recall (exceeding a threshold rt), and satisfies the designated statistical tests in *STAT-TESTS*, then on Line 7 the *change rule* $A_j(a_f, a_t) : [recall(EC)] \leftarrow EC$ is added to the set *Rules*. The change rules generated by CAPE-Learn indicate that, for an agent g, the given change in action attribute A_j from a_f to a_t will occur with a probability of $recall(EC)$ if EC is true.

Proposition 3. *Suppose $CH(g, A_j)$ is a change table for agent g w.r.t. action attribute A_j and $\mathcal{C}$ is the set of all environmental change conditions with maximum size sz composed of the atoms in $CH(g, A_j)$. The complexity of the CAPE-Learn algorithm is*

$$\mathbf{O}(card(\mathcal{C}) \times card(CH(g, A_j))).$$

If $CH(g, A_j)$ contains c columns and each column has at most b values in it, then this complexity boils down to

$$\mathbf{O}(b^c \times card(CH(g, A_j))).$$

Though there is an exponential factor in this computation, c has been quite small in practice. The complexity of the CAPE-Learn algorithm is dominated by the computation of the set $\mathcal{C}$ of environmental change conditions. $\mathcal{C}$ is constructed using only environmental change atoms that occur in the change table, which is usually substantially smaller than the original time series. For instance, the size of the change table, as a proportion of the size of the original table for experiments on 5 terrorist groups in the MAROB dataset is given in Figure 5.3. In other words, the size of the change table is *very small* compared to the size of the original behavioral time series—typically around 3 to 4 % of the size.

Group	T_g Size	$CH(g, A_j)$ Size	$\frac{CH(g,A_j)\ Size}{T_g\ Size}$
1	1224	40	3.27 %
2	1224	37	3.02 %
3	1176	33	2.81 %
4	1176	51	4.34 %
5	1176	35	2.98 %

Figure 5.3: Size of the change table relative to the original behavioral table for 5 MAROB groups

5.2.3 The **CAPE-Forecast** Algorithm

The purpose of the change rules generated by the CAPE-Learn algorithm is to provide a mechanism for forecasting behavioral change at some time y_{k+s} in the

```
Algorithm CAPE-Forecast($T_g, CH^h(g, A_j), dom(A_i), sz, pt, rt,$
$\mathit{STAT\text{-}TESTS}$)
1.      $forecast = null$
2.      $probability = 0$
3.      for each $a_j \in dom(A_j)$ $s.t.$ $A_j[y_k] \neq a_j$ do
4.          $Rules =$ CAPE-Learn($T_g, CH^h(g, A_j), A_j[y_k], a_j, sz, pt, rt,$
                $\mathit{STAT\text{-}TESTS}$)
5.          for each $A_j(a_f, a_t) : [recall(EC)] \leftarrow EC \in Rules$ do
6.              if $y_{k-h}, y_{k-h+1} \in T_g \wedge y_{k-h+1} \models EC \wedge recall(EC) >$
                $probability$ then
7.                  $probability = recall(EC), forecast = a_j$
8.      if $forecast = null$ then
9.          $forecast =$ SitCAST + CONVEX $forecast$
10.         $probability =$ SitCAST + CONVEX $probability$
11.     return $(forecast, probability)$
```

Figure 5.4: The CAPE-Forecast algorithm for forecasting agent behavior, including possible behavioral changes.

future. The CAPE-Forecast algorithm in Figure 5.4 uses these change rules to make such a temporal behavioral forecast. Given a behavioral time series T_g containing data for agent g from time periods $y_1, \ldots, y_k$, and a change table $CH^h(g, A_j)$ with time lag h, this algorithm will forecast the value of action variable A_j from its domain $dom(A_j)$ during the next future time period, i.e., y_{k+1}.

CAPE-Forecast first uses CAPE-Learn to generate all possible change rules for action attribute A_j going from its current value $A_j[y_k]$ to any other possible value in its domain for time y_{k+1}. Each of the environmental change conditions in these rules is compared to the historical context at y_{k-h}; if an applicable rule is found then it is used to forecast the corresponding change in A_j with the probability $recall(EC)$. In the event of a conflict in what the rules predict, a rule with the highest recall is chosen. If a conflict still exists, the first forecasted value is chosen. If this procedure does not forecast a change in the value of A_j from time y_k to y_{k+1}, then CAPE-

Forecast will return the value and probability forecast by SitCAST+CONVEX. CAPE-Forecast, then, is able to produce a comprehensive forecast for each action attribute at a given time point, combining knowledge about behavioral changes with the standard operating procedures of agent g.

5.3 Implementation and Experiments

The CAPE architecture was implemented and applied to study the behavior of seven terror groups in the Asia Pacific region using a specialized subset of the MAROB data [46, 125]. In this data, the contexts and behaviors of these organizations were tracked semi-annually from 1995 to 2006 (i.e., there are 24 time periods in T_g). For these groups, one action was considered: the propensity to engage in rebellion against the state. The goal was to predict whether the group would engage in rebellion at all during a particular time period, as well as the intensity of that rebellious behavior. A time lag of 1 time period was used when creating the change table and generating the change rules in CAPE-Learn.

Two experimental evaluations were conducted. In the first experiment, the data about rebellion by these seven groups indicated only if they were engaged in rebellion or not. In this case, CAPE-Forecast accurately predicted the rebellion status of these groups with 80 % accuracy. Looking specifically at the predictions of a change in rebellion status (i.e., instances of going from no rebellion to rebellion or vice versa), 83.3% of the change forecasts were correct.

In the second experiment, the intensity of rebellion was measured for each time period in T_g on a scale from 0 (no rebellion) to 7 (full-fledged civil war). Sample change rules generated by CAPE-Learn for a group in the Philippines are given in

$$\begin{array}{lllll}
1. & \mathit{ORGREB}(1,0):[1.0] & \leftarrow & \mathit{stateRepression}(3,4). & \\
2. & \mathit{ORGREB}(1,0):[1.0] & \leftarrow & \mathit{representingInterests}(0,1)\ \wedge & \\
 & & & \mathit{terrorAttacks}(1,0). & \\
3. & \mathit{ORGREB}(0,1):[1.0] & \leftarrow & \mathit{representingInterests}(1,0)\ \wedge & \\
 & & & \mathit{terrorAttacks}(0,1). &
\end{array}$$

Figure 5.5: Sample rules indicating conditions when a group will change its rebellious behavior.

Figure 5.5. For example, rules 2 and 3 indicate a relationship between changes in rebellion and changes in the strategy of the group. Specifically, changes in whether they favor representing their interests to officials or terror attacks can be leading indicators of changes in the level of rebellion. When representing their interests becomes a minor strategy, and terrorism is no longer a strategy, the incidence of rebellion declines in the following time period from political banditry (1) to no rebellion (0), and vice versa.

CAPE-Forecast accurately predicted the exact intensity of rebellion 69.32 % of the time; this includes both the "normal" behavioral predictions of the group (i.e., cases where no change rules were applicable and SitCAST+CONVEX was used) as well as the change predictions. Looking only at the cases when a change was predicted, 61.5% of these forecasts correctly predicted the incidence of behavioral changes. Allowing for a margin of error of ±1 regarding the predicted intensity, the change forecast accuracy increases to 69.2%. The results from both of these experiments are shown in Table 5.1.

What these two experiments suggest is that CAPE-Forecast does a very good job of making predictions, and that these predictions are significantly more accurate than an arbitrary guess would be (simple guessing would give 50 % accuracy in the first experiment above, and 12.5 % accuracy in the second experiment). However,

Experiment	Overall Forecast Accuracy	Change Forecast Accuracy
Occurrence	80	83.3
Intensity	69.32	61.5
Intensity ± 1		69.2

Table 5.1: Results from experiments with the CAPE-Forecast algorithm. Here the % accuracy is reported for the overall forecast and change forecast for predicting the occurrence of rebellion, the intensity of rebellion, and the intensity ± 1.

this also suggests that CAPE's accuracy decreases when highly fine-grained forecasts are required. Further experiments are needed in order to assess CAPE's accuracy for various forecast granularities. Moreover, the above experiments only apply to situations one time period ahead—how CAPE's accuracy changes with longer look-aheads also needs to be further studied.

5.4 Conclusions

There are numerous applications where it is necessary to predict not only the ordinary behavior of an agent, but also when that agent will change its behavior. For example, financial organizations are interested in tracking the behavior of major investors, political parties are interested in tracking the actions of various special interest groups, and governments are interested in the behaviors of foreign political and military entities.

This chapter has discussed two important contributions. First, the CAPE-Learn algorithm was developed to learn conditions under which an agent changes a certain behavior of interest. Second, the CAPE-Forecast algorithm was developed to use the rules and conditions generated by CAPE-Learn to forecast what a group will do in future time periods, including when they will exhibit a behavioral shift. The CAPE architecture was experimentally evaluated on one action (the propensity to engage in rebellion) for seven terrorist groups in the Asia Pacific region, and results indicate that CAPE can provide high predictive power, at least in these limited experiments.

Much work remains to be done. Despite the promising results above, CAPE needs to be tested on a much wider range of groups and a much wider set of action attributes. Experiments are also needed to study the accuracy of CAPE in forecasting behavioral change multiple time periods into the future or with different time lags relating context and action changes. In addition, further experiments are necessary to study the accuracy of CAPE where more fine grained, pinpoint forecasts are required. Finally, the CAPE system can be extended in several ways that might make forecasting behavioral changes more accurate. For example, it may be possible to profile clusters of several agents that exhibit similar patterns of behavioral change and pool their change rules to have a potentially more complete model.

The ability to accurately forecast when an agent will change its behavior is a crucial component of developing effective policies and managing contingencies, and is a requirement for any decision-support system. Technologies that might leverage the knowledge provided by CAPE for policy analysis and user applications are described in Chapters 7 and 8. In the following chapter, the many approaches to forecasting in a behavioral time series that have been presented so far are generalized into an overarching theory of forecasting in relational databases.

Chapter 6

Axiomatic Forecast Oriented Reasoning

In the previous chapters, several methods have been described for forecasting and reasoning about the behaviors of an agent in a specific situation or at a given time point. This chapter will integrate these varied approaches into a generalized theory of forecasting in temporal databases using an axiomatic approach for incorporating forecasts as a database operator. This framework encompasses the forecasting algorithms for behavioral time series data—SOMA (Chapter 2), CONVEX (Chapter 3), SitCAST (Chapter 4), and CAPE (Chapter 5)—that have been discussed so far, as well as any other statistical and computational methods that can be applied to temporal databases.

Though temporal logics and temporal reasoning have been studied extensively over the years in AI, there has been little work that merges temporal logic with classical forecasting and time series analysis methods [18]. Clearly, there are numerous applications that require such capabilities. A university might want to forecast research grant income (or expenditures) in the future by examining a database of research projects. A stock market firm might want to include support for various kinds of specialized forecasting algorithms that predict the values of mutual fund portfolios or a single stock over time. A government might want to forecast the number of electricity connections or other development indicators in their country over time. Such forecasts might not just be made about the future, but also used to fill in gaps about the past. For instance, using data about the number of electricity connections in Ecuador from 1990–2000 and 2002–2007, officials may want to interpolate the number of connections there might have been in 2001.

The field of forecasting is extensive and widely studied, with an array of general techniques [18] as well as specialized forecast models for a variety of domains, such as finance [118], epidemiology [57], politics [97, 75, 16, 114], and even product liability claims [111]. All these methods are dramatically different from one another, and even within a restricted domain such as the stock market, there are hundreds of forecasting models available, each with varying strengths and weaknesses.

In this chapter, the question "*what should count as a forecast operator?*" is answered by first providing a set of axioms that such an operator must satisfy. Without loss of generality, assume that forecast operators apply to temporal relational databases—the main reason for this assumption is that in today's world, most (though certainly not all) temporal data is in fact stored in such databases (recall the definition of a behavioral time series in Section 3.1 and the Minorities at Risk Organizational Behavior dataset used in Chapters 2, 3, 4, and 5). Subsequently, three classes of forecast operators—deterministic forecast (DF) operators, probabilistic forecast (PF) operators, and possible worlds forecast (PWF) operators are defined. All DF operators are proved to be a special case of PF operators, which are in turn a special case of PWF operators. Certain classical forecasting methods such as linear regression, polynomial regression, and logistic regression methods are all demonstrated to be special cases of this framework. Some new operators for forecasting are also developed, along with results characterizing the complexity of applying certain forecast operators.

The remainder of this chapter will proceed as follows. Section 6.1 contains two motivating examples—one about forecasting academic grant incomes, and another about electricity connections in developing countries based on real data from the World Bank. Section 6.2 introduces basic notation for temporal relational databases. Section 6.3 provides an axiomatic definition of a forecast operator and then defines the classes of DF, PF, and PWF forecast operators. This section also develops theorems showing relationships between DF, PF, and PWF operators and the complexity of specific types of operator constructions.

6.1 Motivating Examples

Two motivating examples are used throughout this chapter. The *grants* example specifies the total dollar amount ("Amount") of grants and number of employees ("Employees") of a Math and a CS department. Here, we are interested in predicting both of these attributes. The *electricity* example is drawn from real World Bank [1] data about the total expenditures ("Expenditures") on electricity and the number of electricity connections ("Connections") in some Latin American countries. Here, we wish to forecast the number of electricity connections and the amount of total expenditures (which includes operating costs and capital investment).

[1]Benchmarking Data of the Electricity Distribution Sector in the Latin America and Caribbean Region 1995-2005. Available at: http://info.worldbank.org/etools/lacelectricity/home.htm

	Year	**Department**	**Amount**	**Employees**
t_1	2000	CS	6M	70
t_2	2001	CS	6.2M	70
t_3	2002	CS	7M	75
t_4	2003	CS	6M	75
t_5	2004	CS	7.3M	74
t_6	2005	CS	9M	80
t_7	2000	Math	1M	71
t_8	2001	Math	1.1M	74
t_9	2002	Math	1M	73
t_{10}	2003	Math	0.5M	66
t_{11}	2004	Math	1.5M	79
t_{12}	2006	Math	1.2M	77

Table 6.1: The *grants* relation

	Year	**Country**	**Connections**	**Expenditures**
e_1	2000	Brazil	48,000,000	6.8B
e_2	2001	Brazil	50,200,000	7.5B
e_3	2002	Brazil	52,200,000	6.9B
e_4	2003	Brazil	53,800,000	6.3B
e_5	2004	Brazil	56,300,000	7.7B
e_6	2005	Brazil	57,900,000	10.7B
e_7	2000	Venezuela	4,708,215	7.7B
e_8	2001	Venezuela	4,877,084	5.2B
e_9	2002	Venezuela	4,998,433	4.3B
e_{10}	2003	Venezuela	5,106,783	3.3B
e_{11}	2004	Venezuela	5,197,020	3.1B
e_{12}	2005	Venezuela	5,392,500	3B

Table 6.2: The *electricity* relation

6.2 Basic Notation

The proposed forecasting framework discussed here applies only to temporal databases; therefore, some basic temporal DB notation is introduced in this section. Such temporal databases are generalizations of the behavioral time series that were introduced in Section 3.1. Let us assume the existence of a finite set **rel** of relation names, and a finite (disjoint from **rel**) set **att** of attribute names. A *temporal relation schema* for a relation $S \in \mathbf{rel}$ is an n-tuple $(A_1, \ldots, A_{n-1}, A_T)$ where $A_1, \ldots, A_{n-1} \in \mathbf{att}$, and will be denoted as $S(A_1, \ldots, A_{n-1}, A_T)$. Each attribute $A \in \mathbf{att}$ is typed and has a domain $dom(A)$. Assume the existence of a special attribute A_T denoting time whose domain $dom(A_T)$ is the set of all integers (positive and negative). Also assume that each attribute in S is either a *variable* or *invariant* attribute. Invariant attributes do not change with time, while variable attributes might. In the *grants* relation, "Department" is an invariant attribute, while "Amount" and "Employees" are variable attributes. In the *electricity* example, "Country" is invariant, while "Connections" and "Expenditures" are variable.

A *temporal tuple* over $S(A_1, \ldots, A_{n-1}, A_T)$ is a member of $dom(A_1) \times \cdots \times dom(A_{n-1}) \times dom(A_T)$. A *temporal relation instance* R over the relation schema S is a set of tuples over S.

Abusing notation a bit, $S(A_1, \ldots, A_n)$ *will be used instead of* $S(A_1, \ldots, A_{n-1}, A_T)$, *simply assuming that the last attribute in any schema is the time attribute.*

Given a tuple t over $S(A_1, \ldots, A_n)$, $t(A_i)$ (where $i \in [1..n]$) denotes the value of attribute A_i in tuple t. $Attr(S)$ denotes the set of all attributes in S. Given a relation schema S, schema S_e is an *extension* of schema S, denoted $S_e \supseteq S$ iff $Attr(S_e) \supseteq Attr(S)$.

Definition 28 (Equivalence of tuples)**.** *Given a temporal relation instance R over a temporal relation schema S and a set of attributes $\mathcal{A} \subseteq Attr(S)$, $t_1 \sim_{\mathcal{A}} t_2$ iff for each $A_i \in \mathcal{A}$, $t_1(A_i) = t_2(A_i)$. It is easy to see that $\sim_{\mathcal{A}}$ is an equivalence relation—a* cluster *cl is defined for temporal relation R w.r.t. the set of attributes $\mathcal{A}$ to be any equivalence class under $\sim_{\mathcal{A}}$, and $clusters(R, \mathcal{A})$ denotes the set of clusters of R w.r.t. $\mathcal{A}$.*

The following example shows clusters associated with the *grants* and *electricity* examples.

Example 11. *Consider the* grants *relation and suppose $\mathcal{A} = \{Department\}$. Then $clusters(grants, \{Department\})$ contains two clusters $\{t_1, \ldots, t_6\}$ and $\{t_7, \ldots, t_{12}\}$.*

On the other hand, if the electricity *relation and the invariant set $\mathcal{A} = \{Country\}$ are considered, there are again two clusters ($\{e_1, \ldots, e_6\}$ and $\{e_7, \ldots, e_{12}\}$) in $clusters($ $electricity, \{Country\})$.*

6.3 Forecast Operator

In this section, a generic *forecast operator* for any temporal DB is formally defined, and various families of forecast operators are identified. Intuitively, a forecast operator must take as input some historical information and a time period for which to produce a forecast, which might include the future as well as past times where data is missing. The output of a forecast operator, however, can vary dramatically in form. For instance, forecasts can contain a single unambiguous prediction (called deterministic forecasts), or a single probabilistic forecast expressing some uncertainty about its correctness (called a probabilistic forecast), or a set of possible situations (called a possible worlds forecast). For each of these "types" of forecasts, the content can vary widely as well. The following definition accounts for all of these classes of forecasts, but requires that they satisfy specific desired properties.

Definition 29 (Forecast Operator)**.** *Given a temporal relation instance R over the schema S and a temporal interval I defined over $dom(A_T)$, a* forecast operator *ϕ is a mapping from R and I to a set of relation instances $\{R_1, \ldots, R_n\}$ over a schema $S_e \supseteq S$ satisfying the following axioms.*

Axiom A1. *Every tuple in each R_i ($i \in [1..n]$) has a timestamp in I.* This axiom says that the forecast operator only makes predictions for the time interval I.

Axiom A2. *For each relation R_i ($i \in [1..n]$) and for each tuple $t \in R$ such that $t(A_T) \in I$, there is exactly one tuple $t_i \in R_i$ such that $\forall A \in Attr(S)$, $t(A) = t_i(A)$.* This axiom says that tuples of R having a timestamp in I are preserved by the forecast operator (though they can be extended to include new attributes in schema S_e).

Axiom A3. *For each timestamp $ts \in I$ and tuple $t \in R$, there is at least one relation R_i with $i \in [1..n]$ containing the (forecasted) tuple t' such that $t'(A_T) = ts$ and $t' \sim_{\mathcal{A}} t$ where $\mathcal{A} \subseteq Attr(S)$ is a set of invariant attributes.* This axiom says that the forecasting is complete with respect to the timestamps in I and original tuples in R.

Note that axioms (A1) to (A3) above are not meant to be exhaustive. They represent a minimal set of conditions that any forecast operator should satisfy. Specific forecast operators may satisfy additional properties. In addition, to reiterate, the temporal interval I in the above definition can represent both the future and the past, i.e., it can include times that follow and/or precede those in relation R.

Forecast operators may satisfy the following additional properties; however, they are not mandatory for definition as an operator.

Definition 30 (Coherence)**.** *Suppose R is a temporal relation instance over a temporal relation schema S, I is a temporal interval, and $\mathcal{A}$ a set of invariant attributes. A forecast operator ϕ is* coherent *w.r.t. $\mathcal{A}$ iff for each $R_i \in \phi(R, I) = \{R_1, R_2, \ldots, R_n\}$, there is a bijection $\beta_i : clusters(R, \mathcal{A}) \to clusters(R_i, \mathcal{A})$ s.t. for each $cl \in clusters(R, \mathcal{A})$, it is the case that $\phi(cl, I) = \{\beta_1(cl), \beta_2(cl), \cdots, \beta_n(cl)\}$.*

Basically, a forecast operator ϕ is coherent w.r.t. a set of attributes $\mathcal{A}$ if the result of applying ϕ on the whole relation R is equivalent to the union of the results obtained by applying ϕ on every single cluster in $clusters(R, \mathcal{A})$. For instance, consider the *electricity* example and $\mathcal{A} = \{Country\}$. In this case, a coherent forecast operator says that the number of electricity connections and the amount of expenditures in a country only depends on that country. Likewise, in the *grants* example with $\mathcal{A} = \{Department\}$, using a coherent forecast operator implies that the amount of grants and number of employees only depends upon the department. Forecast operators are not required to be coherent because this property may not always be valid in all applications. For instance, there may be a correlation between grant amounts in the CS and Math departments (e.g., decreases in NSF funding may affect both of them proportionately). As a consequence, if the *grants* relation had an additional tuple t_{13} with information on the 2007 grant income of Math, then this may be relevant for a forecast about CS's grant income in 2007, but the coherence assumption would not allow this dependency. As such, coherence is not considered a basic forecast axiom.

Another property that forecast operators may satisfy (but are not required to) is *monotonicity*. Given a relation R, two disjoint sets $\mathcal{A}, \mathcal{B}$ of attributes, and two clusters $cl_1, cl_2 \in clusters(R, \mathcal{A})$, $cl_1 <_{\mathcal{B}} cl_2$ iff $\forall\, t_1 \in cl_1, t_2 \in cl_2, B \in \mathcal{B}$ it is the case that $t_1(B) \leq t_2(B)$. This ordering is used to define monotonicity for forecast operators.

Definition 31 (Monotonicity)**.** *Let R be a temporal relation instance over a schema S, I a temporal interval, and $\mathcal{A}, \mathcal{B} \subseteq Attr(S) \setminus A_T$ two disjoint sets of attributes. A forecast operator ϕ is* monotonic *w.r.t. the pair $\langle \mathcal{A}, \mathcal{B} \rangle$ iff for each $R_i \in \phi(R, I)$, there is a bijection $\beta_i : clusters(R, \mathcal{A}) \rightarrow clusters(R_i, \mathcal{A})$ such that:*

(i) $\forall\, cl \in clusters(R, \mathcal{A})$, *$cl \sim_{\mathcal{A}} \beta_i(cl)$ (i.e., $\forall\, t_1 \in cl, t_2 \in \beta_i(cl), A \in \mathcal{A}$ it is the case that $t_1(A) = t_2(A)$); and*

(ii) $\forall\, cl_1, cl_2 \in clusters(R, \mathcal{A})$ *such that $cl_1 <_{\mathcal{B}} cl_2$, it is the case that $\beta_i(cl_1) <_{\mathcal{B}} \beta_i(cl_2)$.*

A forecast operator is monotonic if a monotonic trend in the values of attributes in $\mathcal{B}$ in the clusters w.r.t. $\mathcal{A}$ of the original relation R implies that this trend is preserved by the clusters w.r.t. $\mathcal{A}$ in the predicted relations $R_1, R_2, \ldots, R_n$.

6.3.1 Deterministic Forecast Operator

A deterministic forecast operator is one that returns a single relation with exactly the same schema as the input relation.

Definition 32 (Deterministic Forecast Operator)**.** *Given a temporal relation instance R over the schema S and a temporal interval I defined over $dom(A_T)$, a* deterministic forecast operator *(DF operator for short) δ is a forecast operator such that $\delta(R, I) = \{R'\}$ with R' also defined over S.*

DF operators can be built on top of any standard time series forecast algorithm. The following example shows how simple linear regression is an instance of the class of deterministic forecast operators.

Example 12. *Suppose the* electricity *example is being used to forecast the amount of connections and expenditures in 2006 and 2007 using simple linear regression.*[2] *The function $LINREG(R, I)$ applies linear regression to each variable attribute in relation R for time interval I. The result of the function $LINREG(electricity, [2006, 2007])$ is the relation* electricity′ *below:*

Year	***Country***	***Connections***	***Expenditures***
2006	*Brazil*	*60,006,666.67*	*9.6B*
2007	*Brazil*	*61,989,523.81*	*10.157B*
2006	*Venezuela*	*5,495,630.8*	*1.353B*
2007	*Venezuela*	*5,623,904.6*	*0.473B*

$LINREG(R, I)$ is an example of a DF operator, as it maps electricity *and a time interval I to the single relation* electricity′ $= LINREG(electricity, [2006, 2007])$. *In this example, $LINREG(R, I)$ also satisfies coherence w.r.t. the set $\mathcal{A} = \{Country\}$ and monotonicity w.r.t. the pair $\langle\{Country\}, \{Connections\}\rangle$.*

6.3.2 Probabilistic Forecast Operator

Deterministic forecasts are 100% certain about the resulting predictions. In contrast, probabilistic forecasts also include information about the probability that a forecast is correct.

[2]The same method shown in this example would allow us to use a variety of other traditional statistical forecasting methods, such as logistic regression, nonlinear regression, etc.

Definition 33 (Probabilistic Forecast Operator)**.** *Given a temporal relation instance R over the schema S and a temporal interval I defined over $dom(A_T)$, a* probabilistic forecast operator *(PF operator for short) μ is a forecast operator such that $\mu(R, I) = \{R'\}$ with R' defined over the schema $S' = Attr(S) \cup \{P\}$ where $dom(P) = [0, 1]$.*

PF operators are just like DF operators except that they have an additional probability attribute P. Each tuple returned by a PF operator includes the probability of that tuple being valid at the associated timestamp. In addition to the general axioms (A1)–(A3), it is often desirable for PF operators to satisfy a property called fact preservation.

Property 6.3.1 (Fact Preservation)**.** *Let R be a temporal relation instance over schema S and I a temporal interval. PF operator μ* is fact preserving *if for each tuple $t \in R$ such that $t(A_T) \in I$, there is a tuple $t' \in R'$ with $R' \in \mu(R, I)$ such that $\forall A \in Attr(S)$, $t(A) = t'(A)$ and $t'(P) = 1$.*

Axiom (A2) ensures that tuples having a timestamp in I are preserved by the forecast operator, i.e., for each tuple $t \in R$ such that $t(A_T) \in I$ there is a tuple $t' \in R'$ such that t and t' have the same values for the attributes in $Attr(S)$. The fact preservation property strengthens axiom (A2) for PF operators since it requires the additional condition that the probability values of the tuples in the resulting relation R' corresponding to those of R (preserved tuples) must be exactly 1.

The fact preservation property should be satisfied by a PF operator when the user trusts what is in the database; in other cases when the user does not trust the content of a database, he may choose to use a PF operator that does not guarantee fact preservation.

Example 13. *Consider the* grants *relation. Suppose a user want to forecast the amount of grants and employees for the CS and Math departments for 2006 and 2007, along with their probabilities. A polynomial regression method $P\_REG(R, \mathcal{A}, I)$ may be applied to variable attributes in each cluster in relation R w.r.t. $\mathcal{A}$ for a time interval I. $P\_REG(R, \mathcal{A}, I)$ is an operator that computes the probability that the actual value will be within one standard deviation of the forecasted value, based on a normal distribution. Assuming independence, the probability of the entire tuple is the product of the probabilities for the individual attributes.*

$P\_REG(R, \mathcal{A}, I)$ is an example of a PF *operator. It first computes the forecasted values for each cluster.*

Year	***Department***	***Amount***	***Employees***
2006	*CS*	*6.929471566*	*74*
2007	*CS*	*6.932925939*	*74*
2006	*Math*	*1.051905341*	*73*
2007	*Math*	*1.052429721*	*74*

The probability of each forecasted value is computed as mentioned above using standard statistics.

<u>CS</u>

$P(Amount = 6.929471566 \pm \sigma | Year = 2006) = 0.68266$

$P(Amount = 6.932925939 \pm \sigma | Year = 2007) = 0.68264$

$P(Employees = 74 \pm \sigma | Year = 2006) = 0.68268$

$P(Employees = 74 \pm \sigma | Year = 2007) = 0.68268$

<u>Math</u>

$P(Amount = 1.051905341 \pm \sigma | Year = 2006) = 0.68268$

$P(Amount = 1.052429721 \pm \sigma | Year = 2007) = 0.68267$

$P(Employees = 73 \pm \sigma | Year = 2006) = 0.68141$

$P(Employees = 74 \pm \sigma | Year = 2007) = 0.6776$

The final relation, $grants'$ is shown below:

Year	***Dept.***	***Amount***	***Employees***	***Prob***
2006	*CS*	*6.929471566*	*74*	*0.46604*
2007	*CS*	*6.932925939*	*74*	*0.46603*
2006	*Math*	*1.051905341*	*73*	*0.46519*
2007	*Math*	*1.052429721*	*74*	*0.46258*

It is clear that every deterministic forecast can be expressed as a probabilistic forecast. Given a DF δ, a temporal relation instance R over schema S, and a time period I, a simple probabilistic forecast operator $\mu^{simp,\delta}(R, I)$ can be defined to return $\{(t, 1) \mid t \in R'\}$ where $\delta(R, I) = \{R'\}$. The following theorem describes the relationships between δ and $\mu^{simp,\delta}(R, I)$.

Theorem 1. *Suppose δ is a DF operator. Then, the following relationships are true:*

(i) *$\mu^{simp,\delta}$ is a probabilistic forecast operator.*

(ii) *If δ is coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. pair $\langle \mathcal{A}, \mathcal{B} \rangle$), then $\mu^{simp,\delta}$ is coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. pair $\langle \mathcal{A}, \mathcal{B} \rangle$).*

(iii) *$\mu^{simp,\delta}$ is fact preserving.*

6.3.3 Possible Worlds Forecast Operator

Probabilistic forecasts still only give one value for each attribute per time period in the prediction interval. However, in general, there may be many possible instances of relation R at a future (or past) time point. Possible worlds forecasts try to return not one instance as the output of a forecast, but a set of relations, each of which is a possible instance of the relation at the time being forecast.

Definition 34 (Possible Worlds Forecast Operator)**.** *Given a temporal relation instance R over the schema S and a temporal interval I defined over $dom(A_T)$, a* possible worlds forecast operator *(PWF operator for short) ω is a forecast operator such that $\omega(R, I) = \{R_1, \ldots, R_n\}$ where each R_i is defined over S and has an associated probability value $\mathbf{P}(R_i)$ such that $\sum_{i=1}^{n} \mathbf{P}(R_i) = 1$.*

Basically, every resulting relation instance R_i represents a possible forecasted world. Observe that, axiom (A2) entails that every possible world includes the tuples representing the facts belonging to temporal interval I that were assumed to be true in the original relation R.

Given any deterministic forecast operator δ, a PWF operator ω^{δ} can be defined in many ways. One such method called the *discretized PWF w.r.t. δ*, denoted $\omega^{disc,\delta}$ is given below. Suppose R is a temporal relation instance over schema S and suppose I is a temporal interval; then, $\omega^{disc,\delta}$ is defined as:

1. Let R' be the relation returned by $\delta(R, I)$. Consider each tuple $t \in R'$. For each variable attribute $A \in Attr(S)$, define $\mathbf{P}(\lfloor t(A) \rfloor) = \lceil t(A) \rceil - t(A)$ and $\mathbf{P}(\lceil t(A) \rceil) = 1 - \mathbf{P}(\lfloor t(A) \rfloor)$. The set of *tuple worlds* $tw(t)$ associated with any tuple $t \in R'$ is now defined to be $\{t' \mid$ for all variable attributes $A \in Attr(S)$, $t'(A) = \lfloor t(A) \rfloor$ or $t'(A) = \lceil t(A) \rceil\}$ and for all invariant attributes $B \in Attr(S)$, $t(B) = t'(B)\}$. Each $t' \in tw(t)$ is called a tuple world.

2. The probability of a tuple $t' \in tw(t)$ is defined to be the product of the probabilities of all the variable attribute elements of t', i.e., if $X \subseteq S$ is the set of all variable attributes in the schema of R, then $\mathbf{P}(t') = \Pi_{A \in X} \mathbf{P}(t'(A))$.

3. The set of *relation worlds* $rw(\delta, R, I)$ is now defined to be the Cartesian product of all tuple worlds, i.e., $\Pi_{t \in \delta(R,I)} tw(t)$. Each member of $rw(\delta, R, I)$ is called a *relation world.* The probability of a given relation world $w \in rw(\delta, R, I)$ is given by $\mathbf{P}(w) = \Pi_{t' \in w} \mathbf{P}(t')$.[3]

4. Return $rw(\delta, R, I)$ and the probability distribution $\mathbf{P}$ on $rw(\delta, R, I)$.

The result below states that this construction is correct.

Theorem 2. *Suppose δ is any deterministic forecast operator. Then, the following relationships are true:*

(i) *$\omega^{disc,\delta}$ is a PWF operator.*

(ii) *If δ is coherent w.r.t. the set of attributes $\mathcal{A}$, then $\omega^{disc,\delta}$ is coherent w.r.t. $\mathcal{A}$.*

A simple example of this construction is provided below.

Example 14. *Let us return to the* electricity *relation and consider using the simple linear regression $LINREG(electricity, [2006, 2006])$ for just the one year 2006. The result of this operator follows immediately from Example 12.*

Year	***Country***	***Connections***	***Expenditures***
2006	*Brazil*	*60,006,666.67*	*9.6B*
2006	*Venezuela*	*5,495,630.8*	*1.353B*

The construction procedure above creates 16 possible relation worlds. The total number of connections in Brazil in 2006 could be 60,006,666 (33%) or 60,006,667 (67%), and the corresponding number in Venezuela could be 5,495,630 (20%) or 5,495,631 (80%). The possible expenditures in Brazil are 9B (40%) or 10B (60%),

[3]This assumes that the events represented by different tuples in $\delta(R, I)$ are independent of one another.

and in Venezuela are 1B (64.7 %) or 2B (35.3 %). The probability of each world is the product of the probabilities of the tuples selected for that world. As an example, consider the world w given below:

Year	***Country***	***Connections***	***Expenditures***
2006	*Brazil*	*60,006,666*	*10B*
2006	*Venezuela*	*5,495,631*	*1B*

$\mathbf{P}(w) = (0.33 * 0.6) * (0.8 * 0.647) = 0.102.$

It is worth noting that, as both DF and PWF operators satisfy axiom (A2), the tuples of the original relation belonging to the predicted temporal interval are preserved by DF operator δ, and then preserved by PWF operator $\omega^{disc,\delta}$ as well.

The following example shows that $\omega^{disc,\delta}$ does not preserve monotonicity.

Example 15. *Assume that for countries C_1 and C_2, electricity connections are almost the same in a given year, differing only in their decimal number, as shown in the following relation el:*

Year	***Country***	***Connections***
2005	C_1	*50,900,800.4*
2005	C_2	*50,900,800.8*

Suppose the result of $\delta(el, [2008, 2008])$ is the relation given below:

Year	***Country***	***Connections***
2008	C_1	*50,900,802.3*
2008	C_2	*50,900,802.9*

Clearly, δ is monotonic w.r.t. the pair $\langle\{Country\}, \{Total\ Connections\}\rangle$. In contrast, $\omega^{disc,\delta}$ is not monotonic w.r.t. $\langle\{Country\}, \{Total\ \ Connections\}\rangle$, since there *is relation world* $rw = \{(2008, C_1, 50, 900, 803), (2008, C_2, 50, 900, 802)\}$ *in* $rw(\delta, el, [2008, 2008])$ *for which the number of electricity connections of* C_2 *is not greater than that of* C_1.

The $\omega^{disc,\delta}$ construction takes exponential time to enumerate the possible relation worlds and compute the associated probability distribution; the number of tuple worlds $tw(t)$ for a tuple t is exponential in the number of variable attributes, and the total number of relation worlds is exponential in the number of tuple worlds.

Theorem 3. *Suppose* R *is a temporal relation instance over schema* S*,* I *is a temporal interval, and* $\mathcal{A} \subset Attr(S)$ *is a set of variable attributes. For any deterministic forecast operator* δ*, the running time of* $\omega^{disc,\delta}$ *is* $O((2^{|\mathcal{A}|})^{|R'|})$*, where* R' *is the relation returned by* $\delta(R, I)$.

From the possible relation worlds produced by $\omega^{disc,\delta}$, a user may only be interested in examining those forecasted relations that are sufficiently probable and contain a given tuple.

Proposition 4. *Suppose* R *is a temporal relation instance over schema* S*,* I *is a temporal interval, and* δ *is a polynomial-time computable* DF *operator. Given a tuple* t *over the schema* S *and probability threshold* k*, deciding whether there is a relation world* $w \in rw(\delta, R, I)$ *such that* $t \in w$ *and* $P(w) \geq k$ *(or* $P(w) \leq k$*) is in* $PTIME$.

Proof Sketch. Let R' be the relation returned by $\delta(R, I)$. First check if there is tuple $t' \in R'$ such that by rounding its value, for each variable attribute A, we obtain t. If no, the answer to our decision problem is "no." Otherwise, keep this tuple t' and find a relation world w_{max} with max probability, i.e., $\forall t'' \in R'$, $t'' \neq t'$ create a maximal tuple world by setting $t''(A) = argmaxP(t''(A))$ for all variable attributes A. If $P(w_{max}) \geq k$, then the answer is "yes." □

A PF operator μ can also be converted to a PWF operator in several different ways. Some possible mechanisms are provided below where R is a temporal relation instance over schema S and I is a temporal interval over $dom(A_T)$.

(i) $\omega^{simp,\mu}(R, I)$ returns just one world as follows. Suppose $\mu(R, I) = \{R'\}$. Then $\omega^{simp,\mu}(R, I) = \{\pi_{Attr(S)}(R')\}$. In other words, it eliminates the probability column in R'. This one world has probability 1 according to the PWF $\omega^{simp,\mu}$.

(ii) $\omega^{ind,\mu}(R, I)$ operates as follows:

1. Compute $\mu(R, I) = \{R'\}$ as above.
2. Return the power set of $\pi_{Attr(S)}(R')$ as the set of worlds.
3. For each tuple $t \in R_i$ in the power set of $\pi_{Attr(S)}(R')$, let $\mathbf{P}(t)$ be the probability attribute of the tuple in R' whose non-probability attributes are identical to those of t. The probability of a particular subset $X \subseteq R'$ is set to $\Pi_{t \in X}\mathbf{P}(t) \times \Pi_{t' \in \pi_{Attr(S)}(R') - X}(1 - \mathbf{P}(t'))$.
4. Return the power set of $\pi_{Attr(S)}(R')$ together with the above probability distribution on this set.

The following theorem shows a strong relationship between a PF operator μ and the PWF operator $\omega^{simp,\mu}$.

Theorem 4. *Suppose μ is any PF operator. Then the following relationships are true:*

(i) *$\omega^{simp,\mu}$ is a PWF operator.*

(ii) *If μ is coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. $\langle \mathcal{A}, \mathcal{B} \rangle$), then $\omega^{simp,\mu}$ is also coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. $\langle \mathcal{A}, \mathcal{B} \rangle$).*

The above theorem holds irrespective of whether the PF operator μ is fact preserving or not. In contrast, $\omega^{ind,\mu}$ will be a PWF operator only if constructed using a fact preserving PF operator. To see this, consider a relation R containing tuple t such that its timestamp $t(A_T)$ belongs to the temporal interval I. If PF operator $\mu(R, I)$ forecasts t' whose invariant attributes are identical to those of t and its probability value is $\mathbf{P}(t') = 0.5$, then there is a possible world returned by $\omega^{ind,\mu}$ that does not contain any tuple having invariant attributes identical to those of t. Hence, A2 would be violated.

Theorem 5. *Suppose μ is any fact preserving PF operator. Then the following relationships are true:*

(i) *$\omega^{ind,\mu}$ is a PWF operator.*

(ii) *If μ is coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. $\langle \mathcal{A}, \mathcal{B} \rangle$), then $\omega^{ind,\mu}$ is also coherent w.r.t. $\mathcal{A}$ (resp. monotonic w.r.t. $\langle \mathcal{A}, \mathcal{B} \rangle$).*

The construction $\omega^{ind,\mu}$ for a PF μ is exponential in the size of the relation returned by $\mu(R, I)$.

Theorem 6. *Suppose R is a temporal relation instance over schema S and I is a temporal interval. For any probabilistic operator μ, the running time complexity of $\omega^{ind,\mu}$ is $O(2^{|R'|})$, where R' is the relation returned by $\mu(R, I)$.*

The following proposition characterizes the complexity of determining whether there is a possible world returned by $\omega^{ind,\mu}$ such that it is sufficiently probable and contains a tuple of interest t.

Proposition 5. *Suppose R is a temporal relation instance over schema S, I is a temporal interval, and μ is a polynomial-time computable PF operator. Given a tuple*

t over the schema S and a probability threshold k, deciding whether there is a world w returned by $\omega^{ind,\mu}$ such that $t \in w$ and $P(w) \geq k$ (or $P(w) \leq k$) is in $PTIME$.

Proof Sketch. First check if $t \in \pi_S(R')$, where R' is the relation returned by $\mu(R, I)$. If $t \notin \pi_S(R')$, then it cannot belong to $\omega^{ind,\mu}$, thus the answer is 'no'. If $t \in \pi_S(R')$, then there is at least one possible world w that contains t. The possible world w_{max} (resp. w_{min}) that contains t is constructed using a strategy similar to that in the proof of Proposition 4. Finally, verify whether $P(w_{max}) \geq k$ (or $P(w_{min}) \leq k$). $\square$

6.4 Conclusions

Though there are numerous works on forecasting in general [18], as well as specialized forecast models for specific domains, such as finance [118], epidemiology [57], politics [97, 16, 114], or behavioral analysis from Chapters 3, 4, and 5, all these methods vary dramatically from one another. However, in this chapter, general axioms were provided that any forecast operator, regardless of domain, should satisfy, together with additional desirable (but not required) properties. Forecast operators, including classical forecasting algorithms and the forecasting methods described in previous chapters, can be classified into three increasingly expressive categories—(i) deterministic forecast operators, (ii) probabilistic forecast operators, and (iii) possible worlds forecast operators—and then embedded as operators in a temporal database. These classes of operators all satisfy the forecasting axioms, and in some cases, the additional desirable properties that were identified. Though forecasting is often complex, many of the techniques reported in this chapter were proven to be polynomially computable.

The framework described in this chapter provides a generic classification scheme for forecasting in temporal databases, but it does not explicitly indicate how such operators should be incorporated into query-answering mechanisms. In order to better understand general forecast operators, their relationship to the standard relational algebra should be explored, as well as policies for combining forecast and standard operators to answer forecast queries. In the next chapter, for example, probabilistic forecasts on event knowledgebases are utilized as part of a framework for developing response policies for agent behavior.

Chapter 7

Optimal Policies from Actionable State Change Attempts

The preceding chapters all discuss methods for forecasting and understanding the behavior of another agent in a particular situation at a particular time. Now, the issue of what can be done with this knowledge to elicit desired behavioral responses from another agent, given data concerning its past behavior, is explored. That is, once the types of actions an agent will take are understood, how can forecasts be generated of what the best policy response or counterstrategy is for the given situation, balancing the likelihood of success against the expected cost? In this chapter a complete formalism—Actionable State Change Attempts—and reasoning algorithms are presented for identifying the optimal policy for interacting with the environment to influence the actions of other agents.

Before the issue of finding the optimal counterstrategy can be addressed, the datasets used to solve this problem must first be understood. A large number of well-known datasets in the social sciences have a tabular form where each row refers to a period of time, and each column represents a variable that characterizes the state of some entity during a time period. These variables naturally divide into those actionable variables that can be controlled ("action variables") and those that cannot ("state variables"). These "state" variables notably include the behaviors of other agents or outcome conditions a user may be trying to influence. For example, data sets regarding school performance for various U.S. states contain "state variables" such as the graduation rate of students in the state and the student to staff ratio during some time frame, while the "action" variables might refer to the level of funding provided per student during that time period, the faculty salary levels, etc. Clearly, a U.S. state can attempt to change the levels of funding per student and/or change the faculty salaries in an attempt to increase the graduation rate, but cannot change the graduation rate directly.

In a completely different setting, political science datasets about the stability of a country (such as the data sets created by the well known Political Instability Task Force [36]) may have "state variables" such as the GDP of a country, the infant mortality rate, and the number of people killed in political conflict in the country during a time period, while "action" variables might include information about the investment in hospitals or education during that time frame, the number of social workers available, and so forth. A government might want to see what actionable policies it can try to implement to achieve a certain goal (e.g., bringing the infant mortality rate below some threshold).

These are just two examples of problems that are not easily solved using current algorithms for reasoning about actions in AI or by AI planning systems. The main reasons are the following: (i) the relationships between the actions and their impact on the state are poorly understood, (ii) a set of actions, taken together, might have a cumulative effect on a state that might somehow be more than a naive combination of the effects of those actions individually, which of course are not known anyway, and (iii) the actions under consideration may not succeed—an attempt to raise hospital funding may be blocked for reasons outside of anyone's control.

In this chapter, the notion of an *event KB* is first proposed in Section 7.1. This description is not novel—indeed it is related to the behavioral time series databases in Chapters 3, 4, and 5—but generalizes several social science datasets, such as those mentioned above. Sections 7.2 and 7.3 present the Actionable State Change Attempts framework and define the concept of "state change attempts" (SCAs for short), formulating various problems related to finding "optimal" (in a sense that will be made precise) SCAs towards a given goal. A host of results on the computational complexity of finding optimal SCAs are also described. In Section 7.4, a straightforward algorithm, called DSEE_OSCA, is first presented to compute optimal SCAs. A vastly improved algorithm, called TOSCA, is then developed in Section 7.6 based on using a trie-structured index on an event KB. Though tries are a well known data structure, the novelty of this work is rooted in how TOSCA uses tries to solve optimal SCA problems with lower computational complexity. Finally, in Section 7.7, an implementation of both algorithms is briefly described, together with an experimental analysis that uses both synthetic and real-world education data to demonstrate that DSEE_OSCA and TOSCA are fast and effective.

7.1 Preliminaries on Event KBs

An event KB is a relational database whose rows correspond to some time period (explicit or implicit) and whose columns are of two types—*state attributes* and *action attributes*. *Throughout this chapter, assume the existence of some arbitrary, but fixed set* $\mathbf{A} = \{A_1, \ldots, A_n\}$ *of action attributes, and another arbitrary,* *but fixed set* $\mathbf{S} = \{S_1, \ldots, S_m\}$ *of state attributes.* As usual, each attribute (state or action) V has a domain $dom(V)$, which is assumed to be finite. In addition, all attributes V are assumed to have domain $dom(V) \subset \mathbb{R}$. $\mathcal{A}$ represents the set $dom(A_1) \times \cdots \times dom(A_n)$ and $\mathcal{S}$ represents the set $dom(S_1) \times \cdots \times dom(S_m)$. A *tuple* w.r.t. $(\mathbf{A}, \mathbf{S})$ is any member of $dom(A_1) \times \cdots \times dom(A_n) \times dom(S_1) \times \cdots \times dom(S_m)$. In the usual way, $t(A_i)$ (resp. $t(S_j)$) denotes the value assigned to attribute A_i (resp. S_j) by a tuple t. A tuple is called an *action tuple* if it contains only values for the action attributes and a *state tuple* if it contains only values for the state attributes. An *event knowledgebase* $\mathcal{K}$ is a finite set of tuples w.r.t. $(\mathbf{A}, \mathbf{S})$.

Example 16. *Throughout this chapter, two sample datasets will be used. Figure 7.1 presents the first, a* school *event KB containing data related to school performance in some region. The columns labeled* $A_1, ..., A_4$ *represent action attributes, while the columns labeled* $S_1, ..., S_5$ *represent state attributes.*

The school dataset contains nine attributes explained at the bottom of Figure 7.1. Math and reading scores obtained from standardized tests are combined into one annual proficiency score*. School administrators have the goal of increasing proficiency and graduation percentages by certain amounts.*

The second dataset, a counterterrorism *event KB, is given in Figure 7.2, and contains information regarding the tactical level behaviors of a terrorist organization as well as characteristics of its sources of support and relations with the state in which is it based. This data is a sample from the Minorities at Risk Organizational Behavior (MAROB) [125, 46] database mentioned in the preceding chapters, which tracks the behaviors and characteristics of 118 ethnopolitical organizations likely to utilize violence or terrorism to address the interests of their ethnic group. Action*

A_1	A_2	A_3	A_4	S_1	S_2	S_3	S_4	S_5
t_1: 9,532	61.6	7.8	4.2	81.1	49.1	51.3	50.6	Yes
t_2: 9,691	63.2	7.8	5.7	82.3	52.1	54.6	53.3	No
t_3: 9,924	63.8	8.1	3.1	82.0	59.8	60.4	60.1	Yes
t_4: 10,148	64.2	7.6	3.4	83.4	60.5	64.2	63.3	Yes
t_5: 10,022	64.0	7.2	2.9	83.2	63.9	68.9	66.9	Yes

Figure 7.1: Small instance of a KB containing hypothetical school performance data. Action variables are A_1: Funding (\$/Student), A_2: Salaries (% of Total Funding), A_3: Student/Staff Ratio, A_4: Proficiency Increase Target. State variables are S_1: Graduation (%), S_2: Math Proficiency, S_3: Reading Proficiency, S_4: Proficiency Score, S_5: Target Reached.

attributes are represented in the columns labeled $A_5, \ldots, A_{11}$, and state attributes are given by the columns labeled $S_6, \ldots, S_{10}$.

The attributes in the counterterrorism dataset are categorical, using integer values to represent different degrees of intensity for a given behavior or characteristic of the organization. Each attribute is described in more detail in Figure 7.2. The international community may want to decrease the occurrence of terrorist violence through diplomatic or other interventions in sources of terrorist support or the actions of a particular state.

7.2 Actionable State Change Attempts

In this section, the notion of an *actionable state change attempt* is formalized for modeling a policy. The idea is that a state change attempt, when successfully applied to a given tuple, will change the actionable attributes with the hope of these changes resulting in a change in the state (the behaviors of the external agent). For example, decreasing class size may lead to better proficiency scores.

	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	S_6	S_7	S_8	S_9	S_{10}
t_1:	0	0	0	0	1	1	0	0	0	0	0	0
t_2:	0	1	0	1	2	1	0	0	0	0	0	0
t_3:	0	1	0	1	4	2	0	1	0	0	1	0
t_4:	0	1	0	1	4	2	0	1	0	0	0	0
t_5:	0	1	0	1	4	2	0	1	0	0	0	0
t_6:	0	1	0	1	4	2	-1	1	1	0	1	0
t_7:	0	1	0	1	4	2	-1	1	1	1	1	1

Figure 7.2: Small instance of a KB containing behavioral and situational data for a terrorist organization. Action variables are A_5: Support from the diaspora $(0 = No, 1 = Yes)$, A_6: Financial support from a foreign state $(0 = No,\ 1 = Yes)$, A_7: Political support from a foreign state $(0 = No,\ 1 = Yes)$, A_8: Non-violent military support from a foreign state $(0 = No,\ 1 = Yes)$, A_9: State repression of the organization ($1 =$ *Org is legal*, $2 =$ *Legal but repressed*, $3 =$ *Illegal but tolerated*, $4 =$ *Illegal and repressed periodically*, $5 =$ *Illegal and repressed*), A_{10}: State violence targeting the organization ($1 =$ *None*, $2 =$ *Periodic violence*, $3 =$ *Consistent violence*), A_{11}: Success of the organization in achieving state agreement ($-1 =$ *Negotiations refused*, $0 =$ *No negotiation sought*, $1 =$ *Negotiated*, $2 =$ *Some state concessions*, $3 =$ *State conceded primary goal*, $4 =$ *State previously conceded goal*). State variables are S_6: Armed Attacks $(0 = No,\ 1 = Yes)$, S_7: Bombings $(0 = No,\ 1 = Yes)$, S_8: Hijackings $(0 = No,\ 1 = Yes)$, S_9: Kidnappings $(0 = No,\ 1 = Yes)$, S_10: Suicide bombings $(0 = No,\ 1 = Yes)$.

Definition 35 (State Change Attempt (SCA)). *A* simple state change attempt *is a triple (A_i, vf, vt) where $vf, vt \in Dom(A_i)$ for some $A_i \in \mathbf{A}$. A (non-simple)* state change attempt *(SCA for short) is a set $\{(A_{i_1}, vf_1, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ of simple state change attempts such that $i_j \neq i_l$ for all $j \neq l$.*

When clear from context, these concepts are referred to as *simple changes* and *changes*, respectively. Intuitively, a *simple* state change attempt modifies one attribute, while a state change attempt may modify more than one.

Definition 36 (Applicability of an SCA). *Given a tuple t, an action attribute A_i, and $vf, vt \in Dom(A_i)$, a simple state change attempt (A_i, vf, vt) is* applicable *w.r.t. t iff $t(A_i) = vf$. The* result of applying a simple state change attempt *that* *is applicable w.r.t. t is t' where $t'(A_i) = vt$ and for action attribute $A_j \neq A_i$, $t'(A_j) = t(A_j)$. $\gamma(t, (A_i, vf, vt))$ denotes the resulting tuple t'.*

A state change attempt $SCA = \{(A_{i_1}, vf_1, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ is applicable *w.r.t. t iff all (A_{i_j}, vf_j, vt_j) for $1 \leq j \leq k$ are applicable w.r.t. t.*

$\gamma(t, SCA)$ denotes the application of SCA to t.

Example 17. *A simple state change attempt w.r.t. the school data from Example 16 could be $a_1 = (A_1, 8700, 8850)$, indicating that funding is increased from \$8,700 to \$8,850 per student, or $a_2 = (A_2, 62.3, 65)$ indicating that salaries are increased from 62.3% to 65% of the budget. Let $SCA = \{a_1, a_2\}$ be a state change attempt. If the values of the action attributes in the current environment are $t = (8700, 64, 7, 3.2)$, then a_1 is applicable w.r.t. t, but a_2 is not. The result of applying a_1 to t is $\gamma(t, (A_1, 8700, 8850)) = t' = (8850, 64, 7, 3.2)$.*

Example 18. *Looking now at the counterterrorism data from Example 16, a simple state change attempt could be $a_3 = (A_6, 1, 0)$, indicating a disruption of foreign state financial support for a terrorist organization, or $a_4 = (A_{10}, 2, 1)$, indicating that the state would stop using lethal violence against the organization. Let $SCA = \{a_3, a_4\}$ be a state change attempt. Assuming that the values of the action attributes in the current environment at time t are $t = (0, 1, 0, 1, 4, 2, -1)$, then both a_3 and a_4 are applicable. The result of applying SCA is $\gamma(t, SCA) = t' = (0, 0, 0, 1, 4, 1, -1)$.*

The result of applying a state change attempt is therefore the result of applying each simple change. However, these changes do not occur without cost.

Definition 37 (Cost of a simple change attempt)**.** *Let $a = (A_i, vf, vt)$ be a simple state change attempt. The* cost of attempting *a is given by a real-valued function* *cost* : $\{A_1, \ldots, A_n\} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, *where* $cost(A_i, vf, vt)$ *is the cost of changing action attribute* A_i *from* vf *to* vt.

Cost functions will be highly dependent on the application domain and are assumed to be provided by a user. The *cost* of an attempt, $cost(SCA) = \sum_{a \in SCA} cost(a)$, is the sum of the costs of the simple state changes in SCA.

Example 19. *Consider the same simple changes* $a_1 = (A_1, 8700, 8850)$, $a_2 = (A_2, 62.3, 65)$ *from Example 17 and a third simple change* $a_3 = (A_4, 3.8, 3.9)$ *(i.e., increment the proficiency increase target from 3.8 to 3.9). A possible cost function could be defined in terms of monetary cost, in which:* $cost(a_1) = 150 * s$ *(where* s *is a constant set to the number of students affected),* $cost(a_2) = 2.7 * A_1$, *and* $cost(a_3) = 0$ *(no monetary cost associated with changing the proficiency increase target).*

Example 20. *A different cost function may be defined for the counterterrorism dataset, perhaps incorporating the political capital or risk necessary to undertake government actions. Consider again the state change attempt* $SCA = \{a_3, a_4\}$ *from Example 18 where* $a_3 = (A_6, 1, 0)$ *and* $a_4 = (A_{10}, 2, 1)$. *Engaging a state that finances terrorism may require more forceful policies, thereby making them both financially and politically more costly:* $cost(a_3) = 420 + 6.2 * d$ *(where* d *is a constant indicating the degree to which the state is an adversary s.t. closer allies are less costly to engage). Influencing a state's policy towards a terrorist organization may require diplomatic actions that are less costly:* $cost(a_4) = 350 + 4.5 * d$.

To this point, state change attempts have been regarded as always successful. However, in general, one cannot expect this to be the case—the funding per student may remain the same in spite of an attempt to change it. Assume, then, that state change attempts are only probabilistically successful—they only induce the change attempted according to a specified probability. Further, assume that the probability of any simple change occurring successfully depends on the entire set of changes attempted. For instance, when attempting to increase the proficiency target alone, one may expect a relatively small probability of the change actually occurring—the teacher's union is unlikely to accept an increase in their expected performance with no additional compensation. However, when attempting to increase the proficiency target along with an increase in teacher salaries, as in Example 19, there may be a higher probability that both changes will actually occur.

Example 21. *Consider the situation described in Example 19. Here the state change attempt a_2 increases teacher salaries from 62.3% to 65%. On its own, attempting this change may anger taxpayers (who would foot the bill for the increase) and may only have a 10% probability of succeeding. Likewise, increasing per student funding might have a 15% probability of success. However, if the taxpayers happen to be willing to increase teacher salaries, then they might also tend to approve per student funding increases, perhaps leading to a joint probability of 9% that both of these will occur when attempted together.*

Similar probability dependencies may be seen in the case of state change attempts in Example 20. A foreign state may be more likely to cease funding a terrorist organization if it knows that the organization will be subject to less lethal violence by its home state, while the propensity of the home state to use violence may decline if it knows the group is receiving less funding.

Let SCA and $SCA' \subseteq SCA$ be state change attempts and suppose the conditional probabilities $pOccur(SCA'|SCA)$ are known; this is the probability that only the actions in SCA' occur given that SCA is attempted. Such probabilities can either be derived from historical data or be explicitly stated by a user. When a state change attempt SCA is "attempted" for a tuple t describing the current situation, this means that each $SCA' \subset SCA$ has the chance $pOccur(SCA'|SCA)$ of being successful, i.e., of having $\gamma(t, SCA')$ be the resulting tuple.

7.2.1 Effect Estimators

The goal of the Actionable State Change Attempts framework is to take an event KB $\mathcal{K}$ and a goal G (some desired outcome condition on state attributes) that the user wants to achieve, and find an SCA—a policy of changes over the actionable attributes—that "optimally" achieves goal G in accordance with some objective function, such as maximizing the probability of goal G being achieved and minimizing cost. Without loss of generality, assume that all goals are expressed as standard conjunctive selection conditions [121] on state attributes. The concept of *effect estimators* is now defined.

Definition 38 (Effect Estimator)**.** *For action tuple t and goal G, an* effect estimator *is a function $\varepsilon(t, G) \to [0, 1]$ that maps a tuple and a goal to a probability $p \in [0, 1]$.*

Intuitively, $\varepsilon(t, G)$ specifies the conditional probability of goal G holding, given that the action attributes are as specified in t. This quantity can be estimated in many ways, some of which will be investigated later in this chapter. As an initial example, one can imagine using some machine learning algorithm as an effect estimator to determine from historical data how often a school's reading score is above some number k given that the student-teacher ratio, funding per student, teacher salaries, etc. have some specific values.

7.2.2 State Change Effectiveness

As described above, the performance of an actionable state change attempt does not necessarily guarantee that all parts of the SCA will occur. That is, in an attempt to change the situation via state change attempt SCA, any subset of SCA may succeed. For instance, trying to decrease the student/staff ratio and increase the funding per student may result in the student/staff ratio increasing as expected, but the funding per student remaining the same. Thus to truly gauge the overall effectiveness of a state change attempt, the probability of each subset of the attempt occurring must be considered, along with its likelihood of achieving the goal.[1]

Definition 39 (State Change Effectiveness)**.** *The probability of a state change attempt $SCA = \{(A_{i_1}, vf_1, vt_1), \ldots, (A_{i_k}, vf_k, vt_k)\}$ satisfying goal G when applied to the action tuple t is*

$$pEff(t, G, SCA, \varepsilon) = \sum_{SCA' \subseteq SCA} pOccur(SCA'|SCA)\varepsilon(\gamma(t, SCA'), G).$$

$pEff(t, G, SCA, \varepsilon)$ works by summing over all the state changes that may occur given that SCA is attempted; since any subset of SCA can occur, this summation ranges over $SCA' \subseteq SCA$. For each SCA' that may occur, its probability of occurring given that SCA was attempted ($pOccur(SCA'|SCA)$) is multiplied by the effectiveness of the given attempt according to ε (recall that $\gamma(t, SCA')$ is the action tuple resulting from the application of the state change SCA' to the original action tuple t). The following result shows that for arbitrary effect estimators, computing state change effectiveness is intractable.

[1]In this work, it is assumed that each simple change attempt either succeeds or fails completely, i.e., there are no partial effects.

Proposition 6. *For condition G, state change attempt SCA, action tuple t, and effect estimator ε, deciding if $pEff(t, G, SCA, \varepsilon) > 0$ is NP-hard with respect to the number of action attributes. Furthermore, if $\varepsilon(.)$ can be computed in polynomial time with respect to the action schema, the problem is NP-complete.*

Proof. Membership In NP: We show that deciding if $pEff(t, G, SCA) > 0$ is in NP with a witness $SCA' \subseteq SCA$ such that $pOccur(SCA'|SCA) \cdot (1 - pOccur(SCA \setminus SCA'|SCA)) > 0$. Since $pEff(s, G, SCA, \varepsilon)$ is a sum of non-negative terms, such an SCA' must exist when $pEff(t, G, SCA, \varepsilon) > 0$, and this can be checked in polynomial time with respect to $|\mathcal{A}|$ if $\varepsilon(.)$ can be computed in this time.

NP-hardness: We show by reduction from the NP-complete subset sum problem, whereby we are given a finite set of integers I and an integer c and are asked to decide if there is a subset $I' \subset I$ such that $\sum_{i \in I'} i = c$ [34]. Let $I = \{i_1, \ldots, i_n\}$ and let $\mathbf{A} = \{A_1, \ldots, A_n\}$ with $Dom(A_j) = \{0, i_j\}$ for $A_j \in \mathbf{A}$. Let action tuple t be the all-zero n-tuple. Define $SCA = \{(A_j, 0, i_j) \mid 1 \leq j \leq n\}$, and define all $pOccur(SCA'|SCA)$ to be zero unless $\sum_{A_j \in \mathbf{A}} \gamma(t, SCA')(A_j) = c$, in which case $pOccur(SCA'|SCA) = 1/2$. Let G be **true**, ε be a constant function that returns 1 for any KB and conditions, and suppose $\mathcal{K}$ is the empty event KB. Under these conditions $pEff(t, G, SCA, \varepsilon) > 0$ iff there is a subset of I summing to c.

($\Rightarrow$): Suppose $pEff(t, G, SCA, \varepsilon) > 0$ to show there is a subset of I summing to c. Since $pEff(t, G, SCA, \varepsilon) > 0$, there is $SCA' \subset SCA$ such that:

$$pOccur(SCA'|SCA)(1 - pOccur(SCA \setminus SCA'|SCA)) > 0.$$

This implies that $pOccur(SCA'|SCA) > 0$, which in turn implies that for $t' = \gamma(t, SCA')$, $\sum_{A_j \in \mathbf{A}} t'(A_j) = c$. Since t is the zero n-tuple, all non-zero $t'(A)$ result from changes in SCA', so we have that $\sum_{(A_j,0,i_j)\in SCA'} i_j = c$. This gives the set $I' = \{i_j|(A_j, 0, i_j) \in SCA'\}$ which describes a subset of I whose sum is c.

($\Leftarrow$): Let $I' \subset I$ be the subset of I such that $\sum_{i_j\in I'} i_j = c$. Now consider the state change attempt $SCA' = \{(A_j, 0, i_j)|i_j \in I'\}$. Clearly $SCA' \subset SCA$ and $(\gamma(t, SCA'), \cdot) \in \sigma_G(\mathcal{K})$, so

$$pOccur(SCA'|SCA) \cdot (1 - pOccur(SCA \setminus SCA'|SCA)) \tag{7.1}$$

will be a term in the sum defining $pEff(t, G, SCA, \varepsilon)$. Since $\sum_{i_j\in I'} i_j = c$ we know that for $t' = \gamma(t, SCA')$, $\sum_{A_j\in\mathbf{A}} t'(A_j) = c$ and therefore that $pOccur(SCA'|SCA) > 0$. Since $(1 - pOccur(SCA \setminus SCA'|SCA))$ is at least 0.5, and this is the only term in Equation 7.1 that may potentially be zero, this proves that Equation 7.1 is non-zero. Further, since all terms in the sum defining $pEff(t, G, SCA, \varepsilon)$ are zero or positive, this suffices to prove that $pEff(t, G, SCA, \varepsilon) > 0$. □

7.3 Optimal State Change Attempts

When finding an actionable state change attempt, there are many possible ways of formulating the problem and defining optimality. Several possibilities for the optimal SCA problem are presented below. Assume that $\mathbf{A} = \{A_1, \ldots, A_n\}$ and $\mathbf{S} = \{S_1, \ldots, S_m\}$, $\mathcal{K}$ is an event KB, t is an action tuple describing the current values of the actionable variables, G is a goal over $\mathbf{S}$, and the functions *cost* and *pOccur*, as mentioned earlier, are both given.

The Lowest Cost SCA Problem. Given real number κ, does there exist a feasible change attempt SCA such that $cost(SCA) \leq \kappa$ and $pEff(t, G, SCA, \varepsilon) > 0$?

The Highest Probability SCA Problem. Given a real number $p \in [0, 1]$, does there exist a change attempt SCA such that $pEff(t, G, SCA, \varepsilon) \geq p$?

The Optimal Threshold Effectiveness Problem. Given a threshold $p \in [0, 1]$, and real number κ, does there exist a change attempt SCA such that $pEff(t, G, SCA, \varepsilon) \geq p$ and $cost(SCA) \leq \kappa$? This problem is the result of combining both the Highest Probability and Lowest Cost problems stated above.

The Limited Cardinality SCA Problems. Given a positive integer h, does there exist a change attempt SCA such that $|SCA| \leq h$ and SCA satisfies one of the conditions from the problems above? For instance, the limited cardinality highest probability SCA will, given a real number $p \in [0, 1]$, tell if there exists a change attempt SCA such that $pEff(t, G, SCA, \varepsilon) \geq p$ and $|SCA| \leq h$.

All of the above problems are stated as decision problems asking whether an SCA satisfying certain conditions exists. Search problems to *find* such an SCA can be analogously stated. Any state change attempt that is a solution to one of these problems (say, problem P) is referred to as an *optimal state change attempt* (OSCA, for short) w.r.t. P.

Theorem 7. *If the effect estimator used can be computed in PTIME, the decision problems associated with the different definitions of optimal state change attempts belong to the following complexity classes, all with respect to* $|\mathbf{A}|$*:*

(i) *The Lowest Cost SCA problem is NP-complete.*

(ii) *The Highest Prob. SCA problem is #P-hard and in PSPACE.*

(iii) *The Optimal Threshold Effectiveness Problem is #P-hard and in PSPACE.*

(iv) *All Limited Cardinality SCA problems are in PTIME.*

Proof. Each part is proved in turn:

(i) *Membership in NP*: Let ε be an effect estimator that can be computed in polynomial time w.r.t. $|\mathbf{A}|$. A witness change attempt SCA, along with $SCA' \subseteq SCA$ such that $cost(SCA) \leq \kappa$, $pOccur(SCA'|SCA)(1-pOccur(SCA \setminus SCA'|SCA)) \cdot \varepsilon(t' = \gamma(t, SCA'), G) > 0$, and $(\gamma(t, SCA'), \cdot) \in \sigma_G(\mathcal{K})$ (implying $pEff(t, G, SCA, \varepsilon) > 0$) can be verified in polynomial time w.r.t. $|\mathbf{A}|$.

NP-Hardness: The NP-hardness proof from Theorem 8 can be extended for this purpose by simply assuming that the only possible values in $Dom(A_i)$ are 0 and i_j (*cf.* the reduction in the proof) and assigning κ to be one greater than the sum of state change attempt costs. Therefore, any subset of SCA as defined for which there exists a subset of I summing to c can be seen as a state change attempt with the required property.

(ii) # *P-hard*: Let F be a SAT formula with variables $v_0, \ldots, v_n$, and N be a number. The problem of determining if the number of solutions to F is greater than or equal to N is #-P complete [51]. Let there be an action attribute A_i for $i = 0, 1, \ldots, n$, with domain $Dom(A_i) = \{0, 1\}$, and let $t(A_i) = 0$ in the action tuple t for all A_i. All feasible simple state change attempts have the form $(A_i, 0, 1)$. Define the cost function to always return 0, $pOccur$ to always be 0.5, and let $\varepsilon(C_1, C_2)$ be one if $C_2 = F$ and C_1 exactly specifies a tuple t', where t' satisfies F, and zero otherwise. Define p to be $N \cdot 0.25$. The number of solutions to F is greater than or equal to N iff there is a feasible state change attempt SCA such that the cost of SCA is less than or equal to 0 and $pEff(t, F, SCA, \varepsilon) \geq p$.

(iii) Since the size of the state change attempt is at most h, the space of possible state change attempts is bounded by a polynomial in $|\mathbf{A}|$. To see why this is the case, it suffices to recall that each $A_i \in \mathbf{A}$ has a *finite* domain $dom(A_i)$, that a state change attempt can specify at most one simple state change attempt per attribute in $\mathbf{A}$, and that $\mathcal{C}_h^n \in O(n^h)$, where $n = |\mathbf{A}|$. Therefore, in the worst case we have to check $\sum_{A \subseteq \mathbf{A}, |A| \leq h} \left(\prod_{A_i \in A} dom(A_i) \right)$, which is in $O(n^h)$.

(iv) Analogous to part (ii).

□

In the proof of the above, all NP-hard reductions use the results of Proposition 6, the $\#P$-hard reductions use $\#SAT$ (the language $\{\langle F, n \rangle\}$, where F is a formula with exactly n solutions), membership in NP is shown by providing a witness. Membership in PSPACE and PTIME are shown below by giving algorithms with the necessary properties.

7.3.1 Basic Algorithms

To show that the problems under limited cardinality are in PTIME, an explicit polynomial time algorithm is given for solving these problems. This algorithm can also be extended to solve all the problems posed in the previous section. The algorithm works by first enumerating each possible state change attempt with size at most h, then choosing the one which solves the appropriate problem. Since there are only $O(|\mathbf{A}|^h)$ such state change attempts, this algorithm runs in PTIME with respect to the number of action attributes $|\mathbf{A}|$.

```
Algorithm limitedSCASet($\mathbf{A}, t, G, \varepsilon, h$)
1.      $R = \emptyset$ // the set to be returned
2.      Add ($\emptyset, 0,$ pEff$(t, G, \emptyset, \varepsilon)$) to $R$ // Initialize $R$
        with empty state change attempt
3.      for each $A_i \in \mathbf{A}$ do
4.         for each value $v \in dom(A_i)$ do
5.            if $v = t(A_i)$ then continue // Go to next
              value, $t$ won't be changed by this SCA

              // iterate over all members of $R$, growing
              those which are small enough
6.            for each $(SCA, c, ef) \in R$ do
7.               if $|SCA| = h$ then continue
8.               Let $SCA' = SCA \cup \{(A_i, t(A_i), v)\}$
9.               Let $c'$ be the cost of $SCA'$ and $ef'$ be
                    pEff$(t, G, SCA', \varepsilon)$
10.              Add $(SCA', c', ef')$ to $R$
11.     return $R$
```

Figure 7.3: The limitedSCASet algorithm.

The limitedSCASet algorithm for enumerating state change attempts of size at most h, along with their cost and probability of effectiveness, is given in Figure 7.3 and runs in time in $O(|\mathbf{A}|^h)$.

Proposition 7. *The limitedSCASet algorithm runs in time in $O(|\mathbf{A}|^h)$ and returns all (SCA, c, ef) where $|SCA| \leq h$, $c = cost(SCA)$ and $ef = pEff(t, G, SCA, \varepsilon)$.*

Proof. The size of R is at most $(|\mathbf{A}| \cdot \max_i(|dom(A_i)|))^h$, and limitedSCASet computes in $O(|R|)$ steps. Since $\max_i(|dom(A_i)|)$ is considered to be a constant, this is $O(|\mathbf{A}|^h)$. Further, to see that R is correct, clearly c and ef are correct for each (SCA, c, ef) (see line 9), so it remains to show that all SCA of size $\leq h$ are included in R. Consider any state change attempt SCA to show there is a (SCA, c, ef) in R. We show this by induction on $|SCA|$. As a base case, when $|SCA|$ is zero, (SCA, c, ef) is in R. Suppose all SCA of size k are in R to show that any SCA_{k+1}

of size $k+1$ is in R. Let $SCA_k \cup \{(A_i^*, vf, vt)\}$ be SCA_{k+1}. Since $|SCA_k|$ has size k there is (SCA_k, c_k, ef_k) in R. Therefore when we run line 10 with $SCA' = SCA_{k+1}$, $A_i = A_i^*$ and $v = vt$, then $(SCA_{k+1}, c_{k+1}, ef_{k+1})$ will be added to R. Thus all SCA of size less than or equal to $k+1$ will be in R as (SCA, c, ef) with correct c and ef. $\square$

Using the limitedSCASet algorithm, solutions to each of the limited cardinality SCA problems can now be computed.

- **Lowest Cost Limited SCA Algorithm**: For cost threshold κ, let R be limitedSCASet$(\mathbf{A}, t, G, \varepsilon, h)$. Eliminate all $(SCA, c, 0)$ from R, then eliminate all (SCA, c, ef) from R where $c > \kappa$. Return true if R is non-empty, false otherwise. $(SCA, c, ef) \in R$ has minimal c.

- **Highest Probability Limited SCA Algorithm**: For probability threshold p, let R be limitedSCASet$(\mathbf{A}, t, G, \varepsilon, h)$, eliminate all (SCA, c, ef) where $ef < p$, and return true if R is non-empty, false otherwise.

- **Optimal Threshold Effectiveness Limited SCA Algorithm**: For probability threshold p and cost threshold κ, let R be limitedSCASet$(\mathbf{A}, t, G, \varepsilon, h)$, and eliminate all (SCA, c, ef) from R where either $c > \kappa$ or $ef < p$. Return true if R is non-empty, false otherwise.

Each of those algorithms correctly computes the associated decision problem from Section 7.3, as a corollary of Proposition 7.

Corollary 1. *Each of the following algorithms correctly computes the associated decision problem: Lowest Cost Limited SCA Algorithm, Highest Probability Limited SCA Algorithm, and Optimal Threshold Effectiveness Limited SCA Algorithm.*

To extend this technique to non-limited, general versions of the various problems, the limited version of the problem simply needs to be solved with h equal to $|\mathbf{A}|$ as follows:

- **Lowest Cost SCA Problem**: For cost threshold κ, let R be $\mathsf{limitedSCASet}(\mathbf{A}, t, G, \varepsilon, |\mathbf{A}|)$. Eliminate all $(SCA, c, 0)$ from R, then eliminate all (SCA, c, ef) from R where $c > \kappa$. Return true if R is non-empty, false otherwise.

- **Highest Probability SCA Algorithm**: For probability threshold p, let R be $\mathsf{limitedSCASet}(\mathbf{A}, t, G, \varepsilon, |\mathbf{A}|)$, eliminate all (SCA, c, ef) where $ef < p$, and return true if R is non-empty, false otherwise.

- **Optimal Threshold Effectiveness Algorithm**: For probability threshold p and cost threshold κ, let R be $\mathsf{limitedSCASet}(\mathbf{A}, t, G, \varepsilon, |\mathbf{A}|)$, and eliminate all (SCA, c, ef) from R where either $c > \kappa$ or $ef < p$. Return true if R is non-empty, false otherwise.

Again, each of these algorithms correctly computes the associated decision problem from Section 7.3, as a corollary of Proposition 7.

Corollary 2. *Each of the following algorithms correctly computes the associated decision problem: Lowest Cost SCA Algorithm, Highest Probability SCA Algorithm, and Optimal Threshold Effectiveness Algorithm.*

7.4 Effect Estimators for Finding Optimal SCAs

As mentioned above, the Actionable State Change Attempts framework uses the probability of effectiveness from Definition 39 to take into account not only the uncertainty of the success of a change attempt, but also its likelihood of achieving the goal. In this section several sorts of effect estimator functions are introduced that are used to compute the likelihood of a given action tuple satisfying a given goal condition G. An effect estimate essentially answers the question: "if I succeed in changing the environment in this way, what is the probability that the new environment satisfies my goal?"

7.4.1 Learning Algorithms as Effect Estimators

Any standard supervised learning algorithm (i.e., neural nets, decision trees, case based learning, etc.) can be applied to an event knowledgebase $\mathcal{K}$ as an effect estimator. Supervised learning algorithms require training data that includes a categorization as either positive or negative instances of a given category. From that training data, they construct a *classifier*, or a procedure that classifies future cases, even those not already seen in the training set.

From a given goal condition G and knowledgebase $\mathcal{K}$, training data can be constructed for any standard machine learning technique. Each member of $\mathcal{K}$ is categorized according to G; that is, if it satisfies G then the tuple is a positive instance, otherwise it is a negative instance. Then, only the actionable portion of the tuple, along with these categorizations, is used to train a decision tree, a neural network, a support vector machine or some other classifier. The resulting classifier is the effect estimator—it will determine for any given action tuple if the resulting state attributes are likely to be a positive instance (satisfying the goal G) or not.

A machine learning algorithm can be abstractly modeled as a *learner*, which, given the appropriate information, will produce a *classifier*.

Definition 40 (Classification Algorithm). *For event KB $\mathcal{K}$ and goal condition G, a classification algorithm is a function* $\textsf{learner} : (\mathcal{K}, G) \mapsto \textsf{classifier}$*, where* $\textsf{classifier}$ *is a function from action tuples to the interval* $[0, 1]$*.*

Example 22. *A neural network fits this definition in the following way [96]. First,* $\textsf{learner}$ *is defined to be a function that generates a neural network with input nodes for each action attribute and exactly one output node with a domain of* $[0, 1]$*. The* $\textsf{learner}$ *function then trains the network via backpropogation according to $\mathcal{K}$ and G, where those tuples in $\mathcal{K}$ that satisfy G are positive instances (expecting the output node to have value* 1*) and those tuples in $\mathcal{K}$ that do not satisfy G are negative instances (expecting the output node to have a value of* 0*). The resulting network is the* $\textsf{classifier}$ *function, and will, given a set of values for the action attributes, return a value in the interval* $[0, 1]$*.*

A classification algorithm can be used to create a *learned effect estimator*.

Definition 41 (Learned Effect Estimator). *Given a classification algorithm learner, a learned effect estimator is defined to be $\varepsilon_{lrn}(learner, \mathcal{K})(t, G)$. The learned effect estimator returns $learner(\mathcal{K}, G)(t)$.*

Example 23. *Using the learning algorithm C4.5, $\varepsilon_{lrn}(C4.5, \mathcal{K})(t, G)$ will first construct a decision tree T according to samples from $\mathcal{K}$ classified according to G. Then the decision tree T will be queried to classify the action tuple t, and this classification will be the return value—1 if t is classified the same as tuples satisfying G, and 0 if not (recall, t is an action tuple and the goal G is a formula over state attributes, so there is no way to check if t satisfies G directly).*

7.4.2 Data Selection Effect Estimators

This section examines another special case of an effect estimator—one that uses selection operations in a database to create a conditional probability estimation. Selection operations will be denoted $\sigma_G(\mathcal{K})$, where $\mathcal{K}$ is an event KB and G is some goal condition on the state tuples. $\sigma_G(\mathcal{K})$ returns the subset of $\mathcal{K}$ satisfying the condition G.

Definition 42 (Data Selection Effect Estimator)**.** *For goal G and action tuple t, a data selection effect estimator is a function that takes an event knowledgebase $\mathcal{K}$ as input and returns an effect estimator: $\varepsilon^* : \mathcal{K} \mapsto (t, G) \mapsto p$, where $p \in [0, 1]$. ε^* must be implemented with a fixed number of selection operations on $\mathcal{K}$, and $\varepsilon^*(\mathcal{K})(t, G)$ is defined to be 0 if there is no tuple in $\mathcal{K}$ whose action attributes match t.*

A data selection effect estimator is distinct in that it depends explicitly on selection from event KB $\mathcal{K}$. While data selection effect estimators are limited to using only selection operations, there are still many ways to specify the relationship between G and the situation described by t (for instance, Definitions 43, and 44).

In the following, the notation used for selection operators in databases is abused slightly by writing $\sigma_t(\mathcal{K})$ to denote the selection of all the tuples in $\mathcal{K}$ that have the values described by t for the corresponding attributes.

Definition 43 (Data Ratio Effect Estimator)**.** *The data ratio effect estimator returns the fraction of the time that G holds out of all times when the attributes in t are matched.*

$$\varepsilon_r^*(\mathcal{K})(t, G) \overset{def}{=} \begin{cases} \frac{|\sigma_{t \wedge G}(\mathcal{K})|}{|\sigma_t(\mathcal{K})|} & : |\sigma_t(\mathcal{K})| > 0 \\ 0 & : |\sigma_t(\mathcal{K})| = 0 \end{cases}$$

The data ratio effect estimator returns the marginal probability of G occurring given that the values specified by the action tuple t occur.

Example 24. *Consider a school metrics database containing only three columns: class size, teacher salary and graduation rate. The class size and teacher salary are action attributes, while the graduation rate is a state attribute. A user may want to determine from the data what fraction of the time a graduation rate is at least* 95% *for an average class size of 20 and an average teacher salary of $60,000. According to* ε_r^**, this fraction is the fraction of tuples in the database with class size 20 and teacher salary $60,000 that have a graduation rate over 95% divided by the total number of tuples in the database with class size 20 and teacher salary $60,000.*

Example 25. *Look also at how a data ratio effect estimator would operate on a subset of the counterterrorism database. Suppose the data only contains the columns foreign state support, state violence, and bombings where foreign state support and state violence are action attribute and bombings is a state attribute. In this case, a security analyst might want to determine what fraction of the time bombings occur when foreign state support is true (i.e., the value equals 1) and the state uses lethal violence against the group (i.e., the value is greater than or equal to 2). Using* ε_r^**, the number of tuples in the database where foreign state support is 1, state violence is greater than or equal to 2, and bombing is 1 is divided by the number of tuples where foreign state support is 1 and state violence is greater than or equal to 2.*

One important feature of the data ratio effect estimator is that when there is no information in the database on a given tuple, the data ratio effect estimator assumes the tuple to be a negative instance. This will allow the quick elimination of possibilities not contained in the database and will reduce the search space needed to compute optimal state change attempts.

Further examples of data selection effect estimators include cautious or optimistic ratio effect estimators, which take a confidence interval into account.

Definition 44 (Cautious Ratio Effect Estimators)**.** *The cautious ratio effect estimator returns the probability of G given t to be the low end of the 95% confidence interval:*

$$\varepsilon^*_{c95}(\mathcal{K})(t,G) \overset{def}{=} \varepsilon^*_r(\mathcal{K})(t,G) - 1.96 \cdot \sqrt{\frac{\varepsilon^*_r(\mathcal{K})(t,G)(1-\varepsilon^*_r(\mathcal{K})(t,G))}{|\sigma_t(\mathcal{K})|}}$$

(if $\sigma_t(\mathcal{K})$ is empty, $\varepsilon^*_{c95}(\mathcal{K})(t,G)$ is defined to be zero).

There is a whole class of cautious ratio effect estimators, one for every confidence level (80%, 90%, 99%, etc.). There are also optimistic ratio effect estimators, which return the high end rather than the low end of the confidence interval.

7.4.3 Computing OSCAs with Data Selection Effect Estimators

Using data selection effect estimators, specific algorithms can be devised for solving the optimal state change attempt problems given in Section 7.3. Since data selection effect estimators are computed via a finite number of selection operations, they can always be computed in time in $O(|\mathcal{K}|)$. The complexity of finding SCAs changes when using data selection effect estimators. Problems that were NP-complete or $\#P$-hard w.r.t. the size of the action schema are polynomial in $|\mathcal{K}|$ when only data selection effect estimators are allowed.

```
Algorithm DSEE_OSCA($\mathcal{K}, G, env, p$)
1.      Let $Dat1 = \emptyset$ // $Dat1$ will contain state change
        attempts and their probability of occurrence

        // Iterate through all tuples satisfying $G \in \mathcal{K}$
2.      for each $t \in \sigma_G(\mathcal{K})$ do
            // Create $SCA$ s.t. $\gamma(\boldsymbol{env}, SCA)$ equals $t$ on
               action attributes
3.          $SCA = \{(A, \boldsymbol{env}(A), t(A)) \mid \boldsymbol{env}(A) \neq t(A)\}$
4.          if $(SCA, \cdot) \in Dat1$ then continue
5.          Let $f = \varepsilon_r^*(\mathcal{K})(t, G)$
6.          Add $(SCA, f)$ to $Dat1$
7.      Let $Dat2 = \emptyset$
8.      for each $(SCA, f) \in Dat1$ do
9.          Let $nextF = \boldsymbol{pOccur}(SCA|SCA) \cdot f$
10.         for each $(SCA', f') \in Dat1$ do
11.             if $SCA' \subsetneq SCA$ then
12.                 $nextF = nextF + \boldsymbol{pOccur}(SCA'|SCA) \cdot f'$
13.         Add $(SCA, ef)$ where $(SCA, nextF)$ to $Dat2$
14.     Remove any $(SCA, ef)$ from $Dat2$ where $ef < p$
15.     return $argmin_{(SCA,ef) \in Dat2}(cost(SCA))$
```

Figure 7.4: The DSEE_OSCA algorithm for solving the optimal threshold effectiveness problem.

Proposition 8. *For goal G, state change attempt SCA, action tuple t, and event KB $\mathcal{K}$, if the effect estimator ε^* is a data selection effect estimator then deciding if $pEff(t, G, SCA, \varepsilon^*(\mathcal{K})) > 0$ takes $O(|\mathcal{K}|^2)$ time.*

Theorem 8. *If the effect estimator is a data selection effect estimator, then the Lowest Cost, Highest Probability, Limited Cardinality, and Optimal Threshold Effectiveness problems can all be solved in $O(|\mathcal{K}|^2)$ time and are therefore in PTIME with respect to the number of tuples in the event KB.*

Figure 7.4 presents the DSEE_OSCA algorithm to solve optimal threshold effectiveness problem using data selection effect estimators. Here, only the data ratio effect estimator (Definition 43) is used. The DSEE_OSCA algorithm works by selecting all tuples in the event KB $\mathcal{K}$ satisfying the goal condition and adding the pair (SCA, f) to a data structure $Dat1$, where f is the chance that SCA, when successfully applied to the current action tuple env, results in a state satisfying the goal G (i.e., $\varepsilon_r^*(\mathcal{K})(t, G)$). In the next loop, two things happen: (i) f is multiplied by the probability that SCA is successful, and (ii) the algorithm iterates through all state change attempts and sums the probability of occurrence of each subset of SCA with that subset's probability of satisfying the goal G, adding the result to data structure $Dat2$. At this point $Dat2$ contains pairs (SCA, ef), where ef is the probability of effectiveness of SCA according to Definition 39. The algorithm then prunes all state change attempts without sufficiently high probabilities of effectiveness, and returns the one with the lowest cost.

The following result asserts the correctness of the DSEE_OSCA algorithm.

Proposition 9. *The DSEE_OSCA algorithm computes state change attempt SCA such that pEff$(env, G, SCA, \varepsilon_r^*(\mathcal{K})) \geq p$ and there is no other feasible state change attempt SCA' such that cost(SCA') < cost(SCA) and pEff$(env, G, SCA', \varepsilon_r^*(\mathcal{K})) \geq p$.*

Proof. To show this, it suffices to show that on line 15 for all $(SCA, ef) \in Dat2$, $ef = pEff(env, G, SCA, \varepsilon_r^*(\mathcal{K}))$. Consider any $(SCA, ef) \in Dat2$ on that line, and note that due to the loop starting on line 8,

$$ef = \sum_{(SCA', f') \in Dat1, SCA' \subset SCA} f' \cdot pOccur(SCA' \mid SCA).$$

Since for all $(SCA', f') \in Dat1$, $f' = \varepsilon_r^*(\mathcal{K})(\gamma(SCA', env), G)$, this suffices to show that $ef = pEff(SCA, G, env, \varepsilon_r^*(\mathcal{K}))$. □

Proposition 10. *The DSEE_OSCA algorithm runs in time $O(|\mathcal{K}|^2)$.*

Proof. The running time of the algorithm can be divided into two parts. First, the loop on line 2 runs at most $|\mathcal{K}|$ times. In each iteration, line 5 may be computed, which takes at most $2 \cdot |\mathcal{K}|$ computations for ε_r^* (in the worst case, both select operations return the entire event KB). This gives a total running time of $O(|\mathcal{K}|^2)$ for the first loop. Then, since $|Dat1|$ can be no larger than $|\mathcal{K}|$, the loop on line 8 will run $O(|\mathcal{K}|)$ times. The contained loop starting on line 10 will, for the same reason, run $O(|\mathcal{K}|)$ times, giving a run time of $O(|\mathcal{K}|^2)$ for that loop. This puts the total run time at $O(|\mathcal{K}|^2)$. □

7.5 A Comparison with Planning under Uncertainty

In order to investigate how the Actionable State Change Attempts approach to solving the proposed class of problems relates to traditional approaches, such as planning under uncertainty, in this section a mapping is discussed from an instance of an OSCA problem to an instance of a *Markov decision process (MDP)* [90, 12]. First of all, recall the elements of the Actionable State Change Attempts framework required to describe an instance of an OSCA problem:

(i) A set of *action attributes* **A** corresponding to the actionable attributes that users can potentially *act upon directly* in order to change their values.

(ii) A set of *state attributes* **S** used to describe *situations* in the environment, including the behaviors of other agents or the outcome attributes users may want to influence.

(iii) A *cost function* for state change attempts describing the cost of changing the values of the action attributes.

(iv) An *effect estimator* function describing the conditional probability that a goal condition is true given an assignment of values to the action attributes.

(v) *Conditional probabilities* for the probability of occurrence of SCAs denoting the probability that a certain state change attempt is successful given that another state change was attempted.

(vi) A *goal* condition specified over the values of a subset of the state attributes that describes the state of affairs the user wishes to accomplish.

Using these components, an instance of an OSCA problem can be specified as $O = (\mathbf{A}, \mathbf{S}, \mathit{cost}, \varepsilon, pOccur, G)$. The goal in this case is to compute an optimal SCA w.r.t. cost and/or probability of effectiveness. Similarly, in order to describe an instance of an MDP, the following items are required:

(i) A finite set S of environment *states*.

(ii) A finite set A of *actions*.

(iii) A *transition function* $T : S \times A \rightarrow \Pi(S)$ specifying the probability of arriving at every possible state given that a certain action is taken in a given state.

(iv) A *reward function* $R : S \times A \rightarrow \boldsymbol{R}$ specifying the expected immediate reward gained by taking an action in a state.

The objective is to compute a policy $\pi : S \rightarrow A$ specifying for each state the action that is optimal w.r.t. the expected utility obtained from executing it.

7.5.1 Obtaining an MDP from the Specification of an OSCA Problem

Given an instance of an OSCA problem as described in Section 7.3 above, the specification of a corresponding MDP can be obtained such that optimal policies for this MDP correspond to solutions to the original OSCA problem.

State Space. The set S_{MDP} of MDP states corresponds to the set of all possible tuples $(v_1, \ldots, v_{n+m}) \in dom(A_1) \times \cdots \times dom(A_n) \times dom(S_1) \times \cdots \times dom(S_m)$, where $\bigcup_{i=1}^{n+m} V_i = \mathbf{A} \cup \mathbf{S}$, the set of all actionable and state attributes.

Actions. The set A_{MDP} of possible actions in the MDP domain corresponds to the set of all possible state change attempts. Without considering the fact that not all SCAs will be applicable in every state, the set of actions can be thought of as containing any subset of h action attributes, each of which can be subject to attempted changes to any other possible value in its domain.

Transition Function. The (conditional) probabilities of occurrence can be used to define the transition function T for the MDP, since it is clear what the effect of a change attempt is when it is successful. Formally, let $s, s' \in S_{MDP}$ and $a \in A_{MDP}$; if $s = (u_1, \ldots, u_{n+m})$, $s' = (u'_1, \ldots, u'_{n+m})$, and $a = ((A_{i_1}, vf_1, vt_1), \ldots, (A_{i_h}, vf_h, vt_h))$ for $i_1, \ldots, i_h \in \{1, \ldots, |\mathbf{A}|\}$. Then,

$$T(s, a, s') = \begin{cases} 0 & \text{if } a \text{ is not applicable in } s, \\ pEff(s, G_{s'}, a, \varepsilon) & \text{otherwise.} \end{cases} \tag{7.2}$$

where $G_{s'}$ denotes the condition that imposes the values in s' on the state attributes as the goal. However, if the OSCA problem only requires the *lowest cost* solution (see Section 7.3), T is simply defined as follows. Let $S_a \subseteq S_{MDP}$ be the set of all states s' for which $\mathit{pEff}(s, G_{s'}, a, \varepsilon) \neq 0$:

$$T(s,a,s') = \begin{cases} 0 & \text{if } a \text{ is not applicable in } s, \\ \frac{1}{|S_a|} & \text{otherwise.} \end{cases} \tag{7.3}$$

Reward Function. The reward function of the MDP, which describes the reward directly obtained from performing action $a \in A_{MDP}$ in state $s \in S_{MDP}$, is defined based on two aspects: (i) the probability that a state satisfying the goal is reached by taking action a in state s (this will depend on the *effect estimator* being used), and (ii) the cost of the change attempt associated with action a. It should be noted that, as in the case of the transition function above, the specific problem to be solved (lowest cost, highest probability, etc.), will directly influence the way in which the corresponding reward function is defined (e.g., for highest probability problems, cost is ignored). Let G be the goal corresponding to the OSCA problem instance and, as above, let $s \in S_{MDP}$ and $a \in A_{MDP}$ such that $s = (u_1, \dots, u_{n+m})$ and $a = ((A_{i_1}, vf_1, vt_1), \dots, (A_{i_h}, vf_h, vt_h))$ for $i_1, \dots, i_h \in \{1, \dots, |\mathbf{A}|\}$. The reward function is defined as

$$R(s,a) = \begin{cases} 0 & \text{if } a \text{ is not applicable in } s, \\ \mathit{pEff}(s, G, a, \varepsilon) & \text{otherwise.} \end{cases} \tag{7.4}$$

Similarly, for lowest cost, the reward function is

$$R(s,a) = \begin{cases} 0 & \text{if } a \text{ is not applicable in } s, \\ \mathit{pEff}(s, G, a, \varepsilon) * \frac{1}{cost(a)} & \text{otherwise.} \end{cases} \tag{7.5}$$

As can be seen by the above mapping, the *key point in which the optimal state change attempt problem differs* from planning problems is that SCAs involve executing actions in parallel which, among other things, means that the number of possible simple SCAs that can be considered in a given state is *very large.* This makes planning approaches infeasible since their computational cost is intimately tied to the number of possible actions in the domain (generally assumed to be fixed at a relatively small number). In the case of MDPs, even though state aggregation techniques have been investigated to keep the number of states being considered manageable [17, 37, 119], similar techniques for *action aggregation* have not been developed.

The following results conclude this comparison with planning under uncertainty using MDPs. The first states that given an instance of OSCA, the proposed translation into an MDP is such that an optimal policy under *maximum expected utility (MEU)* for such an MDP expresses a solution for the original instance. Note, however, that such a policy is actually a *fully contingent* plan in that it prescribes an action for every possible state. This means that the state change attempt prescribed in each state is chosen taking into account what would happen if the goal is not immediately reached, and thus states that have a better utility computed in this manner are preferred. A fair comparison with the approach taken in this work would be to have the $OSCA$ algorithms iterate until the goal is satisfied.

Proposition 11. *Let $O = (\mathbf{A}, \mathbf{S}, cost, \varepsilon, pOccur, G)$ be a specification of an OSCA problem and $M = (S_{MDP}, A_{MDP}, T, R)$ be its corresponding translation into an MDP. If π is a policy for M that is optimal w.r.t. the MEU criterion, then for any state $s \in S_{MDP}$, $\pi(s)$ yields a state change attempt that is a solution for O for the values of the state attributes described by s.*

Proof. (*Sketch*) Assume that the OSCA instance given is a *highest probability* instance (the lowest cost case is analogous). By hypothesis we have that π is MEU-optimal, which means that

$$\pi(s) = \arg\max_a \left(R(s,a) + \max_b \left(\sum_{s' \in S_{MDP}} T(s,a,s') \cdot Q(s',b) \right) \right) \tag{7.6}$$

where Q is the action utility function defined as usual. Now, suppose towards a contradiction that there exists a state s such that the state change attempt corresponding to $a = \pi(s)$ is sub-optimal, i.e., there exists another state change attempt $a' = \pi'(s)$ that has a higher probability of effectiveness; formally, $\mathit{pEff}(s, G, \pi'(s), \varepsilon) > \mathit{pEff}(s, G, \pi(s), \varepsilon)$. As both the reward and transition functions are defined in terms of probability of effectiveness, this directly implies that

$$\left(R(s,a') + \max_b \left(\sum_{s' \in S_{MDP}} T(s,a',s') \cdot Q(s',b) \right) \right) >$$

$$\left(R(s,a) + \max_b \left(\sum_{s' \in S_{MDP}} T(s,a,s') \cdot Q(s',b) \right) \right)$$

However, this contradicts Equation 7.6 above since $\pi(s)$ was selected as the state change attempt that maximizes this sum. The contradiction stemmed from the assumption that $\pi(s)$ is sub-optimal; therefore, $\pi(s)$ is a solution to O. $\square$

Second, the computational cost of taking this approach is analyzed. Since there exists in the literature a large variety of algorithms for solving MDPs, only the size of the MDP resulting from the translation of an instance of OSCA is considered.

Proposition 12. *Let $O = (\mathbf{A}, \mathbf{S}, \mathit{cost}, \varepsilon, \mathit{pOccur}, G)$ be a specification of an OSCA problem and $M = (S_{MDP}, A_{MDP}, T, R)$ be its corresponding translation into an MDP.*

Then, $|S_{MDP}| = \prod_{i=1}^{n+m} |dom(V_i)|$*, where* $V_i \in \mathbf{A} \cup \mathbf{S}$*, and* $|A_{MDP}| \leq \sum_{h=1}^{n}(A_h^n * |V|^h)$*, where* $n = |\mathbf{A}|$*, and* $|V|$ *is the maximum number of possible values for an action attribute.*

Proof. For the size of the state space, simply recall that the MDP's state space corresponds to the set of all possible tuples $(v_1, \ldots, v_{n+m}) \in dom(A_1) \times \cdots \times dom(A_n) \times dom(S_1) \times \cdots \times dom(S_m)$, where $\bigcup_{i=1}^{n+m} V_i = \mathbf{A} \cup \mathbf{S}$.

In order to prove the upper bound on the number of actions, suppose that all SCAs will be applicable in every state. Then, $|A_{MDP}| = \sum_{h=1}^{n} \left(A_h^n * |V|^h\right)$, where n is the number of actions in the action schema $\mathbf{A}$, and V is the set of possible values for the action attributes (assume $|dom(A_i)| = |dom(A_j)|$ for all $A_i, A_j \in \mathbf{A}$ for the purpose of this analysis). Basically, this formula states that any combination of h action attributes can be chosen, and each can be attempted to be changed to any other possible value in its domain. $\square$

Consider that, for instance, the well-known Value Iteration algorithm [90, 12] iterates over the entire state space a number of times that is polynomial in $|S_{MDP}|$, $|A_{MDP}|$, β, and B, where B is an upper bound on the number of bits that are needed to represent any numerator or denominator of the factor β [67]. Now, each iteration takes time in $O(|A_{MDP}| \cdot |S_{MDP}|^2)$, meaning that only for very small instances will MDPs of the size expressed in Proposition 12 be feasible.

```
Algorithm TOSCA$(T, G, env, p)$
1.     Let $Dat1 =$ TOSCA-Helper$(T, G, env)$
2.     Let $Dat2 = \emptyset$
3.     for each $(SCA, f) \in Dat1$ do
4.         Let $nextF = \mathbf{pOccur}(SCA|SCA) \cdot f$
5.         for each $(SCA', f') \in Dat1$ do
6.             if $SCA' \subsetneq SCA$ then
7.                 $nextF = nextF + \mathbf{pOccur}(SCA'|SCA) \cdot f'$
8.         Add $(SCA, ef)$ where $(SCA, nextF)$ to $Dat2$
9.     Remove any $(SCA, ef)$ from $Dat2$ where $ef < p$
10.    return $argmin_{(SCA,ef) \in Dat2}(cost(SCA))$
```

Figure 7.5: The TOSCA algorithm.

7.6 Trie-enhanced Optimal State Change Attempt (TOSCA) Algorithm

In this section, the *Trie-enhanced Optimal State Change Attempt* (TOSCA) algorithm is presented that uses tries [47] to improve the performance of solving problems in the Actionable State Change Attempts framework. In TOSCA, a trie is used to index the event KB to reduce the search space necessary for the data selection effect estimator in the DSEE_OSCA algorithm (Figure 7.4). An internal trie node is a pair $(Atr, Edges)$ where $Atr \in \mathbf{A} \cup \mathbf{S}$ is an attribute and $Edges$ contains (v^-, v^+, N) pairs, where v^- and v^+ are values from $Dom(Atr)$ with $v^- < v^+$ and N is another trie node. A leaf node in a trie maintained by TOSCA is simply a set of tuples from the KB, denoted $tuples(N)$. Tries have a unique root node.

A trie is *data correct* if for any leaf node N there is a unique path from the root $(Atr_1, Edges_1), \ldots, (Atr_{k-1}, Edges_{k-1}), N$ such that for all $t \in tuples(N)$ and all i between 1 and $k-1$, there is $(v^-, v^+, (Atr_{i+1}, Edges_{i+1})) \in Edges_i$ such that $v^- \leq t(Atr_i) < v^+$. That is, the path to a leaf node determines which tuples are

```
Algorithm TOSCA-Helper($T, G, env$)
1.      if $T$ is a leaf node then
           // The following is similar to DSEE_OSCA
              (Figure 7.4)
2.         Let $Dat = \emptyset$
3.         for each $t \in \sigma_G(tuples(T))$ do
              // Create $SCA$ s.t. $\gamma(env, SCA) = t$
4.            Let $SCA = \{(A, \boldsymbol{env}(A), t(A)) \mid t(A) \neq \boldsymbol{env}(A)\}$
5.            if $(SCA, \cdot) \in Dat$ then continue to next $t$
6.            $f = \varepsilon_r^*(tuples(T))(t, G)$
7.            Add $(SCA, f)$ to $Dat$
8.         return $Dat$
9.      else
            // Recursively call for all children of $T$
10.        Let $(A, Edges) = T$
11.        return $\cup_{(v^-,v^+,N) \in Edges}$TOSCA-Helper($N, G, \boldsymbol{env}$)
```

Figure 7.6: The TOSCA-Helper procedure.

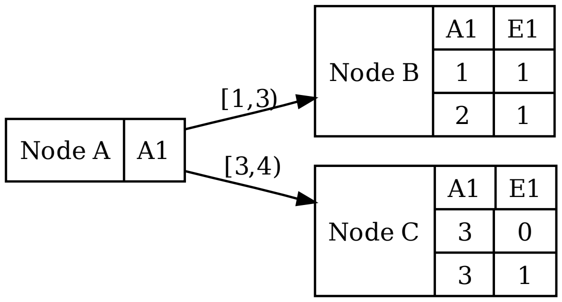

Figure 7.7: The trie used in Example 26.

stored there. A trie is *construction correct* if for all sibling nodes (v_1^-, v_1^+, N_1) and (v_2^-, v_2^+, N_2), $v_1^- \geq v_2^+$ or $v_2^- \geq v_1^+$.

The TOSCA algorithm uses tries as described above to reduce the average case run time for computing optimal state change attempts. TOSCA is divided into the *base* and *helper* procedures, shown in Figures 7.5 and 7.6, respectively. Again, only the data ratio effect estimator (Definition 43) is used in this algorithm.

The following is an example of how TOSCA works.

Example 26. *In this example run of TOSCA, the following simple event KB is used, where $\mathbf{A} = \{A_1\}$ and $\mathbf{S} = \{S_1\}$.*

A_1	S_1
1	1
2	1
3	0
3	1

This event KB can be indexed by the trie T pictured in Figure 7.7. Let the action tuple env $= (A_1 = 0)$ be the environment tuple representing the current value of the actionable attribute A_1, the goal condition be $S_1 = 1$, and the probability threshold be $p = 0.7$. The first step of the TOSCA algorithm (Figure 7.5) is to create Dat1 via the TOSCA-Helper algorithm (Figure 7.6), which recursively traverses the trie, beginning at the root node A. At node B, TOSCA-Helper recognizes a leaf node and selects tuples from that node that satisfy the goal condition, iterating through them in turn beginning with $(A_1 = 1, S_1 = 1)$. The state change attempt that changes the current action tuple env $= (A_1 = 0)$ to $(A_1 = 1, S_1 = 1)$ is SCA $= \{(A_1, 0, 1)\}$. The time saving step of the algorithm occurs at line 6, where ε_r^* is run on the KB tuples(T) instead of the entire event KB (line 5 of DSEE_OSCA in Figure 7.4). Because there is only one tuple in tuples(T) with $A_1 = 1$, and because that tuple also satisfies the goal condition, f is set to 1 and $(\{(A_1, 0, 1)\}, 1)$ is added to Dat. Similarly, $(\{A_1, 0, 2\}, 1)$ is added on the next tuple, $(A_1 = 2, S_1 = 1)$, finishing the call to node B.

The call to node C has slightly different results. The only member of tuples(T) to satisfy the goal condition is $(A_1 = 3, S_1 = 1)$. Further, ε_r^* produces a result of $1/2$, *as of the two tuples with value* 3 *for* A_1*, only one of them satisfies the condition that* $S_1 = 1$*. The returned set from this recursive call contains only* $(\{(A_1, 0, 3)\}, 1/2)$*.*

After merging all recursive calls, the set $\{(\{(A_1, 0, 3)\}, 1/2), (\{(A_1, 0, 2)\}, 1), (\{(A_1, 0, 1)\}, 1)\}$ *is returned and labeled* $Dat1$ *by* TOSCA*. The next loop multiplies the second value of each member of* $Dat2$ *by the probability of the associated state change attempt occurring, which is provided by a user a priori—assume this probability to be* $3/4$ *for all state change attempts. The inner loop then adds the probabilities associated with subsets of the state change attempt (of which there are none in this example). This results in the data structure* $Dat2$ *consisting of pairs* $(SCA, pEff(env, S_1 = 1, SCA, \varepsilon_r^*))$*, or* $\{(\{(A_1, 0, 3)\}, 3/8), (\{(A_1, 0, 2)\}, 3/4), (\{(A_1, 0, 1)\}, 3/4)\}$*.*

At this point, those members of $Dat2$ *with too low a probability of effectiveness are eliminated (only* $(\{A_1, 0, 3\}, 3/8)$*) and the SCA with lowest cost is returned.*

Proposition 13. *The* TOSCA *algorithm computes state change attempt SCA such that* $pEff(env, G, SCA, \varepsilon_r^*(\mathcal{K})) \geq p$ *and there is no other feasible state change attempt* SCA' *such that* $cost(SCA') < cost(SCA)$ *and* $pEff(env, G, SCA', \varepsilon_r^*(\mathcal{K})) \geq p$*.*

Proof. To show this, it suffices to show that on line 10 for all $(SCA, ef) \in Dat2$, $ef = pEff(env, G, SCA, \varepsilon_r^*)$. Consider any $(SCA, ef) \in Dat2$ on that line, and note that due to the loop starting on line 3,

$$ef = \sum_{(SCA', f') \in Dat1, SCA' \subset SCA} f' \cdot pOccur(SCA' | SCA).$$

Since for all $(SCA', f') \in Dat1$, $f' = \varepsilon_r^*(tuple(T))(\gamma(env, SCA), G)$, (from TOSCA-Helper) where $tuples(T)$ is all tuples satisfying $\gamma(env, SCA)$, this suffices to show that $ef = pEff(SCA, G, env, \varepsilon_r^*)$. □

The worst case time complexity of TOSCA is $O(|\mathcal{K}|^2)$. However, the complexity of TOSCA-Helper is $O(|\mathcal{K}| \cdot k)$, where k is the size of the largest leaf node in trie T. Since TOSCA-Helper replaces the loop on line 8 of DSEE_OSCA (Figure 7.4)—a loop that takes time $O(|\mathcal{K}|^2)$—the expected speedup is proportional to $k/|\mathcal{K}|$. Since in the average case, k will be $|\mathcal{K}|/2^h$, (h is the trie's height) this speedup can be large.

While k is bounded by $|\mathcal{K}|$, it is usually much smaller: on the order of $|\mathcal{K}|/2^h$ for a trie of height h. $Dat1$ will have size $O(|\mathcal{K}|)$, as it will be the same as $Dat1$ on line 8 of Figure 7.4. It was produced by at most $2 \cdot |\mathcal{K}|/k$ recursive calls to TOSCA-Helper (there are at most $2 \cdot |\mathcal{K}|/k$ nodes in the trie). When given a leaf node, TOSCA-Helper takes time in $O(k^2)$. Thus the run time of TOSCA-Helper is in $O(|\mathcal{K}| \cdot k)$. The loop on line 3 then runs in time in $O(|\mathcal{K}|^2)$ (it is the same loop as in DSEE_OSCA in Figure 7.4), resulting in an overall run time in $O(|\mathcal{K}|^2)$. However, in practice substantial speedup is gained by using the $O(|\mathcal{K}| \cdot k)$ TOSCA-Helper rather than $O(|\mathcal{K}|^2)$.

7.7 Implementation and Experiments

By conducting an experimental analysis, the outcomes of three major questions can be answered:

(i) Which effect estimator gives the most accurate results?

(ii) Which techniques provide the best runtime with large amounts of data?

(iii) Which techniques provide the best runtime as the number of actionable attributes increases?

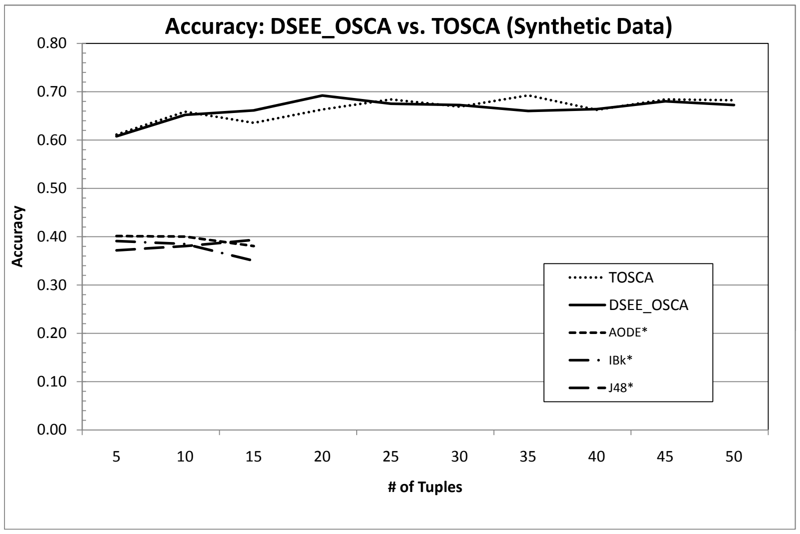

Figure 7.8: A comparison of the accuracy of all algorithms being evaluated, over synthetic data. The learned effect estimators clearly are outperformed by the data selection effect estimator. Algorithms marked with * took longer than two days to complete for inputs of 20 or more tuples.

(iv) Which techniques provide the best runtime with real data?

To address these questions, limitedSCASet (Figure 7.3), DSEE_OSCA (Figure 7.4) and TOSCA (Figure 7.5) were implemented. limitedSCASet was implemented in a modular way so that the effect estimator could be changed, while the other algorithms assume the data ratio effect estimator (Definition 43).

Question 1: Which effect estimator gives the most accurate results? To address this question, the Weka framework's implementation of several machine learning algorithms were used [126]. These include the AODE algorithm, which creates Bayes nets [124]; the IBk algorithm, which uses K-nearest neighbor clustering [4]; and the C4.5 algorithm, which uses entropy minimization techniques to create decision trees that can be used for classification [91]. These algorithms were used to implement respective learned effect estimators (Definition 41) which were combined with limitedSCASet (Figure 7.3) to produce several different methods for

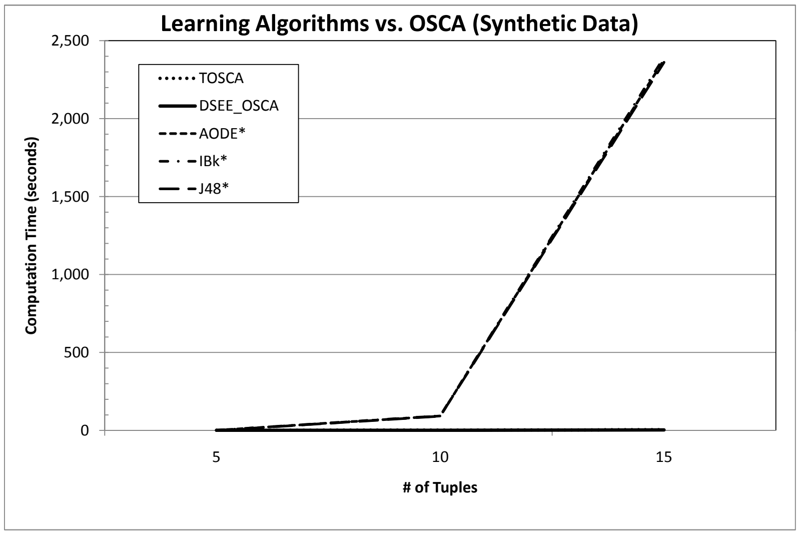

Figure 7.9: A comparison of average running time over at least 60 runs for the data selection (DSEE_OSCA and TOSCA) and learning algorithms (AODE, IBk, and J48) OSCA approaches over synthetic data. The learning algorithms took longer than two days to compute the optimal SCA for 20 or more tuples. DSEE_OSCA and TOSCA's running times correspond to the two curves that are touching the x-axis.

determining the optimal state change attempt. These learned effect estimators were all compared to the data ratio effect estimator.

The data generated for this experiment consisted of k tuples with 4 action attributes and 3 state attributes. Each tuple's value for the action attributes was chosen randomly from $[0, 1]$. To generate the values for the state attributes, random boolean formulas were generated over the action attributes consisting of the operators $<, >, =, \neq$, and $\wedge$, with at most three "$\wedge$" connectives in each formula. In a given tuple, each state attribute value is set to 1 if its associated formula is satisfied by the action attributes in that tuple, and set to 0 otherwise. Because the formula defining the state attributes is known, it is easy to check the accuracy of the state change attempts returned by each algorithm. To do this, the state change attempt is simply applied and the state attribute values determined according to the formulas. The accuracy of a given algorithm will be the fraction of the time the resulting values for the state attributes satisfy the goal condition.

For this experiment, the goal condition was chosen randomly as above, the cost of each simple state change was 1, and the probability of occurrence was set to 1 (all state change attempts always occur as expected).

The results of these experiments are shown in Figures 7.8, and 7.9. In these graphs, two major things are of note. First, the data ratio effect estimator (used in DSEE_OSCA and TOSCA) runs substantially faster than the learned effect estimators. This is due to the algorithms needed for both estimators. Since the data ratio effect estimator assumes that anything not occurring in the event KB is a negative instance, it needs to consider substantially fewer possibilities than the learned effect estimators. The learned effect estimators, on the other hand, allow for any possible tuple to satisfy the goal condition and do not need to limit themselves to those tuples already in the KB. This generality incurs substantial computational costs. Since they consider every combination of all the attribute values in the KB, they are only able to finish computation when there are relatively few tuples. Normally one would hope that this generality would nonetheless result in an increase in accuracy—since learned effect estimators consider more possible solutions than data ratio effect estimators they should be able to find a better solution. However, in these experiments that is not the case. By restricting itself to only those possible solutions currently in the KB, data ratio effect estimators actually increase their accuracy over the various learned effect estimators.

Question 2: Which techniques scale best? Using the same experimental setup, the scalability of DSEE_OSCA and TOSCA were compared when presented with

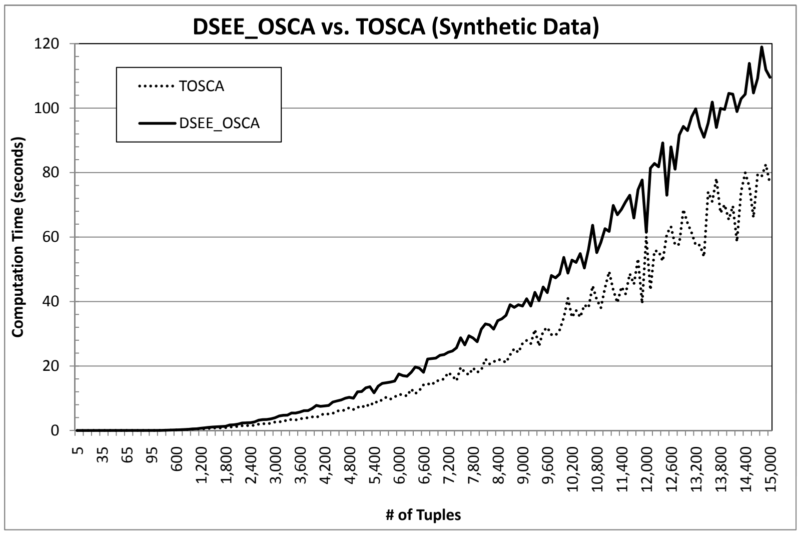

Figure 7.10: A comparison of average running time over at least 60 runs for DSEE_OSCA and TOSCA over synthetic data.

larger amounts of data. Since the algorithms using the learned effect estimators could not scale past 20 tuples, they were not included in this experiment. In this experiment, the algorithms were provided with 1 to 15 thousand tuples. The results are shown in Figure 7.10, where TOSCA performs better than DSEE_OSCA as the event KB increases in size. Note that TOSCA does have a pre-computation step whose running time has been left out of these figures. However, the time needed to compute the trie is several orders of magnitude smaller than the running time of TOSCA; it takes only 91 ms to construct the trie with 10,000 tuples.

Question 3: Which techniques provide the best runtime as the number of attributes increases? Figure 7.11 illustrates how DSEE_OSCA and TOSCA scale as the number of attributes increases in a KB with $8,000$ tuples. None of the learning algorithm effect estimators are depicted because they do not scale to such a large KB. This graph shows TOSCA outperforming DSEE_OSCA, which is an important result because the trie in TOSCA should lose efficiency as the number

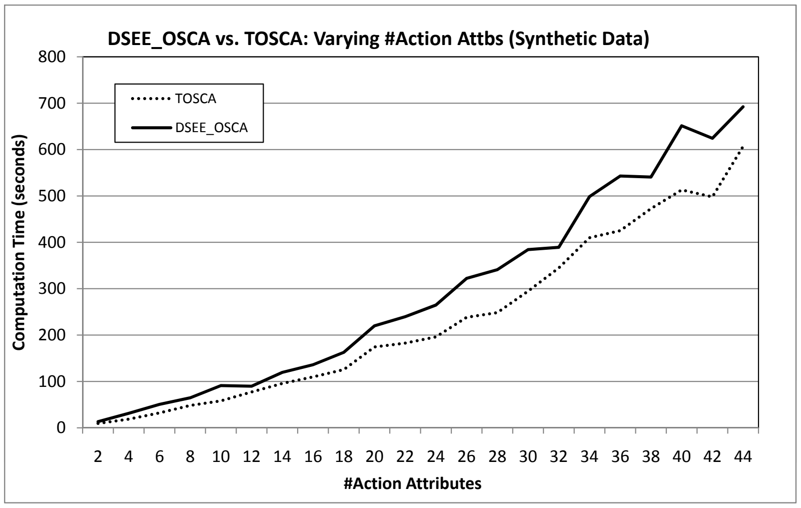

Figure 7.11: The running times of DSEE_OSCA and TOSCA (over synthetic data) as the number of action attributes increases and number of tuples is fixed at $8,000$.

of attributes increases (the trie's depth equals the number of attributes). However, that decrease in the trie's efficiency does not affect its ability to offer TOSCA a speedup over DSEE_OSCA.

The running time of DSEE_OSCA and TOSCA were also compared for varying sizes of the attribute domains. Figure 7.12 shows that TOSCA still consistently outperforms DSEE_OSCA as the size of the attribute domain increases from 2 to 20 possible values for a fixed-size KB.

Question 4: Which algorithms perform best with real-world data? This experiment used the U.S. School Dataset [112], which includes data on school performance, budget, and related variables for all schools in a given state. The state of Arizona was chosen for these tests. The variables can naturally be divided into action and state attributes; to keep the experiments manageable, a subset of about 50 of the most important attributes was chosen, preferring the general variables over the specific ones. The number of attributes of the chosen 50 that would be consid-

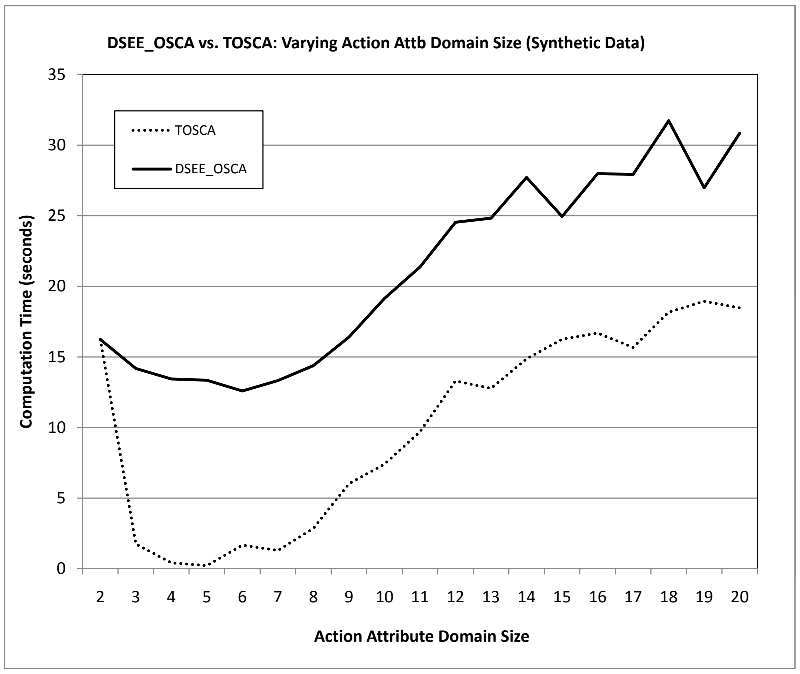

Figure 7.12: Performance of DSEE_OSCA versus TOSCA in synthetic data experiments where the size of the domain of the action attributes is varied. The number of tuples was fixed at $8,000$, action attributes at 4, and state attributes at 3.

ered action attributes were then varied from sets of 6, 12, and 24 action attributes. Again, since the algorithms using the learned effect estimators could not scale past 20 tuples, they were not included in this experiment. In each run, a random subset of the appropriate size from Arizona's data was selected and then a random goal condition was generated based on the domains of the provided state variables. Both DSEE_OSCA and TOSCA were run on the data, keeping track of the running times, including the running time of building the trie for TOSCA. The results of these experiments are shown in Figure 7.13 and clearly show that TOSCA is faster than DSEE_OSCA for any given number of action attributes.

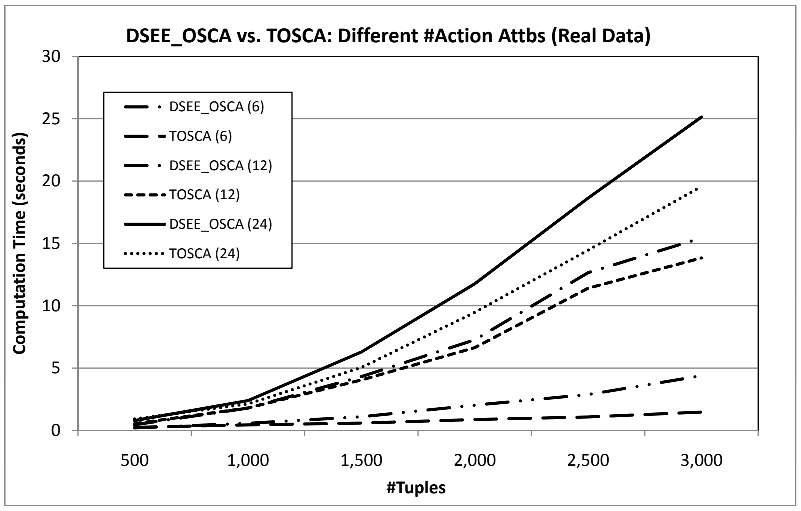

Figure 7.13: Performance of the DSEE_OSCA algorithm versus the TOSCA algorithm in real data experiments. This figure plots three comparisons in one, for 6, 12, and 24 action attributes each.

7.8 Conclusions

The Actionable State Change Attempts framework can leverage knowledge about an agent's behavior or forecasts about potential outcomes to devise possible counterstrategies and response policies. However, as shown in this chapter, determining optimal state change attempts is not an easy problem, since most interesting versions of the optimization task belong to complexity classes widely believed to be intractable. Preliminary experimental results both on synthetic and real-world data show that the TOSCA algorithm, which uses a trie data structure as an index on an event KB, is provably correct, faster than a basic solution to this problem, and tractable for policy analysis on reasonably sized inputs. However, there is still much work to be done in terms of providing deeper explanations and intuitive understandings of the dependencies between the actionable attributes and state attributes that may be implied by the optimal state change attempts .

The formalism and reasoning framework given in this chapter can allow a user to process the information from the various forecasting and modeling methods that have been presented earlier and determine a best response based on his own preferences. Integrating these two pieces and automating the process of policy development has many useful applications—specifically regarding issues of cultural-adversarial reasoning, international security, and social policies—and is addressed by the prototype applications discussed in Chapter 8.

Chapter 8

Prototype Applications and Case Studies

The work in this chapter has appeared in [107], [110], [73], and [74].

In the preceding chapters, several modeling and forecasting methods have been presented that allow for automated analysis of an agent's behavior. Such prediction tools have numerous applications, specifically in the realm of national security and intelligence analysis. The modern global political environment is growing increasingly complex, characterized by webs of interdependency, interaction, and conflict that are difficult to untangle. Technological expansion has led to an explosion in the information available, as well as the need for more sophisticated analysis methods for producing actionable intelligence and understanding various actors in international affairs. In many real-world instances, forecasting and modeling frameworks, such as those presented here, can be used as decision-support tools in conjunction with a human analyst's own experience and knowledge to produce more informed predictions of an agent's actual behavior. This augmented use of analysis technologies is referred to as *man-machine forecasting.*

Several applications for cultural adversarial reasoning and intelligence analysis have been developed that allow users to access the modeling, forecasting, and data analytic tools discussed in prior chapters. The SOMA Terror Organization Portal (STOP) [107] is a prototype decision-support system that also incorporates a social networking component, providing users with the ability to easily collaborate and communicate on particular topics of interest and share interesting analyses throughout the network. This system is described below in Section 8.1. In addition to the development of STOP, the modeling capabilities of the Stochastic Opponent Modeling Agents (SOMA) [103, 62, 61] framework (Chapter 2) have been utilized to analyze several real-world security situations. Section 8.2 contains an example of using SOMA to manually construct behavioral models of actors involved in the Afghan drug economy. In Section 8.3, two case studies are described where terrorism experts have used the automatically extracted SOMA rules provided in STOP to understand and manually forecast the behaviors of Hezbollah and Hamas. Development of decision-support tools such as STOP is an ongoing process, but the ultimate goal is incorporation of such technologies into policymaking and analysis for real-world international security situations; these case studies indicate how such tools can be used to help experts manage the complexity of conflict scenarios.

8.1 STOP: SOMA Terror Organization Portal

The Stochastic Opponent Modeling Agents (SOMA) framework was developed as a paradigm for reasoning about any agent, irrespective of whether it is a terror group, a social organization, a political party, a religious group, a militia, or an economic organization. In this section, a prototype decision-support system, the

SOMA Terror Organization Portal (STOP), is described that allows users to access behavioral models of terror organizations in the Middle East and North Africa that have been constructed using the SOMA framework (Chapter 2). In addition, STOP also provides analysts with access to the CONVEX [75], SitCAST [108], and CAPE [108] forecasting tools described in Chapters 3, 4, and 5, respectively. Because of the overwhelming amount of available raw data, such a system can allow national security or intelligence analysts to better distill the information into actionable policy decisions. In addition, STOP also allows users to browse the raw data directly, which can still prove valuable in comprehensive analyses.

Currently, the study of terror groups in the national security community is hampered by several technological and institutional problems:

(i) **Lack of timely, accurate data.** Much of the data about terror groups—and social science data in general—is collected manually. Due to the shortcomings of this method, any resulting data is often incomplete or out of date (often both). The Minorities at Risk Organizational Behavior (MAROB) [125, 46] dataset described in the preceding chapters is one example, tracking approximately 175 characteristics of 118 ethnopolitical organizations in the Middle East and North Africa from Morocco to Afghanistan. As mentioned in [115], such manual data collections are inherently incomplete because the human coders are only able to process limited numbers of articles in a small number of languages. Furthermore, such data is often coarse grained with variables that are often categorical rather than numerical. For instance, the number of deaths caused by a group might be classified as “none”, “low”, “medium” or “high” instead of giving a numerical estimate. In addition, these data sets are often out of date (Minorities at Risk Organizational Behavior coding is currently complete only until 2004), resulting in data that is non-actionable.

(ii) **Lack of behavioral models.** Analysts are often forced to devise behavioral models in a slow and painstaking manner, often implicitly using such concepts in their analyses without actually specifying the models employed. The APEX algorithm was presented in Chapter 2 for automatically extracting behavioral rules in the form of *ap*-rules. Applying this algorithm to the MAROB data has resulted in extracted rules for approximately 50 organizations, many of which have been included in STOP.

(iii) **Lack of a social network.** Analysts are often unaware of what other researchers have found useful. Information sharing has become a priority in national security decision making, and several technologies—A-Space [23], Intellipedia [23]—have been developed to help facilitate communication throughout the U.S. Intelligence and Defense Communities. However, analysts are often still in the dark about who else is looking at a certain group—either from the same, or different viewpoint as themselves. Additionally, any analytical tools, such as the Analyst's Notebook [56], are completely separate from communication software, and there is no computational mechanism seen to date that incorporates both data analytics and social networking. The SOMA Analyst NEtwork (SANE) provides a social networking framework within STOP, which allows analysts to browse data about the groups of interest and create comment threads and discussions about rules extracted by APEX or the results of forecasts by CONVEX, SitCAST, or CAPE.

(iv) **Lack of a forecast engine.** The SOMA Adversarial Forecast Engine (SAFE) uses a rich probabilistic foundation, with no unwarranted independence assumptions, to forecast the most probable sets of actions that a terror group might take in a given situation. Within the SOMA framework, SAFE implements the most probable world algorithms developed for *ap*-programs in [62] and [61] and described in Chapter 2. Using SAFE, decision makers can analyze the current situation and/or hypothesize new situations based on possible actions the U.S. might be contemplating. However, as these algorithms can be expensive in terms of computation time and space, the CONVEX, SitCAST, and CAPE forecast engines provide more real-time forecasting capabilities.

Through STOP, the issues described in (ii), (iii), and (iv) above can be addressed. Many of the problems relating to real-time data that are encompassed in problem (i) are addressed by the overarching Cultural Adversarial Reasoning Architecture (CARA) described in [117], specifically the OASYS [26] and TREX [5] components.

A prototype version of STOP has been implemented and launched in a web-accessible form. The current system allows users to browse SOMA models for 36 ethnopolitical groups from the MAROB data set, as well as utilize the CONVEX, SitCAST, and CAPE forecasters on any of the 118 Middle Eastern and North African groups in MAROB. SAFE is not operational in the current version of STOP, as the computation time required is more appropriate for a stand-alone system than a web-based one. Currently, the STOP system has users throughout the national security community, including the U.S. Air Force, Army, and Marines, the Departments of State and Defense, the Central Intelligence Agency (CIA), the Defense Intelligence Agency (DIA), the Intelligence Advanced Research Projects Activity (IARPA), the National Counterterrorism Center (NCTC), the Joint Warfare Analysis Center (JWAC), the House Armed Services Committee, and the National Security Agency (NSA).

8.2 Modeling the Drug Economy in Afghanistan

Over the past several decades, Afghanistan has been embroiled in numerous external and internal conflicts—the Soviet invasion in 1979, the struggle for control by rival groups when the Soviets withdrew in 1989, and the subsequent rise of the Islamic fundamentalist Taliban [27]. However, in spite of its tumultuous history, Afghanistan has largely resided on the periphery of international politics. Following the terrorist attacks of September 11, 2001, this remote and isolated country has become a major battleground and has come to the forefront of foreign policy decision-making.

The international community has placed a large emphasis on improving the stability of the country and the new government. High among the many challenges to achieving this goal is the Pashtun tribal culture that defines many aspects of Afghan society and its civil institutions. Enacting a unified policy in a society based on tribal divisions and customary laws is very difficult, especially when these customs and behaviors are only understood by experts or actual Afghan citizens. The necessity for constructing policy initiatives that work within the Pashtun culture, and of anticipating the reactions of members of this society, places the knowledge of such experts at a premium.

With a limited number of experts available, computational models of various agents involved in tribal groups or cross-sections of Afghan society—such as those agents engaged in the drug trade—can be used to make this information accessible to policy makers, helping them construct culturally relevant and viable policy solutions. Agent models can provide information on the types of behavior that can be expected in various scenarios, so decision makers can understand, for instance, how Afghan farmers may react to a policy of destroying opium crops.

In this section the SOMA system described in Chapter 2 is applied as a prototype solution for providing a model of cultural behaviors in Afghanistan. SOMA uses a probabilistic logic framework to represent the behaviors of an agent under certain conditions and several algorithms for reasoning within this framework to determine the most likely actions an agent will perform in a given scenario. In the following sections, Afghan society and the drug economy, which comprises an enormous portion of the country's GDP, are briefly described. Using SOMA, simple behavioral rules are constructed to demonstrate the framework's ability to model the most probable actions of these agents in dynamic situations.

8.2.1 SOMA Case Study: the Drug Trade in Afghanistan

This section focuses on the drug trade in the areas located throughout the Afghan-Pakistan border. This particular geographic region has many features of interest and a rich history explaining the current state of affairs. For instance, [52] states that:

> "Afghanistan's present borders were defined by imperial powers in the nineteenth century and successive Afghan rulers attempted to defend, strengthen, and redefine these borders in response to external aggression or internal pressures. Borders are 'political membranes' and markers of the success of the state-building enterprise. As [98] argues, borderlands are shadow societies, beyond the reach of the state, often with an 'insur-

> rectionary tradition'. In this ongoing 'conversation' between the state and borderlands, violent conflicts have been defining moments of change, shifting the balance of power back and forth between core and periphery. The contemporary political economy of Afghanistan is a product of this history..."

The reality of the situation in post-Taliban Afghanistan is that of warlords competing against each other, as well as with the central authority, to establish 'mini-states' [52]. A great portion of the country's economy is sustained by the cultivation of poppy, which is later used in the production of opium. Initially, most of the poppy fields were in the provinces of Helmand, Kandahar, Uruzgan, and Nangarhar, which are located in Pashtun belts in the south and east, and divided up among many warlords. Now, most cultivation occurs in several provinces in the South and West suffering from the greatest insecurity and violence. Farms tend to be small, but cultivation of poppies can yield about $5,000 per hectare per year, which is more than six times what the cultivation of wheat would yield [83].

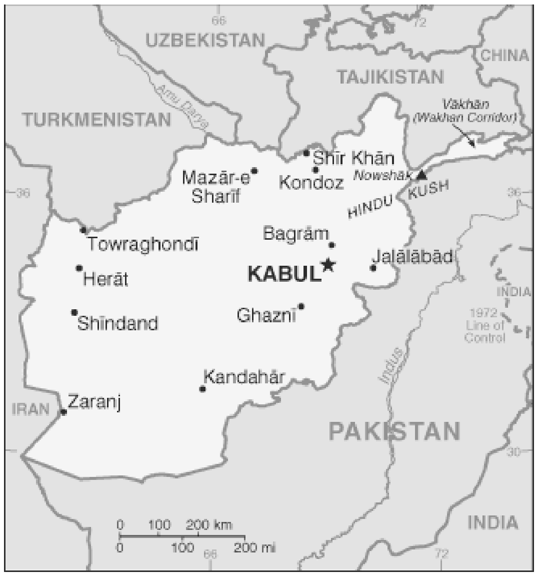

Figure 8.1: Afghanistan and the neighboring area [3].

Afghanistan is the largest producer of opium in the world, comprising 92% of global production. Drug money has been linked to Al-Qaeda terrorism and the Taliban insurgency, and is suspected to be used by Pakistani intelligence to support terrorist activities in Kashmir. The tribes in the border areas play a major role in this scheme, controlling much of the production and trafficking of opium.

There are many actors involved in the Afghan opium economy, among which can mentioned *farmers, itinerant laborers, members of regional militia, landowners, traffickers, warlords,* and *government figures*, not only local but also at the national level. The motives and methods used by each group vary based on their geographic location, economic circumstances, relationships with ethnic groups and external parties, and prevailing political conditions.

For this case study, a small set of actors was considered that are involved in the drug economy of a typical Afghan village and representative of the dominant power-structures in the region. The following characterization of actors was synthesized from [52, 59].

- *Malik.* Maliks are the intermediaries and representatives between the village community and central power/government and are in charge of solving communal disputes and maintaining communal property.

- *Khan.* In Arabic, the word “khan” denotes a ruler or central authority figure. Of course, Arabic is not spoken in Afghanistan—there, the word khan colloquially denotes a large feudal landowner who controls many resources in the community, provides jobs to laborers and land to sharecroppers, and may also arbitrate conflicts.

- *Ulema.* The Ulema shura is a group of religious leaders who lead prayers, give sermons, and have the power of moral judgment in the community. They are involved in resolving conflicts from the point of view of Shariah (Islamic law).
- *Member of Shura.* The Shura is a tribal council that meets only as problems arise in order to solve them. Such imminent problems range from personal disputes to maintenance of communal property.
- *Warlord.* Warlords are large, regionally based commanders—often remnants of past conflicts—who were able to utilize ties based on ethnicity or regional allegiances to build up support and control the area. They collect taxes, control borders and local resources, and produce and sell drugs, arms, etc.
- *Farmer.* Afghan farmers may own the land they work on or rent it from khans or warlords. Cultivating poppies has, in many places, become the main way for farmers to gain access to land or seasonal employment, allowing them to support themselves and the community. In addition, through the tribal salaam system of lending, farmers can gain access to credit; they are given an advance payment on a fixed amount of future production, and opium, with its reliable rates of return, is often favored by the money lenders.

While the micro-economy of each village or region has its own unique properties, poppy cultivation and the opium trade has an impact on the lives and behaviors of the above agents throughout much of the region. Many policies have been developed in an attempt to ameliorate the problems that come with the drug economy. In order to provide a more comprehensive illustration of the scenario, some of these methods are described here.

- *Destroy the poppy fields.* This policy has been tried extensively without much success. The market reacts by treating this as a drop in production (supply) with no corresponding drop in demand. Hence, prices go up.

- *Destroy the production laboratories.* Even though there is a relatively low number of labs in Afghanistan (numbering in the 100s), this method proved ineffective due to warnings of raids by corrupt officials. As in the case where poppy fields were destroyed, this only resulted in raised retail prices.

- *Invest money in basic infrastructure.* An apparent decline in poppy production in the Nihag valley in Pakistan was due to efforts to enhance irrigation, road construction, crops, livestock, and electrification. However, this tactic may push production into other, less developed or more insecure regions [15].

- *Legalize opium production.* This proposal would set up a legal agency to license, produce, and buy opium from farmers at a fixed price and then export it under UN licenses to pharmaceutical companies. Another option is to manufacture morphine/codeine in Afghanistan itself and then export it under UN licenses. The key expected pitfalls in this case include (i) resistance from distributors, and (ii) the difficulty of implementing such a system in Afghanistan in the presence of corruption in the licensing process.

- *Enlist help of fundamental Islamic clerics and Islamic law for enforcement.* Strict fundamentalist Islam forbids consumption of opium. Moreover, the lowest historic opium production levels in Afghanistan coincided with the rule of the Taliban who strongly suppressed the trade. Therefore, enlisting help (with compensation) from fundamentalist clerics to send an anti-opium message could be helpful. In fact, the most common reason cited by farmers for ceasing opium production is religious [83]. However, distributors may be cut out of the loop and can be expected to violently rebel against such a plan or influence the current Afghan administration to deny political support.

- *Take distributors out of the loop.* It is conceivable to make the current opium distributors the distributors of legally exported opium/morphine/codeine. This has many advantages, such as bringing them into the legal regulatory system. Regulatory control also ensures a fair deal for farmers vs. distributors. However, there is always the danger of distributors siphoning off opium to the illegal market at higher prices.

All of this knowledge of the scenario, including facts about how the world is organized, actions taken in the past, etc., can be used to build a model of the major players, which is useful both in understanding and forecasting future actions. The SOMA framework from Chapter 2 was used for such cultural and behavioral modeling of the Afghan drug economy.

SOMA Applied to the Afghan Drug Trade Domain

SOMA is a framework for reasoning with behavioral and cultural models and determining the most probable actions that an agent will take in a given situation. Users can select an agent model and state of the world and use the SOMA Adversarial Forecast Engine (SAFE) described in Section 8.1 to calculate the k most probable worlds (see Definition 4)—or sets of actions—that the agent might take.

SOMA models were developed by experts for several actors involved in the Afghan-Pakistan drug economy. Based on the dynamics of the drug trade, there are countless actions and situations that could be modeled; however, as these models were constructed manually through expert input, rather than automatically using the APEX algorithm (Figure 2.7), models were only created for a small portion of what each agent could possibly do. Below is a brief summary of the role the various actors play in the drug economy, as well as a description of some of the actions included in the model.

- *Malik, Member of Shura, Ulema.* In spite of the rise of opium production, the traditional civil and religious institutions in the villages often remain intact. Tensions do exist between the traditional social structure and the drug economy, as most of the new wealth is controlled by young men in the village who work for warlords or khans. Additionally, powerful warlords can threaten or bribe village officials into supporting the drug trade or local militias.

  Sample Actions: support (warlord, regime, etc.), provide financial backing.

- *Khan.* Because of the relative financial advantage of farming poppies as opposed to other crops, khans have a large incentive to force their sharecroppers to be involved in the drug economy, often determining the price of land rent based on its expected yield of poppies.

  Sample Actions: switch from poppy cultivation to a legal crop, enforce poppy ban, provide financial backing.

- *Warlord.* Warlords and their militias have a great deal of power and control a large portion of the economy. Many of the taxes collected and the revenues obtained through the sale of opium go directly to funding these militias. Because the central government is still relatively weak in many regions, the warlords often have control over border posts and local military leaders, allowing them to successfully smuggle drugs across the Pakistani border.

Sample Actions: enforce poppy ban/ban poppy cultivation, purchase arms or other war materials, collect taxes.

- *Farmer.* Afghan farmers have grown to rely increasingly on the cultivation of poppies for their survival and livelihood. Because opium reliably brings greater revenue than legal crops, money lenders give preferential loans to farmers growing poppies. Even farmers who would like to comply with the anti-drug laws are often influenced or intimidated by the powerful khan whose land they farm or warlords who control the transport and trade of drugs; in many cases the cultivation of poppies is not actually a choice for the farmers.

  Sample Actions: cultivate poppy, switch from poppy to a legal crop.

Figure 8.2 presents a selection of the SOMA rules representing the behavior of several agents involved in the Afghan drug trade. To reiterate, the SOMA framework does *not* make any independence assumptions regarding the actions. In addition, the probability annotations in these rules were derived by analyses of qualitative (as opposed to quantitative) data and discussions with subject matter experts.

The remainder of this section walks through an example using SAFE to forecast the most probable worlds in the Afghan drug trade scenario. The SOMA framework can be used to infer what might happen in a real or hypothetical situation. By changing the existing state to reflect this scenario, SAFE can compute the most probable response that one or more of these groups might have to a given situation, which can be a valuable aid in identifying states of the world that may lead to desired responses from a given group.

Suppose for this example that the user has chosen a farmer as the agent model to reason about, and the state given in Figure 8.3 as the state of the world. When

Farmer

cultivate_poppy(X): [0.7,0.9] ← is_farmer(X) ∧ has_debt(X).
There is a high probability that a farmer will cultivate poppy if he is in debt.

cultivate_poppy(X): [0.8,1.0] ← is_farmer(X) ∧ land_opportunity(X).
There is a high probability that a farmer will cultivate poppy if this gives him a chance to have is own land.

Khan

switch_crop(X,poppy,Z) [0,0.1] ← is_khan(X) ∧ traditional_crop(C) ∧ cultivates(X,poppy) ∧ yields_greater_revenue(poppy,C).
With a low probability a khan will switch crops from poppy to a traditional crop if poppy yields more revenue than the other crop.

enforce_poppy_ban(X): [0.5,0.65] ← is_khan(X) ∧ supports(Y,poppy_ban) ∧ declared_unislamic(Y,opium) ∧ wants_good_relations(X,Y).
With moderate probability, a khan will enforce a poppy ban if a certain group or individual supports such a ban, opium is declared unIslamic, and he desires to have a good relationship with the supporter.

Ulema

support(X,Y): [0.80,0.95] ← is_ulema(X) ∧ is_warlord(Y) ∧ threatens(Y,X).
There is a high probability that an ulema will support a warlord who threatens him.

Warlord

∼ban_poppy_cultivation(X) ∧ ∼enforce_poppy_ban(X): [0.9,1.0] ← is_warlord(X) ∧ is_involved_in_drug_trade(X).
There is a high probability that a warlord will neither ban the cultivation of poppy nor enforce such a ban if he is himself involved in drug trade.

Figure 8.2: A sample set of SOMA rules for some of the classes of actors in the Afghan opium economy.

forecasting with SAFE, the user must decide how to balance the tradeoff between solution quality—the accuracy of the probability values assigned to the worlds—and running time of the computation. As the naive algorithm, which yields exact probability results for the k most probable worlds, can take prohibitive amounts of time even for small SOMA-programs, the user can choose from among the other equivalence class or heuristic algorithms presented in [62, 61] and described in Chapter 2. The user can also choose between solving for the lower bound probability of each world w_i ($\mathsf{low}(w_i)$), the upper bound ($\mathsf{up}(w_i)$), or some other value, such as the average of $\mathsf{low}(w_i)$ and $\mathsf{up}(w_i)$. Choosing to solve for the lower bound is the most conservative approach and gives the most guarantees, as the world has at least a probability of $\mathsf{low}(w_i)$. However, in practice this method often yields a probability of zero for every world; therefore, the upper bound may provide more useful information by returning a nonzero probability.

$$
\boxed{
\begin{array}{l}
s = \{is\_khan(khan1), \\
drought, \\
is\_farmer(farmer1), \\
has\_debt(farmer1), \\
land\_dispute(farmer1, warlord1), \\
is\_warlord(warlord1), \\
yields\_greater\_revenue(drug\_trade, legal\_business), \\
needs(farmer1, money), \\
increases\_quality\_of\_life(peace, farmer1), \\
is\_malik(malik1, village1), \\
solves\_conflicts\_satisfactorily(malik1), \\
helps\_in(malik1, community\_affairs), \\
respects(malik1, sharia), \\
consults(malik1, ulema1), \\
is\_ulema(ulema1), \\
lives(farmer1, village1)\}
\end{array}
}
$$

Figure 8.3: State of the world used by the SOMA framework in calculating the most probable world for a farmer group (farmer1) in a village (village1) containing a warlord (warlord1), malik (malik1), and ulema (ulema1).

Here the 5 most probable worlds are computed for a farmer using the binary heuristic algorithm (Section 2.2.1) and solving for the upper bound probability of the worlds. Figure 8.4 contains the resulting worlds found by SAFE and their respective upper bound probabilities. According to the model, the most likely thing a farmer

$WORLD1 = \{support(farmer1, malik1),$
$arrange\_salaam(farmer1, khan1),$
$support(farmer1, peace),$
$join\_jaba(farmer1),$
$get\_involved\_in\_drug\_trade(farmer1),$
$cultivate\_poppy(farmer1)\}$
Probability: 0.8

$WORLD2 = \{fight(farmer1, land, warlord1),$
$support(farmer1, malik1),$
$arrange\_salaam(farmer1, khan1),$
$get\_involved\_in\_drug\_trade(farmer1)\}$
Probability: 0.25

$WORLD3 = \{fight(farmer1, land, warlord1),$
$support(farmer1, malik1),$
$arrange\_salaam(farmer1, khan1),$
$join\_jaba(farmer1),$
$get\_involved\_in\_drug\_trade(farmer1)\}$
Probability: 0.25

$WORLD4 = \{support(farmer1, malik1),$
$arrange\_salaam(farmer1, khan1),$
$join\_jaba(farmer1)\}$
Probability: 0.225

$WORLD5 = \{fight(farmer1, land, warlord1),$
$arrange\_salaam(farmer1, khan1),$
$support(farmer1, peace)\}$
Probability: 0.225

Figure 8.4: The 5 most probable worlds found by SAFE for a farmer in the state of the world given in Figure 8.3

will do under this state of the world is to support a local malik, arrange a salaam (or loan) with a khan in the village, join a jaba (or local militia), and be involved in the drug trade by cultivating poppies; the farmer will take these actions with a probability of about 80%. Based on the state of the world, this result reflects the power of the khans to influence the behavior of village citizens and control the economy. Even though the farmer may support the malik, which indicates a willingness to support, or at least work with, the central authority, he still becomes involved in the drug trade because of pressure from the landowners and the fact that poppy cultivation yields greater revenues than legal crops. Because the farmer is joining a jaba, this also demonstrates the power that extra-governmental forces have on the villagers, as they find it more advantageous to support local warlords.

While the results obtained by the SOMA rules and SAFE may be somewhat simplistic, they indicate the potential for a tool that will provide policymakers with valuable insights into the behavioral patterns of a cultural group. For instance, in this example the models indicate that neither farmers, warlords, nor khans would be likely to comply with a government ban on poppy cultivation if this interferes with their revenues or their power in the region. The study of computational models to understand cultural behaviors is rather new [117] and has a long way to go. However, decision makers could use results such as these as incentive to develop alternate policies that might be more effective, and test these scenarios using the SOMA framework.

8.3 STOP in Action: Case Studies with Hezbollah and Hamas

The SOMA Terror Organization Portal (STOP) currently contains behavioral models of 36 terrorist organizations in the Middle East and North Africa. These models are constructed using the SOMA framework (Chapter 2) and are automatically extracted from the Minorities at Risk Organizational Behavior (MAROB) [125, 46] dataset using the APEX algorithm given in Figure 2.7. Among the modeled organizations are Hezbollah, a well-known terrorist organization based in Lebanon, and

Hamas, a major Palestinian terrorist organization that has achieved international standing. In this section, interesting results are described that expert terrorism analysts have uncovered by examining the Hezbollah and Hamas models in STOP.

8.3.1 Some Results about Hezbollah's Behavior

Prior to 9/11 Hezbollah was the terrorist organization that had killed the most Americans, carrying out the massive suicide bombings of the U.S. Marine Barracks and U.S. Embassy in Beirut in the early 1980s. Hezbollah also orchestrated a campaign kidnapping Westerners in Lebanon that triggered political crises in the United States and France. The group's terror attacks have, however, extended far beyond the Middle East to Europe and Latin America.

Ideologically, Hezbollah expounds a radical version of Shia Islam that seeks violent confrontation with perceived enemies of Islam, such as the United States and Israel. Hezbollah relies on the support of Lebanon's Shia community and its Iranian and Syrian sponsors, and consequently its actions are shaped by these factors.

Hezbollah is also a multi-faceted organization that engages in a range of activities to further its cause. It participates in Lebanese elections and runs businesses and social services. It maintains a guerilla force that fought a multi-year insurgency against Israel in South Lebanon and conducted platoon and company level operations against Israel during the summer of 2006. Internationally, it provides training to Islamist terrorists, including al-Qaeda and Palestinian terrorist groups, and has links to the Lebanese Shia diaspora, which has a presence on every continent [72].

Hezbollah's combination of formidable capabilities, radical ideology, and international reach makes developing systems to better understand their operations and, if possible, predict them an important priority. More than 14,000 SOMA rules about Hezbollah's behavior were extracted automatically using the APEX algorithm. Here, some core results about Hezbollah are presented that analysts derived using the SOMA behavioral model. SOMA provided probabilities for four different Hezbollah actions: armed attacks, targeting domestic security forces, kidnappings, and transnational attacks. Due to space constraints, this section focuses on rules regarding kidnappings and transnational attacks.

According to analysts, the central condition given in the SOMA rules for the probabilities of kidnapping and for committing transnational attacks is Hezbollah's relationship to Lebanese politics. From 1974 until 1992 Lebanon did not hold elections because of an on-going civil war. Prior to 1992 Hezbollah could not participate in Lebanese elections and did not attempt to represent its interests to Lebanese officials. In 1992, however, Hezbollah had a strategic shift in its relationship with the traditional Lebanese power structures and began to represent its interests to Lebanese officials by participating in elections. Prior to this point, kidnapping was a primary tactic used by Hezbollah to gain stature. With the end of the Lebanese Civil War and Hezbollah's entry into Lebanese politics, the likelihood of kidnapping dropped substantially while the likelihood of committing transnational attacks increased dramatically.

Table 8.1 contains a sample of the conditions and probabilities extracted for the kidnapping action; the *Probability* column is the likelihood of Hezbollah engaging in kidnapping, given that the factors in the *Conditions* column are true. The conditions relating to increased probabilities of kidnapping reflect Hezbollah's capabilities—receiving military support or possessing a standing military wing—or opportunities, such as inter-organizational conflict. Hezbollah and its rival Amal both conducted

Conditions	**Probability**
Does not advocate democracy & solicits external support	.53
No foreign state political support & major inter-organizational conflict	.53
Solicits external support & does not advocate democracy & no foreign state political support	.66
Major inter-organizational conflict & no foreign political support & (foreign state provided non-violent military support OR standing military wing)	.66
Soliciting external support is a minor strategy & (electoral politics is not a strategy OR does not advocate democracy)	.83

Table 8.1: Conditions and probabilities for Hezbollah performing kidnappings

kidnappings as part of their struggle for primacy among the Lebanese Shia community.

The strongest condition linked to a Hezbollah kidnapping campaign is soliciting external support. In the Middle East, kidnapping campaigns against the West and Israel are useful for raising an organization's profile, thereby making it a more attractive candidate for support. Kidnapping also creates bargaining chips, as Hezbollah can either attempt to extract support from the hostages' nation of origin or give potential supporters the opportunity to act as an interlocutor. During the Lebanese Civil War, when Hezbollah's efforts to obtain external support were greater, it appears that they were more likely to curtail their kidnapping activity—possibly in response to pressures from potential supporters.

Table 8.2 illustrates some rules for Hezbollah engaging in transnational attacks. Once in Lebanese politics, transnational attacks became the more attractive strategy. Terrorist attacks outside Lebanon could be denied and did not have a substantial impact on Lebanon itself. Since Lebanon does not have relations with Israel, and many Lebanese resent Israel's long-standing security zone in the south

Conditions	Probability
Pro-democracy ideology	.52
Electoral politics is a minor strategy & no foreign political support	.55
Medium inter-organizational conflict	.58
Electoral politics is a minor strategy & no non-military support from the diaspora	.6
Electoral politics is a minor strategy & (medium rioting OR no foreign state political support)	.6
Electoral politics is a minor strategy	.635
Electoral politics is a minor strategy & medium inter-organizational conflict & no foreign state political support	.67
Electoral politics is a minor strategy & medium inter-organizational rioting	.67
Electoral politics is a minor strategy & medium inter-organizational conflict	.74

Table 8.2: Conditions and probabilities for Hezbollah performing transnational attacks

of Lebanon, Hezbollah's rocket attacks against Israel did not detract from their domestic political standing.

Two factors appear to have substantial impact on the likelihood that Hezbollah will engage in transnational attacks. The most crucial factor is whether or not they are involved in electoral politics as a minor strategy (i.e., they have candidates holding elected office but it is not an election year—that would be major strategy). The other is whether or not there are medium inter-organizational conflicts involving Hezbollah. The positive relationship between inter-organizational conflict and transnational attacks could reflect a rally round the flag phenomenon in which Hezbollah tries to best its local rivals by focusing on the common enemy. This phenomenon does not appear to apply to major inter-organizational conflicts, possibly because these conflicts cannot be defused as easily and require more attention from the leadership and more resources.

The correlation between minor involvement in electoral politics and transnational attacks highlights the tension between Hezbollah's ideology and practical need for public support. The decision to enter Lebanese politics was a contentious one, with the most strident Hezbollah militants opposed because they feared it would corrupt the organization and distract it from its primary role of confronting Islam's enemies [94]. To placate this faction, it is essential that Hezbollah maintain its aggressive stance against Israel, not only by fighting Israeli forces in Lebanon but also by launching attacks into Israel itself. However, the group usually refrains from these attacks during election years. The exception was 1996, when a Hezbollah rocket campaign provoked a particularly harsh Israeli bombardment in which more than a hundred Lebanese were killed and many more were left homeless. In the elections later that year, Hezbollah lost two seats in Lebanon's parliament. This reflects the tension between Hezbollahs core ideology of confronting Israel and their need not to agitate the many Lebanese who are frustrated that their country is being used as a leading front for the Arab-Israeli conflict.

8.3.2 Some Results About Hamas's Behavior

In this section, another case study using the STOP system is examined, this time looking at behavioral rules extracted about Hamas. Hamas, an Arabic acronym for Harakat al-Muqawma al-Islamiyya (Islamic Resistance Movement) is a Palestinian terrorist organization that has carried out sophisticated attacks on Israeli targets. Hamas is also a political organization that provides social services to the Palestinian people, has won elections, has an international presence among the Palestinian diaspora, and has links to states and like-minded organizations throughout the Muslim world. Long an important factor in the Israel-Palestine conflict, since winning the January 2007 Palestinian elections and taking de facto control of Gaza, Hamas has become a major player in Middle East politics.

Despite its overwhelming victory in the Palestinian elections, winning 76 of 132 seats in the legislature, most of the international community refuses to negotiate with Hamas because it will not recognize the existence of Israel. In June 2007, in a series of battles, Hamas took complete control of Gaza [127]. With effective control of a territory, growing military capabilities, and a strong reputation throughout the Arab world for its success in confronting Israel, Hamas has been a rising power in the region.

Using the same MAROB dataset [125, 46] as the Hezbollah study, SOMA rules were extracted from 18 years of data about Hamas's behavior. This section looks specifically at rules relating to the probability of Hamas carrying out suicide attacks and bombings. A basic paradigm that is useful for analyzing a terrorist group's activities is to examine two variables: (i) the group's level of motivation and (ii) its operational capability [50]. The SOMA rules indicate conditions that increase and decrease the probability of a group taking a particular action. These conditions will be examined according to whether they affect a group's capability for the action or its motivation. Some rules may not fit into either category, but may still provide crucial insight into an organization's worldview, intentions, and operations.

The tables below provide a summary of the SOMA rules extracted for various actions undertaken by Hamas. In the left-hand column are the conditions and in the right-hand column is the probability, given these conditions, that Hamas will perform the action. In the column containing the conditions a slash "/" represents a new rule (i.e., the conditions after a slash are not linked to the conditions before the slash in determining the probability of the action). For example, in the fifth row in Table 8.3 below there are two separate sets of conditions in which suicide attacks have a probability of .89—the first set is before the slash and the second set comes after the slash. Parentheses "()" indicate different conditions that, when combined with another condition, have the same probability of an action being carried out. For example, in Table 8.3 in the fourth row, either of the conditions within the parentheses, when combined with the condition before the parentheses, indicate a .91 probability of Hamas carrying out a suicide attack.

Much of Hamas's notoriety has come from its deadly suicide bombings, which have taken hundreds of lives since they adopted this tactic. Examining the SOMA rules in Table 8.3 gives some insight into how suicide attacks reflect the organization's strategies. One interesting correlation is between Hamas's provision of social services and its increased likelihood of launching suicide attacks. These welfare networks have been essential to Hamas's growth in popularity, but providing social services has also increased Hamas's capabilities for carrying out suicide bombings. The infrastructure for providing social services helps recruit members, provide for the families of suicide bombers, and in addition, the service facilities have reportedly been used as safe houses for operations [66].

Some conditions would obviously increase Hamas's motivation to launch suicide attacks, such as being attacked by the Israelis. Other conditions are not as clear. The strong correlation between receiving diaspora support and a high likelihood of suicide attacks may indicate that diaspora support increases Hamas's capabilities. It is also possible that Hamas attracts diaspora support because of its high-profile suicide attacks.

Conditions	**Probability**
Involved in electoral politics	1.0
Providing social services major strategy	.91
Providing social services major strategy & (Periodic lethal violence from state OR Clandestine)	.9
Providing social services is a major strategy & Representing interests to officials is not a strategy / Support from international NGO & No support from international governmental organization	.89
Provision of social services & (Periodic lethal violence from state OR Clandestine)	.83
Provision of social services & Representing interests to officials is not a strategy	.82
State use of lethal violence against organization & (Internal and external bases OR No support from international government organization) / Internal and external bases & Clandestine	.77
Electoral politics is not a strategy	.46
No diaspora support / Electoral politics is not a strategy & Clandestine	.42

Table 8.3: Conditions and probabilities for Hamas carrying out suicide attacks.

Hamas suicide attacks are almost certain in years that the group participates in the Palestinian electoral process, and substantially less likely in years that they are not participating. However, it is probably not accurate to conclude that participating in the electoral process leads to increased suicide bombings. The likelihood of suicide bombings appears to have risen as Hamas has expanded in capabilities, as represented by the establishment of external bases where it could receive training from its sponsors. But Hamas's high likelihood of carrying out suicide attacks in years that it participates in the Palestinian elections does raise questions about the proposition that participation in electoral processes will lead to the organization rejecting violence.

Conditions	**Probability**
Diaspora support / Provision of social services & Inter-organizational conflict / Soliciting external support is a major strategy & Electoral politics is not a strategy	1.0
Provision of social services & Electoral politics is not a strategy	.88
Provision of social services & Representing interests to officials is not a strategy	.82
Providing social services major strategy & State violence against org	.8
Provision of social services & (No support from international governmental organization OR Clandestine) / Representing interests to officials is not a strategy & (Internal and external bases OR Solicits external support	.75

Table 8.4: Conditions and probabilities for Hamas performing bombings.

Bombing refers to attacks with explosive devices that are not suicide attacks (for example, planted car bombs or IEDs). According to the rules in Table 8.4, the likelihood of Hamas carrying out bombings is greatest in the years when they are providing social services. As with suicide attacks, this may reflect how the social service networks augment Hamas's ability to undertake attacks by helping to recruit operatives and providing safe houses for planning operations.

There is a very high likelihood of Hamas conducting bombing attacks in years in which it is engaged in an inter-organizational conflict (most likely with rival Palestinian group Fatah) while also providing social services. The social service networks may play a role in expanding Hamas's capabilities to carry out such an attack, while the inter-organizational conflict may lead Hamas to carry out bombings in order to increase its standing among the Palestinian population.

A review of the SOMA generated rules on Hamas behavior has provided some potentially useful insights. According to the MAROB data, Hamas's efforts to target

Israeli civilians and Israeli attacks on Hamas are virtual constants. However, the tactics Hamas employs do change and the SOMA rules shed some light on possible factors influencing these decisions. Two primary factors appear to shape Hamas's decisions about what tactics to employ: (i) its capabilities, such as the presence of external bases or a social services network, and (ii) its position within Palestinian politics, such as being engaged in a violent confrontation with Fatah or whether or not they are participating in the formal Palestinian political process.

8.4 Conclusions

In this section, applications of the SOMA Terror Organization Portal (STOP)—a decision-support tool that national security analysts can use in order to understand terror threats worldwide—have been described. Not only does STOP include the data analytics and forecasting technologies described in previous chapters, such as SOMA, CONVEX, SitCAST, and CAPE, it also provides a valuable social networking capability that allows analysts to create and expand a network of experts on a given topic. STOP users can leverage this network so that different points of view can be incorporated into the analytic process.

Three case studies—the Afghan drug economy, Hezbollah, and Hamas—demonstrate how the forecasting and behavioral modeling techniques described in the previous chapters can be applied in real-world situations. These cases illustrate the types of man-machine forecasts that can be made by combining human analytical expertise with the data processing capabilities of these computational methods, allowing for better management of the complex dynamics in international security situations.

However, the STOP system and similar decision-support technologies still need to be formally evaluated, especially for incorporation into the U.S. national security community's analysis and decision making processes. In addition to data analytics and behavioral forecasting, the next logical step is to also include the policy forecasting capabilities offered by the Actionable State Change Attempts (Chapter 7) framework in STOP, providing users with a complete system for behavioral analysis and policy development.

Chapter 9

Discussion and Conclusions

In the preceding chapters, numerous modeling, forecasting, and analysis techniques have been presented for understanding and responding to the behavior of other agents. Due to the complexity of human behavior, the need for such methods is particularly salient in international security and conflict situations, where the use of violence is often the result of interrelated systems of cultural, economic, social, historical, and temporal dynamics. With inputs from such a large array of factors, scalable computational methods are crucial for understanding this environment and developing effective conflict management, counterterrorism, and development policies. Based on the methods developed here and other related research efforts in the literature, it seems that a complete system for behavioral analysis must contain four main components:

(i) **Scalable data analytics.** The first step in understanding agent behavior is the development of effective tools that can model, forecast, and explain the behavior of the target agents based on data. Such data can be behavioral and contextual, such as the Minorities at Risk Organizational Behavior [46, 125] data set, event data such as the Worldwide Incidents Tracking System [25] for terrorist attacks, or possibly other types of sources. The crucial aspect of such systems, however, is their ability to scale to real-world situations, which can be extraordinarily complex. The distributed algorithms given in Chapter 2 for reasoning in the Stochastic Opponent Modeling Agents (SOMA) [103, 62, 61] framework allow this robust probabilistic logic-based approach to forecast the most likely behaviors of an agent in a particular state of the world with greater efficiency, scalability, and accuracy. Rather than constructing an intermediate model, the CONVEX [75] and SitCAST [108] algorithms described in Chapters 3 and 4 are extremely efficient and scalable because they are applied directly to time series data to produce either a "what if" forecast for a given situation or a temporal forecast of what an agent will do at some time in the future. In addition, a complete model of behavior must capture when an agent will deviate from its statistically "normal" behavior. In Chapter 5, the CAPE [108] system was presented to address this issue of behavioral change. CAPE uses CONVEX and SitCAST as a foundation, but also incorporates a probabilistic model of behavioral changes to produce an overall more accurate prediction. All of these forecasting analytics are generalized by the axiomatic forecast operators given in Chapter 6.

(ii) **Countermeasure or response policy formulation.** Once the behavior of another agent can be understood, this knowledge can be leveraged to develop effective counterstrategies or policies to induce particular desired behaviors, such as a reduction in violence. The Actionable State Change Attempts framework described in Chapter 7 solves this problem by finding the optimal way of changing the state of the world, balancing cost and likelihood of success, to influence the behavior of another agent. These state change attempts correspond to identifying the best policy—for conflict management, counterterrorism, education development, etc.

(iii) **Decision-support tools.** The ultimate goal of the types of behavioral modeling, forecasting, and analysis techniques presented here is to help manage the complexity of dynamic real-world situations and aid in decision making. In Chapter 8, several prototype applications and case studies were presented where the behavioral modeling and forecasting methods presented here were applied to several security and conflict scenarios. SOMA was used to analyze the cultural, economic, and social dynamics of the Afghan drug economy, while the SOMA Terror Organization Portal (STOP) [107] is a web-based system that makes the above data analytics available to national security specialists studying terrorist group behaviors in the Middle East. This system was used in a detailed case study of Hezbollah and Hamas and a new study on Lashkar-e-Toiba is currently in progress.

(iv) **Technology integration.** Developing modeling frameworks and decision-support software is not the end of the story. Such technologies must be integrated into standard government processes, policy analysis, and conflict management procedures. Though not explicitly addressed here, issues of technology integration and information management in the U.S. Intelligence Community have been explored thoroughly in [109]. Bureaucratic politics, research and development policies, and training programs all have a large impact on how decision-support tools are actually utilized in the policy process.

While the international security domain is of particular interest for behavioral modeling and forecasting, the components outlined above apply to any domain where forecasting may be used to understand relationships in complex systems and make effective decisions. The methods that have been presented here are neither exclusive to conflict models nor even behavioral analysis, but are general enough to be applied to any data set for solving a range of potential forecasting problems.

9.1 Ongoing and Future Work

The behavioral modeling and forecasting techniques developed here have proved very effective in addressing some of the complexities of understanding agent behavior, specifically in international security and conflict situations, providing scalable predictive mechanisms and showing promise as the data analytic foundation of national security decision-support technologies. However, there is still much work to be done in all four components of the forecasting and analysis of agent behavior, especially in complex systems of human interaction.

Temporal-Probabilistic Abduction for Policy Analysis. Similar to the problem addressed by the Actionable State Change Attempts framework presented in Chapter 7, abductive reasoning can be used to identify ways to change the state of the world to induce specific desired actions on the part of an external agent. Beginning with a model of agent behavior, abduction can identify elements that can be added to this model, i.e., actions that can be taken by the user, to cause specific goal behaviors to be taken by the agent. Though abduction has been extensively studied [21, 40, 86, 41], there is no prior work that studies abduction in the context of both probabilities and time.

There has been important work on abduction in temporal logic [20, 33, 39, 42, 100, 60] and abduction under uncertainty [88, 87, 28], which are all devised under various independence assumptions. The only exceptions are [106, 104, 102], which perform abduction in possible worlds-based probabilistic logic systems where independence assumptions are not made.

A new approach [77] to abduction in the Annotated Probabilistic Temporal (APT) logic [99] is being developed, where no independence assumptions are made, and moreover, time is included in addition to probabilities. Triples of the form $\langle \Pi, H, g \rangle$ are considered where Π is an APT logic program, H is a set of formulas that can be added (i.e., the possible policy options), and g is a goal the user wishes to achieve (i.e., reduce the use of violence by 10%). The *Basic APT Abduction Problem* (BAAP) tries to find (if possible) a set $S \subseteq H$ such that $\Pi \cup S$ is consistent and entails the goal, while the *Minimal APT Abduction Problem* (MAAP) tries to find a "minimal" subset $S \subseteq H$ such that $\Pi \cup S$ is consistent and entails the goal. Such sets S correspond with the potential countermeasures that policy-makers could take to try to ensure the desired goal occurs at the desired time (with some probability). Preliminary algorithms have been developed to solve both the Basic and Minimal APT Abduction Problem that utilize a geometric interpretation of the problem—finding a policy requires "slicing" the convex polytope defined by linear constraints associated with the environment so it is entirely enclosed in the polytope defined by the goal conditions.

As a policy analysis mechanism, the use of APT logic will allow for the encapsulation of more information—specifically temporal relationships—regarding the policy implementation and provide users with a more thorough understanding of the results. However, this initial framework is only preliminary and scalable implementations of reasoning algorithms for BAAP and MAAP are necessary. Even under various assumptions, the complexity of checking for the existence of a solution was shown to be Σ_2^P-complete for both BAAP and MAAP, and the complexity of checking whether a given set is a solution is D^P-complete for BAAP and Π_2^P-complete for MAAP [77]. Randomized and sampling-based algorithms are in development in the spirit of [62], which scale well to very large numbers of possible worlds. Furthermore, distributed approaches such as those described in Chapter 2 may improve the efficiency and quality of the results, ensuring that the policy analysis is reliable. Such scalability will allow for the application of APT Abduction to real-world situations, such as conflict management, counterterrorism, or international development.

Stochastic Contingency Planning. The modeling and forecasting methods presented here have all addressed the issue of finding the most likely behavior that an agent will take in a given situation or at a particular time, including times when it is likely that they will change their behavior. However, in many decision-making environments, it is crucial to also have an understanding of behaviors or events that may be low probability but extremely high impact, such as the use of nuclear weapons, a major rebellion, or a very severe terrorist attack. Because of the complexity of agent behavior and the situations in which it occurs, automated techniques are necessary for forecasting and analysis of these contingency situations.

Currently, the Stochastic Opponent Modeling Agents (SOMA) [103, 62, 61] framework consists of several reasoning algorithms described in Chapter 2 to find the most probable world (i.e., most probable set of actions) that an agent will take in a given situation. However, this same probabilistic logic formalism for representing agent behavior can be used with different valuations of worlds, focusing on those that might correspond with high impact, low probability events. In [105], particular actions of interest are specified and only worlds containing those actions are included in the most probable world computation; the excluded possible worlds are accounted for by a refinement algorithm that modifies the set of linear constraints associated with a probabilistic logic program. While, this method still operates as a “what if” forecaster for a particular situation, iterative applications to a variety of scenarios can help identify situations where catastrophic events are more likely.

In addition, the most probable world computations given in Chapter 2 are capable only of returning a lower and upper bound for the probability of a world, which often range from very close to zero to very close to one, possibly distorting the apparent likelihood of severe events. [19] provides a new query answering semantics for probabilistic logic, using a series of volume ratios in a convex polytope to return a histogram of the probability mass, rather than just the bounds. Several new semantics are currently being formalized, such as finding an expected value of the probability or using geometric concepts such as the centroid or insphere of a polytope to identify particular probability distributions over the possible worlds. In addition, distributed algorithms are being developed to make this technique scalable for real-world behavioral analysis. By providing more information to the user about the probability of each world, these methods may more clearly represent the actual probability of large events. It would also be interesting to explore how the upper and lower probability bounds in SOMA are related to the frequency-severity power law statistics discovered for various conflict situations such as wars [95] and terrorist attacks [32, 29, 31].

Another approach could involve the identification of possible “critical states” where the probability of an event of interest crosses above some threshold in a sufficiently high number of future possible scenarios. Temporal-probabilistic abduction techniques as described above may be used to indicate the best time to act and what policies to employ to influence the chain of events and avoid these critical states. Such contingency analysis and planning tools can provide decision-support to policymakers that must be prepared for a major event.

Expanded Application Domains. In addition to the above projects, the methods presented here can be further generalized and expanded for application in other behavioral domains related to security or conflict issues. Prototype applications have been developed that apply these data analytic techniques to education logistics and development planning in Nigeria, highlighting the need for decision-support technologies in international development as well as security. Such technologies have potential not only as tools for policymakers, but also as a means of facilitating democracy or civic engagement through e-government applications. Similar new domains might also involve behavioral modeling and forecasting in disaster recovery or post-conflict reconstruction efforts.

The cyber landscape is also proving to be a challenging new realm for international security, as the offense-dominated nature of attacks are poorly addressed by existing policies. Behavioral models and forecasts can potentially find similarities in cyber strategies and identify categories of agents rather than specific actors. Such models can allow security experts to reason about effective responses or targeted deterrents to categories of cyber attacks, identify leading indicators to forecast strategies already underway, and provide empirical support for a new policy framework for cyber conflict.

Again, all of these forecasting methods are extremely flexible and can be used for analysis in any domain. However, the complexities and dependencies of the international arena are only increasing, presenting ample new opportunities for automated modeling and forecasting techniques to help decision-makers better understand and respond to the challenges of managing global conflict and contributing to a more peaceful and secure world.

Bibliography

[1] Sat4j: Bringing the power of sat technology to the java platform.

[2] Bruce Abramson and Anthony Finizza. Probabilistic forecasts from probabilistic models: A case study in the oil market. *International Journal of Forecasting*, 1:63–72, 1995.

[3] The World Factbook: Afghanistan. “afghanistan” [map] visual scale, 2007. http://www.cia.gov/cia/publications/factbook/geos/ af.html.

[4] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[5] M. Albanese and V.S. Subrahmanian. T-rex: A system for automated cultural information extraction. In *Proc. 2007 Intl. Conf. on Computational Cultural Dynamics*, pages 2–8. AAAI Press, August 2007.

[6] M. Antunes, M. A. Amaral Turkman, and K. F. Turkman. A bayesian approach to event prediction. *Journal of Time Series Analysis*, 24(6):631–646, November 2003.

[7] David Applegate, William Cook, Sanjeeb Dash, and Monika Mevenkamp. Qsopt linear programming solver: Java library, 2003.

[8] Tsz-Chiu Au and Dana Nau. Is it accidental or intentional? a symbolic approach to the noisy iterated prisoner’s dilemma. In *The Iterated Prisoners’ Dilemma: 20 Years on*, pages 231–262. World Scientific, 2007.

[9] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 1984.

[10] Chris L. Baker, Joshua B. Tenenbaum, and Rebecca R. Saxe. Bayesian models of human action understanding. In *Advances in Neural Information Processing Systems 18*, pages 99–106. MIT Press, 2006.

[11] Chitta Baral and Le chi Tuan. Reasoning about actions in a probabilistic setting. In *AAAI 2002*, pages 507–512. AAAI Press, 2002.

[12] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

[13] L. Mark Berliner and Mark Berliner. Monte carlo based ensemble forecasting. *Statistics and Computing*, 11:269–275, 2001.

[14] Yudhijit Bhattacharjee. Pentagon asks academics for help in understanding its enemies. *Science Magazine*, 316(5824):534–535, 2007.

[15] Christopher M. Blanchard. Afghanistan: Narcotics and us policy. *CRS Report for Congress*, 2005.

[16] J. Bond, V. Petroff, S. O'Brien, and D. Bond. Forecasting turmoil in indonesia: An application of hidden markov models. In *International Studies Association Convention, Montreal*, pages 17–21, March 2004.

[17] Craig Boutilier, Richard Dearden, and Mosiés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1–2):49–107, 2000.

[18] B. Bowerman, R. O'Connell, and A. Koehler. *Forecasting, Time Series and Regression*. Southwestern College Publishers, fourth edition, 2004.

[19] Matthias Broecheler, Gerardo I. Simari, , and V.S. Subrahmanian. Using histograms to better answer queries to probabilistic logic programs. In *25th International Conference on Logic Programming (ICLP 2009)*, July 14-17, 2009.

[20] Vittorio Brusoni, Luca Console, Paolo Terenziani, and Daniele Theseider Dupré. An efficient algorithm for temporal abduction. *AI*IA*, pages 195–206, 1997.

[21] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson. The computational complexity of abduction. *Artificial Intelligence*, 49((1–3)):25–60, 1991.

[22] Castelfranchi C. Modelling social action for ai agents. *Artificial Intelligence*, N 103:pp.157–182, 1998.

[23] Massimo Calabresi. Wikipedia for spies: The cia discovers web 2.0. *TIME*, April 8, 2009.

[24] David Carmel and Shaul Markovitch. Incorporating opponent models into adversary search. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 120–125. AAAI, 1996.

[25] National Counterterrorism Center. Worldwide incidents tracking system, 2011.

[26] C. Cesarano, B. Dorr, A. Picariello, D. Reforgiato, A. Sagoff, and V.S. Subrahmanian. Opinion analysis in document databases. In *Proc. AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs, Stanford, CA*, March 2006.

[27] Sarah Chayes. *The Punishment of Virtue: Inside Afghanistan After the Taliban.* The Penguin Press, New York, NY, USA, 2006.

[28] Henning Christiansen. Implementing probabilistic abductive logic programming with constraint handling rules. In *Constraint Handling Rules*, pages 85–118. 2008.

[29] Aaron Clauset and Kristian Skrede Gleditsch. The developmental dynamics of terrorist organizations (pre-print), 2009.

[30] Aaron Clauset, Lindsay Heger, Maxwell Young, and Kristian Skrede Gleditsch. The strategic calculus of terrorism: Substitution and competition in the israel-palestine conflict. *Cooperation and Conflict*, 45(1):6–33, April 9, 2010.

[31] Aaron Clauset and Frederik W. Wiegel. A generalized aggregation-disintegration model for the frequency of severe terrorist attacks. *Journal of Conflict Resolution*, 54(1):179–197, 2010.

[32] Aaron Clauset, Maxwell Young, and Kristian Skrede Gleditsch. On the frequency of severe terrorist events. *Journal of Conflict Resolution*, 51(1):58–87, February 2007.

[33] T. Console, P. Terenziani, and D.T. Dupré. Local reasoning and knowledge compilation for temporal abduction. *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1230–1248, 2002.

[34] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[35] Paul Dagum, Adam Galper, Eric Horvitz, and Adam Seiver. Uncertain reasoning and forecasting. *International Journal of Forecasting*, 11:73–87, 1995.

[36] J.L. Davies and T.R. Gurr. *Preventive Measures: Building Risk Assessment and Crisis Early Warning Systems*. Rowman and Littlefield, 1998.

[37] Thomas Dean, Robert Givan, and Sonia Leach. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In Dan Geiger and Prakash Pundalik Shenoy, editors, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 124–131, San Francisco, August 1–3 1997. Morgan Kaufmann Publishers.

[38] James Delgrande, Aaron Hunter, and Torsten Schaub. Coba: A consistency-based belief revision system. In Sergio Flesca, Sergio Greco, Giovambattista Ianni, and Nicola Leone, editors, *Logics in Artificial Intelligence*, volume 2424 of *Lecture Notes in Computer Science*, pages 509–512. Springer Berlin / Hei-

delberg, 2002.

[39] Marc Denecker, Lode Missiaen, and Maurice Bruynooghe. Temporal reasoning with abductive event calculus. In *In Proc. of the European Conference on Artificial Intelligence*, pages 384–388. John Wiley & Sons, 1992.

[40] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):3–42, 1995.

[41] Thomas Eiter, Georg Gottlob, and Nicola Leone. Semantics and complexity of abduction from default theories. *Artificial Intelligence*, 90:90–1, 1997.

[42] U. Endress, P. Mancarella, F. Sadri, G. Terreni, and F. Toni. The ciff proof procedure for abductive logic programming with constraints. *Logics in Artificial Intelligence/Lecture Notes in Computer Science*, 3229:31–43, 2004.

[43] Ronald Fagin and Joseph Y. Halpern. Uncertainty, belief, and probability. *Computational Intelligence*, 7(3), August 1991.

[44] Ronald Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87(1/2):78–128, 1990.

[45] Michael Thomas Flanagan. Flanagan java scientific library.

[46] Center for International Development and Conflict Management. Minorities at risk organizational behavior dataset, minorities at risk project, 2008. Retrieved from http://www.cidcm.umd.edu/mar.

[47] Edward Fredkin. Trie memory. *Communications of the ACM*, 3(9):490–499, 1960.

[48] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[49] Alan M. Frisch and Peter Haddawy. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122, 1994.

[50] B. Ganor. The counter-terrorism puzzle: A guide for decision makers. *Transaction*, page 30, 2005.

[51] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 1 edition, April 2008.

[52] Jonathan Goodhand. Frontiers and wars: the opium economy in afghanistan. *Journal of Agrarian Change*, 5(2):191–216, 2005.

[53] T. Hailperin. Probability logic. *Notre Dame Journal of Formal Logic*, 25 (3):198–212, 1984.

[54] A. C. Harvey. A unified view of statistical forecasting procedures. *International Journal of Forecasting*, 3(3):245–275, 1984.

[55] Michael C. Horsch and David Poole. A dynamic approach to probabilistic inference using bayesian networks. In *In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 155–161, 1990.

[56] i2. Analyst's notebook, 2010.

[57] Nicholas P. Jewell. *Statistics of Epidemiology*. Chapman & Hall/CRC, 2003.

[58] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1995.

[59] Palwasha L. Kakar. Fine-tuning the nsp: discussions of problems and solutions with facilitating partners. *Afgan women and the National Solidarity Programme*, 2005.

[60] A. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[61] S. Khuller, V. Martinez, D. Nau, G I Simari, A. Sliva, and V S Subrahmanian. Computing most probable worlds of action probabilistic logic programs: Scalable estimation for $10^{30,000}$ worlds. *Annals of Mathematics and Artificial Intelligence*, Nov 2007.

[62] S. Khuller, V. Martinez, D. Nau, G I Simari, A. Sliva, and V S Subrahmanian. Finding most probable worlds of probabilistic logic programs. In *Proc. 2007 International Conference on Scalable Uncertainty Management*, volume 4772, pages 45–59. Springer Verlag Lecture Notes in Computer Science, 2007.

[63] Sarit Kraus, Penina Hoz-Weiss, Jonathan Wilkenfeld, David R. Andersen, and Amy Pate. Resolving crises through automated bilateral negotiations. *Artificial Intelligence*, 172(1):1–18, 2008.

[64] V. Kumar and D. S. Nau. A general branch-and-bound formulation for and/or graph and game tree search. *Search in Artificial Intelligence*, pages 91–130, 1988.

[65] John D. Lafferty, Andrew Mccallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 01)*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers.

[66] M. Levitt. Hamas from cradle to grave. *Middle East Quarterly*, 9(1), Winter 2004.

[67] Michael Lederman Littman. *Algorithms for Sequential Decision Making.* PhD thesis, Department of Computer Science, Brown University, Providence, RI, February 1996.

[68] J. W. Lloyd. *Foundations of Logic Programming, Second Edition.* Springer-Verlag, 1987.

[69] Thomas Lukasiewicz. Probabilistic logic programming. In *European Conference on Artificial Intelligence*, pages 388–392, 1998.

[70] Thomas Lukasiewicz and Gabriele Kern-Isberner. Probabilistic logic programming under maximum entropy. *Lecture Notes in Computer Science (Proc. ECSQARU-1999)*, 1638, 1999.

[71] Eren Manavoglu, Dmitry Pavlov, and C. Lee Giles. Probabilistic user behavior models. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 203, Washington, DC, USA, 2003. IEEE Computer Society.

[72] A. Mannes. *Profiles in Terror: The Guide to Middle East Terrorist Organizations*. Rowman & Littlefield Publishers, 2004.

[73] Aaron Mannes, Mary Michael, Amy Pate, Amy Sliva, V.S. Subrahmanian, and Jonathan Wilkenfeld. Stochastic opponent modeling agents: A case study with hezbollah. In *First International Workshop on Social Computing, Behavioral Modeling, and Prediction*, April 2008.

[74] Aaron Mannes, Amy Sliva, V.S. Subrahmanian, and Jonathan Wilkenfeld. Stochastic opponent modeling agents: A case study with hamas. In *Proceedings of the Second International Conference on Computational Cultural Dynamics (ICCCD)*, September 2008.

[75] Maria Vanina Martinez, Gerardo I. Simari, Amy Sliva, and VS Subrahmanian. Convex: Context vectors as a similarity-based paradigm for forecasting group behaviors. *IEEE Intelligent Systems*, July/August 2008.

[76] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[77] Cristian Molinaro, Amy Sliva, and V. S. Subrahmanian. Abduction in annotated probabilistic temporal logic. In *International Conference on Logic Programming*, 2011 (under review).

[78] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann, San Francisco, CA, USA, 2004.

[79] Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[80] Raymond T. Ng and Venkatramanan Siva Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. In Koichi Furukawa, editor, *Proceedings of ICLP '91*, pages 565–580. The MIT Press, 1991.

[81] Liem Ngo and Peter Haddawy. Probabilistic logic programming and bayesian networks. In *Asian Computing Science Conference*, pages 286–300, 1995.

[82] Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[83] United Nations Office on Drugs and Crime. *Afghan Opium Survey*, 2010.

[84] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*, chapter 3: Markov and Bayesian Networks: Two Graphical Representations of Probabilistic Knowledge, pages 77–131. Series In Representation And Reasoning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[85] Judea Pearl. Reasoning with cause and effect. *AI Mag.*, 23(1):95–111, 2002.

[86] Y. Peng and J. Reggia. *Abductive Inference Models for Diagnostic Problem Solving*. Springer-Verlag, 1990.

[87] David Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, 1993.

[88] David Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):7–56, 1997.

[89] David Poole. Logic, knowledge representation, and bayesian decision theory. In *CL '00: Proceedings of the First International Conference on Computational Logic*, pages 70–86, London, UK, 2000. Springer-Verlag.

[90] M. L. Puterman. *Markov decision processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, Inc., New York, 1994.

[91] Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[92] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[93] Adrian E. Raftery, David Madigan, and Jennifer A. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92:179–191, 1998.

[94] M. Ranstorp. The strategy and tactics of hizballahs current lebanonization process. *Mediterranean Politics*, 3(1):103–134, Summer 1998.

[95] Lewis F. Richardson. Variation of the frequency of fatal quarrels with magnitude. *Journal of the American Statistical Association*, 43(244):523–546, December 1948.

[96] R. Rojas. *Neural Networks: A Systematic Introduction.* Springer, 1996.

[97] P. Schrodt. Forecasting conflict in the balkans using hidden markov models. In *Proc. American Political Science Association meetings*, 31 August - 3 September 2000.

[98] James C. Scott. *The moral economy of the peasant: rebellion and subsistence in Southeast Asia.* Yale University Press, Boston, MA, USA, 1976.

[99] P. Shakarian, A. Parker, G. I. Simari, and V. S. Subrahmanian. Annotated probabilistic temporal logic. *ACM Transactions on Computational Logic*, 2010.

[100] M. Shanahan. An abductive event calculus. *Journal of Logic Programming*, 44(1–3):207–240, 2000.

[101] Barry G. Silverman, Michael Johns, Ransom Weaver, Kevin Brien, and Rachel Silverman. Human behavior models for game-theoretic agents: Case of crowd tipping. *Cognitive Science Quarterly*, 2:273–301, 2002.

[102] G. Simari, J. Dickerson, A. Sliva, and V.S. Subrahmanian. Parallel abductive query answering in probabilistic logic programs. *submitted to a technical journal*, Dec. 2010.

[103] Gerardo Simari, Amy Sliva, Dana Nau, and V. S. Subrahmanian. A stochastic language for modelling opponent agents. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 244–246, New York, NY, USA, 2006. ACM.

[104] Gerardo I. Simari, John P. Dickerson, and V. S. Subrahmanian. Cost-based query answering in action probabilistic logic programs. In *SUM*, pages 319–332, 2010.

[105] Gerardo I. Simari, Maria Vanina Martinez, and Amy Sliva andV.S. Subrahmanian. Focused most probable world computations in probabilistic logic programs. *Annals of Mathematics and Articial Intelligence*, Accepted-To Appear.

[106] Gerardo I. Simari and V. S. Subrahmanian. Abductive inference in probabilistic logic programs. In *ICLP (Technical Communications)*, pages 192–201, 2010.

[107] A. Sliva, V. S. Subrahmanian, V. Martinez, and G. I. Simari. The SOMA terror organization portal (STOP): Social network and analytic tools for the real-time analysis of terror groups. In *Proc. 2008 First Intl. Workshop on Social Computing, Behavioral Modeling and Prediction*, pages 9–18. Spring Verlag Lecture Notes in Computer Science, April 2008.

[108] A. Sliva, V.S. Subrahmanian, V. Martinez, and G.I. Simari. *Mathematical Methods in Counterterrorism*, chapter CAPE: Automatically predicting changes in group behavior, pages 247–263. Springer Verlag, 2009.

[109] Amy Sliva. Information management and technology integration in the intelligence community. Master's thesis, University of Maryland School of Public Policy, 2010.

[110] Amy Sliva, Vanina Martinez, Gerardo I. Simari., and VS Subrahmanian. Soma models of the behaviors of stakeholders in the afghan drug economy: A preliminary report. In *First International Conference on Computational Cultural Dynamics (ICCCD 2007)*. ACM Press, 2007.

[111] Eric Stallard. Product liability forecasting for asbestos-related personal injury claims: A multidisciplinary approach. In *National Institute on Aging Conference: Demography and Epidemiology: Frontiers in Population Health and Aging, Washington, D.C.*, 2001.

[112] State Education Data Center. http://www.schooldatadirect.org, 2008.

[113] Bernhard Von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14:220–246, 1996.

[114] D. Subrahmanian and R. Stoll. Events, patterns, and analysis. In *Programming for Peace: Computer-Aided Methods for International Conflict Resolution and Prevention*. Springer Verlag, 2006.

[115] V S Subrahmanian. Cultural modeling in real-time. *Science*, 317(5844):1509–1510, Sep. 14 2007.

[116] V S Subrahmanian, M. Albanese, V. Martinez, D. Nau, D. Reforgiato, G I Simari, A. Sliva, and J. Wilkenfeld. Cara: A cultural adversarial reasoning architecture. *IEEE Intelligent Systems*, 22(2):12–16, April 2007.

[117] V.S. Subrahmanian, M. Albanese, V. Martinez, D. Reforgiato, G. Simari, A. Sliva, O. Udrea, and J. Wilkenfeld. CARA: A Cultural Reasoning Architecture. *IEEE Intelligent Systems*, 22(2):12–16, 2007.

[118] S. J. Taylor. *Modelling Financial Time Series*. World Scientific Publishing Company, 2nd edition, 2007.

[119] John Tsitsiklis and Benjamin van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1/2/3):59–94, 1996.

[120] J.D. Ullman. *Principles of Database and Knowledge Base Systems*, volume 2. Computer Science Press, Maryland, 1989.

[121] Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I.* Computer Science Press, 1988.

[122] M.H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 4:37–53, 1986.

[123] Scott A. Wallace and John E. Laird. Behavior bounding: Toward effective comparisons of agents & humans. In *In Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 727–732, 2003.

[124] G. Webb, J. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, 2005.

[125] Jonathan Wilkenfeld, Victor Asal, Carter Johnson, Amy Pate, and Mary Michael. The use of violence by ethnopolitical organizations in the middle east. Technical report, National Consortium for the Study of Terrorism and Responses to Terrorism, February 2007.

[126] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco, 2005.

[127] M. Yaghi. Hamass authoritarian regime in gaza. *The Washington Institute for Near East Policy*, 2007.

[128] Hui Zou and Yuhong Yang. Combining time series models for forecasting. *International Journal of Forecasting*, 20:69–84, 2004.

Index