Answering Questions about Complex Events

Steve Sinha

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-175
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-175.html

December 19, 2008

Answering Questions about Complex Events

by

Steven Kumar Sinha

B.S. (Carnegie Mellon University) 2001

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Jerome Feldman, Chair
Professor Srini Narayanan
Professor Charles Fillmore
Professor Dan Klein

Fall 2008

Answering Questions about Complex Events

by

Steven Kumar Sinha

Abstract

Answering Questions about Complex Events

by

Steven Kumar Sinha

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jerome Feldman, Chair

Reasoning about event structure is a fundamental research problem in Artificial Intelligence. Event scenarios and procedures are inherently about change of state. To understand them and answer questions about them requires a means of describing, simulating and analyzing the underlying processes, taking into account preconditions and effects, the resources they produce and consume, and their interactions with each other. We propose a novel, comprehensive event schema that covers many of the parameters required and has explicit links to language through FrameNet. Based on the event schema, we have implemented a dynamic model of events capable of simulation and causal inference. We describe the results of applying this event reasoning platform to question answering and system diagnosis, providing responses to questions on justification, temporal projection, ability and 'what-if' hypotheticals, as well as complex problems in diagnosis of systems with incomplete knowledge.

Professor Jerome Feldman

Dissertation Committee Chair

Contents

List of Figures

Acknowledgments

My work

First and foremost, I wish to thank my advisor, Srini Narayanan. A Berkeley Ph.D. alum himself, Srini represents what the rest of us in this graduating batch should hope to be as researchers: broadly and deeply curious, highly capable, and ridiculously smart. I, for one, feel truly privileged that he volunteered his time to mentor and advise me, and I greatly appreciate that I can call him my friend.

I also wish to thank my other advisor, Jerry Feldman, for providing me an amazing environment for learning and conducting research. It is his research vision that gave genesis to our group, and I am grateful to him for the opportunity to be a part of the team. Jerry has always been there any time I needed him, and I thank him for that.

In addition, I would like to thank Chuck Fillmore and Dan Klein for providing their insights during my qualifiers and beyond. I especially appreciate Chuck and his FrameNet team for their frame semantics research, the fruits of which I used in many components of my own research efforts.

My families

In my time at Berkeley, I have come to have many extended families, each of which has played a significant part in making my experience here so enjoyable.

Of course, I wish to thank my direct family first. My parents have been unwavering in their support. I could always call them for love and understanding, and I very much appreciated that. To Terry and Vinod: I *think* I've turned out pretty well, and that is due in no small measure to your efforts. Thank you. I love you both.

And though my parents live far away, my other relatives keep me feeling connected: Joyce & Ed, Chris & Julie & Elisha, Sangeeta & Ken & Ryan & Alisha; and further away: Leela & Janine, and Dolly & Neeraj & Muski & Vijaya – thank you, too.

Day in day out, though, my housemates have been most present in my life. To: Yatish Patel, Sonesh Surana, and Melissa Ho, as well as their significant others and stream of friends, especially Jen Sloan and Divya Ramachandran: you're my family, too, and I love you dearly.

I also wish to acknowledge my computer science brethren from the early years. From my first office: Satrajit Chatterjee, my brother in arms; Sailesh Krishnamurthy, my older, wiser brother; and I have to add, Manikandan Narayanan, truly a most wonderful human being (seeing him always made me happy). And my CS social committee: Shyam Lakshmin! and Rachel Rubin, Hayley Iben, Pushkar Joshi, Kaushik Datta, Brian Milch, Mark Whitney, Nemanja Isailovic… and many others. Good times.

More recently, I have had the pleasure of the company of my ICSI colleagues and friends: Eva Mok, Joe Makin, Nancy Chang, Johno Bryant, Leon Barrett, Michael Ellsworth, Josef Ruppenhofer, Collin Baker… and the aforementioned Srini Narayanan, Jerry Feldman, and Chuck Fillmore. I especially want to thank Eva, who has been a significant source of friendship and support throughout my time at ICSI. And Joe Makin, for saying the darndest things that sparked the best conversations not to get work done to.

Also, from all my activities away from work, I thank my MOT China people (Jihong Sanderson and our whole team, most notably Andreas Schmidt), my GSC people (top of the list: Carolyn White), and most recently, my Obama campaign people (my brother Gregory H. Smith and his sister Polly Furth, Nick & Scott, the

Tompkins, the Holidays, the Gonders, Doug, and all of the amazing volunteers) for all of the wonderful times.

And I have to add in here (because they don't really fit into any category above): Ruth Keeling, who's awesome, and whose friendship, support, and tea I have greatly appreciated. And my dearest Mary Zimmerle, who knows she's awesome. Though I haven't seen her frequently enough in these recent years (in some part because of this dissertation work), her friendship is one I always cherish.

My support

ARDA / DTO / IARPA: no matter how many times it's changed its name, it has continued to support me throughout my education here at Berkeley, and I appreciate that. Putting a face to the organization: Steve Maiorano has been a wonderful friend to our group and to me, and I want to thank him, especially.

My home

Finally, I wish to thank and acknowledge the University of California, Berkeley. I have gotten so much out of my time here. In a sense, my Ph.D. work was just my day job. Here I have had the privilege of taking courses in business, political science, public policy, Mandarin, economics, rhetoric, linguistics, geography, music... (Thanks to all the incredible professors who have allowed me to sit in on their courses!) Beyond classes, I have had the opportunity to run student groups, I have had the flexibility to travel the world... I have lost track of the number of amazing experiences I have had while here. My years here have been some of the most personally enriching yet. It's somewhat depressing to think about life without the Berkeley community, but if the litany of friends above is any indication, I don't think it will be leaving me any time soon, no matter where I go.

1 Introduction

Imagine that you are an analyst for General Motors. The company has been facing stiff competition from foreign auto makers who are producing significant numbers of smaller, more fuel-efficient vehicles. The Boss comes in and asks: "Is our Lansing, Michigan plant *capable* of producing hybrid cars? I can give you a list of the inventory, the machine types, the employee list..." Later, he comes back and says, "If needed, we could invest $10 million dollars, could it be upgraded sufficiently?"

Information analysts are swamped with data, from which they are being asked to answer complex questions such as these. The questions are, implicitly or explicitly, about events and their interactions. Currently, there is no system, or for that matter, no framework from which a system can be built, to tackle the complex reasoning required to answer them. This task of reasoning about events is a fundamental research problem in Artificial Intelligence.

1.1 Solution Approach

Events unfold over time, changing state based on conditions in their environment. To reason about events requires a means of describing, simulating, and analyzing their underlying dynamic processes. For our dissertation work, we designed an event modeling and inference framework to answer event-related questions. The task required certain features of the framework. It must capture event interactions, including contingent causal and temporal relations between multiple events. It must be able to model contextual information about events. In addition, it must also be able to reason about evolving events with uncertain and partially ordered trajectories, be able to represent sequentiality and concurrency, and be able to support asynchronous control. To be effective, the framework should be general enough to represent events across multiple domains. All of this will facilitate answering questions[1] about how states evolve over time and how states and actions interact with one another.

1.1.1 Events & Event Questions

What do we mean by events, though? As we discuss in Chapter 2, there is a rich history of event reasoning work in AI employing many different representations of events. For example, representations for planning, for event recognition, and for event extraction, frequently include the event's name, its preconditions and effects, and perhaps its participants. Each of these features is necessary, but even together they are not sufficient for answering typical complex questions asked about events.

Specifically, in Section 3.2, we identify five prominent question types relating to events from which we wish to be able to infer answers when having only incomplete information.

- *Justification*: questions asking about a state that must be verified and justified

  e.g. "Did Toyota complete production of its next-generation Priuses?"
- *Temporal Projection / Prediction*: questions projected a state into the future

  e.g. "Will Ford start manufacturing fuel-cell cars in the next five years?"
- *Ability*: questions asking about whether an actor *can* (as opposed to *will*) commit an action or reach a state

  e.g. "Can Chrysler produce electric cars?"

[1] Note: in this work, we will be using the term "question" loosely, to cover queries including but not limited to those in natural language form.

- *'What-if' Hypothetical*: questions hypothesizing changes to a state, and then asking a *Justification*, *Temporal Projection*, or *Ability* question about that new state

  e.g. "If GM had access to cheap rechargeable batteries, could it manufacture plug-in hybrids for under $20,000?"
- *Hypothesis Disambiguation*: questions about the structure of an event when only limited segments of it are observable and multiple hypotheses are possible

  e.g. "Is Honda prototyping a new luxury car or a sports car?"

To represent the underlying events in questions like these, we designed an event schema that describes structure beyond preconditions and effects. As we detail in Section 3.3, the parameters of a basic event also include resources that the event produces and consumes, its data inputs and outputs, the duration of the event, and the time and place of the event, among others. Events are not isolated from each other, either, and their relationships with one another go beyond generic causation. Our schema distinguishes events that 'enable', 'disable', 'suspend', 'resume', 'abort', etc., other events. It also captures patterns of sub-events that compose a larger-scale event ('sequentially', 'concurrently', in a 'loop', as 'alternatives' to one another, etc.). In addition, the schema recognizes the different levels of granularity at which events can also be construed ("Trip to Europe" vs. "Catch a cab", "Drive to Airport", "Fly to Europe"...). This thorough representation of events provides valuable details about the dynamic characteristics of an event.

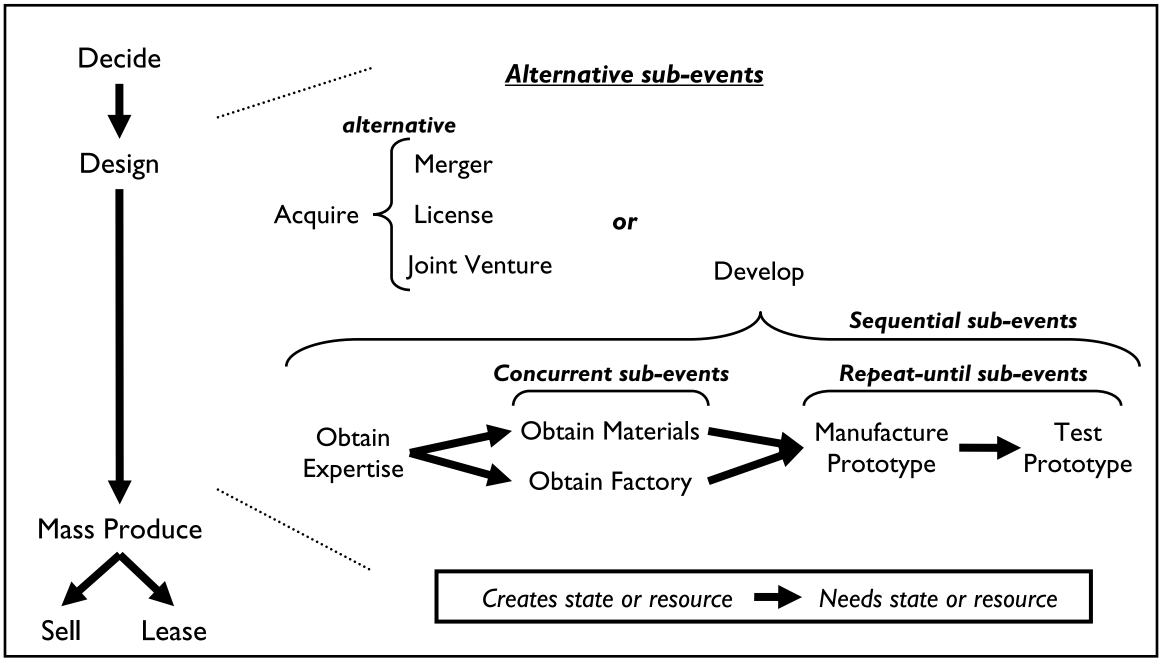


Figure 1-1: Event schema instance example – Car Design

In Figure 1-1, we show an example of an event schema instance, depicted in graphical form. The figure shows an expansion of our car design and production event, a complex scenario similar to one we used in an evaluation of our work. On the left side of the figure, there is a simplified high-level sequence of events: *Decide* (to design a new car), *Design* (the car), and *Mass Produce* (the car), followed by an implicit choice, between *Sell* (the car) or *Lease* (the car). *Design* can be broken down into alternative sub-events: *Acquire* or *Develop*, which in turn can be further broken down into sub-events organized using the various patterns shown. There are multiple alternative methods to *Acquire* a design, including *Merging* the company with another firm with a design, *Licensing* a design, or entering a *Joint Venture* with another company with a design. To *Develop* a design requires a sequential series of sub-events: *Obtain Expertise* (e.g. learn about new technologies); *Obtain Materials* and *Obtain Factory* (which can be done concurrently); and *Manufacture Prototype* and *Test*

Prototype (which are repeated until a successful test). The end result of both the *Acquire* and *Design* actions is a car design that can be mass produced.

1.1.2 Inferences with events

A static event schema instance, though, does not have the runtime semantics to simulate the execution of an event scenario unfolding over time, as is necessary for our task. We require a dynamic model of events. In Section 3.4, we discuss our use of X-nets, a formalism extending Generalized Stochastic Petri Nets (GSPNs: Bause and Kritzinger 1996), originally used by Srini Narayanan in his dissertation work on aspect and metaphor understanding (Narayanan 1997). X-nets can represent sequentiality, as well as concurrency and synchronization, alternatives, stochasticity, and asynchronous control (see 4.6.3 for an X-net representation of a scenario of similar structure to our Car Design event). In addition, X-nets can support a number of analysis routines that can test the potential of a state to evolve into a new state of interest given an event structure. At the end of Chapter 3, we describe procedures we developed to infer answers to our five target question types using these event analysis routines.

1.1.3 Language

To reason about complex events also requires an interface from our event models to data sources. Events, while independent of language themselves, are frequently discussed in natural language, yielding copious data in that form. We exploit semantic frames as an intermediate structure and interface between event descriptions in natural language and event models that produce inferences to answer questions. The link between events and semantic frames (specifically through FrameNet:

http://framenet.icsi.berkeley.edu) is made explicit in Chapter 4. The frame structure in language provides a bi-directional mapping from language to event models, enabling us to link information found in text about an event of interest to models that represent that event.

1.2 Applications

From the work we describe in Chapters 3 and 4, we have: 1) a means of describing events comprehensively; 2) a modeling system that captures the inherent dynamics of events; 3) a set of useful inference techniques we can use on the event models; and 4) specific procedures to answer our target question types.

To test this event modeling and reasoning framework, we applied it to three applications: Answer Selection, Question Answering, and Pathway Classification.

1.2.1 Answer Selection

Before applying our event modeling framework to a full-fledged NLP application, we decided to test it on a simpler problem. Answer selection is the process of choosing the best answer to a natural language question from a list of pre-chosen candidates. It is closely related to calculating the relevance of a returned answer for a search query. The difficulty in the task is assessing the relative relevancy of candidates that each contain the relations queried in the question asked.

For event-related questions, we hypothesized that an ontology of event models would improve the relevancy assessment. Specifically, we theorized that:

1) Keywords extracted from a question are related to one another and thus extracting relational information in a question and in a set of answer

candidates should enable higher precision measurement of the relevancy of the candidates; and

2) Analyzing these relations in the context of a relevant, expressive model of the event in question provides a valuable link between the information sought in the question and the information contained in a good answer.

For us, the Answer Selection task provided an initial test of our approach for modeling events. We describe the details of our experiments in Chapter 5.

1.2.2 Question Answering

Boosted by initial successes with Answer Selection, we applied out event reasoning framework to the full task of automated Question Answering (QA). The challenging goal in QA is to find or create focused answers to natural language questions. The system we designed used the preprocessed language about events in question to select a relevant event model. It then uses the model to select relevant evidence and simulation to infer missing information required for the answer. We implemented the algorithms of Chapter 3 to solve *Justification*, *Temporal Projection*, *Ability*, and *Hypothetical* questions. These question types had previously not been answerable with automated systems. We describe the details of this system and the results of three demonstrations in Chapter 6.

1.2.3 Pathway Classification

Our third system tackles a different task that can be addressed using event modeling and reasoning: the classification of dynamic system pathways. A pathway can be any set of activities, organized around temporal and causal structure, that serves a particular goal. Car production is one example. The biological pathway shown in

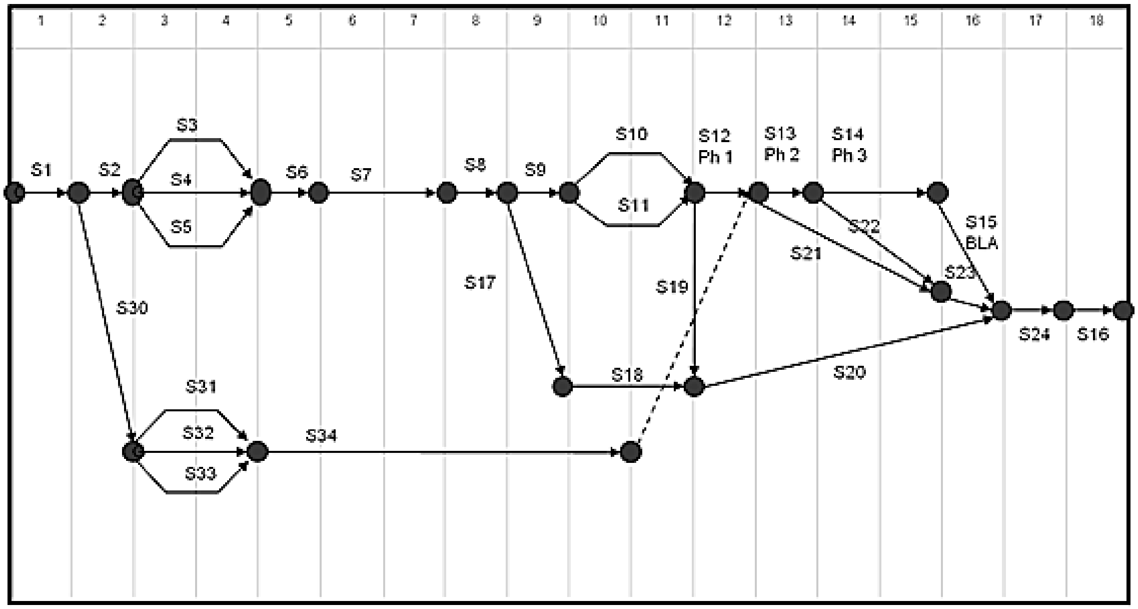


Figure 1-2: Pathway Classification example – Biological Pathway

Figure 1-2, is another. Here, each line connecting two dots represents a pathway segment (a particular activity, e.g. research, testing, production, etc.) unfolding over time (each vertical bar represents one month in the 18 month pathway). Based on the resources allocated by an external actor to the segment, the progress towards completion of the segment may be slowed or stopped.

To classify an unknown, partially-observable pathway requires comparing the behavior of the unknown pathway to the expected behavior of each pathway hypothesis. The task distinguishes itself because uncertainty exists about the actual structure and use of the events of interest. In addition, interventions on the pathway (called probes) may be required to elicit outputs that distinguish the hypotheses. We designed a system to test probes and classify pathways. Our simulation system is specially designed to handle data input that changes over time steps based on the incremental results of the simulated pathway. We describe the details of this system and three demonstrations of it, in Chapter 7.

1.3 Contributions

The work of this dissertation produced the following novel results:

- An expressive framework and relational schema for representing event structure, a dynamic model of pathway processes and complex events capable of simulation and inference, and a method to that uses semantic frames to translate from a natural language description of a particular event to a dynamic model form.
- A system to solve *Justification*, *Temporal Projection*, *Ability*, and *'What-if' Hypothetical* questions with incomplete information. With caveats, this is the first such system to demonstrate promise at answering these question types in the Question Answering domain in an evaluation.
- A system to model and simulate complex dynamic system pathways, and evaluate probes, in the service of answering the *Hypothesis Disambiguation* questions at the heart of Pathway Classification tasks.

In this work, we showed not only the benefits of our event framework approach, but also areas for future work needed to facilitate the development of more robust automated systems that answer questions about complex events.

1.4 Road map

Chapter 2 situates this work within the event modeling and Question Answering literature. Chapter 3 puts forward the objectives of the dissertation, specifying five event-related question types we wish to be able to answer. It then describes our comprehensive event schema, our dynamic event modeling representation (X-nets), and methods for converting descriptions to dynamic models. In addition, it lays out analysis routines that can be performed on X-nets and specific procedures we used to answer our target question types. Chapter 4 describes the glue that connects the theoretical work of Chapter 3 to the applications of Chapters 5 and 6. Specifically, Chapter 4 discusses our interface between language and event models, and provides solutions for selecting, instantiating, and filling models with data based on a question asked. Chapter 5 describes our first evaluation of our event modeling framework: Answer Selection. Chapter 6 follows with a description of our work on Question Answering. Chapter 7 details our final system designed for Pathway Classification. Finally, Chapter 8 reviews our conclusions and future work.

Readers interested in the theoretical underpinnings of our work are directed to Chapter 3. Those interested in applications of language and FrameNet will wish to also consult Chapter 4. Readers interested in our Question Answering system may concentrate on Chapters 3, 4, and 6. Readers exclusively interested in pathway modeling and probe design may read only Chapter 3 (concentrating on Section 3.6) and Chapter 7. Chapters 5, being a preliminary test of our work before tackling the full Question Answering task described in Chapter 6, may be skipped by most people.

2 Reasoning about Events

Before we dive into our model and its applications, we first describe our notion of events and situate it in the wider context of work within Artificial Intelligence (AI) and Natural Language Processing (NLP). There is a rich history of event reasoning work in AI starting from early work on situation calculus (McCarthy and Hayes 1969) and robot planning (STRIPS: Fikes and Nilsson 1971) to work in probabilistic models of planning, events and causality (Astrom 1965; Pearl 2001), employing many different representations of events. In recent years, there has been an increased interest in event recognition and extraction within Natural Language Processing (Bethard 2007; Chambers, Wang et al. 2007; Chambers and Jurafsky 2008). Work in planning frequently focuses on the preconditions and effects of events leaving implicit the cause/effect relationship between two events. Work on event extraction focuses on the predication features (the name of the event and the participants of the events). These features are all necessary, but not sufficient for answering the complex questions we care about, as we will describe in Chapter 3.

We start by situating our work in the context of event representation and reasoning in other fields of AI including logical AI and planning, Web services on the semantic web, and in Computational Linguistics.

2.1 Event Models in AI

Reasoning about actions and events is a significant subfield within AI. Our aim here is not to survey the field, but to outline significant approaches and developments that are relevant to our goal of connecting inference and reasoning about processes to answering questions about events.

We start with a brief description of the logical approach to reasoning about events, which identified important problems and provided a solid understanding of the issues involved. We then describe critical progress in probabilistic reasoning about actions that directly informed our modeling approach from the very beginning. We follow by describing recent efforts related to the semantic web that significantly enhance interoperability and reuse of models across different event and process descriptions, ontologies, and models.

2.1.1 Logical AI

In AI, formal approaches to model the ability to reason about changing environments have a long tradition. This research area was initiated by McCarthy (McCarthy 1958) who claimed that reasoning about actions plays a fundamental role in common sense. A decisive advantage of deductive approaches to reason about actions is the universality inherent in any logical framework. A purely logical axiomatization which is suitable for temporal prediction can just as well be employed to answer more general questions, such as postdiction (i.e. what can be concluded form the current state as to states in the past) and planning (i.e. how to act in order that the system evolves into a desired state). Deductive approaches have suffered from several well known problems, the most famous of which is the Frame problem pertains to compactly specifying aspects of world that are unchanged when an action is executed. A number of associated problems and proposed solutions can be found in (Lifschitz 1989). Additionally, formal theories of action and change often make the assumption that (Gelfond and Lifschitz 1993) environmental changes only occur as a result of some agent initiated action. This assumption is unduly restrictive and renders such systems incapable of dealing with complex environments where state transitions may occur independent of agent initiated action. In most complex environments, the world continues to evolve between agent actions. These problems combined with the need to generate real time behavior in complex and dynamic environments renders deliberative reasoning about the effects of low level action too expensive and thus impractical. Recognition of this fact in AI and Robotics has resulted in various proposals for the representation and use of compiled plans and behaviors (Nilsson 1984; Rosenschein 1985; Brooks 1986; Agre and Chapman 1987; Arkin 1990). The basic idea is that rather than divide overall process of acting in the world into functional components (such as planning, execution, and perception) one could instead divide the process into task specific pre-compiled plans for various behaviors. The basic insight here is that the necessity to act fast in an uncertain and dynamic world requires reactive planning agents (biological or robotic) to develop representations that can tightly couple action, execution monitoring, error correction and failure recovery. Narayanan (1999) contains a full comparison of our event model to these techniques in logical AI and classical planning.

2.1.2 Graphical Models of Actions

There has been a considerable amount of work initiated by Pearl (2001) and colleagues on graphical models of actions. The basic idea is to use a causal network graph to represent a set of autonomous mechanisms where inter-mechanism interactions are captured through links in the graph. Each action implies an intervention on this graph resulting in isolating an autonomous mechanism through local surgery on the causal graph. This allows for a uniform structure that can reason about both observations and actions. Pearl develops an extensive framework that can deal with many issues in causal inference including predictions, diagnosis, and counterfactual reasoning. Our work derives a lot from the work in probabilistic representation of actions. Specifically, (Narayanan 1999) outlines an initial integration of the two lines of work that specifically addresses the frame problem (and the qualification and ramification subproblems) in a framework that combines our representation and model of event structure with the graphical model for inference. Chapter 8 outlines a companion thesis that is focused more directly on a more expressive probabilistic representation and manipulation of complex behavior. The main difference between our approach and the work of Pearl and colleagues is in the representation of actions and events. In previous approaches, actions are typically treated as atomic (either effect axioms in the logical approach, or single mechanism in the probabilistic work). Our approach, in contrast, models actions with internal structure that includes resource production and consumption, internal phases, and composite actions that include concurrency and synchronization. As shown in subsequent chapters of the thesis, such an expressive model of event structure is essential to handle the kinds of event and action related questions that users routinely ask of complex scenarios. The work here also connects to recent work on planning using Partially Observable Markov Decision Processes (POMDP) and to work on concurrent models of computation such as Stochastic Petri Nets (Bause and Kritzinger 1996) for representing complex processes. Chapter 3 describes the connection to concurrent models in greater detail. A companion thesis by Leon Barrett (see Chapter 8 for a brief description) is exploiting this connection for planning and reasoning about complex behaviors.

2.1.3 Event Models for Transactions on the Semantic Web:

More recently, there have been efforts toward the idea of creating a database out of the web. This idea (called the Semantic Web) has the goal of enabling access Web resources by content rather than just by keywords. In the last decade, the Semantic Web community has developed a new generation of Web markup languages such as OWL (McGuinness, Van Harmelen et al. 2004). These languages enable the creation of ontologies for any domain and the instantiation of these ontologies in the description of specific Web sites. These languages are also amenable to efficient reasoning procedures and thus reasoning applications can be built to automatically determine the logical consequences of the ontological statements.

Among the most important Web resources are those that provide services. Web services are web sites that do not merely provide static information but allow one to effect some action or change in the world, such as the sale of a product or the control of a physical device. To make use of a Web service, a software agent needs a computer-interpretable description of the service, and the means by which it is accessed. An important goal for Semantic Web markup languages, then, is to establish a framework within which these descriptions are made and shared. Web sites should be able to employ a standard ontology, consisting of a set of basic classes and properties, for declaring and describing services, and the ontology structuring mechanisms of OWL provide an appropriate, Web-compatible representation language framework within which to do this.

A collaborative effort by researchers at several organizations has resulted in just an ontology called OWL-S or OWL for Services (Martin, Burstein et al. 2004). OWL-S is a markup language for describing both atomic and composite transactions on the web using a formal model of events and processes. OWL-S service descriptions enable programmatic access for service discovery, monitoring, simulation, analysis, and composition.

The OWL-S model of web services is closely related to the event model described in this thesis. Indeed, our event model implemented the first Distributed OPErational (DOPE) semantics of the OWL-S language (Narayanan and McIllraith 2002). Models that are described in OWL-S can be compiled automatically to our event models. Details of the process and our use of the models are described in Chapter 4. The connection to OWL-S allows our event models to be compiled directly from process and event descriptions on the Semantic web.

2.1.4 Event Models in Linguistics and NLP

Events are central to language and a subject of obvious interest to linguists. The description of events in language can rely on a variety of devices including predication, modification, and reference. Vexing problems in event interpretation such as linguistic aspect and contingent causal relations have led researchers in Formal Semantics to propose ontological schemes such as the Vendler-Dowty-Taylor (VDT) verb classification. Extensions and computational models of event structure have been proposed to handle some of the compositional issues in interpreting expressions about events (Moens and Steedman 1988; Steedman 1996). More recently, basic ideas from Frame Semantics and Formal Lexical Semantics have been used to build wide-coverage open domain semantic resources such as FrameNet and VerbNet. Resources such as FrameNet function as a semantic basis for extracting event information from textual sources. Chapters 3-6 details our use of FrameNet as a structured intermediate representation to connect language to event models. (Narayanan 1999) details an expressive model of event structure using a structured dynamic system (called X-schemas) that was used in solving hard problems linguistic aspect and metaphoric language about events. Our work is a direct follow-on to Srini Narayanan's thesis work on active event representation for understanding aspect and metaphor (Narayanan 1997). His work on X-schemas is a cornerstone of our QA engine.

2.2 Question Answering (QA) research

The typical formulation of the QA task is: a user inputs a natural language question, and the system responds with an answer (the exact answer, or a passage or document containing the answer). The task comes in two flavors: open-domain (where a question can be on any topic) and closed-domain (where a question is on a prespecified topic, e.g. one about cooking, travel, or international terrorism). Open domain QA frequently leverages the Web as a large corpus. With broader coverage required, though, it is a challenge to get and exploit deep domain specific knowledge; for that reason, we chose to work in a closed-domain environment for our research. Event scenario QA represents a specialization of the general QA problem, focusing on complex question types that need additional semantic resources, as we detail in the Chapters 3 and 4.

2.2.1 Motivation to use event models in QA

The driving rationale for our approach is that humans appear to have limited need for factoid question answering, and instead much more need to have systems that can deal with complex reasoning about causes, effects, and chains of hypotheses.

This is borne out through a number of studies, as well as our own empirical analysis. As cited in (Yin 2004; Aouladomar 2005), procedural questions make up the second largest set of queries on web search engines, after factoids. In our own analysis of the AQUINAS AnswerBank corpus. The corpus, produced by the University of Texas, Dallas (UTD), has 2692 question/answer pairs generated to resemble intelligence analysts' queries about weapons programs and proliferation issues, approximately half of the questions were found to be event related.

As discussed in 2.2.3, most work in QA during the first half of this decade was focused on improving performance in factoid answering systems. Even in recent years, only a few groups have been working on solving event related questions. It is still a problem waiting for an adequate solution.

2.2.2 State of the Art

We used the UTD-LCC state-of-the-art QA System (Pasca and Harabagiu 2001) as a baseline for our task. The UTD QA system uses a component-based flexible architecture with modules that a) process the question by linking it to an entry in an ontology of answer-types, b) use a variety of IR techniques to retrieve relevant answer passages, and c) extract the answer passage ranked highest amongst the candidates. The system is trained to handle questions related to a closed domain of documents from the Center for Nonproliferation Studies.

State-of-the-art QA systems such as the UTD system and others (Ramakrishnan, Chakrabarti et al. 2004) rely on standard IR techniques (like TF-IDF; Salton, Wong et al. 1975) along with enhancements that expand the query. Such modifications include search patterns and heuristics based on word clusters, synonym sets, and lexical chains, which are a) derived using machine learning techniques, b) extracted from lexical resources such as WordNet or c) a combination of a) and b). Selecting answer passages relies on a quantitative measure that evaluates the degree to which a passage shares the words in the expanded query keyword set.

2.2.3 Related Work in QA

While question answering as a domain is not a new area of research (e.g. Lehnert 1978), it is only over the past half dozen years that the field has really taken off, stimulated by new funding (AQUAINT) and large evaluations (TREC QA Track). Many projects during this period have focused on improving factoid QA performance, which has, indeed, improved as a result (Pasca and Harabagiu 2001).

Using lexico-semantic resources, such as WordNet, PropBank, and FrameNet, has become more popular, though the use of FrameNet is still relatively rare (Fellebaum 1998; Kingsbury and Palmer 2002; Palmer, Gildea et al. 2005; Ruppenhofer, Ellsworth et al. 2006). One group actively experimenting with FrameNet for QA is at Saarland University. Gerhard Fliedner, in cooperation with the SALSA group, is using frame annotations as a basis of comparison between questions and answers in a prototype QA system for German (Fliedner 2006). He has developed a number of innovative techniques to merge frames that we plan on looking at further for possible adoption (Fliedner 2004; Fliedner 2005).

Attempts at answering procedural questions has some history, the most recent of which comes from France and England. Farida Aouladomar, a computational linguist at IRIT in France, has done significant work in analyzing questions about procedures and texts which explain those procedures (Aouladomar 2005). Her goal is to answer "How" questions (mostly geared towards French). Compared to the system proposed in this document, her work is most similar to our model selection component (outlined in Chapter 5). Aouladomar's analysis of procedural texts may assist us in the future for extracting event model structure in additional test domains. Ling Yin at ITRI at the

University of Brighton, UK, has also done some procedural question analysis (Yin 2004).

One other group has worked on answering prediction questions using simulation. Back in 1994, Jeff Rickel and Bruce Porter, at UT Austin, built a system for answering questions of scientific models (Rickel and Porter 1994). In this system, there was no tie to natural language. Instead, interactions were through predicate calculus. Their system focused on automatically building models of necessary and sufficient differential equations chains for solving a question posed to the system. Our work may be able to supply a means of interacting with and using natural language with their scientific model solver. In more recent work, Porter and colleagues have been using an ontology and lexicon to build on this early work. They have been working in the area of question answering for scientific domains. The input to their system is still a logical representation of the textual material (Barker, Porter et al. 2001; Barker, Chaudhri et al. 2004). They have built a library of generic concepts and use a Knowledge-Base and reasoner to answer questions about advanced placement Chemistry. Our system differs from these previous systems in the complexity of question types, the connection to language using a wide coverage resource (FrameNet) and the ability to reason with dynamic and uncertain input and knowledge.

3 Expressive schema and dynamic models of events

Events and procedures are complex. They are dynamic, unfolding over time. They are inherently about change of state, with outcomes contingent on resources. What makes them interesting is their structure, which dictates the conditions required to bring an event to a particular state, and the possible and likely evolutions going forward from a state. To understand events and reason about them requires a means of describing, simulating, and analyzing the underlying processes, taking into account preconditions and effects, the resources they produce and consume, and their interactions with each other.

We developed a novel, comprehensive event schema that covers many of the parameters required to do complex reasoning about events (Section 3.3). Based on this schema, we further designed a dynamic model of events capable of simulation and causal inference (3.4). With this framework in hand, we have formulated and adapted existing algorithms (3.5) to infer the information required to answer common questions about events (detailed in 3.2).

3.1 Motivating example

Here is a motivating example. Take a process like the one that commences when a crime is reported. Here in the United States, the police investigate the alleged crime and come to one of two conclusions: there was or there was not a crime. If there was a crime, suspects are identified and pursued, often leading to the arrest of the suspect. The suspect is brought before a judge and notified of the charges, after which he enters a plea. If needed, the judge sets bail and the case goes to trial. A jury is sometimes selected, the case is deliberated, and a verdict is entered. If found guilty, the suspect is sentenced and pays for the crime: monetarily, in prison, or with his life. If innocent, the suspect is released and the investigation is potentially restarted.

3.2 Objectives

As humans, we can understand the complexities of this judicial process. How, though, can a computer reason about it? What specific aspects of reasoning would we like a computer to tackle?

As mentioned, events evolve over time. As they progress, clues are left behind regarding the path taken. If a suspect in a crime is in front of a judge, there is a trail of police reports and court filings preceding this court appearance. Given a set of information, like court filings, we would like a computer system to be able to determine if there is a causal relationship between that information and a particular target state of interest. This will allow us to answer a question like, *Did Joe appear in traffic court for speeding?* These are **justification** questions: questions with an underlying premise (Joe appeared in traffic court for speeding) that require verification with evidence and a trace of reasoning.

Events can progress in potentially multiple ways. In addition to looking at past effects, we would like to be able to reason about the possible evolution of a state going forward. This can come in two forms: a **temporal projection** or a test of the **ability** of an event to reach a target state. The former is an assessment of the likely evolution of a state: *Will Sam testify in front of a jury?* The latter is a test of what is possible: *Can Jill evade jail?*

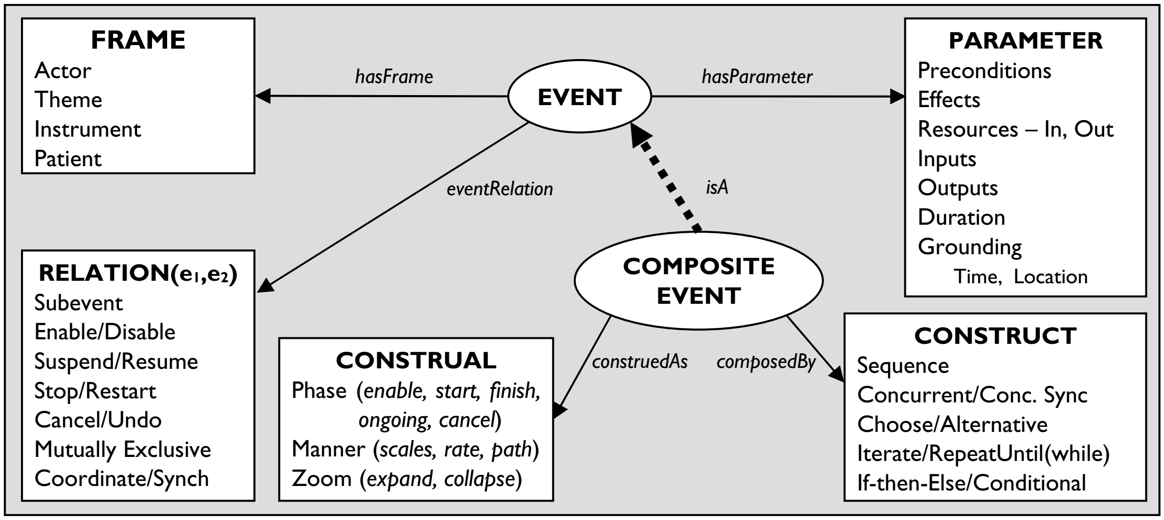


Figure 3-1: Event Schema

Not all reasoning, though, is based on states as they actually exist(ed). Many times we wish to reason about what could have been or could be. We wish to be able to find causal chains from an altered set of evidence (a **hypothetical** state) to a target state, such as: *If the murder weapon is thrown out as evidence, will Mike be convicted?*

Also, our knowledge about the structure of an event is not always complete. In analyzing an event, we may not be in a position to observe it actually happening; instead we may only have access to incomplete and uncertain snapshots of the progress. We would like to be able to **disambiguate between hypotheses** of the type of event occurring based on that evidence. *Based on the court proceedings, is the defendant being tried under a civil law or religious law system?*

These are each types of event-related questions we seek to be able to answer: justification, temporal projection, ability, hypothetical, and hypothesis disambiguation. We identify solutions to each in Section 3.5, after describing in the upcoming sections a common framework for representing and analyzing the events.

3.3 Describing events

Having a standardized event representation is crucial to being able to reason about a significant number of event-related scenarios. It allows us to develop algorithmic solutions to common complex questions, such as those described in Section 3.2.

In designing an ontology of events, we have identified a number of critical features required to provide significant coverage and usability for reasoning tasks. First, an event representation must be *fine-grained* to capture the wide range of possible events and their interactions. Next, it must be *context-sensitive* in order to adapt to a dynamic and uncertain environment. Thirdly, it should be *cognitively motivated* to allow humans to easily add content, query, and make sense of the information returned. Finally, event descriptions should allow for *elaboration*, so that new domain models can specialize existing representations without changing the basic primitives.

We have developed a parameterized model of the structure of events and processes that meets these criteria. Figure 3-1 shows the basic schema of events. We describe its main elements here.

1) *All events have a basic structure*: A basic event is comprised of a set of inputs, outputs, preconditions, effects (direct and indirect), and a set of resource requirements (consuming, producing, sharing and locking). Events are grounded at a time and place and have a duration. The *hasParameter* link in Figure 3-1 depicts the set of parameters in the domain of the basic event type. We discuss this further in Section 3.4.2.

2) *Events have a frame semantic structure*: Events, though occurring independent of language, are described in language using Frame-like relations (Fillmore 1982). Frames are labeled entities comprised of a collection of roles that include major

syntactic and semantic sub-categorization information. The relation *hasFrame* in Figure 3-1 is a many-to-many link since an individual event may have multiple frames and a single frame could capture multiple events. Many frame roles used when describing an event have corresponding event parameters, allowing for tight integration of language and event structure. See Chapter 4 for more details.

3) *Composite events have rich temporal structure and evolution trajectories*: The fine-structure of events is composed of key states (such as enabled, ready, ongoing, done, suspended, canceled, and stopped) and a partially ordered directed graph of transitions that represents possible evolution trajectories between these states (transitions include: prepare, start, interrupt, finish, cancel, iterate, resume, restart). Each of these transitions may be atomic, timed, stochastic or hierarchical (with a recursively embedded event-structure).

4) *Composite events are composed of process primitives*: Verbal aspect (the temporal structure of events (Narayanan 1997)) discriminates between events that are punctual, durative, (a)telic, (a)periodic, (un)controllable, (ir)reversible, ballistic, or continuous. Each type of event relates to a particular internal structure, which draw upon a set of process primitives and control constructs (sequence, concurrent, choice, conditionals, etc.). These primitives specify a partial execution ordering over subevents. The *composedBy* relation in Figure 3-1 shows the various process decompositions. Figure 3-6 describes them in greater detail.

5) *Composite events support various construals*: Composite events can be viewed at different granularities using operations for *elaboration* (zoom-in) and *collapse* (zoom-out) (Narayanan, 1997). In addition, specific parts and participants of a

composite event can be *focused on, profiled* and *framed*. Construal operations are shown in Figure 3-1 through the *construedAs* relation.

6) *Events relate to each other in regular patterns*: A rich theory of inter-event relations allows sequential and concurrent enabling, disabling, or modifying relations. Examples include interrupting, starting, resuming, canceling, aborting or terminating relations, as shown in Figure 3-1 through the *eventRelation* relation.

Example

Any specific action can be described in structured form, as an instantiation of the event schema. We can use feature-structures to formally specify the components of the event. (A feature-structure is a collection of attributes, values, and constraints.) Take our example of going to trial.

Composite Event: CriminalTrial				
Precondition:	crimeAlleged, suspectApprehended			
Effect:	verdictRendered			
Resource-In:	[judge, 1], [court, 1], [clerk, 1], [prosecutionTeam, 1], [defenseTeam, 1], [defendant, 1], [juryPool, 24]			
Input:	charges			
Duration:	3-hours			
Subevent:	[*sequence* JurySelection, EvidenceExamination, VerdictFinding]			
Frame:	Trial			
		Element:	Defendant	- <Person>
		Element:	Judge	- <Judge>
		Element:	Prosecution	- <DistrictAttorney>
		Element:	Defense	- <DefenseAttorney>
	...			

A trial is a composite event, having fine-grained structure. At the basic level, it has certain preconditions (Precondition), such as a crime being alleged (crimeAlleged) and a suspect being apprehended (suspectApprehended). The effect (Effect) of the trial is that a verdict is rendered (verdictRendered). These are states – they occur or they don't. The mechanics of a jury trial require a set of resources (Resource-In): a judge, a court, a clerk, a prosecution team, a defense team, a defendant, and pool of potential jurors. Resource requirements are specified with tuples of the resource name and the amount required. Certain information is also required (Input), namely the charges against the defendant. A prototypical trial will take 3 hours, in this example (Duration). In that time, certain subevents are executed. The trial is composed of a sequence of subevents, namely jury selection (JurySelection), the examination of witnesses and evidence (EvidenceExamination), and the jury deliberation and finding of verdict (VerdictFinding). Each subevent may, itself, be a composite event. With these different levels of granularity, we can construe the trial event as either a single, high-level event, or as a sum of its sub-parts. In addition to this structure, each action and sub-action can be described in frame form, in this case, the Trial frame provides one such grounding in language, with its various elements and bindings (see Chapter 4 for more information about frames).

This representation provides the first step in the apparatus needed for complex reasoning about events: for tasks ranging from planning to simulation.

3.4 Dynamic modeling of events

Complex reasoning about event interactions requires not only an event description, but also a dynamic model that can simulate the execution of the event unfolding over time. We can instantiate such a model with facts about a particular event, enabling us to project which situations are likely or possible based on the consumption and production of resources and the creation and elimination of states.

Take, for example, our simplified trial process. In a particular instance, a suspect, Joe, may be arrested for theft. We may wish to know whether it is possible for him to

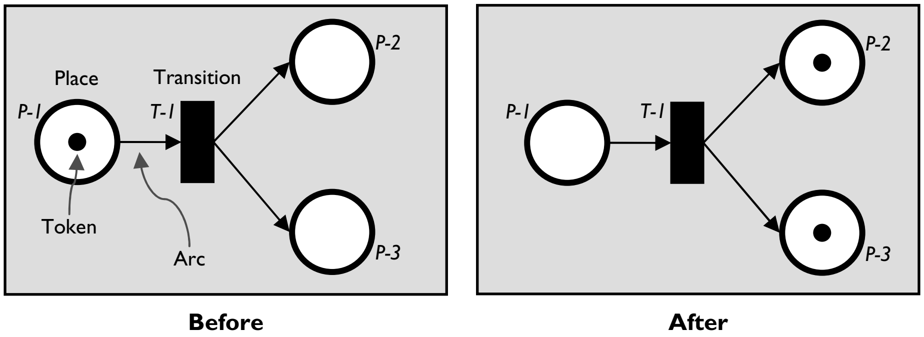


Figure 3-2: X-net basic simulation

be convicted. If we know that there is no judge available, an attempt to simulate a trial will fail, and it won't be possible to reach a verdict. If the necessary resources for a trial do exist, then an analysis should determine that after three hours in court, the defendant may be convicted.

To create dynamic models to analyze this and more complicated scenarios, we use a computational modeling framework known as X-nets (Section 3.4.1). They support many of the event primitives described in the event schema (Section 3.4.2). Most importantly, they support both simulation of the underlying event and algorithms for analysis and inference (Section 3.4.3).

3.4.1 X-nets

Events, as mentioned, are about state changes. A method for dynamic modeling of events requires two key pieces, then: a way of storing state and a way of changing state. X-nets have these two main components. **Places** hold resource and condition state. **Transitions** are active elements that create, destroy, and test the resources and conditions encoded in Places. Together, they provide a solution for representing the dynamics of events and the means of simulating them.

Figure 3-2 shows the basic semantics with a subset of the features of X-nets. In the first frame (Before), Place P-1 is shown containing one **token**, representing either

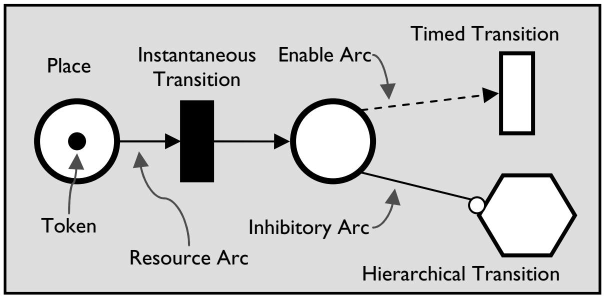


Figure 3-3: Full X-net graphical key

one unit of a resource or the existence of a condition. Places P-2 and P-3 have no tokens. An **Arc** exists *from* P-1 *to* Transition T-1 representing a resource dependence. P-1 must have adequate resources (one token by default) for T-1 to execute (also known as "firing"). T-1 will fire in this case, resulting in the second frame (After). Here T-1 has consumed the resource from P-1. The Arcs *from* T-1 *to* P-2 and P-3 represent resource production; T-1 creates resources in P-2 and P-3 when firing. Each firing thus changes the overall state of the X-net. We call this distribution of tokens over Places a **marking**, in this case going from [1, 0, 0] to [0, 1, 1] (for [P-1, P-2, P-3]).

This simple, abstract model can represent a specific event. For example, Joe buys a can of soda. P-1 represents Joe's money, in this case, one token is $1. T-1 represents the buying operation. P-2 represents Joe having a soda, and P-3 is the receipt. Joe has a $1, he executes the buying operation, the $1 is consumed, and he gains a soda and a receipt. Dynamic modeling allows us to make different inferences based on the evidence available. Had Joe had no money, he would not have been able to buy the can of soda.

In full, the X-net representation directly captures a number of additional features of events. (See Figure 3-3 for some.)

- Events unfold in uncertain ways. X-nets support stochasticity in transitions.

- Some events take time and some do not. X-nets provide instantaneous Transitions and timed Transitions.
- For conceptualization and display, it can be useful to abstract away subevents, collapsing an X-net as a special Transition in another X-net. We flatten these "hierarchical" Transitions at X-net creation time.
- The existence of certain resources and conditions can have both positive and negative impact on an event's execution. X-nets provide inhibitor arcs for those cases where a satisfied condition should prevent a Transition from executing.
- Not all resources and conditions required for the execution of an event should be consumed during the execution of the event. Enable arcs can be used to test-but-not-consume resources and conditions.

All of these features combined provide us the building blocks to represent the complex event structures described in Section 3.3. At their core, X-net Transitions represent simple events. We are able to chain several events together through their common conditions and resources to represent a larger scenario. In the next section (Section 3.4.2), we will show the mapping from event description to X-net.

Background

Narayanan, in his 1997 work on aspect and metaphor understanding, proposed using modified Petri Nets for modeling actions. His Action Execution Schema (X-schema) extended basic Place-Transition Nets (P-T Nets) with stochastic transitions and hierarchical transitions, bounded colored tokens (tokens that are typed), and enable and inhibitor arcs (for k-safe nets). The resulting semantics could cleanly and efficiently capture sequentiality and concurrency, while providing asynchronous control. (Narayanan 1997; Narayanan 1999)

Simulating and analyzing events to answer questions has computational requirements similar to the features provided in Narayanan's X-schema design. In following up his work, we use a subset of the proposed Petri Net extensions. Our design supports stochasticity and enable and inhibitor arcs, but we leave out colored tokens and hierarchical transitions (which can be decomposed and flattened with finite nets). The resulting design, as with X-schemas, is built upon Generalized Stochastic Petri Nets (GSPNs) (Ajmone Marsan, Balbo et al. 1995; Narayanan 1999). We describe our extended version of GSPNs, below.

Formal definition

Formally defined, a GSPN is an 8-tuple, GSPN = $(P, T, T_1, T_2, I^-, I^+, W, M_0)$ where

- $P = \{p_1, \ldots, p_n\}$ is a finite, non-empty set of places,
- $T = \{t_1, \ldots, t_n\}$ is a finite, non-empty set of transitions,
- $T_1 \subseteq T$ is the set of timed transitions, $T_1 \neq \varnothing$
- $T_2 \subset T$ denotes the set of immediate transitions, $T_1 \cap T_2 = \varnothing$, $T = T_1 \cup T_2$
- $P \cap T = \varnothing$
- $I^-, I^+ : P \times T \rightarrow \mathbb{N}_0$ are the backward and forward incidence functions, respectively (these are the incoming and outgoing resource arc weights, w.r.t. the transition)
- $W = (w_1, \ldots, w_{|T|})$ is an array whose entry w_i $\mathbb{R}^+$
  - is a rate of negative exponential distribution specifying the firing delay when transition t_i is a timed transition, or

- is a firing weight when transition t_i is an immediate transition

- $M_0 : P \rightarrow \mathbb{N}_0$ is the initial marking; for each place, a number of tokens

Note: if $T_1 = \varnothing$, then the net is a weighted Place-Transition net. If $T_2 = \varnothing$, then the definition of GSPN coincides with the definition of a Stochastic Petri Net. (definition adapted from Bause and Kritzinger 1996) Our design relaxes the constraint that $T_1 \neq \varnothing$, thus supporting both GSPNs and basic P-T Nets.

Modifications to traditional GSPNs that we use in X-nets:

- $C = \{c_1, \ldots, c_{|P|}\}$ is an array of place capacities whose entry $c_i \in \mathbb{N}_0$, if present, represents an inclusive upper bound on marking value for place p_i,
- $E : P \times T \rightarrow \mathbb{N}_0$ are enable arc weights, not separately defined from I^+ and I^- as they are syntactic sugar, providing equal additive influence to I^+ and $I^=$,
- $N : P \times T \rightarrow \mathbb{N}_0$ are inhibitor arc weights
- $\Pi : T \rightarrow \mathbb{N}$ are transition priority levels

Real-time simulation semantics

Petri Nets follow a well-defined simulation semantics that specify how states change. With any given marking, the next state is determined by executing the net firing rule.

- At $state_i$ with marking M_i, a transition t_j is enabled iff, for each place p_k, $M_i[k] \geq I^-[k, j]$ (i.e. resource and enable arcs satisfied), $M_i[k] \neq c_k$ (i.e. place capacity has not been reached), $M_i[k] \,!\!\geq N[k, j]$ (i.e. not disabled by inhibitor arcs), and no other transition t_h can be enabled with $\Pi_h > \Pi_j$ (i.e. highest priority transition with necessary resources).

- When an enabled transition t_j fires, for each place p_k, $M_{i+1}[k] = M_i[k] - I^-[k, j] + I^+[k, j]$ (i.e. consume and produce tokens)
- For all enabled immediate transitions, firing order is determined by sampling from a discrete uniform distribution
- When no immediate transitions are enabled, timed transitions can fire after a delay determined by sampling the negative exponential distribution ($F_{X_i}(x) = 1 - e^{-\lambda_i x}$), where the rate of transition t_i is λ_i. Race condition semantics are employed (when enabled, a transition waits for its sampled delay, then, if its incoming tokens are still available, it fires).

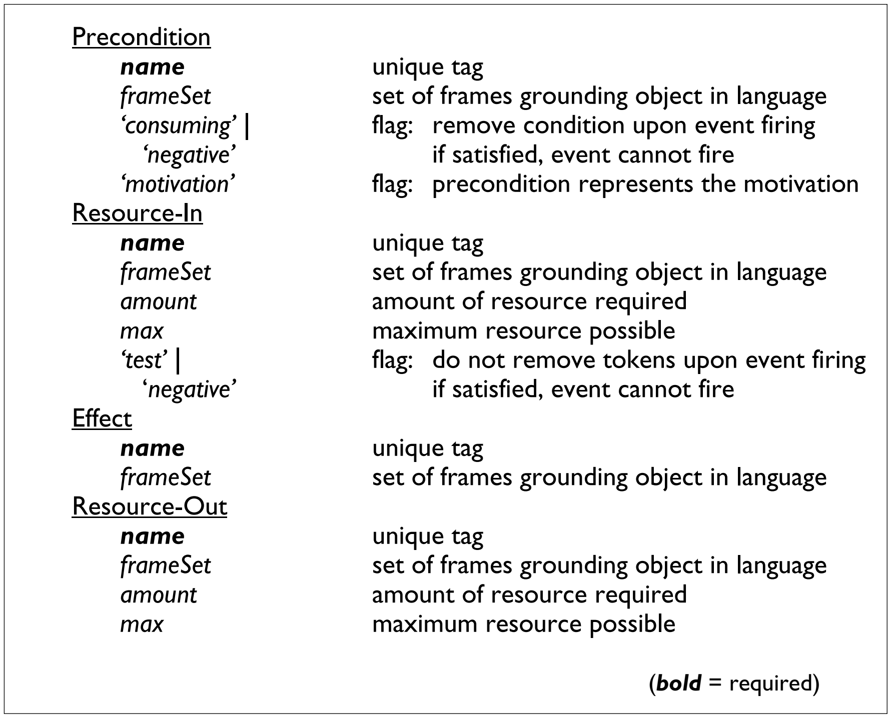
3.4.2 Representing complex events with dynamic models

Our event schema provides a structured form that explicitly captures many of the features of events. Our X-net framework provides the raw materials for representing complex events in a dynamic model. Next, we detail how to map the structure described in event schema instances to dynamic X-net models.

Basic Event

In Section 3.3 (Figure 3-1), we described a number of event parameters that provide a description of the basic structure of an event. They include inputs, output, preconditions, effects, resources, duration, time, and place. Each can be translated into a piece of X-net structure.

In the simple atomic case, the event itself translates into a single Transition t, representing the action of executing that event. The event parameters are attribute tuples, as defined in Figure 3-4, which constrain the mapping from description to X-net structure.



Precondition	
name	unique tag
frameSet	set of frames grounding object in language
'consuming' |	flag: remove condition upon event firing
'negative'	if satisfied, event cannot fire
'motivation'	flag: precondition represents the motivation
Resource-In	
name	unique tag
frameSet	set of frames grounding object in language
amount	amount of resource required
max	maximum resource possible
'test' |	flag: do not remove tokens upon event firing
'negative'	if satisfied, event cannot fire
Effect	
name	unique tag
frameSet	set of frames grounding object in language
Resource-Out	
name	unique tag
frameSet	set of frames grounding object in language
amount	amount of resource required
max	maximum resource possible

(***bold*** = required)

Figure 3-4: Basic event schema

- **Precondition**: translates into an incoming Place p (with a directed arc from p to t), with Capacity $c_p = 1$. By default, the arc connecting p to t ($I^-[p,t]$) is an Enable arc (s.t. $I^+[p,t] = I^-[p,t]$). If flagged as 'consuming', the arc will be a standard Resource arc. If set as 'negative', the arc is an Inhibitor arc. The arc weight is 1 ($I^-[p,t] = 1$).
- **Resource-In**: translates into an incoming Place p. By default, Capacity c_p is not set (i.e. infinity), and the arc weight is 1 ($I^-[p,t] = 1$), unless otherwise specified by 'max' and 'amount', respectively. If flagged as 'test', the arc is set to be an Enable arc. If flagged as 'negative', the arc is set to be an Inhibitor arc.
- **Effect**: is similar to a Precondition, but instead translates into an outgoing Place p (with a directed arc from t to p), with Capacity $c_p = 1$ and arc weight 1 ($I^+[p,t] = 1$).

- **Resource-Out**: translates into an outgoing Place p. Again, by default, Capacity c_p is not set, and the arc weight is 1 ($I^+[p,t] = 1$), unless otherwise specified by 'max' and 'amount'.
- An event **Duration** can be directly translated into the rate parameter of t. The existence of a duration implies Transition t is timed; the lack of a duration parameter implies t is immediate.
- **Input**, **Output**, and **Grounding** parameters (and the frameSet attribute of the aforementioned parameters) are not taken into account by the simulation firing rule, and are thus not mapped over to the simulatable model. (Inputs are, for example, credit card numbers needed for a Buying event. The Precondition of having a credit card is sufficient for simulation; the exact number is not necessary.)

Event-Event Relations

Events can be connected to other events implicitly or explicitly. Implicitly, events sharing states and resources are automatically connected to one another by common Places. Explicitly, events can be connected through stated event-event relations (described in Figure 3-1).

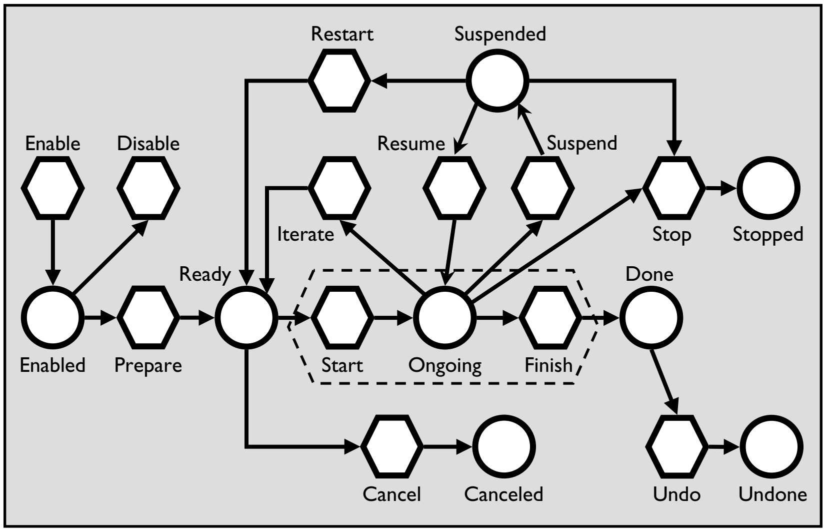


Figure 3-5: Event controller

We extend Narayanan's work on linguistic aspect to provide guidance on how to connect events related by an evolutionary trajectory. In Figure 3-5 (adapted from Narayanan 1999), we show a general event controller, where the event in focus is represented with three components, the *starting* action, the *ongoing* state, and the *finishing* action (shown contained in the dashed-line hierarchical transition). Other events related to the focal event are linked according to the design shown. For example, an event that *suspends* the focal event will be linked so as to remove the *ongoing* control token when fired, preventing the focal event from finishing. (Note: each hierarchical transition represents a sub-net that can be expanded; thus our *suspend* is a placeholder for the event that is suspending the focal event.) The control states (*ready*, *suspended*, *stopped*, etc.) are added to connect the related events and to mark event evolution progress; they assist in answering questions about the current phase an event may be in. (Not all relations shown may be used in a description; the unused portions of the design are collapsed.)

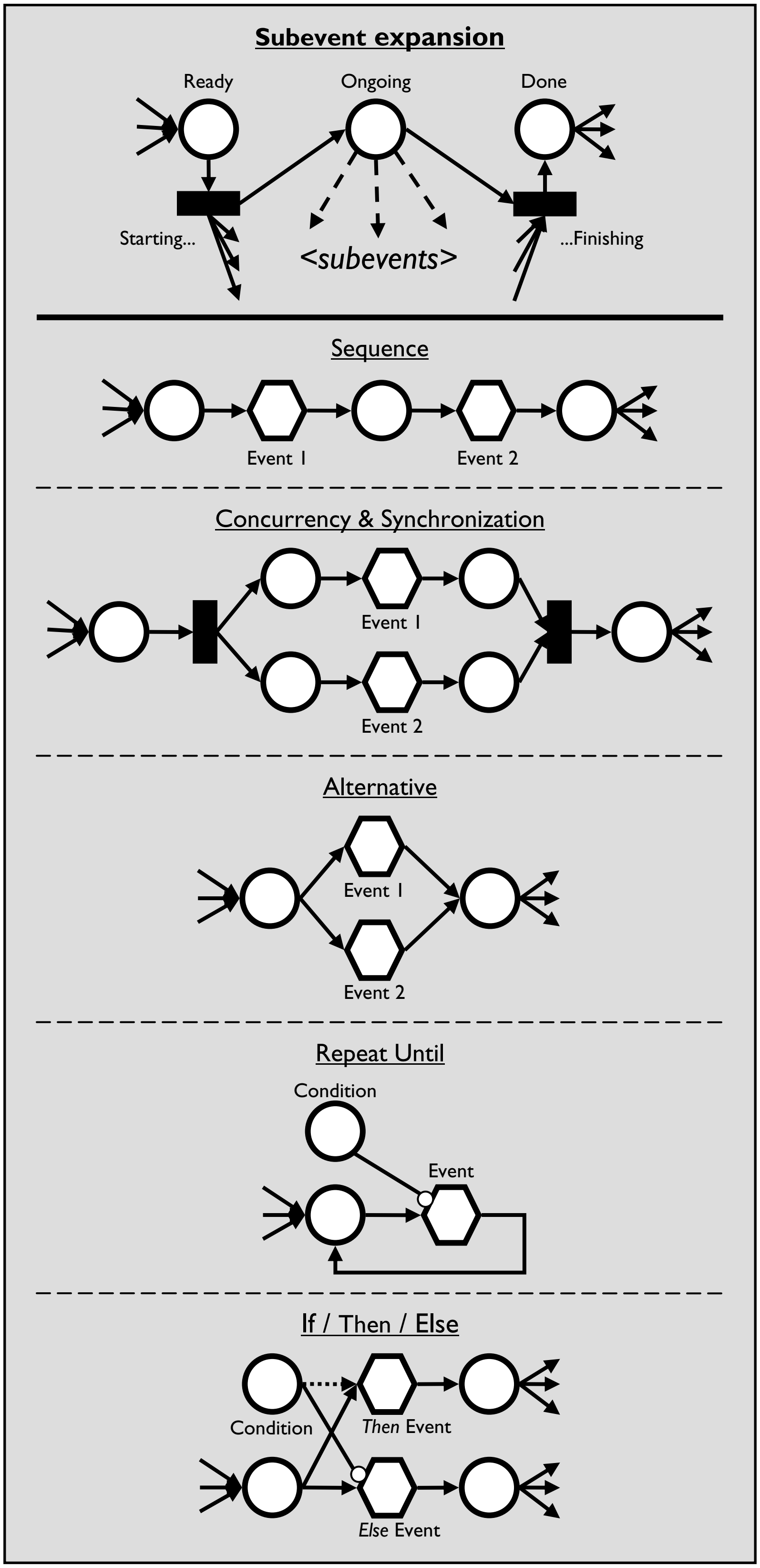


Figure 3-6: Composite Event Control

Composite Events

Control constructs in composite events can also explicitly define the layout of a set of (sub)events. In an event description, composite events can be specified as a recursive 3-tuple in the form of <*constructType event event*>, where each '*event*' can be atomic or composite. In Figure 3-6 (adapted from Narayanan 1997), we demonstrate the expressive power of X-nets to represent these control constructs for composite events.

Frames

Frames capture the way in which language describes events; specifically, they capture roles and profiled participants, aspectual structure, instruments, and other grounding constructs such as location and time. These frame elements, when extracted from the linguistic input, can provide parameters to an event simulation. In Section 4.1, we will discuss our use of frames to help setup event simulations. To enable this effort, X-nets created from Event Schema instances must retain links to all linguistic descriptions of the underlying events.

As mentioned in Section 3.3, frames can be associated with the main action of the event, or its parameters. Frames associated with the main action are directly linked to the corresponding Transition. Those associated with a particular parameter (as specified in the attribute tuple under 'frameSet') are directly linked to the corresponding Place.

Linking and Tuning Event Models for Larger Scenarios

Event schema instances are a starting point for creating dynamic X-net models. Our method provides a default mapping from a schema instance to a model, but it requires a model designer to tune models to meet an analysis goal.

First, a model designer may have access to an event ontology consisting of relatively primitive event schema instances. Building up a larger event scenario may requires piecing together smaller composite and basic events. This can be semi-automated based on common resources shared between these pieces, leaving control structure linking to the designer.

The designer must then tune X-net parameters such as immediate transition weights and transition priorities, to achieve accurate stochastic behavior model-wide.

3.4.3 Analysis

Questions about events, as discussed in Section 3.2, require complex reasoning to answer. In a question about the ability of an actor to do something not yet done, for example, we wish to be able to figure out if a particular state is reachable given the current state. It may be likely or not. It may be inevitable or it may be one of ten-thousand possible evolutions.

X-net modeling, by virtue of its simulation semantics, gives us the direct ability to simulate single paths of potential evolution of events. While this is an important building block for more complex analysis, it is insufficient for our needs on its own. GSPNs (and by extension, X-nets) are amenable to a number of analysis techniques, chief among them for our purposes: reachability.

Reachability (Forward)

Reachability analysis produces an exhaustive set of states that are achievable through simulation given the initial marking. As we will discuss in Section 3.5, this is an important component of our algorithms to solve justification, temporal projection, and ability questions.

Our method to calculate the forward reachable set of states is to construct a coverability tree. Coverability analysis is a special form of reachability analysis that handles both bounded and unbounded nets. (To be bounded, the number of tokens in the net cannot grow beyond some fixed value.) The deterministic version of the algorithm we use to construct coverability trees is adapted from (Murata 1989).

```
1. Label the initial marking $M_0$ as the root and tag it "new"
2. While "new" marking exist, do the following:
   2.1. Select a new marking $M$
   2.2. If $M$ is identical to a marking on the path from the root to
        $M$, then tag $M$ "old" and go to another new marking.
   2.3. If no transitions are enabled at $M$, tag $M$ "dead-end"
   2.4. While there exist enabled transitions at $M$, do the following
        for each enabled transition $t$ at $M$:
        2.4.1.Obtain the marking $M'$ that results from firing $t$ at $M$
        2.4.2.On the path from the root to $M$ if there exists a
              marking $M''$ such that $M'[p] \geq M''[p]$ for each place $p$
              and $M' \neq M''$, i.e., $M''$ is coverable, then replace $M'[p]$
              by $\omega$ for each p such that $M'[p] > M''[p]$ && $c_p = \varnothing$
              ($\omega$ represents "infinity")
        2.4.3.Introduce $M'$ as a node, draw an arc with label $t$ from
              $M$ to $M'$, and tag $M'$ "new"
```

adapted from Murata 1989

Line 2.4.2 of the algorithm is responsible for handling unbounded nets. It relies on the monotonicity property of Petri Nets, that transitions that can fire with a certain number of tokens can also fire with more tokens. This property is violated by the existence of inhibitor arcs. As such, for models with inhibitor arcs, our design only supports analysis of bounded nets; in that situation, we ignore line 2.4.2. There is literature on subclasses of Petri Nets with inhibitor arcs where modified coverability analysis is still possible. This is not a part of our design, but we reference readers to (Busi 2002), for one approach.

For nets with stochasticity, we can convert the net into a Markov Chain and use Forward-Backward and Viterbi algorithms to estimate the likelihood of particular reachable states or paths (see Bause and Kritzinger 1996).

Backwards Reachability

Justification, ability, and other analyses do not only rely on projecting evidence forward. As we will discuss in Section 3.5, information about a current state can provide clues of what has already happened, not just what may happen in the future. Backwards reachability analysis is an important tool that can provide an exhaustive set of states that may have generated a given state. With it, in our simplified judicial process example, we can infer that someone who was convicted must have gone through a trial.

Backwards reachability is more complicated than forward reachability, unfortunately. Unlike determining the forward reachable set, where standard Petri Net firing rule semantics are employed, techniques for calculating the backwards reachable set require structural model intervention or a significant change to the simulation firing rule. We provide a brief explanation of solutions customized for our task of answering event-related questions. A full solution is outside the scope of this work.

There are two types of Petri Net circuits that require attention, as seen in Figure 3-7. In the first circuit, forward analysis requires tokens on all incoming arcs, leading to token creation on all outgoing arcs when the transition is simulated to fire. In backward analysis, though, any token on an outgoing arc is taken as evidence that the Transition may have fired, which requires projecting tokens on each of the incoming arcs. This is a relaxed constraint for cases of incomplete information. Likewise, in

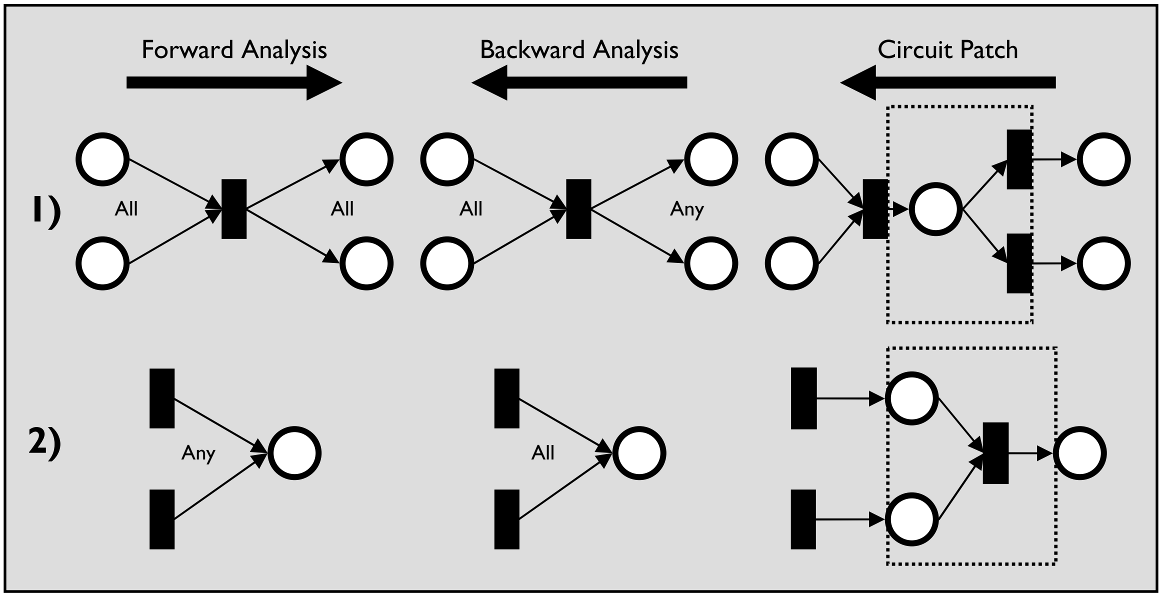


Figure 3-7: Reachability Analysis Token Requirements

the second circuit, forward analysis requires that only one of the Transitions must have fired to create a token in the outgoing Place. In backwards analysis, any token in the Place means either Transition may have fired and thus both possibilities are reachable.

To run a backwards direction analysis, we can reverse the incidence matrices (swap I^+ and I^-) and consult a secondary firing function that reflects this different logic when running reachability analysis. Alternatively, we can keep a modified version of the net that has been patched following the template shown in Figure 3-7 (unit arc weights on added section). Then we can reverse the incidence matrices and run a standard reachability analysis on this net. Both of these methods are computationally expensive compared to reachability in the forward direction, since the new firing semantics (and modified net alternative) lead to more state paths being followed.

The structural solution is unnecessarily expensive for our design, though. In our task, we focus on the state of a limited number of Places (typically only one) per analysis. Rather than analyzing an X-net to determine if a particular *full* marking is reachable from a given state, only a partial marking is relevant; the parts of a full marking that do not influence the distribution of tokens over the Places in focus can be ignored. This assumption allows us to devise a much more efficient algorithm.

We use a modified breadth-first search (BFS) on the net in the forward direction from the target, at each step checking if any Place on the search path has a token in the marking. There are two types of paths we cut short at a transition: a) those where we know the transition cannot have fired due to a known lack of required tokens, and b) those where we know the transition must have been inhibited. The pseudo-code is shown here. Note with respect to case (a), in certain situations, we have evidence that tokens do not exist and can specify this in the net by setting the related Place's capacity to 0. This is further explained in Section 4.4.

```
1.  For each target X-net object (Place or Transition)
    1.1.  Push target on new Queue q
    1.2.  While q not empty (and timeout depth not reached)
          1.2.1.Pull out object o
          1.2.2.If o is a Place,
                1.2.2.1.    For each Transition t where $I^-[o,t] > 0$
                     If
                       a)  for each Place p where $I^-[p,t] > 0$, $c_p \neq 0$ (i.e. p
                           is not guaranteed to have no tokens, which
                           would be specified by setting place capacity to
                           0), and
                       b)  for each Place p where $N[p,t] > 0$ && $\notin t'$
                           where $I^+[p,t'] > 0$, $M[p] < N[p,t]$ (i.e. t is not
                           guaranteed to be inhibited),
                     then push t onto q
          1.2.3.If o is a Transition,
                1.2.3.1.    For each Place p where $I^+[p,o] > 0$
                     If $M[p] > I^+[p,o]$,
                     then tag target as reachable and return to 1.
                     else push p onto q
2.  Return reachability status of targets
```

3.5 Inferring target information

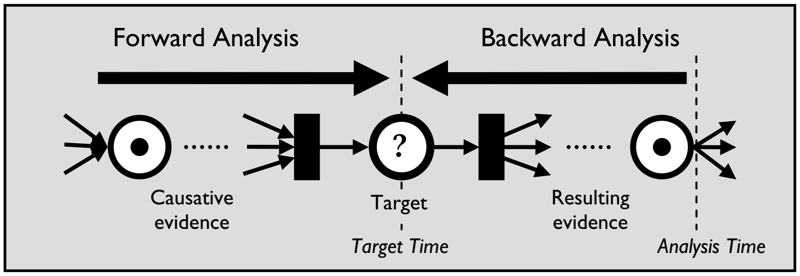
With the ability to transform detailed descriptions of events into active representations on which analysis can be performed, we come back to our main goal of answering complex event-related questions. For four question types (*Justification*, *Temporal Projection*, *Ability*, and *Hypothetical*), we have developed analysis routines that calculate causal chains from evidence to information targets.

While a query may ask about multiple pieces of information and our routines can be modified to meet such a request, we currently focus on and optimize for single targets per question. That target can be an event or a state or resource; for example, the question could be about the act of running (did/can/will *X* run?) or it could be about the state of reaching a goal. For basic event models, this translates into analyzing evidence about the X-net Transition corresponding with an event in question or the X-net Place corresponding with a state in question. When X-nets model the control structure of events (Figure 3-5), questions can be about a fine-grained internal state of an event, again corresponding to a specific Place.

In Chapter 4, we will discuss how to acquire and automatically place evidence into a model. Next, though, we describe how to process evidence based on the question type.

3.5.1 Justification

In Justification questions, Target Time (the temporal point at which the event in question took place) is before Analysis Time (the point at which evidence is collected and analyzed to answer the query). Thus, some available evidence related to the event may have been generated before the target event (information about preconditions and required resources) and some after (information about effects and generated resources), both of which are potentially relevant to the task of attempting to justify a claim in question.



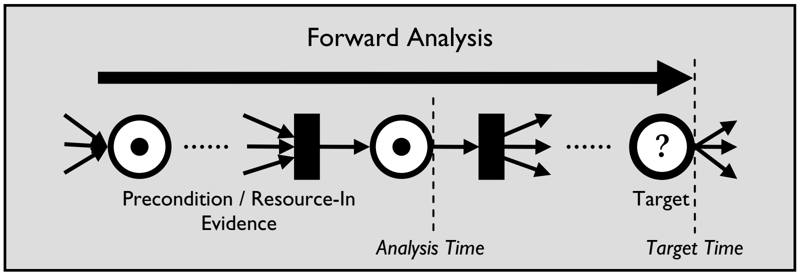
To apply both sets of evidence, we combine forward and backward reachability analysis, to find viable causal chains that would show the possible existence of the state in question. Forward analysis can extend from the 'earliest' evidence in the model up to the target ('early', in terms of causally preceding). Backward analysis can extend from analysis time back to the target (evidence will not exist for events that have not yet occurred).

We can handle negative evidence (evidence that a state did not occur or an action did not take place) by disabling the corresponding Place or Transition. With a Place, we can set its capacity to 0; with a Transition, we can set its rate to 0. This prevents non-viable causal chains from being inferred.

If we find a viable causal chain, we can conclude that there is a possible justification of the claim in question. With a stochastic X-net, we can calculate the probability of that chain.

3.5.2 Temporal Projection

In Projection questions, Target Time is ahead of Analysis Time. We have evidence from before Target Time, but not between Target Time and Analysis Time, nor after Target Time.

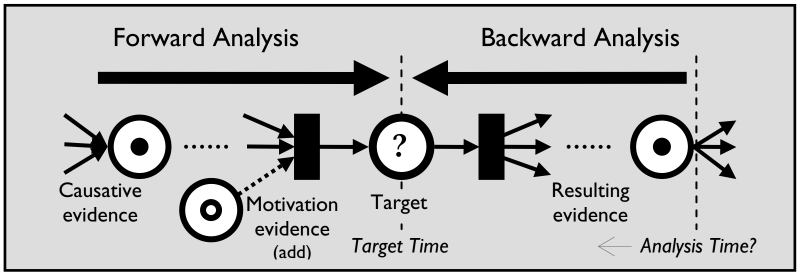


Here we only use forward reachability analysis to find causal chains, using evidence available before Analysis Time to project what may occur going forward, up until the target event.

Negative evidence that relates to events occurring before analysis time is handled similarly to Justification questions, by disabling the related Place or Transition. Negative evidence that relates to events occurring after Analysis Time is ignored (the rationale being, though it did not occur does not mean it will not occur).

3.5.3 Ability

The unique feature of Ability questions is that they take the motivation of the actor in question out of the analysis. Analysis Time can come before or after Target Time. The question can be asking about an actor's ability to do an action in the future (can *X* do *Y*?) or in the past (could *X* have done *Y*?). The latter has an underlying Justification question and can be handled similarly. The former can be handled in one of two ways: 1) taking into account evidence of previous occurrences of the action in question, making the ceteris paribus assumption that previous occurrences do not affect the actor's ability to execute again; or 2) ignoring the evidence. The first choice leads to similar analysis to Justification questions; the second, more restrictive choice leads to similar analysis to Projection questions. An analyst choosing between the two options can look to the resource requirements of the target action for guidance: actions consuming few or easily replaceable resources, for example, may lead to a bias towards (1) – an external Bayesian reasoning system can assist here.



No matter the underlying question type, we need to make two manipulations to the model to prepare it for reachability analysis. For all Precondition *Places* that are tagged as representing motivation of the actor in question that come causally before the target, we add a token and convert all outgoing arcs to Enable arcs. This removes the influence of the motivation, or, more accurately, assumes motivation.

There is one exception: Preconditions that are positive preconditions to some actions (linked by Enable and Resource arcs) and negative preconditions to other actions (linked by Inhibitor arcs). To maintain the modeled mutual exclusivity between the positive and negative cases, analysis has to be run under both cases: 1) Resource and Enable arcs as Enable arcs, with Inhibitor arcs as Inhibitor arcs; and 2) Resource and Enable arcs as Inhibitor arcs, with Inhibitor arcs as Enable arcs. The analysis grows exponentially with the number of mutual exclusion cases related to the motivation of the actor in question. (For most models, this will tend to be very small.)

Once the model is manipulated, if the underlying question is a Justification question or we choose to use evidence of previous ability, we use forward and backward reachability analysis, just as in the Justification question case. For underlying Projection questions, we use forward analysis up to the target.

3.5.4 "What-if" Hypothetical

Ability questions can be construed as one type of hypothetical: if *X* has/had the motivation, will/did *X* do *Y*? Our design can handle other similar questions: if *X* has/had some resource or precondition, will/did/can *X* do *Y*? Specifically, our design supports simple cases of hypothesized added evidence. We add the hypothesized evidence to the known evidence and then analyze the underlying question, be it Justification, Projection, or Ability. Analyzing both the base case and the hypothetical and calculating the delta determines the salient effect of the hypothetical.

There is plenty of future work to be done here around more complex model intervention (Pearl 2001), specifically about identifying stale evidence to be manipulated given hypothesized new evidence. Such a solution is crucial to solving counterfactuals.

3.6 Pathway classification

In Section 3.5, we described analysis routines to infer information targets described within a particular model. Our active event modeling framework also lends itself to analysis for classifying a partially observable model, based on its output, as fitting one of a given set of hypotheses about the its structure and use. The questions that can be answered are thus not only about a specific state or action within a known model, but also about the makeup and use of the model itself. This additional question type requires a different analytic approach, which we detail below.

3.6.1 Pathways and Hypotheses

Consider a complex dynamic system with multiple uses. Let us assume that each use constrains the possible evolution trajectories. We refer to a set of possible evolution trajectories that serve a particular goal as a **pathway**. A pathway is thus an assemblage of complex events that share a single goal. For instance, a metabolic pathway in biology refers to the set of processes that result in a specific set of interacting proteins and possibly a set of outputs. Similarly an automobile production pathway corresponds to a set of related processes that result in the production of an automobile. We can encode one or more related pathways in an event model.

Assume we know multiple related different development pathways and we are interested in ascertaining which of these pathways is actually being pursued in a certain situation. We can call each such known type of pathway a **hypothesis** for the unknown pathway instance. In many circumstances, only some of the individual pathway processes (**pathway segments**) are observable, and further even those segments that are observable can have noise in the measurements of relevant features of the pathways. This can complicate analysis to disambiguate between the hypotheses when attempting to classify the pathway of interest.

3.6.2 Hypothesis Disambiguation

If there are no shared segments between pathway hypotheses, we can just look at the values of observable segments (whether they occur, or how quickly they proceed), and based on this set, decide which of the hypotheses is being pursued.

The more interesting question occurs when some of the segments (both observable and unobservable) are shared between the pathways. In this case, the question is essentially one of hypothesis disambiguation. Given the data and hypothesis, we construct a set of simulations under the different hypotheses and use this as training data for a classifier. The features are pathway segment initiation and durations under the different hypotheses. The classes are the different hypotheses. Using this data, we can train a classifier (e.g. SVM, Naïve Bayes) to learn the mapping from features to classes. Then in an unseen situation (test case), we can use the classifier to assign the most likely class to the observed data.

3.6.3 Probes and optimal probe design

Given the default set of inputs for an unknown pathway, the observations of the evolution of the pathway, made using the default set of observable pathway segments, may be insufficient for hypothesis disambiguation. The observations expected under each hypothesis for that input set may be indistinguishable.

The question then becomes one of constructing **probes**. Probes are external events that are interventions designed to produces effects on the observable segments. They can be **passive** (measurements on observables; e.g. monitoring an additional resource) or **active** (structural or input changes to the pathway; e.g. changing the resource profile for a specific segment). The effect of a probe can manifest itself differently for each hypothesis, ideally creating better separation in the expected observations and thus greater diagnosticity.

Probes inherently trade the cost of probing (how easy is it, how detectable it is, monetary expense, etc.) with the value of the information obtained. We propose a method for calculating the information value of probes. When weighted by the costs of the probes (which are application specific), we can use the information value metric to choose the optimal probe to help us answer *Hypothesis Disambiguation* questions. In Chapter 7, we describe a multi-university project that use our event model to build biological pathway simulations which are then used by another team (from CMU) to build optimal probes for hypothesis disambiguation.

The mutual information between a hypothesis set and a probe

There are two sets of evaluation measures that have to be combined to produce an overall system diagnosticity measure. One is the utility of the model and the second is the utility of a probe given a model.

1. Model Utility: how good is the model in evaluating the posterior probability of multiple hypothesis given a set of observable values.
2. Probe Utility: given the model, how diagnostic are a chosen set of noisy measurements (passive probes) and interventions (active probes).

We define the overall system utility as a combination of the model and probe utility. Next, we outline the evaluation metrics for each of the component utilities.

Evaluating model utility

Our basic question pertains to the degree to which the model is able to diagnose a given sets of observations correctly as evidence of specific hypotheses. Our baseline for evaluation is a prior assignment of probabilities to each competing hypothesis (based on the current best information available). The question then is how much can the model improve over the baseline if it has access to specific observations (time series of values for specific observable variables).

The evaluation method is as follows:

1. An expert provides a prior over the hypotheses.
2. A case generator generates cases that are values of subsets of observables. The case generator (independent from the model) may use a (weighted) combination of:
    a. Expert knowledge
    b. Historical gold standard cases
    c. Sampling techniques (Latin Hypercube Design, model MCMC)

    (The first two have a high weight, the third low weight.)
3. The ability of the model to evaluate the hypotheses (given the prior) is evaluated. We can use any of the following criteria:
    a. Ordinal rankings
    b. ROC curves[2]
    c. Brier scores[3]

[2] Curves of True Positive rate (TPR or Sensitivity) verses False Positive Rate (FPR = 1 - specificity). Here we assume the prior over hypotheses is P(H) and the True Positive probability $P(TP) = P(h) * P(True \mid P(h)) = P(h) * TPF$.

[3] Least Mean Square of misclassification. $\sum_i (P(i) - X(i))^2$, where $P(i)$ and $1 - P(i)$ are the classifier probability for a binary hypothesis and $X(i) = 0,1$ is the correct hypothesis for the instance i

d. AUC (Area under the ROC).

ROC and AUC are especially good: a) they are invariant to skews in the class distributions (change over time); b) they allow discrete and continuous classifiers; c) can be extended to the multiclass case, and c) per example costs can be incorporated. They are widely used in medical diagnosis and in machine learning and classification (Swets 1988; Mossman 1999; Lachiche and Flach 2003).

Evaluating probe utility

With the model utility measurement as a baseline, we wish to determine how much we can improve diagnosticity by a careful design of measurements and interventions. The value is of the information obtained by the probe is thus this *improvement* of diagnosticity.

There has been an explosion of statistical approaches to measuring value of information in recent years within the fields of machine learning and planning (e.g. NIPS 2005 Workshop on Value of Information in Inference, Learning and Decision-Making). We propose the use of entropy reduction for determining the value of probes.

The basic method for measuring entropy reduction is determining the difference between the prior uncertainty over a given set of hypotheses, and the posterior uncertainty over those hypotheses after the use of a given probe (measurement or intervention). Take $H(C)$ as the prior entropy over the different hypotheses C $(c_i \in C)$. The conditional entropy is defined as the posterior entropy given the probe, P, and the result of the probe is a set of observables $Q \in O$ with post probe distributions of their values ($q_i \in Q$).

The reduction in uncertainty (the probe utility) is thus given by the mutual information $I(C,P)$ between the hypothesis C and the probe P:

$$I(C,P) = H(C) - H(C \mid P) \tag{3.1}$$

Decomposing the terms, we get:

$$H(C) = \sum_{c_i} P(c_i) * log \frac{1}{P(c_i)} \tag{3.2}$$

Using the chain rule, we decompose the second term:

$$H(C \mid P) = H(C \mid Q) + H(Q \mid P) \tag{3.3}$$

Here, the first term is the conditional entropy in the hypotheses (C) given the observation set (Q), and the second term is the sensor model that provides the conditional entropy of the observations (Q) given the probe (P).

Expanding the first term of Equation (3.3), we get:

$$H(C \mid Q) = \sum_{q_j} P(q_j \mid Q) * H(C \mid q_j) \tag{3.4}$$

$$H(C \mid q_j) = \sum_{c_i} P(c_i \mid q_j) log \frac{1}{P(c_i \mid q_j)} \tag{3.5}$$

Combining Equations (3.4) and (3.5) we get:

$$H(C \mid Q) = \sum_{q_j} P(q_j \mid Q) * \sum_{c_i} P(c_i \mid q_j) log \frac{1}{P(c_i \mid q_j)} \tag{3.6}$$

Using the definition of entropy, $H(Q \mid P) = P(Q \mid P) log \frac{1}{P(Q \mid P)}$ we get the overall conditional entropy of the hypothesis given the probe ($H(C \mid P)$).

$$H(C \mid P) = \sum_{q_j} P(q_j \mid Q) * \sum_{c_i} P(c_i \mid q_j) log \frac{1}{P(c_i \mid q_j)} + P(Q \mid P) log \frac{1}{P(Q \mid P)} \tag{3.7}$$

Plugging this back into Equation (3.1), we get the total change in entropy for the probe given a model which is the diagnosticity of the probe.

Still to be explained is the sensor model, $P(Q \mid P)$. The two types of probes, measurement (passive probe) and intervention (active probe), have different sensor models, which we describe in turn.

For an intervention (into the real world), we do not know which observables are going to be impacted, nor how they will be impacted (what the new distribution is over the impacted observables). By contrast, in the case of a measurement, we know (by definition) which variables are being observed and can have a model of measurement errors to generate the new distribution over possible values of the observed variables.

Thus, for interventions, there is uncertainty over which variables are impacted and the distributions over the observables given the impacted variables. Let O be the set of observables, and $P(Q \mid P)$ $(Q \in O)$ the probability of an observable variable Q being impacted by the probe.

$$P(Q \mid P)_i = \sum_{Q \in O} P(Q \mid P) * \sum_{q_j \in Q} P(q_j \mid Q, P) \qquad (3.8)$$

This is the sensor model of the intervention probe and its impact on observables. This model has to be designed using understanding of the effect of the probe on the system. The model can be explicit or can be acquired from domain expertise.

In the case of a measurement, we know which variables are impacted (Qs), leading to:

$$P(Q \mid P)_m = \sum_{q_j \in Q} P(q_j \mid Probe(q_j)) \qquad (3.9)$$

Here $P(q_j \mid Probe(q_j))$ is the measurement error model (what is the probability that the value q_j obtains when a probe returns a value q_j). This model can come from domain expertise and sensor models of measurements, but also from model exploration to estimate the measurement parameters.

Plugging Equation (3.8) as the value of $P(Q \mid P)_i$ into Equation (3.7) gives us the measure of the diagnosticity of an intervention. Plugging Equation (3.9) as the value of $P(Q \mid P)_m$ into Equation (3.7) gives us the measure of the diagnosticity of a measurement probe.

Calculating the optimal probe

We can thus use Equation (3.1) (expanded, as above) to test the value of information of each potential probe. Weighting $I(C,P)$ for each P by the cost of P, we can calculate the overall utility for the probes, and select the best one for a given question. Chapter 7 details the use of the pathway model in a multi-university effort to determine optimal probes for a complex set of biological pathways.

4 Bridge to Applications

Chapter 3 laid out the key theoretical components of our event modeling and reasoning framework, as well as our goals for utilizing this platform to answer complex-but-common questions related to the structure of events (Section 3.2). To fulfill our goal and build applications to answer such question, though, requires the addition of a number of tools and techniques for managing and interacting with event models.

Our design uses a structured representation of language based on FrameNet frames to mediate between natural language and event models (4.1). Our representation extends the general structure of frames with domain specific ontologies and named entities, allowing us to better constrain the information that is extracted from a question and any text used to answer it (4.1.3). Frames of this form extracted from a question can be used for selecting a model of the event in question (4.3). Furthermore, frames of this form extracted from evidence related to a question can be used to add data to a pertinent model (4.4).

The ability of an application to process questions using our representation requires certain preprocessing of the natural language input, which we have identified (4.2). We have also devised a mechanism by which an application can extract information related to an answer, after it has been inferred using our event reasoning framework (4.5). In addition, we have guidance for model designers on methods to populate the event ontology, which is necessary for any event reasoning to take place using our framework (4.6).

This work comes to fruition in the applications we describe in Chapters 5 (Answer Selection), 6 (Question Answering), and 7 (Pathway inference).

4.1 Interface between language and models

Event analysis requires a dynamic event model that a) captures the structure and the set of possible evolutions of an event and b) incorporates state information resulting in a specific instantiation for simulation and analysis. An application thus need a means of finding an appropriate event model for a desired analysis, and a means of finding and incorporating state information into the model.

Natural language is both the easiest form for users to specify their analysis requests and it is also the form in which significant amounts of event-related knowledge is available (corpora, etc). Unfortunately, natural language is capable of encoding an unrestricted set of concepts in a wide variety of forms. In its raw form, it is too unwieldy to map directly to models; we need a formal, precise intermediate representation to work with, instead.

Fortunately, there are now robust efforts to develop open domain semantic resources that we can exploit for this purpose. As we have discussed, events inherently have frame semantic structure. We chose frames as the intermediate structured language in our design. We can use this structured representation on two fronts: to describe queries and data, and to describe event models. In our frame-based approach to answering event-related questions, we use FrameNet (Fillmore, Johnson et al. 2003), which we explore next.

4.1.1 Background on Frames and FrameNet

In 1976, Charles Fillmore proposed that the meaning of a word in a phrase is based on its relation to its semantic frames (Fillmore 1976). Frames he defined as "schematic representations of the conceptual structures and patterns of beliefs, practices, institutions, images, etc. that provide a foundation for meaningful interaction in a given speech community" (Fillmore, Johnson et al. 2003). For us, frame annotations of a phrase are formal, precise structures that can capture the meaning from the words in the phrase and the relational form that binds them together.

In our work, we employ FrameNet, a lexico-semantic database of frames in the English language. The FrameNet project works to identify significant frames in language, their frame elements, as well as lexemes that evoke those frames. In addition, the project maintains a corpus of annotated sentences that demonstrate the use of these frames in language.

FrameNet is a mature research project devoted to the specification and analysis of frames in English and other languages. A number of other groups research tools to improve and simplify automated frame annotation of natural language text (e.g. University of Saarlandes' Shalmaneser tool). Coverage and performance of these resources should only improve over time.

FrameNet frame attributes

FrameNet frames have a number of key components. Each frame is tagged with a name: a mnemonic descriptor related to the target concept represented. In addition, frames have an argument structure of thematically-tagged variables known as **Frame Elements** (FEs), each describing the types of entities that can participate in the frame. Certain FEs are labeled as being **core** to the frame, having the property that without a participant semantically filling the specified role, the event described in the frame could not occur. Some FEs have **Semantic Types** that provide guidance of acceptable roll fillers using very high level ontological categories (mappable to WordNet's synset hierarchy). FrameNet also documents frame-to-frame relations, including inheritance, causation, part-of, perspectivization, and temporal ordering. Each frame comes with the **Lexical Units** (LUs) that **evoke** the frame, and annotated sentences demonstrating the use of the frame in various syntactic forms. A full description of these attributes can be found here (Ruppenhofer, Ellsworth et al. 2006).

The end result is a frame like Commerce_buy, which describes "a basic commercial transaction involving a buyer and a seller exchanging money and goods, taking the perspecitive of the buyer". It is evoked by lexical units like 'buy' and 'purchase'. (FrameNet: http://framenet.icsi.berkeley.edu)

Frame: Commerce_buy

Element (core):	Buyer
Element (core):	Goods
Element:	Duration (Type: Duration)
Element:	Manner (Type: Manner)
Element:	Means (Type: State_of_affairs)
Element:	Money
Element:	Place (Type: Locative_relation)
Element:	Purpose (Type: State_of_affairs)
Element:	Purpose_of_goods
Element:	Rate
Element:	Reason (Type: State_of_affairs)
Element:	Recipient
Element:	Seller (Type: Source)
Element:	Time (Type: Time)
Element:	Unit
Inherits From:	Getting
Is Inherited By:	Renting
Perspective on:	Commerce_goods-transfer

Frame annotations

Frames are, in and of themselves, abstract schemas. It is through frame annotation that meaning in a sentence is extracted in frame form.

Take, for example, the sentence "Alice and Bob purchased a car from the dealer at the Ford showroom." Evoked by the predicate, "purchase", a frame annotation of this sentence will result in the frame:

Frame: Commerce_buy

Buyer:	"Alice and Bob"
Goods:	"a car"
Seller:	"the dealer"
Place:	"the Ford showroom"

This frame instance **binds** a subset of its elements to words in the sentence that fulfill corresponding roles. In this case, "Alice and Bob" are determined to fill the Buyer slot; "a car" fills the Goods slot, "the dealer" is the Seller, and "the Ford showroom" is the Place. The frame thus captures the actors, the theme, their roles, and the relations between them in this sentence.

Frames abstract away (or provide the means to abstract away) from a number of natural language idiosyncrasies, including:

- Predicate word choice – collapsing synonyms to a single representation
- Syntactic phrasing – collapsing various syntactic realizations of the same concept
- Perspective of actors – collapsing to the unperspectivized event

The above sentence could have used the predicate "to buy" rather than "to purchase", without affecting the annotation. So too could it have moved some of its clauses around, like moving the location to the beginning ("Alice and Bob, at the Ford showroom, purchased a car from the dealer."), without significant effect to the meaning of the sentence and thus no change to that frame annotation. Moreover, it could have been written from the perspective of the dealer selling Alice and Bob a car, which would connect back to the same unperspectivized 'Commerce_goods-transfer' frame that 'Commerce_buy' does, with approximately the same role bindings.

(Methods used in the annotation process are beyond the scope of this dissertation.)

4.1.2 Frame matching

We use frames with bindings as the functional units for linguistic descriptions in our designs: for event-related queries, for evidence required for deriving answers to a query, and for event models.

Frames provide us with structured representations of language. We can compare and match not only which frame is used to describe two pieces of data, but also the bindings of roles essential to a particular analysis. This allows us to extract crucial relational information from language to select and instantiate event models.

In the simplest case, we can attempt to match query frames to evidence frames directly. To demonstrate the approach and its drawbacks, let us consider one of the questions that was used in an evaluation we will discuss later (Chapter 6), regarding the subject of weapons production:

> What countries have provided Iran with ballistic missiles and missile-related technology?

The passage provided with a possible answer to the question was:

> The continued willingness of the Democratic People's Republic of Korea (DPRK), the People's Republic of China (PRC), and Russia to provide Iran with both missiles and missile-related technology that at the very least exceed the intentions of the Missile Technology Control Regime (MTCR). This has been complemented, to a lesser extent, by the willingness of other nations (e.g., Libya and Syria) to cooperate within the realm of ballistic missile development.

Frame analysis of the question leads to the following pertinent frame:

Frame: Supply

Supplier:	"What countries"
Recipient:	"Iran"
Theme:	"with ballistic missiles and missile-related technology"

and analysis of the provided passage leads to this frame:

Frame: Supply

Supplier:	"the Democratic People's Republic of Korea (DPRK), the People's Republic of China (PRC), and Russia"
Recipient:	"Iran"
Theme:	"with both missile and missile-related technology"

The annotation extracts structures that humans can see yields an answer to the question posed: the relational constraints of the question frame are fulfilled by the passage frame. Unfortunately, while we can automatically determine that the frames are of the same type (Supply), the role bindings are still unstructured and do not match up perfectly. The Recipient and Theme bindings refer to common objects across the two frames, but in the case of the Theme, use slightly different strings. Matching frames to frames directly is insufficient for our needs.

4.1.3 Linking entities to bindings

We would like to be able to match related frames, but, as mentioned above, raw frame annotations bind strings to frame roles. There are many surface realizations of the concepts underlying the strings, which are difficult to anticipate in advance, making it difficult to find links between frames that match in meaning but not form. Furthermore, frame representations of natural language phrases do not automatically incorporate domain related knowledge such as domain ontologies and constraints. To get to a form for our intermediate representation that provides for robust semantic comparison between items, we extended FrameNet frames to incorporate domain ontologies of entities that capture relations between entities.

Named entity classes

Named entity (NE) tagging provides an off-the-shelf solution for collapsing strings to a unified representation of underlying concepts, in this case named entity types. We can link the text string bound to frame elements with the NE class associated with the semantic head of the string. Unfortunately, our investigation of easily available named entity recognizers (NERs) found the range of classes produced by each to be inadequate. For example, the ACE program specifies five entity types (Person, Geo-Political Entity, Organization, Facility, and Location); MUC-7 specifies seven; IREX has eight (Chinchor 1998; Sekine and Isahara 2000; Doddington, Mitchell et al. 2004). With the number of classes in the range of five to ten, and the fact that NERs are technically only supposed to capture rigid designators (like proper names), the best frame-and-entity annotation of our question, "What countries have provided Iran with ballistic missiles and missile-related technology?", would be:

```
Frame: Supply
  Supplier:     <  >
  Recipient:    <Geo-Political Entity>
  Theme:        <  >
```

with analysis of the passage providing the frame:

```
Frame: Supply
  Supplier:     <Geo-Political Entity, Geo-Political Entity,
                Geo-Political Entity>
  Recipient:    <Geo-Political Entity>
  Theme:        < >
```

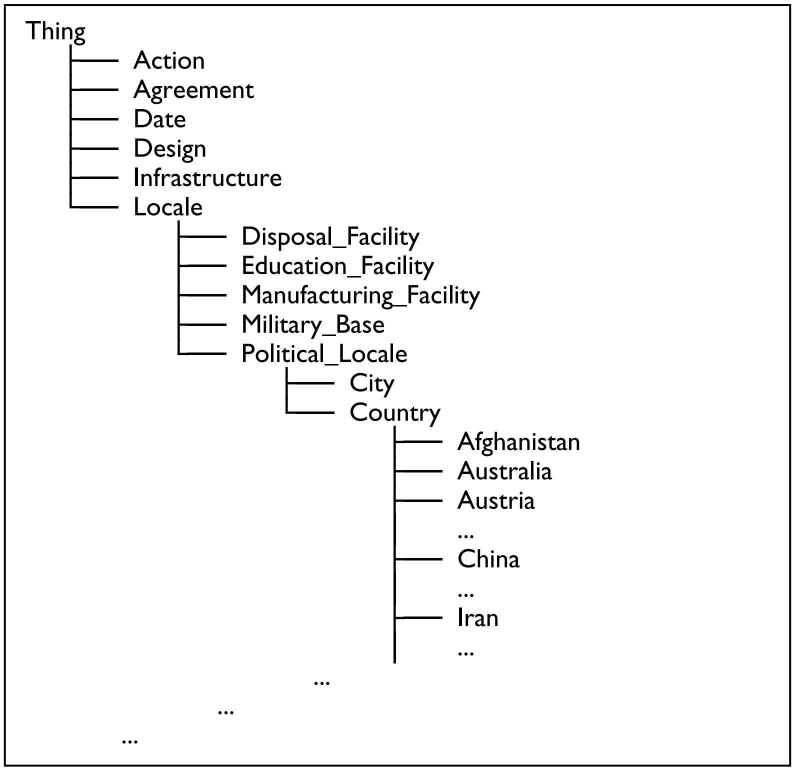
(As here, we use brackets to denote entity classes. These classes shown are ACE tags.)

Obviously, the representation is impoverished in both breadth and depth of coverage of concepts. These named entities are insufficient for representing important fine-grain distinctions in any particular domain.

Hierarchical domain ontology

Our design, instead, requires the use of fine-grained hierarchical domain ontologies that cover detailed information about the domains in question. Such an ontology captures general and specific types and super-type/sub-type relationships, amongst others.

For our evaluation dealing with the topic of Weapons Production, we developed our own hierarchical concept ontology of approximately 250 classes, up to seven levels deep, coded in OWL (McGuinness, Van Harmelen et al. 2004). It encodes inheritance relationships, as well as part-of and representative-of relationships that provide a basic level of support for metonymy reasoning. Here is a partial snapshot of the ontology:



The level of detail of the ontology, frame and ontological entity analysis of our example question leads to the following frame:

```
Frame: Supply
  Supplier:     < Country >
  Recipient:    < Iran >
  Theme:        < Ballistic_missile >
```

and analysis of the passage leads to this frame:

```
Frame: Supply
  Supplier:     < North_Korea, China, Russia >
  Recipient:    < Iran >
  Theme:        < Missile >
```

With the domain ontology, we are able to automatically reason that a) North Korea, China, and Russia are all subtypes of Country, and thus match with the question frame's Supplier element; and b) that ballistic missiles are a type of missile; thus the passage frame's theme may be relevant.

Access to hierarchical domain ontologies and tools to tag strings with ontological entities is *essential* for effective frame matching.

To provide a starting point for domain ontology creation, there are publicly available ontologies to use and extend, including WordNet's synset hierarchy, the SUMO/MILO ontology, and Cyc (Lenat and Guha 1990; Fellebaum 1998; Niles and Pease 2001). In our work, we only did a cursory exploration of using these ontologies; it is an area for future work.

4.1.4 Semantic relevance matching

A structured representation of language content facilitates the comparison of questions and data, as we have shown. For a representation using frames with entity fillers, we have developed a simple, formal method of comparison to determine similarity and relevance of a candidate frame to a target frame. (Here, the *target* frame represents the search template that we use to find data of relevance; potentially relevant data is represented by the *candidate* frame.)

We require that a target frame and candidate frame match in terms of frame name. Then, for each frame element (FE), we look at the entities bound and compare and score using a set of scoring parameters ($\text{penalty}_{\text{gen}}$, $\text{penalty}_{\text{spec}}$, $\text{penalty}_{\text{miss}}$, $\text{penalty}_{\text{mismatch}}$; different penalties can be used for core FEs and non-core FEs). FE match scores range from 0 to 1, real-value.

1. If the entities match perfectly, that is a full FE match.
   $\text{score}_{\text{FE}} = 1$
2. If the target's entity type is more general than the candidate's entity, the FEs match imperfectly, and the FE match score is penalized, proportional to the degree of separation in the type super/sub-type hierarchy.
   $\text{score}_{\text{FE}} = (1 - \text{penalty}_{\text{gen}})^{\text{deg}}$
3. Similarly, if the target's entity type is more specific than the model entity's, the FEs match imperfectly, and the FE match score can be penalized, proportional to the degree of separation in the type hierarchy.
   $\text{score}_{\text{FE}} = (1 - \text{penalty}_{\text{spec}})^{\text{deg}}$
4. If the target frame has an element filled that the candidate frame does not, either the FE match is 0 or the full frame match is rejected or otherwise severely penalized.
   $\text{score}_{\text{FE}} = 0 \mid \text{score}_{\text{Frame}} = 0 \mid \text{score}_{\text{Frame}} \mathrel{-}= \text{penalty}_{\text{miss}}$
5. If the candidate frame has an element filled that the target frame does not, the FE is ignored.
6. If the target's entity and candidate's entity do not match and have no inheritance relationship, either the full frame match is rejected or it is severely penalized.
   $\text{score}_{\text{Frame}} = 0 \mid \text{score}_{\text{Frame}} \mathrel{-}= \text{penalty}_{\text{mismatch}}$

Note that relevance is asymmetric, as a candidate frame describing a state or action more specific than a target frame may be deemed more relevant than a case where a candidate frame is more general than a target. Frame match scores are normalized sums of their constituent FE match scores (normalized to 1 by dividing by number of FEs in target frame).

The algorithm can be extended to use relations beyond super-type and sub-type to establish a partial match between entities. As mentioned, relations like part-of and representative-of, are two that we have explored that help in many metonymic situations. (<George Bush> instead of <United States>, for example, matching a <Country> entity restriction.)

4.1.5 Linking event scenario models to frames

The ability to use frames to directly connect questions about complex events to 'smoking-gun' answer data only exists in limited situations. Our work focuses on using simulation and dynamic event analysis to facilitate the use of indirectly related information to answer such queries. We design for the use of frame annotated queries to guide the selection of a relevant event model; this model, in turn, provides guidance on the (frame annotated) data that can be used to answer the question.

As briefly mentioned in Section 3.4.2 (Figure 3-4), our dynamic models contain a number of frame 'hooks' that attach to the Places and Transitions. These frame hooks act as linguistic descriptions of the states, resources, and actions represented by the Places and Transitions. For our purposes, the hooks are also patterns that match the entity-bound frames of questions and data. The hooks themselves are entity-restricted frames. We distinguish "entity-bound" frames from "entity-restricted" frames to denote the difference in derivation. "Entity-bound" frames are typically extracted from analyzed text and are as specific as possible in terms of the constituent frame and entities used to represent the desired meaning. "Entity-restricted" frames are defined at the time of model creation, and are as general as possible, yet specific enough to match only what is relevant to the model. They are structurally the same: frames with entity role fillers linked to hierarchical domain ontology entities.

When designing a model, a modeler has a desired meaning in mind for each state, resource, and action they depict. Attaching frame hooks to these units formalizes the meaning such that the model can interact with queries and data. With the ability to select specific frames and entities to describe each unit, the modeler can calibrate which information matches and which information is deemed too general or irrelevant.

Continuing our weapons production example, an event model can depict the high-level process of manufacturing a weapon or the specific process of manufacturing a ballistic missile. The first model is relevant to a all weapons, while the second is only relevant to ballistic missile production. The same frame, Manufacturing, can be used in describing both models, but its Product element should be restricted to the ontological entity Weapon in the first case, and Ballistic_missile in the second.

Abstract models

A SME may wish to design an abstract model, like one of the process of a person buying a car, that can cover multiple instances with different buyers and different cars. To do so requires not just using a general entity category restriction in all relevant frame hook slots across the model, like <Person> for all buyer slots. Also, it requires linking all such restrictions together such that if one restriction is tightened, all related restrictions are tightened – if we wish to specialize the Person Buying Car scenario model to apply specifically to Joe, any place the buyer has a role should have a tightened restriction of <Joe>.

Our approach is to use a layer of indirection, setting entity-restrictions in frame hooks to an entity category *variable*; subsequently, that variable can be set to, in this example, originally <Person>, then <Joe>.

```
Frame: Commerce_buy
  Buyer:    v_buyer
  Goods:    v_car

Frame: Commerce_pay
  Buyer:    v_buyer
  Goods:    v_car

Specialize
v_buyer      = <Person>  →   = <Joe>
v_car        = <Car>     →   = <Ferrari>
```

Using this technique, it is easy to manage interrelated entity restrictions across a model. Note that this only works in one direction; models can be made more specific in their entity restrictions and still be structurally accurate, but that need not be true in the reverse.

Abstract models are a key component of our overall event modeling framework, providing a tool for tractable coverage over multiple event instances without the need of a SME to create unique models for each instance.

4.2 Determining data and simulation goal: Question Analysis

Before simulation and event analysis can commence in the service of answering a question, certain processes are required of other systems to analyze the question. For one, the frames and entities in the question have to be extracted, so as to provide a form that can be compared to model frame hooks and frame-encoded data. In addition, the question type has to be ascertained, be it *Justification*, *Prediction*, *Ability*, *Hypothetical*, or *Hypothesis Disambiguation*. Also, some determination has to be made on what, if any, post-processing is required to extract the exact answer from the structure resulting from event simulation.

Question Frame and Entity Analysis

Question frames (extended with domain entities) can be used to help determine which event model out of a set would be most useful for analyzing an event in question. It can furthermore pinpoint which action or state in that model is specifically in question. To acquire frames and entities from a raw text query requires extraction tools for each.

Many groups are actively working in the area of semantic parsing (aka semantic role labeling, semantic analysis), and for our work we wished to use the best existing tool to extract frames from text. Specifically, we looked to the work of University of Texas at Dallas (Harabagiu, Bejan et al. 2005) and Saarland University (Erk and Pado

2006) for two approaches for automated frame analysis. Our work used the resulting analyses of the UTD system, as well as hand generated, gold standard parses by the FrameNet group.

Entity tagging is also an active area of research. Stanford University has recently done some interesting work (Klein, Smarr et al. 2003; Krishnan and Manning 2006), as one example. Unfortunately, no off-the-shelf approaches were able to perform with sufficient accuracy at the fine-grained level we needed. For our work using our Weapons Production ontology, we used a customized in-house entity tagger created by a member of our research group, Javier Rey. We used this tagger in conjunction with hand generated and corrected analysis.

Question Type Classification

The question type directs which event analysis method is used, as described in Section 3.5. While the complete technique of classifying the question type of a raw text query is outside the scope of our research, note that there are a number of lexical cues that highly correlate with certain question types. For example, ability questions typically have the words "can", "could", "able", "capable", etc. in them. Hypotheticals have words like "if". Predication questions are frequently in the future tense. For our work, we used mostly hand generated question type analysis, created internally and by UTD.

Support for peripheral event-related questions

Many event-related questions ask a fundamental question about whether a state or action did, can, or will exist. Many other event-related questions ask about a specific detail of an event that is not critical to simulation or other dynamic analysis, but may be contained in structured data generated by simulation. An example of this might be a question about the location of an event ("where *x*"). To take structured data resulting from an event analysis and extract from it one particular piece of interest requires a postprocessing step to be applied to that structure.

That step can be a relational lookup, like for a location. It can be a token count, like for a question about the size of a resource ("how many"). It could also be a simple token existence test, like for a basic *Justification* question ("did *x*"). For example, in the question *"Did Joe buy a car?"*, the test is simply, was the *Joe buy car* action reached? Alternatively, the question could be, *"What car did Joe buy?"*, in which case the answer should be a relational argument, specifically the information bound to the Goods frame element of the Commerce_buy frame attached to the *Joe buy car* action.

In this work, we delegated post-processing of data for these type of questions to an external presentation engine that used hand generated functions.

4.3 Selecting Models for Questions

To use event analysis to assist in answering a question, we require a method to select an event model based on the question being asked. Our approach, as alluded to in Section 4.1.5, is to search for matches between the frame representation of the question and frame-form descriptions of our event models in our event ontology.

Scoring models

Our simple heuristic approach extends scoring semantic relevance of an individual frame to another, to scoring a match of frames from a question to frames from a model. The easiest method is to look at question frames as one set and model frames (all the frame hooks connected to states, resources, and actions for a particular dynamic model) as another set, and then calculate the largest set intersection. Scoring a set intersection is taking the sum total of frame matches (Section 4.1.4) between the sets.

```
For each model
    For each question frame
        Find potential matches of qFrame in model
            totalFrameScore + = thisFrameScore
    Score[modelID] = totalFrameScore
Return $\text{argmax}_{ID}$ Score[ID]
```

This method values a match between any question frame and any event model frame equally, not taking into account whether the model frame represents a peripheral or core component of the model. The scoring approach can be further strengthened by weighting the value of a frame match by a measurement of the relative relevance to the model of the particular state/resource/action represented by the event frame matched. This can be dynamically approximated by the centrality of the model component in the model graph; or it can be supplied as an extra model parameter by the model designer. The highest scoring event scenario model is used for analyzing the event in question.

Instantiating an abstract model

Abstract models (Section 4.1.5) must be instantiated after selection, binding the model to the specific event instance asked about in the question. This is carried out based on the frame matches used to select the model. For each frame match between question frames and model frames (i.e. the frame match score is above zero), the frame element restriction of the model is compared to the frame element binding of the question, and altered according to the following algorithm:

```
1.  For each pair of matching frames {modelFrame, questionFrame}
    1.1.  For each FE, fe, of the questionFrame
        1.1.1.If questionFrame[fe] binding is more specific than
              modelFrame[fe] restriction,
            1.1.1.1.    If modelFrame[fe] restriction is basic domain entity,
                         Set modelFrame[fe] restriction to questionFrame[fe]
                         binding
            1.1.1.2.    If modelFrame[fe] restriction is through a variable,
                         Set modelFrame[fe] variable to questionFrame[fe]
                         binding
        1.1.2.If modelFrame[fe] restriction does not exist,
              Set modelFrame[fe] restriction to questionFrame[fe] binding
```

Note, as stated in Section 4.1.5, restrictions in a model are only made more specific; never more general.

4.4 Acquiring and Incorporating data

An event model selected for a question, whether abstract or specific, requires data about the event in question in order to provide analysis. These data can be pieces of contextual background information (e.g. Alice owns a car) or control information (e.g. at the time to be analyzed, Alice is at home). Combined, we can simulate and analyze the particular event at a particular state to infer new information (e.g. Alice is able to drive to work).

Data for analysis

Pieces of information in X-nets are represented by tokens. These tokens represent either, a) a condition being satisfied, or b) a quantity of a resource, depending on the meaning of the Place where it resides (recall Section 3.4.2). The goal in searching for information about an event instance is finding evidence that informs the model about the state or resource represented by each Place of interest in an X-net.

4.4.1 Acquiring relevant data through query expansion

In selecting a model based on question frames, not only is a model chosen, but also information targets within the model: specific Places and Transitions within frame hooks that match the query frames. If direct evidence about these information targets can be found, no further event reasoning is needed to answer the question. However, for those situations where X-net analysis is needed to infer these information targets, the analysis routines (described in Section 3.5) require contextual information. The more information that can be added to an X-net simulation, the more thorough an analysis can be completed.

Models, though, can be large, and processing information requests about every aspect of a large model can be expensive. Each piece of data with the potential of enlightening an X-net about contextual information sought, requires frame and entity extraction. To limit the scope of information requests, we can dynamically select subsections of the model most relevant to the information targets of the question. Relevancy of a particular Place or Transition in the X-net is directly proportional to the causal influence that X-net object has to the X-net objects representing the information targets.

We select the *contextual* information targets using a depth-limited breadth-first search:

```
1.  Initialize new Set queryExpansion
2.  Push all query targets on new Queue currentExpansionBwd
3.  For depth = 0..max                // search causal predecessors
    3.1. Initialize new Queue nextExpansion
    3.2. For each target object o in currentExpansionBwd
         3.2.1. Add o to queryExpansion
         3.2.2. For each object o′ where
                    (if o is a *Place*) $I^+[o,o'] > 0$ (or)
                    (if o is a *Transition*) $I^-[o',o] > 0$,
                add o′ to nextExpansion
    3.3. currentExpansionBwd = nextExpansion
4.  Push all query targets on new Queue currentExpansionFwd
5.  For depth = 0..max                // search causal successors
    5.1. Initialize new Queue nextExpansion
    5.2. For each target object o in currentExpansionFwd
         5.2.1. Add o to queryExpansion
         5.2.2. For each object o′ where
                    (if o is a *Place*) $I^-[o,o'] > 0$ (or)
                    (if o is a *Transition*) $I^+[o',o] > 0$,
                add o′ to nextExpansion
    5.3. currentExpansionFwd = nextExpansion
6.  Return combined set of frame hooks from queryExpansion set
```

The algorithm selects all of the X-net objects that have a causal path to the objects representing the information targets of the query, within a certain path distance. (The depth parameter quantifies the tradeoff between search completeness and expense.) These context objects are chosen because of their potential affect on the analysis routines used to infer the question targets. The algorithm harvests the frame-hooks for each object, combining them into a single set. This set represents a query expansion (beyond the original information targets), containing structured information requests in entity-restricted frame form for contextual data required for analysis.

Note that the algorithm should be adapted slightly when selecting contextual information targets for answering a *Temporal Projection* question. As discussed in Section 3.5, our method to analyze models for *Projection* questions only touches X-net objects that are causally before the objects of the analysis targets. As such, sections 4 and 5 are not required in the above algorithm, because that information is not used.

4.4.2 Incorporating data

Data found matching information targets of a model (from the question or from context), can be incorporated into the model in the form of tokens or changes to the behavior of the model. When doing so, it is important to distinguish evidence that explicitly supports a condition being satisfied, evidence that explicitly states that a condition is not satisfied, and a situation where there is no evidence. We have developed the following table for incorporating data when matches are found between model query frames (e.g. frames from a model-based expanded query, as above) and data frames (e.g. frames extracted from data retrieved based on the expanded query):

1. If the query is about a **state**,
   evidence of the state existing will result in a token added to the corresponding Place.
2. If the query is about a **resource**,
   evidence of the amount of resource existing will result in a comparable number of tokens (in $\mathbb{N}_0$) added to the corresponding Place.
3. If the query is about an **action**,
   evidence of the action taking place will result in a pseudo-'firing' of the corresponding Transition: without consuming tokens on incoming arcs, resources and states created by the action are added to the Places on the outgoing arcs. (recall X-net semantics: Section 3.4.1)
4. If the retrieved evidence is explicitly negative
   (e.g. the queried fact is not true; there is no resource)
   - For *Justification* questions:
     - a state or resource:
       the capacity of the corresponding Place is set to 0
     - an action:
       the firing rate of the corresponding Transition is set to 0
   - For Non-*Justification* questions:
     - a state or resource:
       any existing marking is removed from the corresponding Place
5. If there are multiple references to evidence about:
   - a state: only one token is added
   - a resource: the largest referenced quantity is used
   - an action: the action is only fired once

4.5 Extracting and composing answers

Applications using our event reasoning framework require a means to extract data relevant to the answer of a question after completing an analysis of the underlying event. While we are not tackling natural language generation, we have developed a method to return more detailed and complete answer data.

Users rarely want a one bit answer, "yes" or "no", to a question; instead they frequently wish for a small amount of context that provides insights to answer. Similar to the query expansion algorithm described above in Section 4.4, it is possible to dynamically and automatically select a set of contextually relevant actions and states closely causally related to (and including) the state or action in question. We call this the Answer Structure. The Answer Structure is a feature structure with slots for information related to each of the designated contextually relevant actions and states. The goal of a system can then be to fill this Answer Structure, instead of just the information directly targeted in the question. After all analysis is complete, information about each element of the Answer Structure can be returned. This comes in a few flavors: the information was retrieved from a data source, the information was inferred through X-net analysis, or no information was found. The last group (no information found) can act as feedback and can lead to a second round of searching for the missing information (e.g. from additional resources). How the elements of the Answer Structures can best be presented is application dependent and an open question beyond the scope of our research.

4.6 Building Models

Event scenario models can represent a wide range of situations – everything from basic embodied actions, like *walking*, *grasping*, *kicking*, etc., to higher level processes, like device manufacturing, to specific historical accounts, like the D-Day Battle of Normandy in WWII. We require a tractable solution to populate our event ontology with models such as these, as needed. This entails a means to manually generate models and/or mechanisms to semi-automate or fully automate the creation of basic models within a domain of interest.

4.6.1 Manual generation

The process of creating a model can be split into two stages: describing the event structure, and providing the frame hooks to ground the events in structured language. A Subject Matter Expert (SME) is required for each.

A model designer needs either a tool for describing events (Section 3.3) and a tool for converting the description into an X-net (Section 3.4.2), or tool to directly create X-nets. In addition, the designer also needs a tool to load descriptions or X-net models and append frame hooks to individual event objects.

PIPE2 Editor and OWL Translator

For our work, we modified a version of the Platform Independent Petri-net Editor 2 (PIPE2) (Akharware 2005), a Petri Net editing package, as described in Section 6.6. This provided us a GUI for the creation, saving, loading, and editing of X-nets. Our package also provides for adding entity-restricted frame-based linguistic descriptions of each Place and Transition in a model.

In addition, a member of our team, John Denero, created an OWL-S–to–X-net translator, providing the means for event descriptions encoded in OWL-S (the semantic markup language for services: Martin, Burstein et al. 2004) to be translated into X-nets. This allows for 1) event models to be described using popular packages like the Protégé Ontology Editor (http://protege.stanford.edu); and 2) those descriptions to automatically generate X-nets.

Model-language design approach

In creating models, each model element requires a linguistic description using frames and entities that describe the represented state, action, or resource. We used a couple of techniques to generate those entity-restricted frame descriptions.

For frames, first, we take a word or a few different words that describe the state or action and look up which frame(s) they evoke, using FrameNet's lexical-unit-to-frame map. Then, for each frame, we use the FrameNet frame-to-frame relations to find related frames that may also be employed in language used to describe this action (many related frames will be relevant to the event at hand, some will not).

Once a set of frames is selected, the frame elements need to be restricted with the appropriate domain entity category, the most general entity category that still restricts the model to appropriate data. We use keyword search to find entities in our domain entity ontology.

4.6.2 Auto-generation

Frame Translation

As mentioned, events have frame semantic structure: frames have many of the same parameters as events, allowing us to directly map between frame elements and event parameters (see Chapter 3). This also means we can use pre-existing frames as a starting point to generate primitive event models. With FrameNet available in OWL format, we can use the OWL-S-to-X-nets tool described in the previous section to semi-automate this translation process. There are challenges to this, though, as

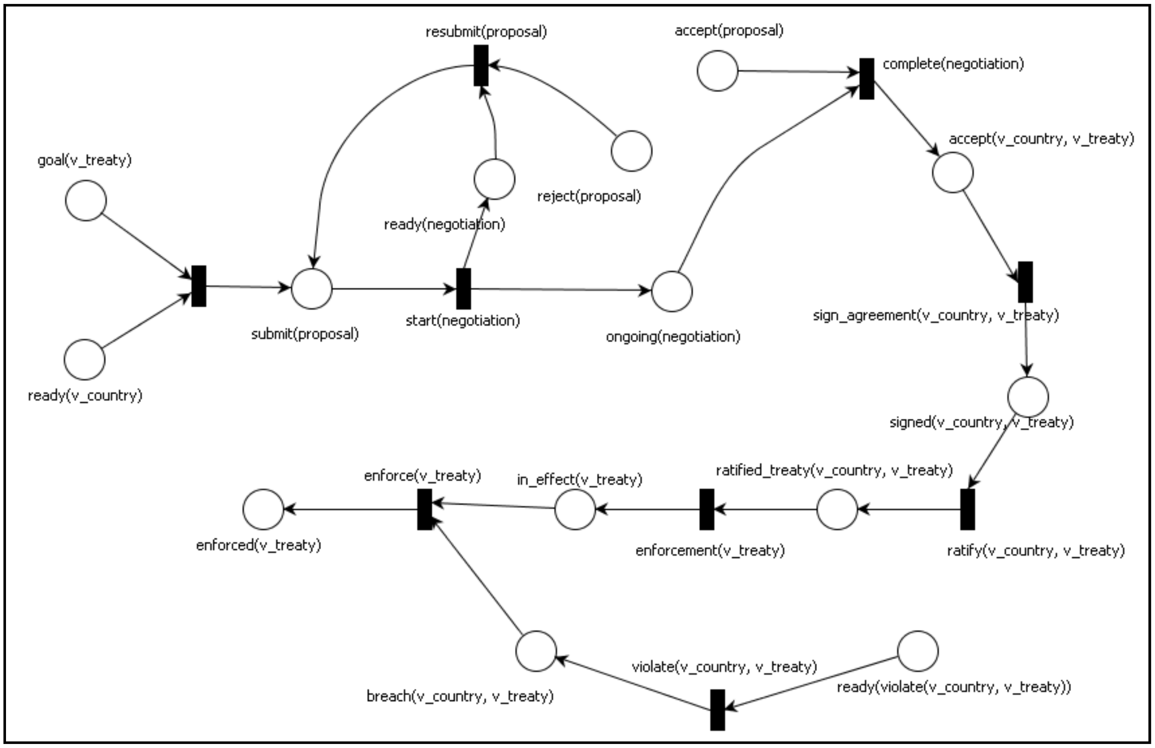
FrameNet tags do not automatically map onto event model parameters. This is laid out in greater detail in (Chang, Narayanan et al. 2002). For OWL-S-to-X-nets translations, manual correction is required.

4.6.3 Models built

We have developed several dynamic event models using our framework. With them, we have been able to test our systems' inference ability on a number of topics of interest. The X-net models we built include those of:

- Treaty negotiation and enactment
- General weapon procurement by a country
- Biological weapon production by a country
- The events of the 2006 Lebanon War
- Technology pathway alternatives (three sets of models)
- The judicial process
- The stages of employment
- and others

The breadth of models is representative of the coverage capability of our design. We highlight a few of these models, below. We describe the technology pathways in Section 7.5.1.



Treaty model

The treaty model is an abstract model that describes the process of creating, enacting, and executing a treaty with respect to a country. Before use, this model must be instantiated with a particular country and a particular treaty. Above is a screenshot of the model from our X-net editor.

The process starts with the requirement that the country has the **goal** of creating the treaty and is **ready** to proceed. This enables event progress to the point of a proposal for the treaty to be **submitted**. This spins off **negotiation** on the proposal, which continues until a proposal is **accepted**. At this point, the county's representative **signs the agreement** and then **ratifies** it. The treaty is then **in effect**. Should the treaty be **violated**, leading to a **breach**, the treaty will be **enforced**.

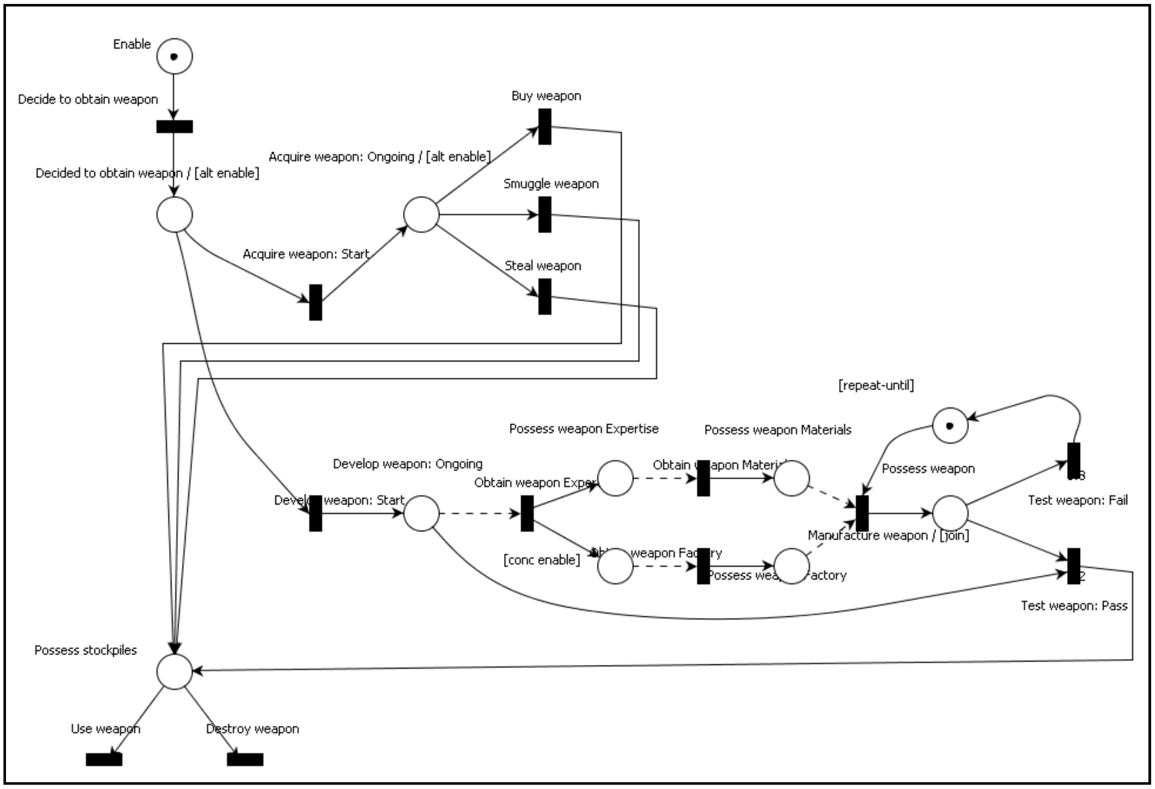
The Places and Transitions have names corresponding to the state or action represented. This is just aesthetic, though. More importantly, each Place and

Transition has a set of entity-restricted frames that provide linguistic descriptions of the particular state or action. For example, the "sign_agreement(v_country, v_treaty)" Transition is linked to the following frame:

```
Frame: Sign_agreement
  Signatory:     <v_country>
  Agreement:     <v_treaty>
```

In this abstract model, v_country is a variable initially restricted to be of the domain entity type, <Country>; and v_treaty is a variable restricted to <Treaty> (see more on abstract models Section 4.1.5). This means the action of signing the agreement can match with a specific instance of any particular country (like <England>, which is a subtype of <Country>) signing any particular treaty (like <Geneva Conventions>, which is a subtype of <Treaty>).

This simple model was manually generated.



Weapons procurement

The weapons procurement model describes two major paths a country can take in obtaining a weapon: acquiring a ready-made weapon or developing one, in-house. This abstract model, shown above, has to be instantiated with a country and a weapon sought before being used.

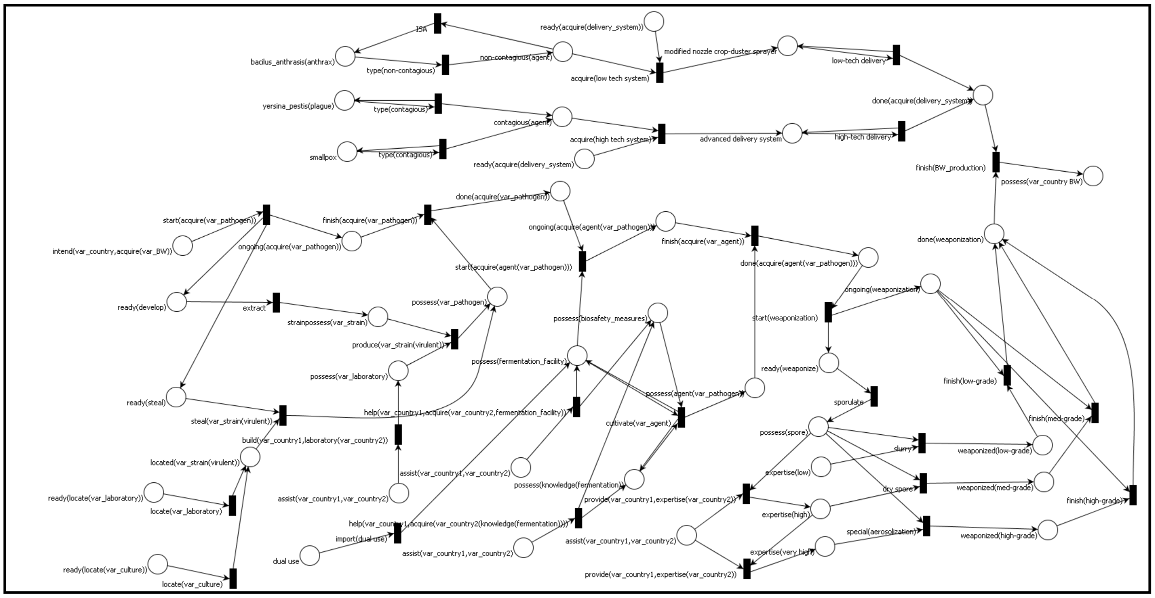
The model starts when **enabled** (shown enabled with a token). Enabling the model leads to a **decision to obtain a weapon**. A choice is encoded: choice 1) **acquire** a ready-made weapon, and choice 2) **develop** weapon in-house. A decision to acquire a weapon leads to another choice: to **buy**, **smuggle**, or **steal** the weapon. A decision to develop the weapon in-house leads to concurrent actions, to **obtain weapon expertise** and **materials**, and at the same time to **obtain a weapon factory**. With both sets of actions completed, weapons can be **manufactured** and

tested repeatedly until a weapon passes the test. At that point, weapons can be **stockpiled**, and then **used** or **destroyed**.

In this model, we constrain most Places to a capacity of one token, because each Place represents a state that is either true (one token) or currently false (no tokens). We use enable arcs (dashed arrows) in the weapons development track so as to not remove states that continue to hold as the process proceeds (e.g. obtained expertise is not lost as the weapon is built). In addition, we bias some of the choice points, like the simulation of testing the weapon; there we set an 80% fail rate per attempt.

Here, too, we use entity-restricted frames for linguistic descriptions of the constituent states and actions. In this model, there is greater frame support, with many actions having multiple descriptions. "Buy weapon", for example, uses seven entity-restricted frames: Commercial_transaction, Commerce_goods-transfer, Commerce_money-transfer, Commerce_buy, Commerce_sell, Commerce_collect, and Commerce_pay.

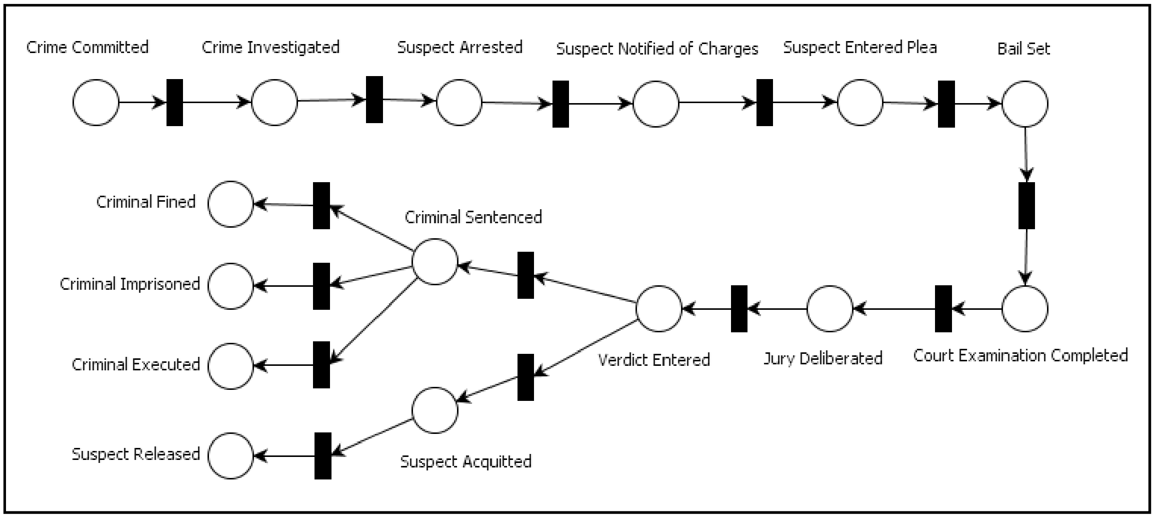
This model was manually generated in consultation with Subject Matter Experts.


ISA
ready(acquire(delivery_system))
bacilus_anthrasis(anthrax)
type(non-contagious)
non-contagious(agent)
acquire(low tech system)
modified nozzle crop-duster sprayer
low-tech delivery
yersina_pestis(plague)
type(contagious)
contagious(agent)
done(acquire(delivery_system))
smallpox
type(contagious)
acquire(high tech system)
advanced delivery system
high-tech delivery
ready(acquire(delivery_system)
finish(BW_production)
possess(var_country BW)
done(acquire(var_pathogen))
start(acquire(var_pathogen))
finish(acquire(var_pathogen))
ongoing(acquire(agent(var_pathogen)))
finish(acquire(var_agent))
intend(var_country,acquire(var_BW))
ongoing(acquire(var_pathogen))
done(weaponization)
start(acquire(agent(var_pathogen)))
done(acquire(agent(var_pathogen)))
ongoing(weaponization)
ready(develop)
extract
possess(var_pathogen)
strainpossess(var_strain)
possess(biosafety_measures)
start(weaponization)
produce(var_strain(virulent))
possess(fermentation_facility)
ready(weaponize)
possess(var_laboratory)
finish(low-grade)
ready(steal)
possess(agent(var_pathogen))
sporulate
steal(var_strain(virulent))
help(var_country1,acquire(var_country2,fermentation_facility))
cultivate(var_agent)
finish(med-grade)
build(var_country1,laboratory(var_country2))
possess(spore)
slurry
weaponized(low-grade)
located(var_strain(virulent))
assist(var_country1,var_country2)
expertise(low)
possess(knowledge(fermentation))
ready(locate(var_laboratory))
assist(var_country1,var_country2)
provide(var_country1,expertise(var_country2))
dry spore
weaponized(med-grade)
finish(high-grade)
locate(var_laboratory)
expertise(high)
help(var_country1,acquire(var_country2(knowledge(fermentation))))
import(dual use)
assist(var_country1,var_country2)
special(aerosolization)
weaponized(high-grade)
assist(var_country1,var_country2)
dual use
expertise(very high)
ready(locate(var_culture))
provide(var_country1,expertise(var_country2))
locate(var_culture)

Biological weapons production

The biological weapons production model is significantly more complicated than the weapons procurement model, as can be gleaned from the above screenshot. Without going into detail, the model depicts both the production of the pathogen (the bottom of the model), and the delivery system (the top of the model), to produce a full weapon (the right-most Place).

This model, too, was manually generated with input from Subject Matter Experts.



Judicial process

The judicial process model is an abstract model that follows the scenarios described in our motivating example from Section 3.1. It tracks a suspect accused of a crime through the judicial process. It was derived directly from FrameNet frames, demonstrating the potential for semi-automated translation. Each Place represents a frame in FrameNet. The Places were linked together based on the FrameNet Frame-to-Frame relations between the frames that helped generate those Places. The most pertinent relations are: Subframe-of, Precedes, Is-preceded-by.

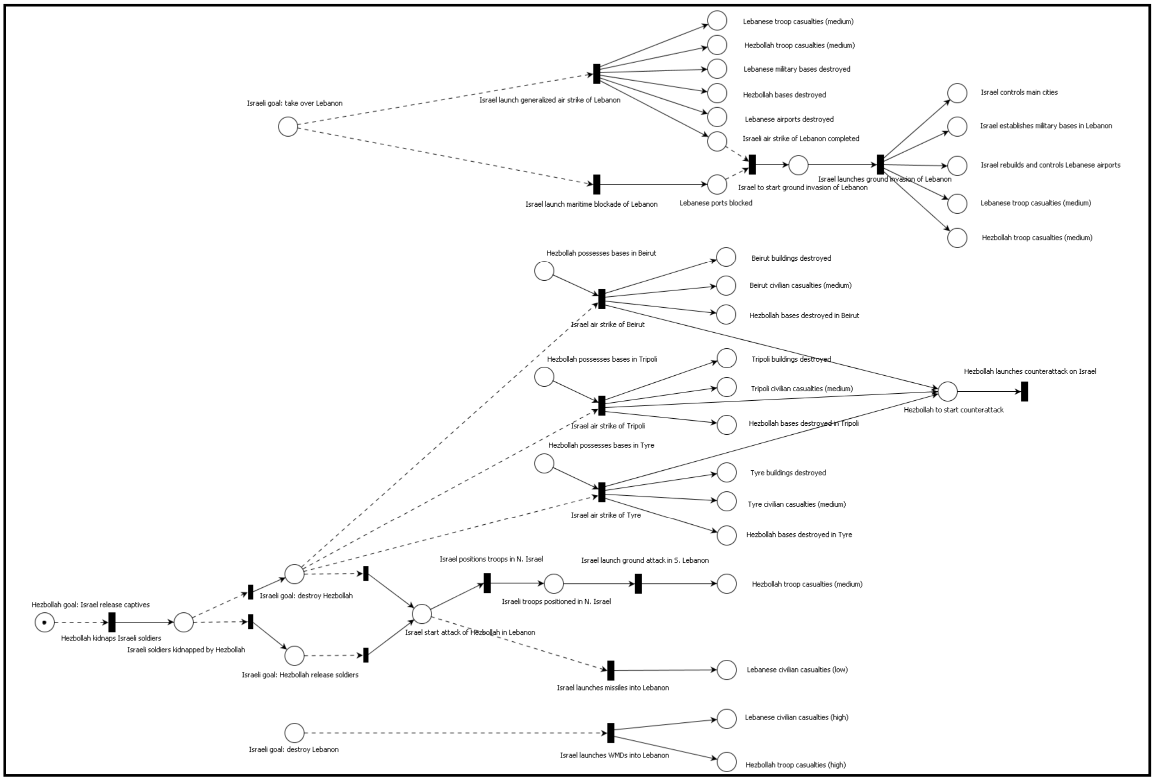
The process starts with a **crime committed**:

```
Frame: Committing_crime
  Crime:          <v_crime>
  Perpetrators:  <v_suspect>
```

Again, like the Treaty model, we use variables to link constraints across the model. Here we use v_crime, initially set to a type of <Crime>, and v_suspect, initially set to a type of <Person>. It continues with the **investigation of the crime**:

```
Frame: Criminal_investigation
  Incident:      <v_crime>
  Suspect:       <v_suspect>
```

and so on, through the **arrest**, **notification of charges**, **plea**, **bail**, **trial**, **jury deliberation**, **verdict**, **sentencing**, and **possible penalty**. Each Place is linked to the frame that was used to generate it.


Lebanese troop casualties (medium)
Hezbollah troop casualties (medium)
Lebanese military bases destroyed
Israeli goal: take over Lebanon
Israel launch generalized air strike of Lebanon
Hezbollah bases destroyed
Lebanese airports destroyed
Israeli air strike of Lebanon completed
Israel controls main cities
Israel establishes military bases in Lebanon
Israel rebuilds and controls Lebanese airports
Israel launches ground invasion of Lebanon
Israel to start ground invasion of Lebanon
Israel launch maritime blockade of Lebanon
Lebanese ports blocked
Lebanese troop casualties (medium)
Hezbollah troop casualties (medium)
Hezbollah possesses bases in Beirut
Beirut buildings destroyed
Beirut civilian casualties (medium)
Hezbollah bases destroyed in Beirut
Israel air strike of Beirut
Hezbollah possesses bases in Tripoli
Tripoli buildings destroyed
Tripoli civilian casualties (medium)
Hezbollah launches counterattack on Israel
Hezbollah to start counterattack
Israel air strike of Tripoli
Hezbollah bases destroyed in Tripoli
Hezbollah possesses bases in Tyre
Tyre buildings destroyed
Tyre civilian casualties (medium)
Israel air strike of Tyre
Hezbollah bases destroyed in Tyre
Israel positions troops in N. Israel
Israel launch ground attack in S. Lebanon
Hezbollah troop casualties (medium)
Israeli goal: destroy Hezbollah
Israeli troops positioned in N. Israel
Hezbollah goal: Israel release captives
Hezbollah kidnaps Israeli soldiers
Israeli soldiers kidnapped by Hezbollah
Israel start attack of Hezbollah in Lebanon
Israeli goal: Hezbollah release soldiers
Israel launches missiles into Lebanon
Lebanese civilian casualties (low)
Lebanese civilian casualties (high)
Israeli goal: destroy Lebanon
Israel launches WMDs into Lebanon
Hezbollah troop casualties (high)

Lebanon War of 2006

The 'Lebanon War of 2006' model is a specific model, requiring no instantiation. It is designed to map out the major events of the conflict by that name, as well as a few alternative scenarios that did not take place.

The main scenario starts with **Hezbollah's goal to get Israel to release Hezbollah captives** (the Place on the furthest left of the screenshot of the model). This begat Hezbollah's **kidnapping Israeli soldiers** to use as collateral. Israel formulated their own **goals of destroying Hezbollah** and attempting to **get Hezbollah to release the Israeli soldiers**. This escalates into a number of attacks and counter attacks, with casualties and property damage on both sides.

In addition to mapping out the scenario that did happen, we also modeled a few simple additional scenarios, one relating to an **Israeli goal to destroy Lebanon** (bottom of model), and another related to an **Israeli goal to take over Lebanon** (top of model). This enables an analyst to test what may have happened if conditions had been different. (We explore two such questions in Section 6.7.1.)

5 Applying Event Modeling to Answer Selection

Before applying our event modeling framework to a full-fledged NLP application, we decided to test it on a simpler problem. Answer selection is the process of choosing the best answer to a natural language question from a list of pre-chosen candidates. The difficulty in the task is assessing the relative relevancy of candidates that each contain the keywords used in a question asked.

For event-related questions, we hypothesized that our event ontology would improve the relevancy assessment. Specifically, we theorized that:

3) Keywords extracted from a question are related to one another and thus extracting relational information in a question and in a set of answer candidates should enable higher precision measurement of the relevancy of the candidates; and

4) Analyzing these relations in the context of a relevant, expressive model of the event in question provides a valuable link between the information sought in the question and the information contained in a good answer.

For us, the Answer Selection task provided an initial test of our approach for modeling events.

The system we designed relies on our event description work (Section 3.3), as well as our methods for linking language to event models (Chapter 4). In Section 5.1, we discuss the external tools that provide our system the answer candidates from which to select. In Section 5.2, we provide an example scenario that motivates our approach. We then describe our system at a high level (Section 5.3) and in detail (Section 5.4), before discussing the results of a simple assessment of the system (Section 5.5).

Answer Selection is a stepping stone to a full Question Answering system. We will describe our Question Answering system in Chapter 6 that builds upon our Answer Selection results.[4]

Early version of event modeling framework

The Answer Selection system was designed using an early incarnation of our event modeling framework; it tests only a portion of our framework's eventual functionality and uses some methods later improved. This is observable in two ways: 1) this system uses event models composed of event schema instances (as described in Section 3.3) and does not utilize our X-net design (Section 3.4); and 2) the system uses PropBank predicate-argument structures rather than FrameNet frames to linguistically describe events (Kingsbury and Palmer 2002; Palmer, Gildea et al. 2005).

In Chapter 4, we laid out our methodology for linking language to event models using frames. The key features of frames that are required for these algorithms also exist in PropBank predicate-argument structures. FrameNet frames are (and can be thought of as) a richer type of predicate-argument structure: predicates evoke frames and arguments to frames are frame elements. For the remainder of the chapter, we will reference these algorithms with the caveat that we use predicate-argument structures, but leave to the reader the mental substitution of 'predicate-argument structure' for 'frame' in the algorithm.

[4] Due to the overlap in the design of the Answer Selection system and Question Answering system, there is some repetition in our discussion of the two systems. For clarity, Chapter 5 and Chapter 6 do not rely on one another.

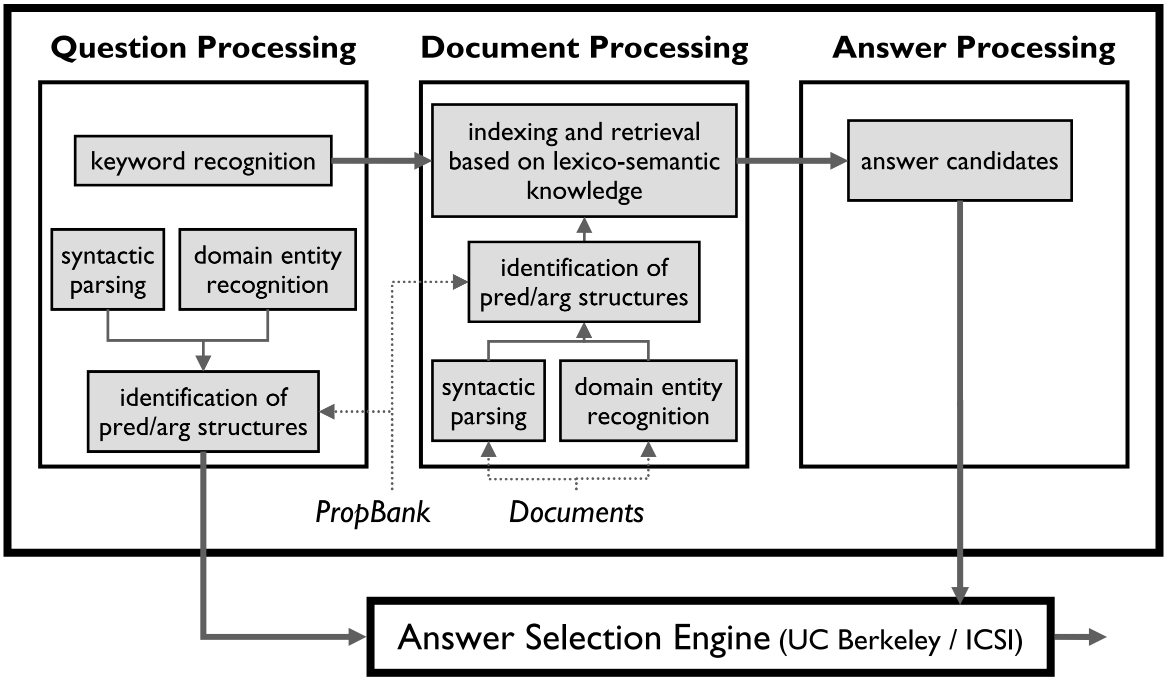

Question Processing
Document Processing
Answer Processing
keyword recognition
syntactic parsing
domain entity recognition
identification of pred/arg structures
indexing and retrieval based on lexico-semantic knowledge
identification of pred/arg structures
syntactic parsing
domain entity recognition
answer candidates
PropBank
Documents
Answer Selection Engine (UC Berkeley / ICSI)

Figure 5-1: AQUINAS Answer Selection System flowchart

5.1 System context

This Answer Selection work was a component of a larger project, AQUINAS. Our partner team from the University of Texas, Dallas (UTD), supplied and managed a suite of NLP tools that provided a front-end to our work. As shown in Figure 5-1, this component of the overall system has a three-stage pipeline: **question processing**, **document processing**, and **answer processing**. Its output, predicate-argument structure annotated question and answer data, is passed to our **Answer Selection Engine**, which ranks the answer candidates and selects the best.

The Question Processing stage of the AQUINAS system has a few key components. Three operations apply to the question: 1) its keywords are extracted for the Document Processing stage; 2) the question is syntactically parsed; and 3) any domain entities are tagged. The syntactic analysis and entity tags are used to help identify

PropBank predicate-argument structures (pred-args) in the question. These pred-args are then fed as the first of two inputs to the Answer Selection Engine.

During the Document Processing stage, the system selects answer candidates using the keywords extracted from the question. The documents are pre-processed with the same tools used in the Question Processing stage: they are syntactically parsed and domain entity tagged, before pred-args are identified. The documents are indexed, stored both as text and in predicate-argument form. From this database, answer candidates are chosen using a number of heuristics.

In the Answer Processing stage, the AQUINAS system forwards the answer candidates, annotated for predicate-argument structures, to the Answer Selection Engine.

The Answer Selection Engine thus requires candidate answers in a structured form. This needs: an information retrieval (IR) system, syntactic and semantic parsing capabilities, and domain entity tagging. In the AQUINAS system, these tools are provided by UTD. Their work is outside the scope of our research, but can be read about here (Harabagiu, Bejan et al. 2005).

5.2 Motivating example

In this chapter, we will use the following example question from an evaluation of the system to drive the explanation:

> Does Pakistan possess the technological infrastructure to produce biological weapons?

In addition, we will focus on one of the answer candidates provided for the question:

> While Pakistan is not known to possess biological weapons (BW), it has talented biomedical and biochemical scientists and well-equipped laboratories, which would allow it to quickly establish a sophisticated BW program, should the government so desire. *(Pakistan Country Profile, CNS 2004)*

5.3 System modules and processing flow

Shown in Figure 5-2, the Answer Selection Engine executes when the question and answer candidates are input. The system uses the question to select a relevant event model, which extends the range of what the system knows to be relevant in an answer. The system uses this information to score the relevancy of each answer candidate, finally ranking them and selecting the best answer to return.

Each phase of the system accomplishes a distinct task:

- Activating the system: The question and the answer candidates are each presented in two sets: 1) PropBank-defined pred-args; and 2) domain entity tags.
- The **alignment** phase: The pred-args and entities are interrelated. The system combines them, as appropriate, into entity-bound pred-args.
- The **model selection** phase: The question pred-args act as keys for looking up event models that describe the main underlying state or action in question.
- The **answer candidate scoring** phase: The pred-args of each answer candidate are compared to the pred-args of the model and scored accordingly.
- The **candidate ranking** phase: The system rank orders the answer candidates based on the score, and outputs the ranking, tagging the highest ranked candidate as the "selected" answer.

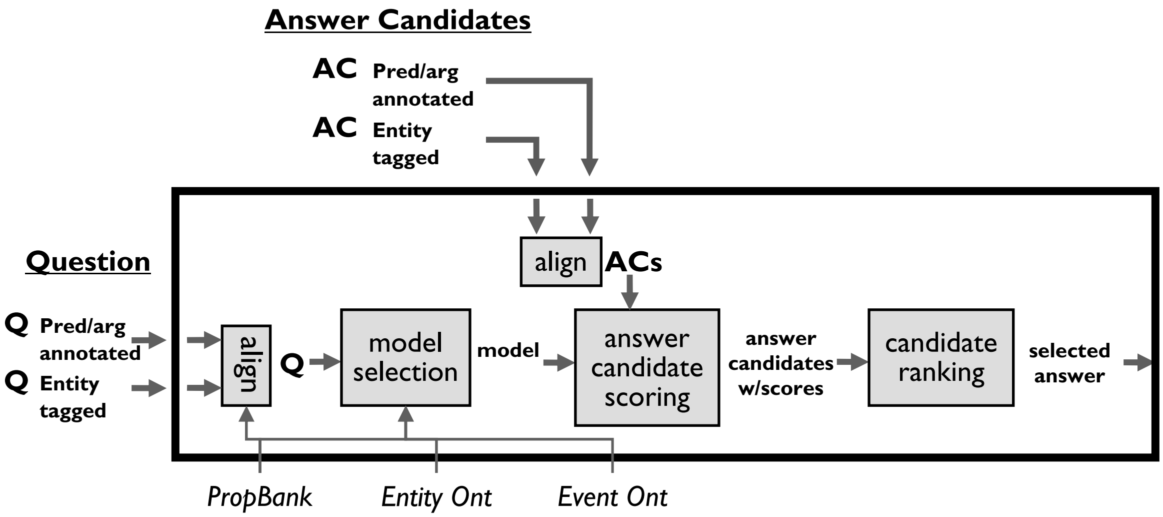


Figure 5-2: Answer Selection Engine

Example

In our example, the questioner asks about the current state of Pakistan's biological weapons infrastructure. The UTD front-end system provides to the Answer Selection Engine this question, as well as a set of candidate answers to the question. Those data come annotated for pred-args and tagged for domain entities. The pred-args and entity tags of the question are aligned, and the resulting entity-bound pred-args key into our event ontology, selecting our Biological Weapons Production model as being most relevant to the question. (A later version of our Biological Weapons Production model is shown in X-net form in Section 4.6.3.)

The pred-args of each answer candidate are compared against the pred-args of the model, scoring the candidate's relevance to the topic of Pakistan's biological weapons program. Among the candidates considered is the one mentioned in Section 5.2. Unlike many other candidates, this one has a number of pred-args that overlap with the those in the model's description, resulting in a high score. This high score places it at the top of the candidate ranking, and the candidate is selected as the best answer to the question.

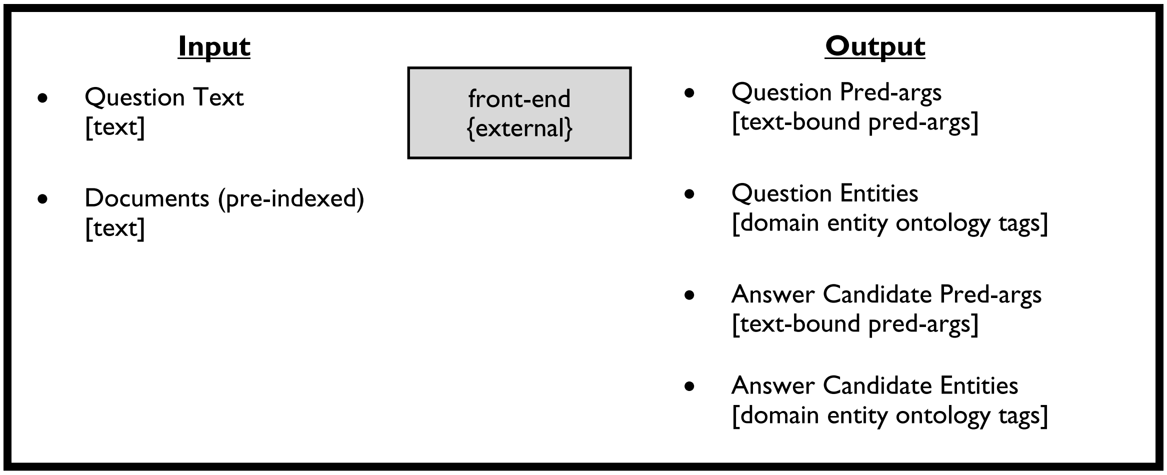


Figure 5-3: Front-end interface

5.4 Modules

In this section, we discuss the main modules of our Answer Selection Engine. Section 5.4.1 details the API of the front-end component of the AQUINAS system. Each subsequent subsection details one module shown in Figure 5-2. The subsections are concluded by a discussion of our running Pakistan infrastructure example, explaining the progress of the system during the stage discussed.

5.4.1 Front-end

The front-end system performs a number of critical tasks, as explained in Section 5.1. At a high level, it processes the question and provides candidate answers to the question. As shown in Figure 5-3, it provides both the question and answer candidates in predicate-argument and domain entity form. The processed question will allow the system to select an event model used in the answer selection analysis.

The pred-args annotated in the question and answer candidate text are PropBank pred-args. For each sentence, the analysis produces the predicates of the sentence with arguments bound to the text strings fulfilling the semantic roles of the predicate.

Input

- Question Text

"Does Pakistan possess the technological infrastructure to produce biological weapons?"

- Documents

Including passage:
"While Pakistan is not known to possess biological weapons (BW), it has talented biomedical and biochemical scientists and well-equipped laboratories, which would allow it to quickly establish a sophisticated BW program, should the government so desire."

Output

- Question Pred-args

possess ("Pakistan", "the technological infrastructure")
produce ("Pakistan", "biological weapons")

- Question Entities

  - "Pakistan" → <Pakistan>
  - "the technological infrastructure" → <Technology Infrastructure>
  - "biological weapons" → <Biological WMD>

- Answer Candidate Pred-args (shown: Candidate #5)

possess ("Pakistan", "biological weapons", "not known")
has ("Pakistan", "biomedical scientists", "talented")
has ("Pakistan", "biochemical scientists", "talented")
has ("Pakistan", "laboratories", "well-equipped")...

- Answer Candidate Entities (shown: Candidate #5)

  - "Pakistan" → <Pakistan>
  - "biomedical scientists" → <Biomedical Scientist>
  - "biochemical scientists" → <Biochemical Scientist>
  - "laboratories" → <Laboratory>...

Figure 5-4: Front-end example

The domain entities extracted from the text include traditional named entities, as well as more general types, as described in Section 4.1.3. These extracted entities are fine-grained categories, representing the most specific relevant type in the domain ontology.

(Note again: the methods used for predicate-argument annotation and domain entity tagging are outside the scope of this work.)

Running example: Front-end

Figure 5-4 shows the results of the front-end system on the text of our example. The question analysis produces two pred-args and three domain entities. The documents indexed in the system include the passage extracted as Candidate #5. That candidate has, among others, the four pred-args and four domain entities shown.

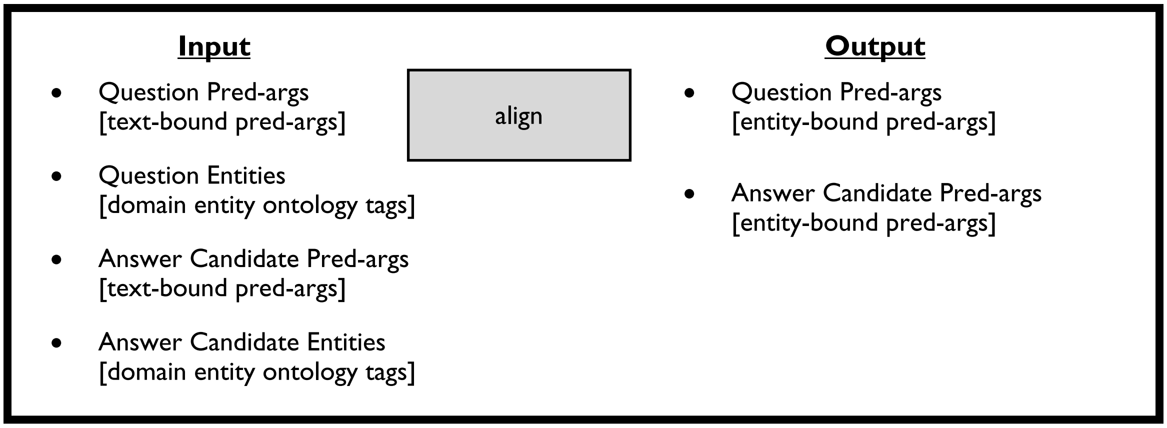


Figure 5-5: Alignment interface

5.4.2 Alignment

The form of linguistic description used in our Answer Selection Engine is entity-bound pred-args. As shown in Figure 5-5, though, the inputs to the Engine are text-bound pred-args and domain entities. Text bindings have too many surface realizations to facilitate direct comparison of the underlying meaning, which is required in later modules. The purpose of the Alignment module is to substitute each text-binding in a pred-arg with the entity tagged to the semantic head of the binding text. When no tag is available, the system does not set an entity binding, and the information is not carried on. (See Section 4.1.3 for a further discussion on using domain entity bindings.)

Running example: Alignment

In Figure 5-6, we show the entity-bound pred-args that result from aligning the text-bound pred-args and entities that are entered into the Alignment module in our ongoing example.

Input

- Question Pred-args (see Figure 5-4 output)
- Question Entities (see Figure 5-4 output)
- Answer Candidate Pred-args (see Figure 5-4 output)
- Answer Candidate Entities (see Figure 5-4 output)

Output

- Question Pred-args (entity-bound)

```
possess (<Pakistan>, <Technology Infrastructure>)
produce (<Pakistan>, <Biological WMD>)
```

- Answer Candidate Pred-args (entity-bound)

```
possess (<Pakistan>, <Biological WMD>)
has (<Pakistan>, <Biomedical Scientists>)
has (<Pakistan>, <Biochemical Scientists>)
has (<Pakistan>, <Laboratory>)
```

Figure 5-6: Alignment example

5.4.3 Model Selection

The Model Selection module: 1) selects a model that describes the type of event in question; and 2) "instantiates" the model, tuning it to the specific event instance in question. As shown in Figure 5-7, this is accomplished with the aid of the pred-args of the question and the event model ontology. We use a number of the methodologies described in Chapter 4 to accomplish these tasks (with minor modification, due to use of pred-args instead of frames in linguistic descriptions and semantic analysis).

Selecting a model from the Event Ontology

The Model Selection module selects a model from the Event Ontology. The Event Ontology for this Answer Selection system is a database of event model descriptions, each composed of Event Schema instances (discussed in Section 3.3, and shown in Figure 3-4). Each Event Schema instance has linguistic descriptions, in entity-restricted pred-arg form, of an event's constituent states and actions (see Section 4.1.5 for explanation of "entity-bound" vs. "entity-restricted" distinction).

In selecting a model, we implement the algorithm of Section 4.3. The system treats each model in the database as a single set of entity-restricted pred-args, grouping

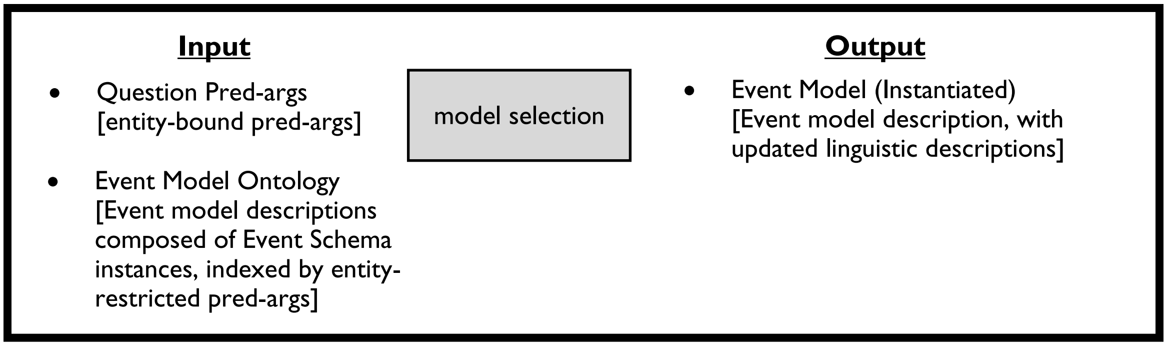

Input
Question Pred-args [entity-bound pred-args]
Event Model Ontology [Event model descriptions composed of Event Schema instances, indexed by entity-restricted pred-args]
model selection
Output
Event Model (Instantiated) [Event model description, with updated linguistic descriptions]

Figure 5-7: Model Selection interface

all of the pred-args related to that model. It then computes the set intersection between the pred-arg set of each model in the database and the set of entity-bound pred-args in the question. The model with the 'largest' intersection is selected. The underlying rational is, relations in the question related to any part of the model is proof of relevance of the model to the question, and the larger the connection the better.

The method of scoring the intersection between a model set and the question set is to sum all of the pred-arg match scores between the pred-args of the two sets. We implement the frame matching algorithm described in Section 4.1.4 (applied to pred-args instead of frames), extending it (as mentioned) to use part-of and relationship-of relations in the domain entity ontology, just as it does subtype-of relations.

Instantiating an abstract model

A model selected may be abstract or specific, as discussed in Section 4.1.5. An abstract model is applicable to multiple scenarios and ideally uses linguistic descriptions that are general enough to encompass all of the scenarios intended, but specific enough to *only* cover those scenarios (e.g. the Treaty model, Section 4.6.3). A specific model will cover only one scenario (e.g. a model of a particular historical event).

Abstract models must be instantiated to be used. If an abstract model is chosen, the linguistic descriptions that made it applicable to multiple scenarios must be focused to only apply to the scenario asked about in the question. As mentioned in Section 4.1.5, abstract models use variables in their linguistic descriptions, restricting (in this case) argument bindings in those pred-arg-based descriptions, not directly to domain entity categories, but instead to variables set to domain entity categories. When the value of any one variable is changed, it can affect multiple linguistic descriptions across the model. The selection of which variables are changed is based on the matches between question pred-args and model pred-args. We use the algorithm of Section 4.3 to update the model, making it as specific as the question's entity-bindings.

Once a model is instantiated, the variable values will not change. As such, for the rest of the chapter, we will ignore this redirection of restrictions through variables and use reified specialized entity restrictions.

Input

- Question Pred-args (see Figure 5-6 output)
- Event Model Ontology

Includes:
- Bioweapons Production model
  restricted to:
  <Country>,
  <Biological WMD>

Output

- Event Model (Instantiated)

```
o  Bioweapons Production model
   restricted to: <Pakistan>,<Biological WMD>
   pred-args include:
   // Preconditions for the Develop Expertise stage
   possess(<Pakistan>, <Bio Expert> )
   possess(<Pakistan>, <Bio Laboratory> )
   has(<Pakistan>, <Bio Expert> )
   has(<Pakistan>, <Bio Laboratory>)
   …

   // Process of building weapon
   manufacture(<Pakistan>, <Biological WMD> )
   produce (<Pakistan>, <Biological WMD> )
   …

   // Effect of building weapon
   possess(<Pakistan>, <Biological WMD> )
   has(<Pakistan>, <Biological WMD> )
```

Figure 5-8: Model Selection example

Running example: Model Selection

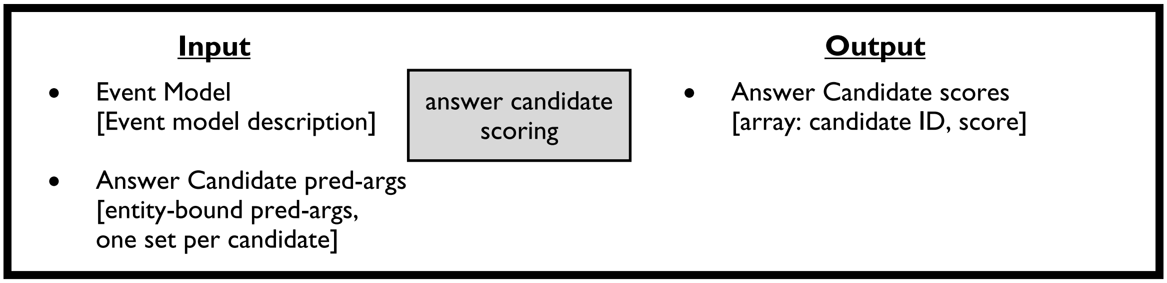
Figure 5-8 shows the model selected based on the pred-args of the question. Specifically, the question has two pred-args:

- possess (<Pakistan>, <Technology Infrastructure>)
- produce (<Pakistan>, <Biological WMD>)

The first, when used to index into the event model database, is too general and returns nothing. The second matches with the Bioweapons Production model, which contains:

- produce (v_country, v_weapon),

where v_country is initially set to <Country>, and v_weapon is initially set to <Biological WMD>). This will produce a model match score of 0.95, higher than any other potential match. As such, the Bioweapons Production model is chosen. Then,



Input

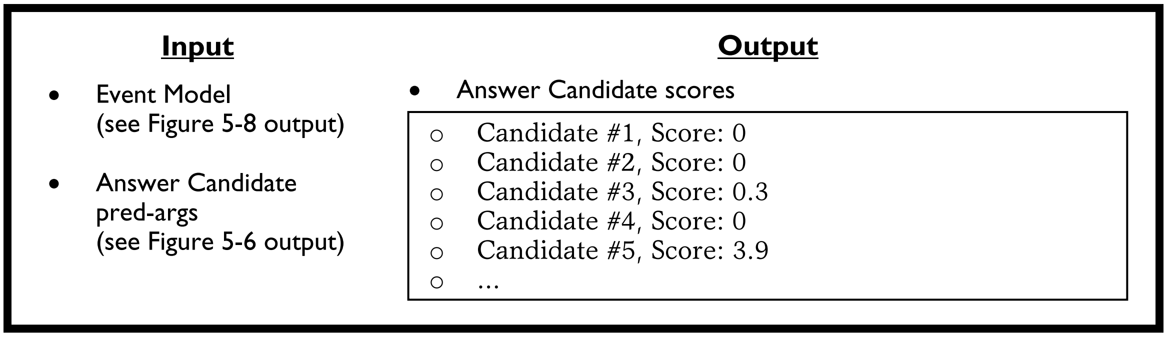
- Event Model
  [Event model description]
- Answer Candidate pred-args
  [entity-bound pred-args,
  one set per candidate]

answer candidate scoring

Output

- Answer Candidate scores
  [array: candidate ID, score]

Figure 5-9: Answer Candidate Scoring interface



Input

- Event Model
  (see Figure 5-8 output)
- Answer Candidate
  pred-args
  (see Figure 5-6 output)

Output

- Answer Candidate scores
  - Candidate #1, Score: 0
  - Candidate #2, Score: 0
  - Candidate #3, Score: 0.3
  - Candidate #4, Score: 0
  - Candidate #5, Score: 3.9
  - …

Figure 5-10: Answer Candidate Scoring example

v_country is updated to <Pakistan> and v_weapon stays as <Biological WMD>. The resulting model is shown.

5.4.4 Answer Candidate Scoring

To rank the answer candidates and select a final answer, the system must first score the relevance of each candidate. As shown in Figure 5-9, the Answer Candidate Scoring module uses the event model selected to test and score each answer candidate, returning an array of candidate scores to the Candidate Ranking module.

The scoring algorithm used by the module is a modified form of the algorithm used to select a model. Instead of scoring the entity-bound pred-arg set of the question against the entity-restricted pred-arg set of each model the system considers for selection (as is done in the Model Selection module), the Answer Candidate Scoring module scores the entity-restricted pred-arg set of the selected model against the entity-bound pred-arg set of each answer candidate. We use the following scoring function (similar to our algorithm in Section 4.3, using a pred-arg matching algorithm similar to frame matching algorithm of Section 4.1.4):

```
For each answerCandidate
    For each acPredArg
        Find potential matches of acPredArg in model
                totalPredArgScore + = thisPredArgScore
    Scores[acID] = totalPredArgScore
Return Scores
```

Running example: Answer Candidate Scoring

In Figure 5-10, we show the results of scoring the set of answer candidates for the question. Specifically, we highlight the scoring of Candidate #5: "While Pakistan is not known to possess biological weapons (BW), it has talented biomedical and biochemical scientists and well-equipped laboratories, which would allow it to quickly establish a sophisticated BW program, should the government so desire."

Each of the candidate's four pred-args matches with a pred-arg in the Bioweapons Production Model. Using pred-arg match scoring where penalty$_{\text{spec}}=0$ and penalty$_{\text{gen}}$ = 0.2 (see Section 4.1.4):

- possess (<Pakistan>, <Biological WMD>) matches exactly, for a score of 1
- has (<Pakistan>, <Biomedical Scientists>) & has(<Pakistan>, <BioExpert>) have a more-specific match, for a score of 1
- has (<Pakistan>, <Biochemical Scientists>) & has(<Pakistan>, <Bio Expert>) have a more-specific match, for a score of 1
- has (<Pakistan>, <Laboratory>) & has (<Pakistan>, <Bio Laboratory>) have a less-specific match, for a score of 0.9

The total score is: 3.9.

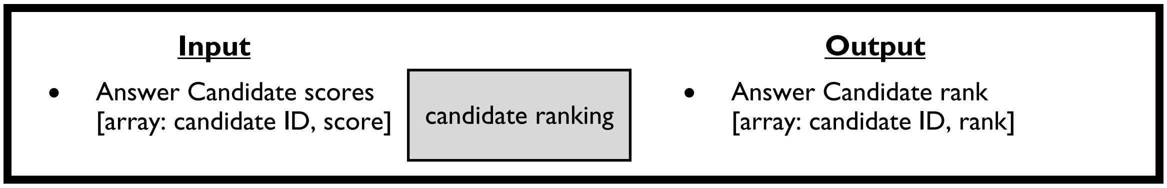


Figure 5-11: Candidate Ranking interface

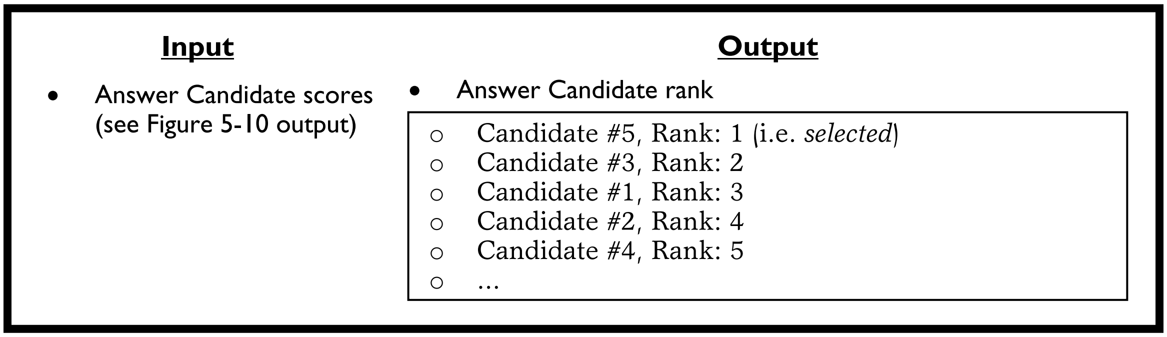


Figure 5-12: Candidate Ranking example

5.4.5 Candidate Ranking

The Candidate Ranking module collects the answer candidate scores, ranks the candidates in order of decreasing score, and packages and returns this list to the user (Figure 5-11).

Running example: Candidate Ranking

As shown in Figure 5-12, the final results in our example put Candidate #5 at the top of the ranking.

5.5 Evaluation

The Answer Selection task , as stated at the onset of the chapter, provided an initial test of our event modeling approach. Specifically, we wished to evaluate whether the use of relational structures (in this case, predicate-argument structures) could improve precision in the selection of answers, while the use of event models could extend recall of relevant answers.

To facilitate our experiment, we used the AnswerBank corpus (UTD-created for AQUINAS). The corpus has approximately 2700 questions with gold-standard answers drawn from a large set of public documents related to national security. More than half of the questions are event-related, and are either *justification*, *projection*, *ability*, or *hypothetical* questions (as defined in 3.2).

We setup a test of 12 event-related questions, each randomly selected from the AnswerBank corpus, filtered to pertain to the topic of weapons procurement or production. We submitted the questions to the UTD front-end system and received its top seven ranked answer candidates for each. In addition to the seven candidates returned, we included the gold-standard answer from AnswerBank, when not already included in the original seven (in all but one case, we added the gold-standard answer). The pred-args and domain entities were hand-corrected to ensure gold-standard annotations. For each question, we entered its pred-args and domain entities, and those of its answer candidates, into our Answer Selection Engine. In each case, the system cleanly separated the gold-standard answer from the rest, selecting the gold-standard answer as best.

An analysis of the test set of questions and answer candidates revealed that most (78 of 84) of the answer candidates returned by the front-end system were irrelevant to the question, though they contained the question's keywords. While this validated our approach to increase precision in answer selection and demonstrated the difficulty of answering complex event-related questions with traditional NLP tools, we concluded that the design of the next stage Question Answering system (Chapter 6) should emphasize mechanisms to increase the retrieval of information relevant to the question.

6 Applying Event Reasoning to Question Answering

Automated Question Answering (QA) is a challenging research problem in the Natural Language Processing (NLP) community. The task is to find or create focused answers to natural language questions. This can be contrasted with a typical information retrieval search that retrieves a list of documents or passages that contain a given set of keywords. For our research, the QA problem is a good test of the event reasoning framework we developed, to see if our techniques can help provide solutions for complex, event-related queries (in a closed domain).

In Chapter 3, we discussed our design for describing, modeling, simulating, and analyzing events. In Chapter 4, we described our solution for linking language semantics with event models, providing a means for selecting a model capable of reasoning about an event of interest, and a mechanism for incorporating linguistic evidence about an event in a relevant model. We put a subset of these capabilities together in our Answer Selection system (of Chapter 5)[5]. To build a full-fledged QA system, though, requires incorporating all of these features, as well as a number of additional tools.

In Section 6.1, we describe the context in which we built our QA system, including a review of the auxiliary tools required to enable our system to work. Subsequent to that, we provide a high level overview of the system (Section 6.3), before detailing each of its components (Section 6.4). Then we briefly discuss back-off

[5] Due to the overlap in the design of the Answer Selection system and Question Answering system, there is some repetition in our discussion of the two systems. For clarity, Chapter 5 and Chapter 6 do not rely on one another.

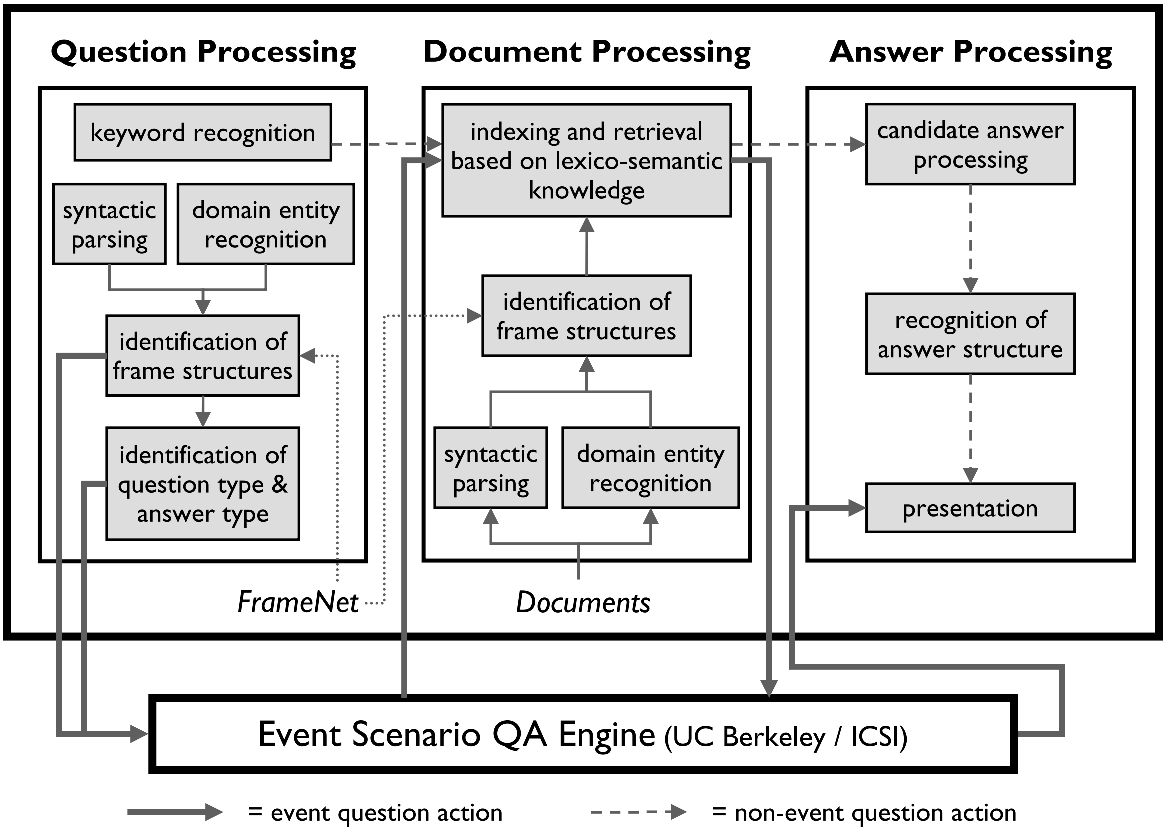

Question Processing
Document Processing
Answer Processing
keyword recognition
syntactic parsing
domain entity recognition
identification of frame structures
identification of question type & answer type
indexing and retrieval based on lexico-semantic knowledge
identification of frame structures
syntactic parsing
domain entity recognition
candidate answer processing
recognition of answer structure
presentation
FrameNet
Documents
Event Scenario QA Engine (UC Berkeley / ICSI)
= event question action
= non-event question action

Figure 6-1: AQUINAS QA System flowchart

strategies to handle cases where certain resources are not available (Section 6.5) and a few implementation notes (Section 6.6), before describing the results of evaluations of the system (Section 6.6).

6.1 System context

Our QA research was conducted within the context of a larger project (AQUINAS). Our partner team from the University of Texas, Dallas (UTD), supplied and managed a suite of NLP tools that complimented our work (and others). As shown in Figure 6-1, we broke down the QA task into a three-stage pipeline: **question processing**, **document processing**, and **answer processing**. Our work, the **Event Scenario QA Engine**, supports and interacts with the pipeline at various stages.

Question Processing has a number of components. A question needs to be syntactically parsed, and its domain entities need be to recognized from an ontology and extracted. This assists in identifying: the frame structures in the question, the type of question being asked, and the form of answer being requested. For event related questions, our Event Scenario QA Engine is activated for further processing of these inputs. For non-event related questions, the AQUINAS system extracts keywords from the question and continues on to Document Processing.

The Document Processing stage returns passages with information relevant to answering the question asked of the system. To do so, the AQUINAS system indexes the documents with lexico-semantic information, like frames, entities, and other keywords, ahead of time. It syntactically parses them, recognizes domain entities, and identifies frame structures. With this retrieval capability, for event related questions, the Event Scenario QA Engine can request documents using frame encoded information requests. For non-event related questions, the keywords extracted from the question in the Question Processing stage can be used to index into the document database.

In the final stage, Answer Processing, the AQUINAS system either presents the information inferred by the Event Scenario QA Engine (for event related questions), or it uses its own mechanism to process the retrieved documents and extract answers (for non-event questions).

We thus found that, to build a full QA system capable of answering event related natural language questions, our event reasoning framework needs to be bolstered by: an information retrieval (IR) system, syntactic and semantic parsing capabilities, domain entity tagging, question analysis, and answer presentation modules. In the AQUINAS system, we used the UTD IR and parsing tools. Their work, done in coordination with the Language Computer Corporation (LCC), is outside the scope of our research, but can be read about here (Harabagiu, Bejan et al. 2005). For our design of the Event Scenario QA Engine, we assumed gold standard quality output from these tools. (The form and nature of that output is part of our research, and is discussed in Section 4.2 and further in this chapter.)

6.2 Motivating example

Throughout the next sections of the chapter, we describe the details of our Event Scenario QA Engine. We use the following example question, taken from an evaluation of the system, to drive the explanation:

> Is Iran a signatory to the Chemical Weapons Convention?

The following relevant passage will also be used to help infer an answer to the question:

> Being one of the few countries in the world that has experienced chemical warfare on the battlefield, Iran ratified the Chemical Weapons Convention in 1997.

6.3 System modules and processing flow

In Figure 6-2, we show our Event Scenario QA Engine flowchart. The system is broken down into two stages: question processing (**Stage I**), and data processing (**Stage II**). The processed question suggests which data should be used in Stage II to infer an answer. There is a break in control at the end of the first stage when the

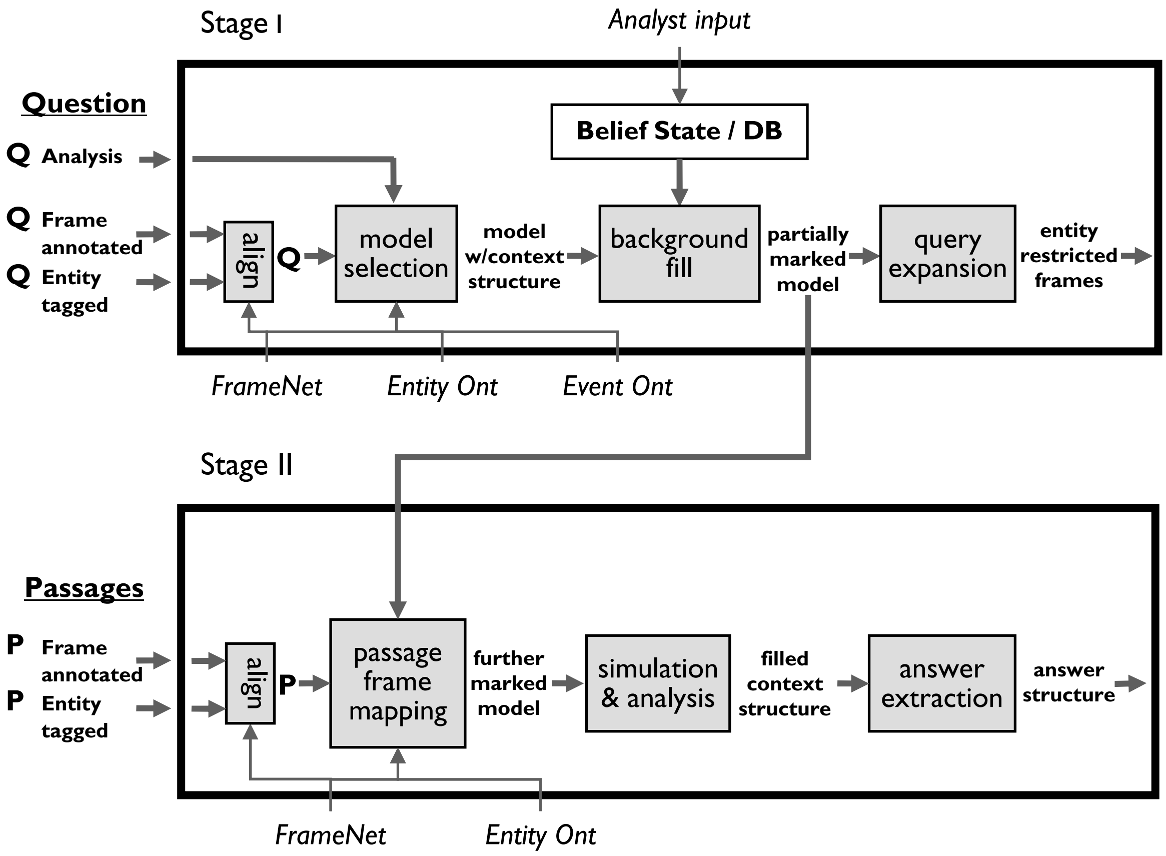


Figure 6-2: Event Scenario QA Engine

system sends out an expanded query to the AQUINAS program to retrieve relevant data.

In the first stage, the system attempts to identify an appropriate model of the type of event in question. It then tries to fill in any information it already knows about the scenario based on internal information resources. That which it cannot retrieve from internal resources but recognizes as valuable for inference, it requests. More specifically:

- Activating the system: This stage starts by accepting three pieces of information about the question: the frames of the question, the domain entities in the question, and the type of the question (prediction, hypothetical, justification, etc.).

- The **alignment** phase: The frames and entities, though analyzed by two separate external algorithms, are interrelated. The system combines them into a unified (partial) semantic representation of the question, composed of entity-bound frames.
- The **model match** phase: The entity-bound frames act as keys for looking up event models that describe the main underlying state or action in question. Centering on the main state or action in question, the system dynamically determines a **context structure** for the answer, which includes slots for information causally-relevant to the answer being sought. The most significant subset of the context structure will eventually be returned as the answer; this subset is called the **answer structure**.
- The **background fill** phase: The system adds in as much of the information to the context structure as is available from internal resources that have relational querying capability (e.g. RDBMSs). (This is especially valuable for reusing cached data.)
- The **query expansion** phase: Typically, the context structure cannot be filled with internal information alone. The system requests all remaining missing information from an external program. (As mentioned in Section 6.1, we use the UTD IR program.)

In the second stage, the system incorporates new information retrieved from the external program and uses it to infer information to fill the answer context structure.

- The **alignment** phase: The frames and entities analyzed from the retrieved passages are once again aligned to produce entity-bound frames.

- The **passage frame mapping** phase: With the data now in a processable form, the system attempts to fill in as many remaining empty context structure slots as possible.
- The **simulation & analysis** phase: Typically, some information will still not be available. To derive the full answer, the system attempts to simulate the event with the information it has, leading to inferences about the missing information. The exact method of inference in the simulation phase is based on the type of question asked, as discussed in Section 3.5.
- The **answer extraction** phase: The system extracts the answer structure from the context structure and returns it to be presented to the user.

The system is capable of using models with information pre-coded in them. In those cases, we can collapse the two stages into one, bypassing the passage retrieval and processing steps.

Example

We can use our example about Iran signing the Chemical Weapons Convention (CWC) to detail the components of the Event Scenario QA Engine.

This engine can handle questions about complex events and processes. Were this Iran question asked of the AQUINAS system (Section 6.1), it would detect that the question is about an event (the signing of the CWC), and would forward it to our Event Scenario QA Engine after processing the question text into frames and domain entities.

In this scenario, our event system aligns the frames and entities of the question, and uses the resulting entity-bound frames to key into our event ontology, selecting our Treaty model (described in Section 4.6.3) as being most relevant to the question.

After selecting the model, the system determines a context structure for the answer, focusing on the signing action in the Treaty model. During the Background Fill phase, the system fills the context structure with as much contextual information as possible about Iran signing the CWC from internal resources, after which information still missing in the context structure is requested from an external system.

That search returns passages like the one described in Section 6.2, discussing Iran's ratification of the CWC. Using this passage, the system add the ratification fact into the context structure and maps it onto the model. Being a 'justification' question, the system then runs an analysis projecting the data forwards and backwards, as dictated in Section 3.5. This shows that ratification of the CWC can only occur after signing the CWC. That information is then packaged and returned for presentation to the questioner.

6.4 Modules

In this section, we detail each of the main modules of our Event Scenario QA Engine, as depicted in Figure 6-2. Each subsection on a module is concluded by a discussion of our running Treaty example, explaining the progress of the system during the stage discussed.

6.4.1 Pre-system: Question Analysis

The correct input form of the question is critical to enabling the functionality of the event reasoning framework. The system relies on external analysis of the question text to produce three pieces of information: 1) the frames in the question, 2) the domain entities in the question, and 3) the question type (see Figure 6-3). Together they form a structured, partial semantic representation of the question text that will

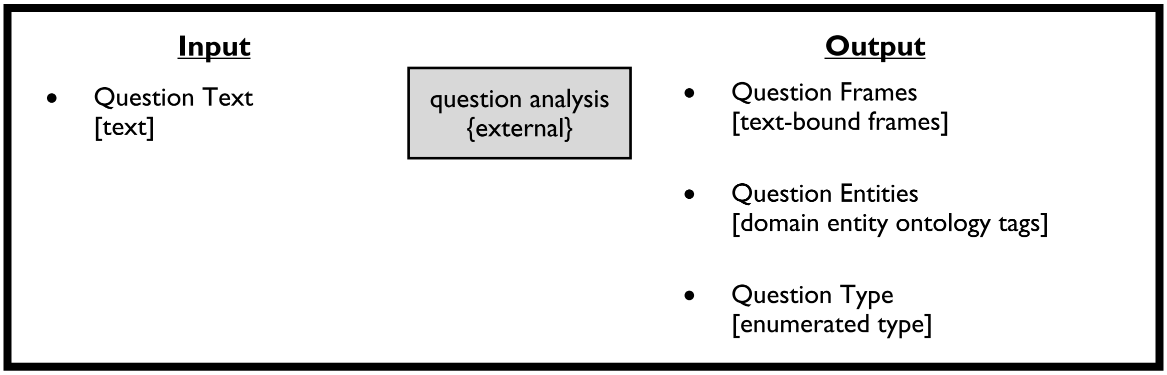


Figure 6-3: Question Analysis interface

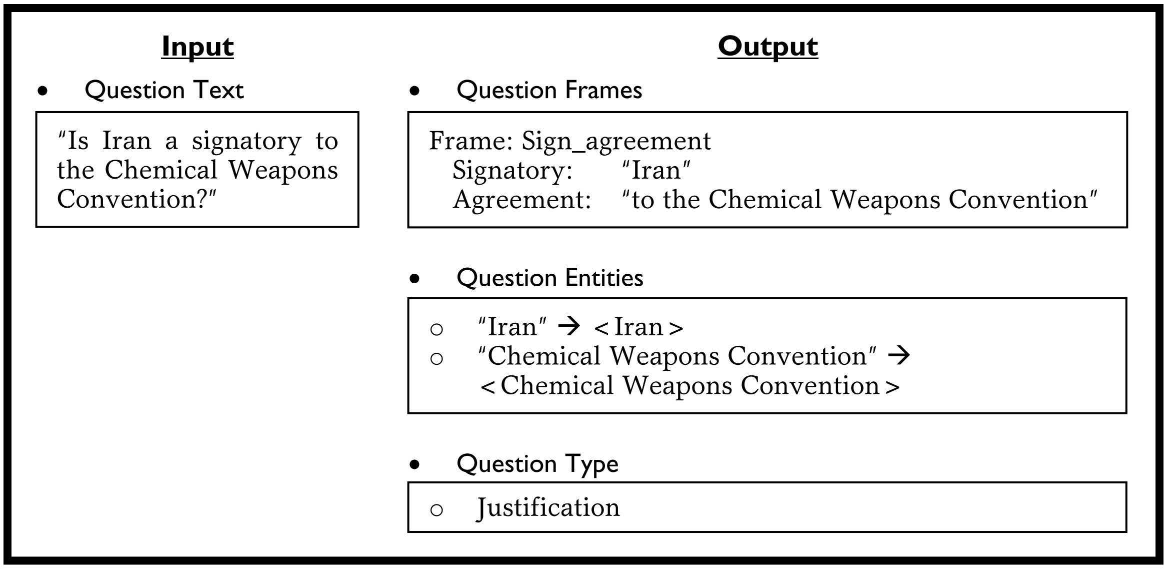
allow the system to: select an appropriate event model, request relevant data, and properly process that data to answer the question.

The frames analyzed in the questions are FrameNet frames (as discussed in Section 4.1.1). The output of the analysis is, for each frame evoked, the frame name and its constituent frame elements (FEs) bound to text ("text-bound frames").

The domain entities extracted from the question include traditional named entities, as well as more general types, as described in Section 4.1.3. These extracted entities are fine-grained categories, representing the most specific relevant type in the domain ontology.

Finally, the system receives question type information. Event questions our QA system handles come in one of four types: *justification*, *prediction*, *hypothetical*, and *ability* (defined in Section 3.2).

(Note again: the methods used for parsing, tagging, and analysis of the question are outside the scope of this work.)



Input

- Question Text

"Is Iran a signatory to the Chemical Weapons Convention?"

Output

- Question Frames

Frame: Sign_agreement
Signatory: "Iran"
Agreement: "to the Chemical Weapons Convention"

- Question Entities

  - "Iran" → <Iran>
  - "Chemical Weapons Convention" → <Chemical Weapons Convention>

- Question Type

  - Justification

Figure 6-4: Question Analysis example

Running example: Question Analysis

Our question, "Is Iran a signatory to the Chemical Weapons Convention?", when entered into the AQUINAS system, is analyzed in three ways (see Figure 6-4). The UTD front-end (ideally):

1) extracts the *Sign_agreement* frame, with the *Signatory* FE bound to "Iran", and *Agreement* FE bound to "to the Chemical Weapons Convention";

2) tags the words "Iran" and "Chemical Weapons Convention" with <Iran> and <Chemical Weapons Convention>, respectively; and

3) determines that the question is a *justification* question. (Recall from Section 3.2, *justification* questions have an underlying premise (Iran is a signatory to the CWC) that require verification.)

All three analysis results are passed as input to the Event Scenario QA Engine.

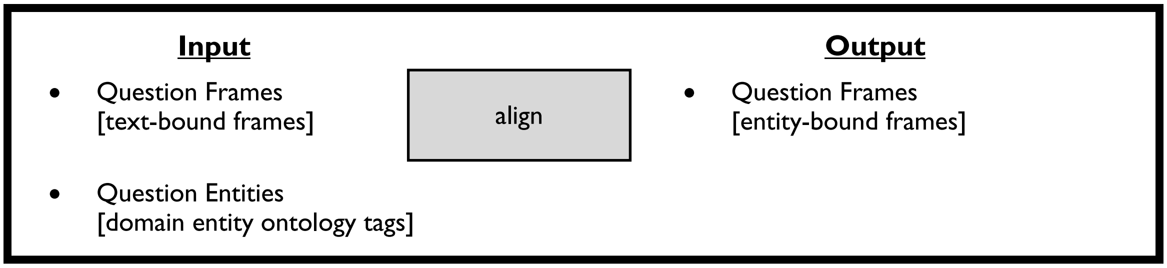


Figure 6-5: Question Alignment interface

6.4.2 Alignment (Question)

In order to connect questions with relevant event models, and event models with relevant data, we require a consistent, precise structural form to describe the question, the data, and the event model. As discussed in Section 4.1, we chose frames (with elements linked to domain entities) as the common form for capturing semantic information across our system.

The Alignment module is responsible for taking raw text-bound frame annotations (frames with frame elements bound to text) and domain entity tags of a particular phrase, and combining them into entity-bound frames (frames with frame elements bound to domain entities), as shown in Figure 6-5.

Text bindings have too many surface realizations to facilitate direct comparison of the underlying meaning, which is required in later modules. The system, instead, replaces the textual frame element (FE) bindings in the frame parse with the "best" entity tagged to words enclosed in the binding text. The most significant issue is how to determine what constitutes the "best" entity amongst multiple options. We use one of two heuristics:

- Selecting a proper subtype of the FrameNet FE's Semantic Type
- Selecting the semantic head of FE binding

Input	**Output**
- Question Frames *(see Figure 6-4 output)* - Question Entities *(see Figure 6-4 output)*	- Question Frames (entity-bound) Frame: Sign_agreement Signatory: <Iran> Agreement: <Chemical Weapons Convention>

Figure 6-6: Question Alignment example

FrameNet frames provide Semantic Types for some FEs. As described in Section 4.1.1, Semantic Types provide guidance of acceptable roll fillers using ontological categories. Unfortunately, not all FEs are tagged with these, and those that are use tags that provide limited guidance due to their high-level of generality. In any case, the Semantic Types must map onto whichever domain entity ontology the system references; this enables the system to check that a selected entity is a proper subtype of the Semantic Type. When choosing among multiple valid subtypes, the most specific entity in the ontology is selected.

When the Semantic Type is unavailable, insufficient, or not mappable to the domain ontology, we use the entity capturing the semantic head of the FE binding text.

Note that there are not always appropriate entities tagged within an FE binding text string. In these cases, the system does not set an entity binding, and the information is not kept.

Running example: Question Alignment

In our example, the frame and entities input into the system would produce the aligned entity-bound frame seen in Figure 6-6. There is only one entity enclosed in the text, "Iran", bound to the *Signatory* role of the *Sign_agreement* frame. For the phrase, "to the Chemical Weapons Convention", there are in fact five entities we have in our domain entity ontology: <Chemical>, <Weapon>, <Convention>,

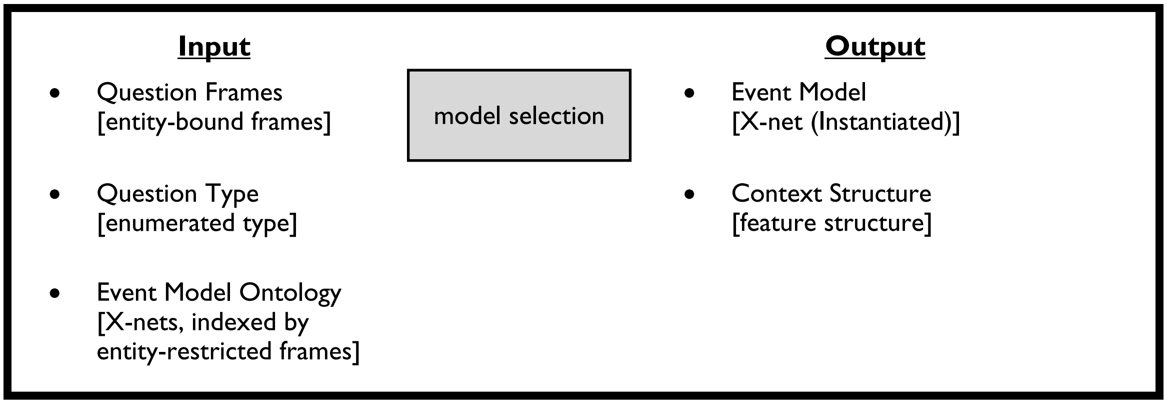


Figure 6-7: Model Selection interface

<Chemical Weapon>, and <Chemical Weapons Convention>. The most specific type in the semantic head is <Chemical Weapons Convention>, so it is selected.

6.4.3 Model Selection

The Model Selection module accomplishes a number of tasks for the system: 1) it selects a model that can simulate the event in question; 2) it instantiates the model, tuning it to the specific event instance in question; and 3) it calculates the context structure for determining the answer. (See the module interface in Figure 6-7.) In Chapter 4, we discussed a number of key methodologies to accomplish these tasks. In this section, we put them together.

Selecting a model from the Event Ontology

To select a model requires access to an ontology of event models, in the form of a database of X-nets. (The models described in Section 4.6.3 exemplify the models that would be available in such a database.) The system queries this database using the frames of the question as search keys. Each X-net is composed of Places (representing states and resources) and Transitions (representing actions), which in turn have a number of linguistic descriptions in entity-restricted frame form. Based on the semantic similarity between the entity-bound frames of the question and the entity-restricted frames used to describe the states and actions of each model, the output of the search is the most relevant model for the question in X-net form (see Sections 4.1.5 for explanation of "entity-bound" vs. "entity-restricted" distinction; recall that, structurally, they are equivalent, which facilitates the matching algorithm).

We implemented the model matching algorithm of Section 4.3. The system treats each model in the database as a single set of entity-restricted frames, grouping all of the frames related to that model (those representing its Places and Transitions). It then computes the set intersection between the frame set of each model in the database and the set of entity-bound frames in the question. The model with the 'largest' intersection is selected. The underlying rationale is that relations in a question related to any part of a model is an indication of relevance of the model to the question, and the larger the connection the better.

The method of scoring the intersection between a model set and the question set is to sum all of the frame match scores between the frames of the two sets. We implement the frame matching algorithm described in Section 4.1.4, extending it (as mentioned) to use part-of and relationship-of relations in the domain entity ontology, just as it does with subtype-of relations.

Instantiating an abstract model

The model selected from the event model database may be "abstract" or "specific". An abstract model, as discussed in Section 4.1.5, is applicable to multiple scenarios and ideally uses linguistic descriptions that are general enough to encompass all of the scenarios covered by the X-net structure, but specific enough to *only* cover those scenarios (e.g. the Treaty model, Section 4.6.3). A specific model will cover only one scenario (e.g. the Lebanon War of 2006 model, Section 4.6.3).

Abstract models must be instantiated to be used. If an abstract model is chosen, the linguistic descriptions that made it applicable to multiple scenarios must be bound to the scenario asked about in the question. As discussed in Section 4.1.5, abstract models use variables in their linguistic descriptions, restricting frame element bindings in those frame-based descriptions not directly to domain entity categories, but instead to variables set to domain entities categories. By using this indirection, a change to entity restrictions used in the linguistic description of one action or state can be propagated throughout the model. The selection of which variables are changed is based on the matches between question frames and model frames. We use the algorithm of Section 4.3 to update the model, restricting the model to the scenario referenced in the question's entity-bindings.

The model output by the module will a standard X-net, but the frame descriptions will have changed from the X-nets originally selected from the ontology. The original form is appropriate for selection of the model; the latter form is appropriate for selecting relevant data to analyze to find an answer to the question. For an abstract model using variables to accomplish this, once the model is instantiated, the variable values will not change. (As such, for the rest of the chapter, we will ignore this redirection of restrictions through variables and use reified specialized entity restrictions.)

Calculating the Context Structure and Answer Structure

The Model Selection module also provides a Context Structure for the answer. Using the question frames to pinpoint the areas of interest within the selected model, the

```
Input parameters: AS_depth, CS_depth
        (assert CS_depth > AS_depth)
Output variables: AS_set, CS_set
Foreach model state/action matching a question frame
    Add state/action and each ancestor* state/action to
        • AS_set, up to AS_depth state/action generations
        • CS_set, up to CS_depth state/action generations
    For non-prediction questions, add each descendant*
    state/action to
        • AS_set, up to AS_depth state/action generations
        • CS_set, up to CS_depth state/action generations
*ancestor states/actions are those that causally precede this action/state
*descendant states/actions are those that causally succeed this action/state
```

Figure 6-8: Context Structure algorithm

module uses simulation trajectories to determine which information, if available, would be relevant for inferring an answer. The format of the context structure is a feature structure with slots for each piece of information sought. This information is stored in frame-encoded form and tagged with its source.

The context structure is directly linked to and is not independent of the X-net model. Each slot directly corresponds to a Place or Transition in the model. Information found to fill a slot directly translates to tokens or restrictions on the related Place or Transition in the model.

In addition, the system marks a subset of the slots of the Context Structure as belonging to the Answer Structure. While event reasoning for the system requires substantial contextual information to infer information targeted in a question (the Context Structure), we also believe the questioner will appreciate a subset of this context accompanying any direct answer (the Answer Structure). (See Section 4.5 for further discussion). The Answer Structure will be extracted during the Answer Extraction phase, and returned from the system.

To compute the Context Structure (CS) and Answer Structure (AS), we implement a slight variant of the query expansion algorithm described in Section 4.4; this algorithm, shown below, creates slots for each X-net object traversed, rather than

Input

- Question Frames
  (see Figure 6-6 output)
- Question Type
  (from Figure 6-4 output)
  - ○ Justification
- Event Model Ontology
  (see Section 4.6.3 for e.g.)

  Includes:
  - ○ Treaty model
    restricted to:
    <Country>,
    <Treaty>

Output

- Event Model (Instantiated)
  - ○ Treaty model
    restricted to:
    <Iran>, <Chemical Weapons Convention>
- Context Structure (w/Answer Structure †)
  - ○ ongoing(negotiation)
  - ○ accept(proposal)
  - ○ complete(negotiation)
  - ○ accept(Iran, CWC) †
  - ○ sign_agreement(Iran, CWC) †
  - ○ signed(Iran, CWC) †
  - ○ ratify(Iran, CWC)
  - ○ ratified_treaty(Iran, CWC)

Figure 6-9: Model Selection example

compiling a list of frames to be included in an expanded query (our system accomplishes this in the Query Expansion phase).

Running example: Model Selection

In our continuing example, shown in Figure 6-9, the Event Ontology includes the Treaty model (described in detail in Section 4.6.3). The original Treaty model is relevant to situations where the principle actor of interest was any *country* and the theme was any *treaty*. This is exemplified in the frame hooks used to linguistically describe the states and actions represented by the Places and Transitions in the model. The *"Sign Agreement"* Place, for example, has the following frame hook:

```
Frame: Sign_agreement
  Signatory:     <Country>
  Agreement:     <Treaty>
```

The entity restrictions of the frame require that any question about agreement signing will be about a country signing a treaty in order to match with the model. This is specific enough to exclude, for example, home buyers signing mortgage papers.

In the question, "Is Iran a signatory to the Chemical Weapons Convention?", the frame *Sign_agreement* matches the *Treaty* model hook of the *"Sign Agreement" Transition* mentioned above. In our Weapons Production domain ontology (Section 4.1.3), as well as WordNet, we find that the *Signatory*, <Iran>, is a subtype of <Asian Country>, which is a subtype of <Country>. The *Agreement*, <Chemical Weapons Convention>, is a subtype of <Convention>, which is a subtype of <Treaty>. Thus, for each FE, there is an inexact, but strong match. We use frame match scoring parameters ($\text{penalty}_{\text{spec}} = 0.1$; see Section 4.1.4) that only penalize greater specificity a small amount, leading to an overall frame match score of close to 1 (1 = exact match). This being the only frame extracted from the question, the model match score is the same score. Finding no other models which match this question frame, the *Treaty* model is selected as the most relevant model for the question.

The Treaty model begins as an abstract model. In our example, we match the question's *Sign_agreement* frames of the question and model to find the relevance of the model. After selection, though, the model still only describes how an unnamed country negotiates and completes an unnamed treaty. The restrictions on the "*Sign Agreement*" Transition can easily be made specific, changing the restriction on the *Signatory* from <Country> to <Iran> and the restriction on the *Agreement* from <Treaty> to <Chemical Weapons Convention>, as specified in the question's frame. However, we would like these restrictions to propagate as appropriate to the frame hooks of other Places and Transitions (i.e. the linguistic descriptions of other states and actions depicted in the model). Not only is the signing about Iran and the CWC, but so too is the negotiation, ratification, enforcement, etc. To do this, the model designer can use entity variables to link the FE restrictions of multiple frames in a model. The Treaty model, instead of having <Country> as the restriction on the *Signatory* of the *Sign_agreement* frame, will have a variable, *var_country*, as the restriction; the initial setting of *var_country* will be <Country>. Similarly, the restriction on the *Ratifier* of the *Ratification* frame hook of the "*Ratification*" Transition will not be <Country>, but will be *var_country*. Then, instead of changing the restriction on the *Signatory* from <Country> to <Iran>, the model instantiation operation will change the value of the variable attached to *Sigantory*, *var_country*, from <Country> to <Iran>, thereby in effect changing the restriction on both *Sign_agreement* and *Ratification* (and others). Note that specialization of a restriction only goes in one direction, taking a general restriction and making it more specific; restrictions in the model that are already more specific than those in the question frames are left as is.

Finally, the system computes the Context Structure and Answer Structure of the model for this question, with settings: *CS_depth* = 3 and *AS_depth* = 1. With the *Sign_agreement* frame match, the Context Structure algorithm centers its search on the *Sign Agreement* Transition of the model. The results are a Context Structure of eight unfilled slots, shown in Figure 6-9, with the three slots shown with (†) a part of the Answer Structure.

6.4.4 Background Fill

During the Background Fill phase, the system searches internal resources for facts sought in the Context Structure. This may include cached results of prior event analyses, encyclopedic information, expert-inserted background information,

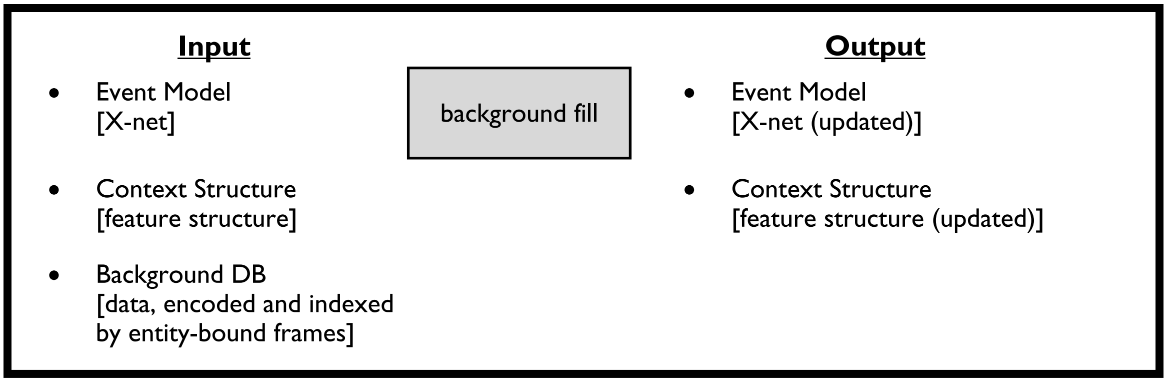


Figure 6-10: Background Fill interface

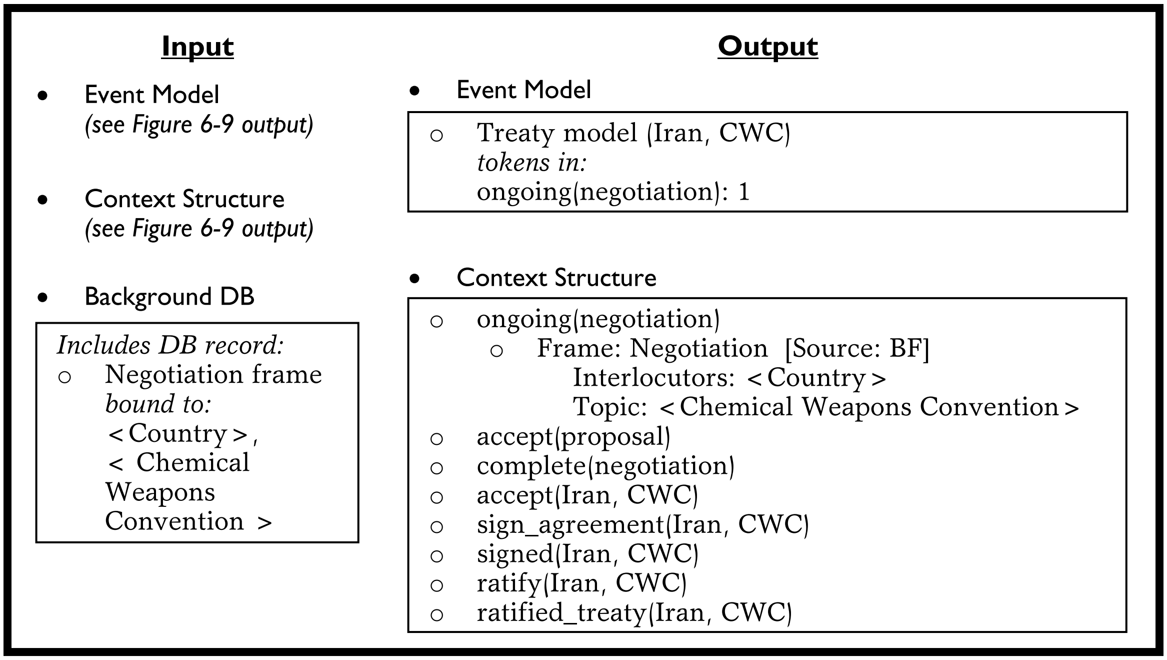
depictions of the belief state of the user, etc. The information is added to the event model in the form of tokens (creating a partial marking), and added to the Context Structure in the form of frame-encoded information with source tags.

Database

In our system, internal background information is stored in entity-bound frame form in a basic relational database. This information is indexed by frame name. The system searches the database using entity-restricted frames. For each frame instance returned, the FE restrictions of the query are checked against the instance. For each frame match score (see Section 4.1.4) above threshold (default condition is non-zero), the frame in the database is returned.

Retrieval

The Background Fill module collects all of the entity-restricted frame hooks of the states and actions in the Context Structure. The system then searches the database for information matching any frame hook. Following the template laid out in Section 4.4, that information is added to the Context Structure and X-net.



Input

- Event Model
  (see Figure 6-9 output)
- Context Structure
  (see Figure 6-9 output)
- Background DB

Includes DB record:
- Negotiation frame *bound to:* <Country>, < Chemical Weapons Convention >

Output

- Event Model

- Treaty model (Iran, CWC)
  tokens in:
  ongoing(negotiation): 1

- Context Structure

- ongoing(negotiation)
  - Frame: Negotiation [Source: BF]
    Interlocutors: <Country>
    Topic: <Chemical Weapons Convention>
- accept(proposal)
- complete(negotiation)
- accept(Iran, CWC)
- sign_agreement(Iran, CWC)
- signed(Iran, CWC)
- ratify(Iran, CWC)
- ratified_treaty(Iran, CWC)

Figure 6-11: Background Fill example

Running example: Background Fill

In our example, the fact that '165 countries negotiated the Chemical Weapons Convention' is in the internal database. The *ongoing(negotiation)* Place in the Treaty model represents whether a treaty was negotiated or not, in this case the Chemical Weapons Convention. A token will be added to that Place because of the information available in the relational database. The context structure will keep track of that data and its source (BF = Background Fill).

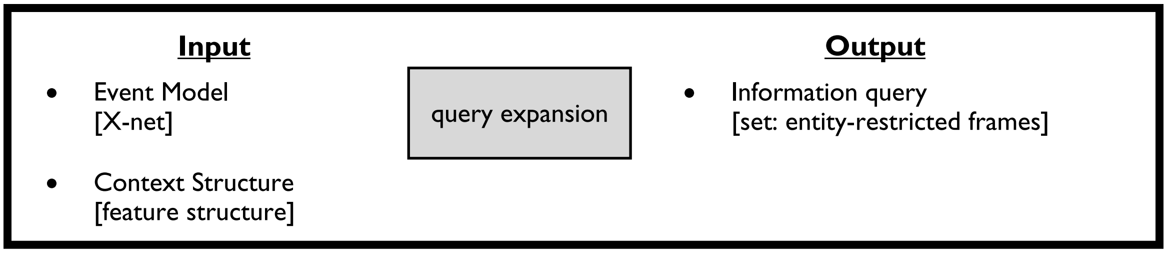
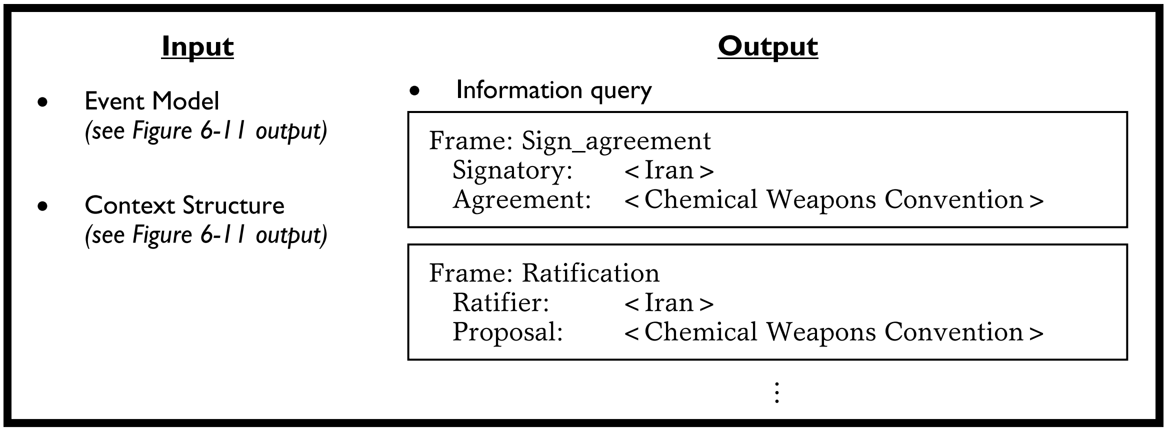


Figure 6-12: Query Expansion interface

6.4.5 Query Expansion system (Output of Stage I)

The input to the Query Expansion module is a partially filled, instantiated event model. Some information may have been added during the Background Fill phase, but most likely there is significant information sought in the context structure still missing. The Query Expansion phase extracts the linguistic descriptions from the Places and Transitions linked to the empty slots of the context structure (see Figure 6-12). These entity-restricted frames are output as a query for an external program that then searches for resources that match the information requests. This query represents an expansion of the original information directly sought in the question.



Input

- Event Model
  (see Figure 6-11 output)
- Context Structure
  (see Figure 6-11 output)

Output

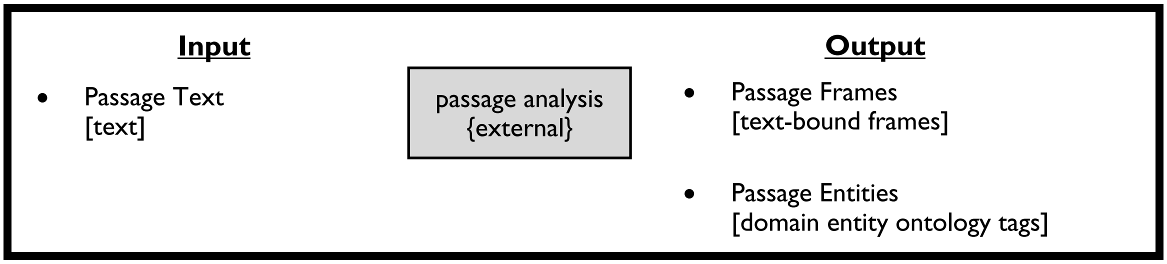
- Information query

Frame: Sign_agreement
Signatory: <Iran>
Agreement: <Chemical Weapons Convention>

Frame: Ratification
Ratifier: <Iran>
Proposal: <Chemical Weapons Convention>

⋮

Figure 6-13: Query Expansion example



Input

- Passage Text
  [text]

passage analysis
{external}

Output

- Passage Frames
  [text-bound frames]
- Passage Entities
  [domain entity ontology tags]

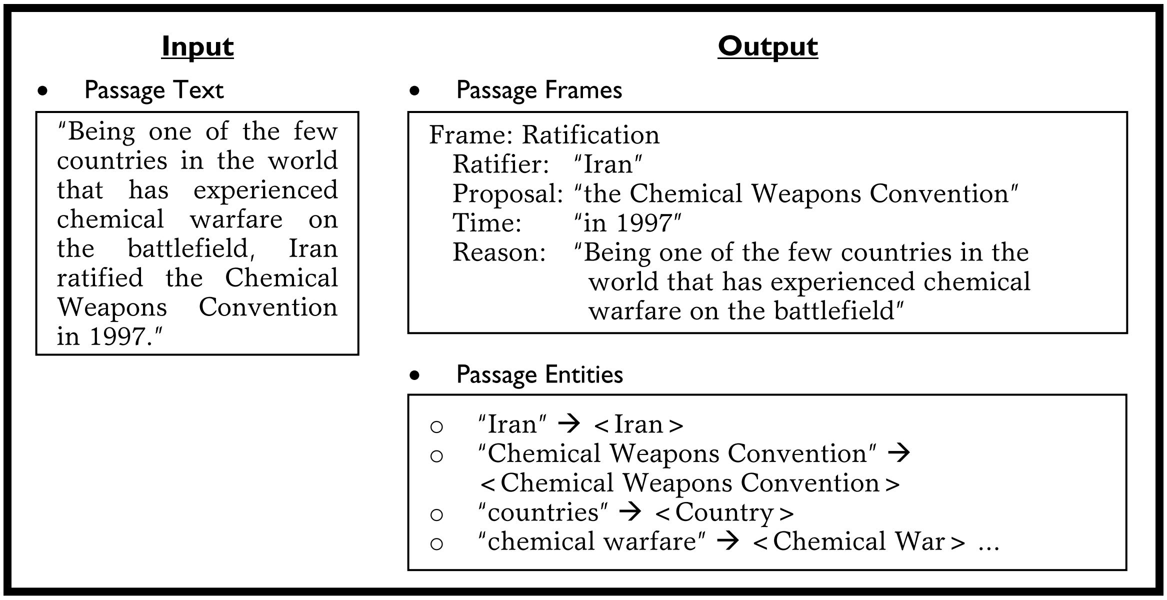
Figure 6-14: Passage Analysis interface

Running example: Query Expansion

In our example, there are seven slots yet to be filled in the context structure. Each slot corresponds to a Place or Transition in the Treaty X-net model, each of which, in turn, has a set of linguistic descriptions in entity-restricted frame form. These frames are gathered and output by the system. As shown in Figure 6-13, they include frame descriptions of the *sign_agreement(Iran, CWC)* Transition, directly referenced in the original question, and the *ratified_treaty(Iran, CWC)* Place, which is indirectly related.

6.4.6 External: Passage Analysis

The main source of data used to infer an answer to the question asked comes in the passages retrieved. Similar to the interface to Stage I, the input to the system is text, analyzed for frames and entities (see Figure 6-14). Rather than just the one sentence



Input

- Passage Text

"Being one of the few countries in the world that has experienced chemical warfare on the battlefield, Iran ratified the Chemical Weapons Convention in 1997."

Output

- Passage Frames

Frame: Ratification
  Ratifier: "Iran"
  Proposal: "the Chemical Weapons Convention"
  Time: "in 1997"
  Reason: "Being one of the few countries in the world that has experienced chemical warfare on the battlefield"

- Passage Entities
  - "Iran" → <Iran>
  - "Chemical Weapons Convention" → <Chemical Weapons Convention>
  - "countries" → <Country>
  - "chemical warfare" → <Chemical War> ...

Figure 6-15: Passage Analysis example

entering Stage I (the question), the system receives multiple, potentially multi-sentence passages in Stage II.

Running example: Passage Analysis

Continuing our treaty example, one passage retrieved is: "Being one of the few countries in the world that has experienced chemical warfare on the battlefield, Iran ratified the Chemical Weapons Convention in 1997." Frame parsed, this will include the *Ratification* frame, shown in Figure 6-15. Tagging the entities of the passage will produce: <Country>, <Chemical War>, <Iran>, <Chemical Weapons Convention>, and more.

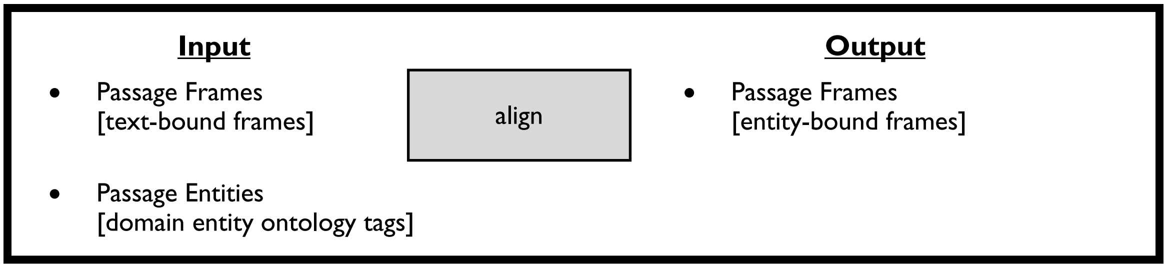


Figure 6-16: Passage Alignment interface

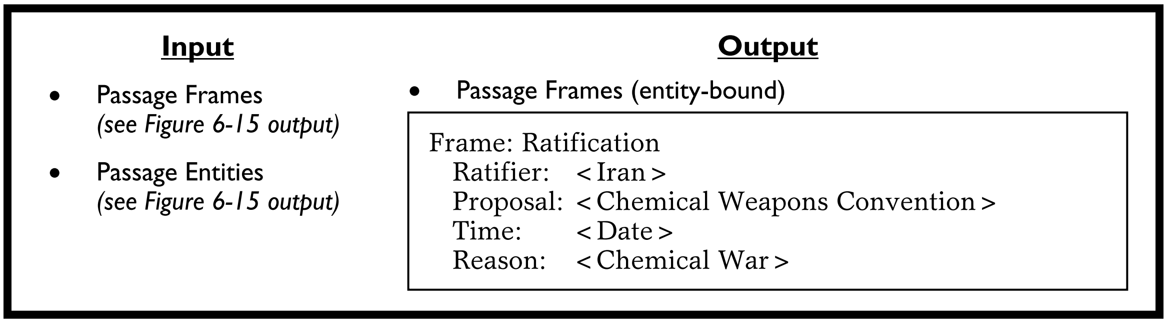


Figure 6-17: Passage Alignment example

6.4.7 Alignment (Passage)

As in Stage I, the frames and entities are aligned before being further analyzed, in this case, for facts relevant to the question (see Figure 6-16). See Section 6.4.2 for a description of the alignment method used.

Running example: Passage Alignment

In our example, the system aligns the text-bound *Ratification* frame and the entities of the passage to create an entity-bound frame, as seen in Figure 6-17.

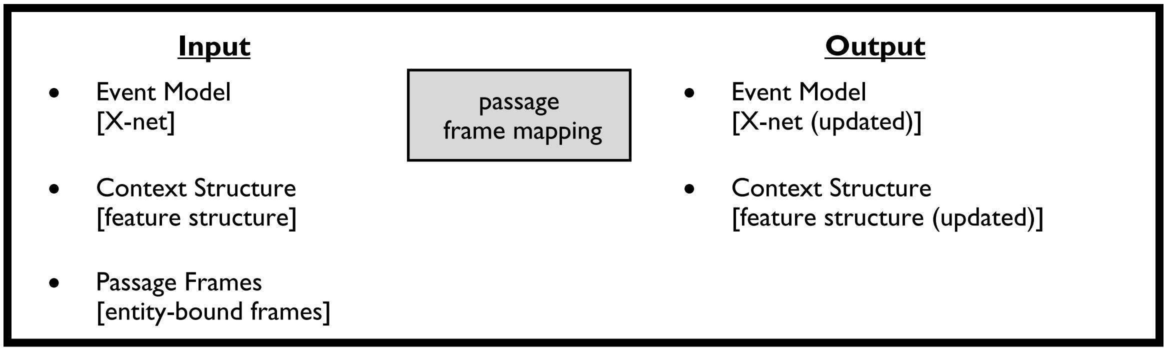


Figure 6-18: Passage Frame Mapping interface

6.4.8 Passage Frame Mapping

In the Passage Frame Mapping module, new data from the passages retrieved is added to the model and context structure (see Figure 6-18). The model and context structure is routed from the Background Fill module, and the new data is passed from the Alignment module, which provides the data in entity-bound frame form.

The entity-bound frames are matched to individual Places and Transitions in the X-net. The system inserts evidence into those components using the same scheme (depicted in Section 4.4.2) as in the Background Fill phase.

Input

- Event Model
  (see Figure 6-11 output)
- Context Structure
  (see Figure 6-11 output)
- Passage Frames
  (see Figure 6-17 output)

Output

- Event Model
  - Treaty model (Iran, CWC)
    tokens in:
    ongoing(negotiation): 1
    ratified_treaty(Iran, CWC): 1
- Context Structure
  - ongoing(negotiation)
    - Frame: Negotiation [Source: BF]
      Interlocutors: <Country>
      Topic: <Chemical Weapons Convention>
  - accept(proposal)
  - complete(negotiation)
  - accept(Iran, CWC)
  - sign_agreement(Iran, CWC)
  - signed(Iran, CWC)
  - ratify(Iran, CWC)
  - ratified_treaty(Iran, CWC)
    - Frame: Ratification [Source: Passage #123]
      Ratifier: <Iran>
      Proposal: <Chemical Weapons Convention>
      ...

Figure 6-19: Passage Frame Mapping example

Running example: Passage Frame Mapping

In our example, the *Ratification* frame matches with the *ratified_treaty* slot in the context structure (see Figure 6-19). The system marks the source of the data (e.g. "Passage #123"). The data translates into a token in the *ratified_treaty* Place in the model.

6.4.9 Simulation and Analysis

The Simulation & Analysis module processes the event model, based on the question type, with the marking (the group of tokens) derived from background and passage information. The result is used to infer as much of the missing information in the Context Structure as possible, tagging the inferences with the source of the data used to make the inference. The Simulation & Analysis module therefore uses the event

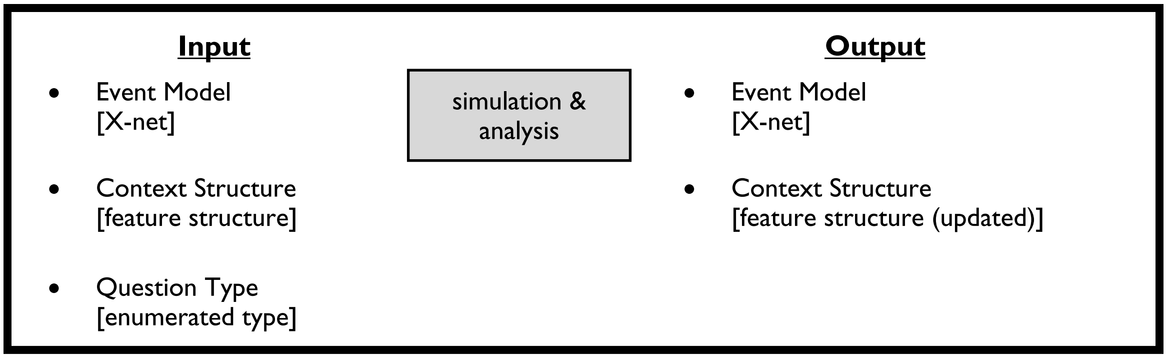


Figure 6-20: Simulation and Analysis interface

model, context structure, and question type, to make changes to the context structure (see Figure 6-20).

We implemented the analysis methods for four question type: *Justification*, *Temporal Projection*, *Ability*, and *Hypothetical*, as described in Section 3.5. The system has two main steps in the analyses: depending on the question type, the system will make necessary changes to the model to fit assumptions in the question; once complete, it will use targeted reachability analysis to infer possible causal chains from the data in the Context Structure to the data missing from the Context Structure. A full review of the reachability algorithm we implemented can be found in Section 3.4.3.

Hypotheticals

Hypothetical questions are treated slightly differently. Recall from Section 3.5, *Hypothetical* event questions hypothesize an altered state about which a more basic question is asked. The questions input into the system thus have two components: the hypothesis and the underlying question. The underlying questions we handle are of the other three question types supported in this system: *Justification*, *Projection*, and *Ability*. The hypotheses we handle are simple additions of information: that some condition holds or some resource is available (e.g. the addition of a car for the question: "if Alice had a car, could she get to work?").

Hypotheticals frequently are asked to understand the difference between the normal state (without the hypothesized change) and the altered state (with the hypothesized change). As such, the system runs analyses twice for *Hypotheticals*: once on the base case (normal state) and once for the hypothesized state. For the hypothesized state, the information from the hypothesis of the question is first added to the model, using the same techniques as used for frames extracted from passages in the Passage Frame Mapping module. The result of the analyses is two copies of the Context Structure: one for the base case and one for the hypothesized case.

Input

- Event Model
  (see Figure 6-19 output)
- Context Structure
  (see Figure 6-19 output)
- Question Type
  (from Figure 6-4 output)
  - Justification

Output

- Event Model
  - Treaty model (Iran, CWC)
    tokens in:
    ongoing(negotiation): 1
    ratified_treaty(Iran, CWC): 1
- Context Structure
  - ongoing(negotiation)
    - Frame: Negotiation [Source: BF]
      Interlocutors: <Country>
      Topic: <Chemical Weapons Convention>
  - accept(proposal)
    - [Source: Inferred - @ratified_treaty]
  - complete(negotiation)
    - [Source: Inferred - @ratified_treaty]
  - accept(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - sign_agreement(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - signed(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - ratify(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - ratified_treaty(Iran, CWC)
    - Frame: Ratification [Source: Passage #123]
      Ratifier: <Iran>
      Proposal: <Chemical Weapons Convention>
      ...

Figure 6-21: Simulation and Analysis example

Running example: Simulation and Analysis

In our example, analysis of the model showed that if the treaty was ratified by Iran, each of the states and actions that causally proceeded it must have also occurred (see Figure 6-21). The missing slots in the context structure are marked accordingly, referencing the ratification data as justification for the inference.

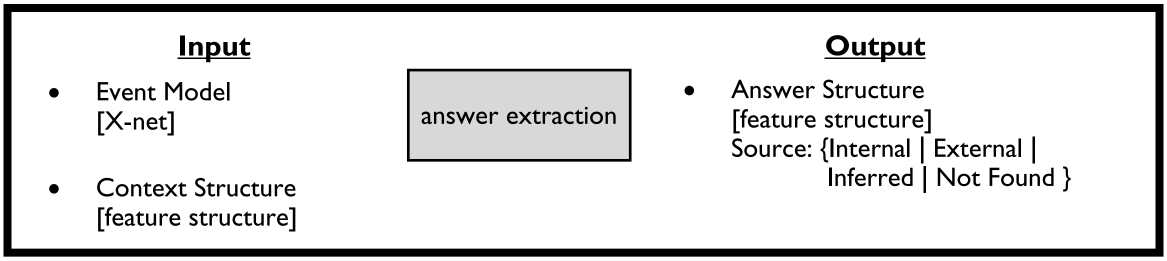

Input
Event Model [X-net]
Context Structure [feature structure]
answer extraction
Output
Answer Structure [feature structure]
Source: {Internal | External | Inferred | Not Found }

Figure 6-22: Answer Extraction interface

6.4.10 Answer Extraction (Output of system)

The principle output of the entire system is a filled Answer Structure (see Figure 6-22). The Answer Structure represents the full set of relations deemed relevant to the question's answer. As mentioned earlier, the Answer Structure is a subset of the Context Structure that the system has been filling throughout the input processing phases described earlier. By the end of simulation, as much of the Answer Structure is filled as possible. A portion is filled with background information ("Internal"), a portion is filled with information retrieved from passages ("External"), and a further portion is inferred through event model analysis ("Inferred"). Some information also may not be found ("Not Found"), which may signify that the relational fact is not true, or the event it represents did not happen; alternatively, it may mean information about the event was unavailable for retrieval. The answer structure slots are each tagged with one of the four Source tags, to help the user with further analysis.

Hypotheticals

The Answer Structure is recomputed for *Hypothetical* questions. Instead of the procedure used in the Model Selection phase, the Answer Structure is determined by comparing the Context Structures output from the Simulation & Analysis phase: one from the base case and one from the hypothesized case. Any differences between the

Input

- Event Model
  (see Figure 6-19 output)
- Context Structure
  (see Figure 6-19 output)

Output

- Answer Structure
  - accept(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - sign_agreement(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - signed(Iran, CWC)
    - [Source: Inferred - @ratified_treaty]
  - *ratified_treaty(Iran, CWC)*
    - *Frame: Ratification [Source: Passage #123]*
      Ratifier: <Iran>
      Proposal: <Chemical Weapons Convention>
      ...

Figure 6-23: Answer Extraction example

two Context Structures is deemed pertinent to the answer. The answer is then extracted as above.

Running example: Answer Extraction

In our example, the answer structure spanned three slots in the context structure (see Figure 6-9): *accept(Iran, CWC)*, *sign_agreement(Iran, CWC)*, and *signed(Iran, CWC)*. No background or passage information was found to fill in any of these slots. Instead, each was inferred through analysis of the data retrieved for the *ratified_treaty(Iran, CWC)* slot (see Figure 6-23). In addition to returning the information on the three slots of the answer structure, the source information (on ratification) is also returned.

To articulate the information about each slot in the answer structure, the system can provide a linguistic description of the slot in frame form. For example, *sign_agreement(Iran, CWC)* can be translated to:

```
Frame: Sign_agreement
  Signatory:     <Iran>
  Agreement:     <Chemical Weapons Convention>
```

and *accept(Iran, CWC)* can be described using the frame:

```
Frame: Respond_to_proposal
  Speaker:        < Iran >
  Proposal:       < Chemical Weapons Convention >
```

6.5 Back-off strategies

The system is dependent on a number of external resources, each with varying coverage. To handle cases where critical resources are not available, we developed a few back-off strategies

- Event Models: If no model matches the question asked, the system backs off to searching for direct frame matches between the question frames and the passage frames. (Fliedner offers a similar solution: Fliedner 2004; Fliedner 2005) The set of frames searched for can be increased by using FrameNet's inherent frame-to-frame relations, though precision will decrease. This latter technique was not implemented in our system
- Frames: FrameNet is an ongoing project covering a significant portion of the frames in the English language, but far from all. While not as rich a semantic source, event models can be grounded in language through PropBank predicate-argument structure (Palmer, Gildea et al. 2005), as was done in Chapter 5, and even just a bag of words optionally linked to WordNet. The latter significantly degrades the precision of the system.

6.6 Implementation details

We make note of the following details of our implementation:

- We coded our system using the Sun Java 6 SDK.

- For the Background Fill module, we also used the Java DB (Apache Derby) database engine to store known facts.
- For our simulation analysis and for creating models, we modified Platform Independent Petri-net Editor 2 (PIPE2), version 2.4 (Akharware 2005), a Petri Net editing package. The modifications provide the ability to: simulate simultaneous transition firings, apply capacity restrictions to *Places*, create test and inhibitory arcs, and run coverability analysis. In addition, our version of the editor supports linking frames to Places and Transitions, with the ability to store the frame descriptions in a custom XML format along with the base X-net representation (in PNML format).
- For references to frames, we used FrameNet 1.3 (Ruppenhofer, Ellsworth et al. 2006).

6.7 Results

In Chapter 6, we have described an intricate QA system capable of utilizing structured linguistic input to catalyze event reasoning to solve difficult event-related questions. In Section 6.7.1, we demonstrate how the system handles *Prediction*, *Ability*, and *Hypothetical* questions. In Section 6.7.2, we describe the system's performance under a number of public evaluations.

6.7.1 Examples of answering target question types

Throughout the chapter, we have walked through an example *Justification* question, regarding Iran's signing of the Chemical Weapons Convention. Next, we will use two additional examples to demonstrate the system's handling of *Prediction*, *Ability*, and *Hypothetical* questions, using our model of the Lebanon War of 2006.

Input

- Question Text

"If Israel wished to conquer Lebanon, would Israel destroy Lebanon's airports?

Output

- Question Frames

Frame: ! Desiring
Experiencer: "Israel"
Event: F-Conquering
Frame: Conquering
Conquerer: "Israel"
Theme: "Lebanon"
Frame: Destroying
Destroyer: "Israel"
Undergoer: "Lebanon 's airports"

- Question Entities

  - "Israel" → <Israel>
  - "Lebanon" →<Lebanon>
  - "airports" → <Airport>

- Question Type

  - Hypothetical; Underlying type: Prediction

Figure 6-24: Hypothetical Prediction example – Part 1

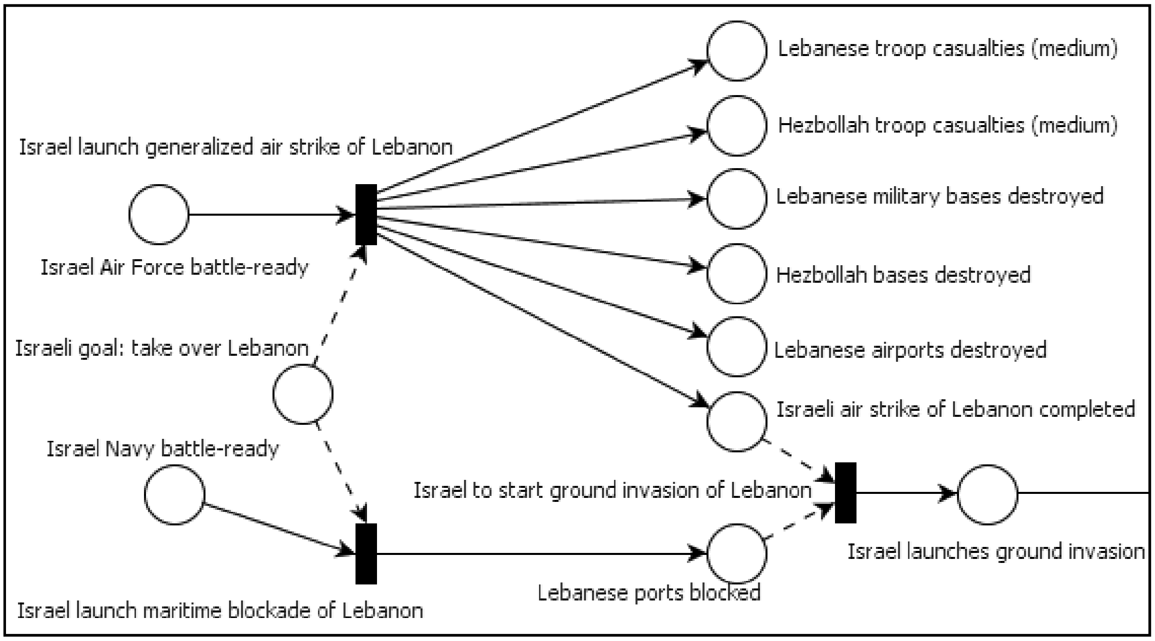
Hypothetical Prediction question example

In our third major evaluation of our system (described below: "Demo 3"), it answered a number of complex questions related to the Lebanon War of 2006. A simple version of one of the questions is: "If Israel wished to conquer Lebanon, would Israel destroy Lebanon's airports?" This question is a *Hypothetical* with an underlying *Temporal Projection/Prediction* query, explicitly hypothesizing Israel's motivation to conquer Lebanon, and then asking whether Lebanon's airports would be destroyed. Our system handles this question as follows:

The idealized front-end system analyzes the question for its frames, entities, and question type (see Figure 6-24). The frames of the question (after alignment) key into the model database and select the Lebanon War model. The Lebanon War Model, as mentioned in Section 4.6.3, is designed to map the major chronology of the war, as well as contain a number of alternative scenarios. (This question asks about one of the alternative scenarios.) It is a specific model, requiring no instantiation (6.4.3).

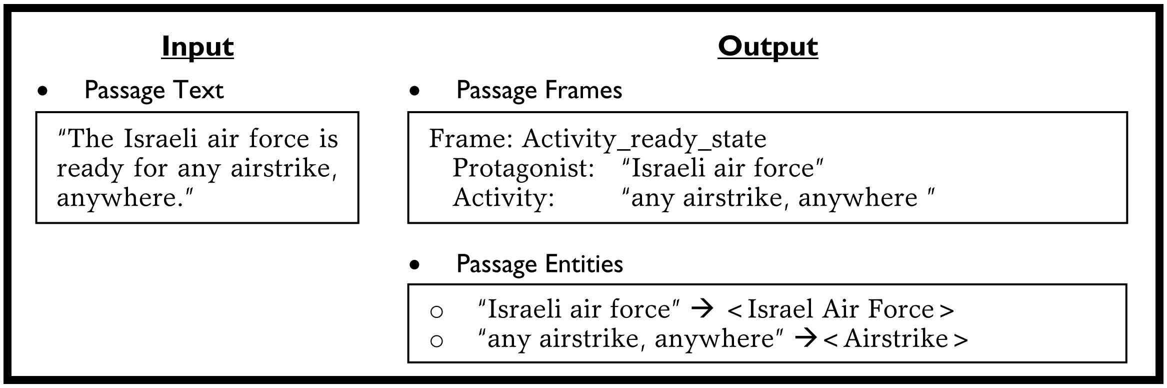
To answer a hypothetical (recall from Section 6.4.9), our system compares the results of analyzing the underlying question under the hypothetical and baseline scenarios; specifically, the system constructs the Answer Structure by calculating the delta between the slots filled in Context Structure between the two scenarios.

An invasion with the intent to occupy is a much larger endeavor than Israel actually took on during the Lebanon War. As such, the outcomes inferred by assuming an intent to occupy will not occur in the baseline case. In the hypothesized case, though, the Lebanese airports would likely be destroyed, if the Israeli Air Force is battle-ready. This is depicted in the relevant section of the model, below:


Lebanese troop casualties (medium)
Hezbollah troop casualties (medium)
Israel launch generalized air strike of Lebanon
Lebanese military bases destroyed
Israel Air Force battle-ready
Hezbollah bases destroyed
Israeli goal: take over Lebanon
Lebanese airports destroyed
Israeli air strike of Lebanon completed
Israel Navy battle-ready
Israel to start ground invasion of Lebanon
Israel launches ground invasion
Lebanese ports blocked
Israel launch maritime blockade of Lebanon

The following steps transpire in the system to achieve this conclusion:

1. Question Analysis & Alignment: the question is entered as frames and entities (Figure 6-24), and they are aligned



Input

- Passage Text

"The Israeli air force is ready for any airstrike, anywhere."

Output

- Passage Frames

Frame: Activity_ready_state
Protagonist: "Israeli air force"
Activity: "any airstrike, anywhere "

- Passage Entities

○ "Israeli air force" → <Israel Air Force>
○ "any airstrike, anywhere" → <Airstrike>

Figure 6-25: Hypothetical Prediction example – Part 2

2. Model Selection:

   2.1. the *Destroying* frame keys into the model database, matching the "Lebanese airports destroyed" Place in the Lebanon War model

   2.2. the Context Structure is created, with a depth of 2 (Section 6.4.3)

3. Background Fill: no information is available

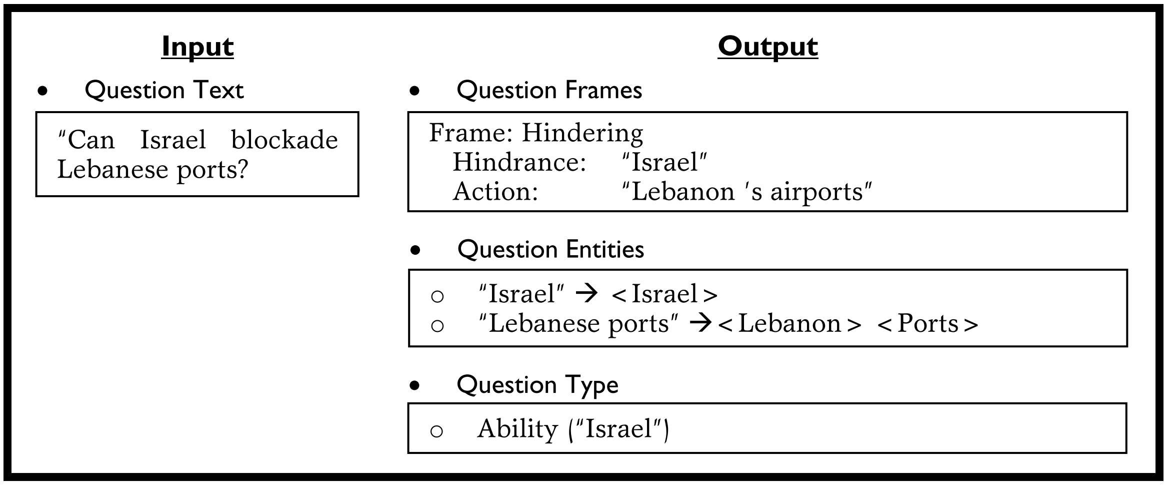
4. Query Expansion: the remaining Context Structure slots regarding "Israel launch generalized air strike of Lebanon", "Israeli goal: take over Lebanon", and "Israel Air Force battle-ready" are requested

5. Passage Analysis, Alignment, and Mapping: With the passage shown in Figure 6-25, a token is added to the "Israel Air Force battle-ready" Place.

6. Simulation & Analysis: The question type, *Hypothetical*, forks off two processes: one for the base case, and one for the hypothetical case

   6.1. The base case runs as a normal prediction question, using Forward Reachability analysis to ascertain that the Lebanese airports will not be destroyed.

   6.2. The hypothetical case adds in the influence of the hypothetical. The frames of the antecedent of the question match with the "Israeli goal: take over Lebanon" Place. Because of this is information, the Place received a token. Reachability



Input

- Question Text

"Can Israel blockade Lebanese ports?

Output

- Question Frames

Frame: Hindering
Hindrance: "Israel"
Action: "Lebanon 's airports"

- Question Entities
  - "Israel" → <Israel>
  - "Lebanese ports" →<Lebanon> <Ports>
- Question Type
  - Ability ("Israel")

Figure 6-26: Ability example

analysis is run again, this time reaching the conclusion that Lebanese airports will be destroyed.

7. Answer Extraction: The different resulting Context Structures from the two X-net analyses are contrasted; the Lebanese airport destruction is computed to be different, and that fact is returned for presentation to the user, along with a reference to the passage used in the inference.

Ability question example

Ability questions are handled quite similarly. As stated in Section 3.5, *Ability* questions are implicit *Hypotheticals*: they use a hypothesis of the existence of an actor's intent, in order to ascertain whether the actor can reach a target state. For the question, "Can Israel blockade Lebanese ports?", we can assume any necessary precondition towards the state in question that involves the motivation of Israel.

The system approaches this question after the initial question analysis by the front-end system (shown in Figure 6-26). Note that the question type analysis identifies both the question type (*Ability*), as well as the principle actor whose ability is in question ("Israel"). This allows the system to find objects in the model eventually selected that are marked as representing the motivation of that actor. For each such Place, a token is added, and outgoing arcs are made to be *enable* arcs (see Sections 3.4.1 and 3.5 for review).

In this situation, once again, the Lebanon War model is selected based on the question. The "Israeli goal: take over Lebanon", tagged by the designer as motivation ("Israeli goal"), is assumed to be true and a token is added to its Place. With the addition of the knowledge that the "Israel Navy is battle-ready", a simple inference leads the system to conclude that Israel could blockade the Lebanese ports.

This transpires as follows:

1. Question Analysis & Alignment: the question is entered as frames and entities (Figure 6-26), and they are aligned
2. Model Selection:
   - 2.1. the *Hindering* frame keys into the model database, matching the "Lebanese ports blocked" Place in the Lebanon War model
   - 2.2. the Context Structure is created, with a depth of 2 (Section 6.4.3)
3. Background Fill: the fact that "Israel Navy is battle-ready" is retrieved from internal resources.
4. Query Expansion: the remaining Context Structure slots regarding "Israel launch maritime blockade of Lebanon" and "Lebanese ports blocked" are requested
5. Passage Analysis, Alignment, and Mapping: no passages are found
6. Simulation & Analysis: The question type, *Ability* uses basic Forward and Backward Reachability analysis to ascertain that the Lebanese ports could be blockaded.
7. Answer Extraction: The resulting Context Structure from the X-net analysis provides that the Lebanese port could be blockaded based on the assumed

motivation and the background fact about the Israeli Navy, and that fact is returned for presentation to the user.

6.7.2 Evaluations and Demonstrations

We conducted our Question Answering work in partnership with our teammates at the University of Texas, Dallas, Stanford University, and the Language Computer Corporation, as a part of the AQUINAS project (Answering Questions Using INference and Advanced Semantics). The AQUINAS project was funded under the ARDA AQUAINT program (Advanced QUestion Answering for INTelligence). As a part of this program, we took part in a number of evaluations of our system. Many of the preceding examples used in this dissertation were pulled from these evaluations.

At the outset, it must be made clear that these evaluations were pilot studies and made assumptions that are detailed further in individual subsections. Taken together, they point to the potential of our techniques to significantly improve the state of the art in semantic question answering.

Demo 1 – Entailment Evaluation

After our initial Answer Selection success (discussed in Section 5.5), we tested the preliminary version of our full QA system on an entailment task evaluation. Ten research teams from universities and industry participated, split into two groups of five. Each team supplied, ahead of time, their own data set of 30 to 40 questions to be tested by all the teams in their group.

In the entailment task, each question is paired with a short passage, generally not longer than three sentences. The objective for each question is to determine if the passage entails an answer to the question, and if so, what that answer is. Most questions required only "yes", "no", or "unknown" (not knowable with available information) answers, though a handful (like our example in Section 4.1.2) required a specific named entity. A typical example from the evaluation is:

Passage: In the early 1990s, Iran reportedly acquires 120 tons of castor beans, used in the production of the toxin ricin.

Question: Is there evidence that Iran has attempted to produce ricin?

Answer: "Yes"
(Because): Acquisition of a component of something is a necessary step in producing something.

(providing the reasoning was not required).

We tested our system against the data sets of our evaluation group: two teams from the Language Computer Corporation (LCC), Stanford University, and the University of Texas, Dallas (UTD), as well as our own data set, for a total of 131 questions. Across the five teams (and five data sets), we performed the best on four of the five sets. This table shows the percent (%) correct answers achieved by each team on each dataset:

	ICSI Data	LCC 1 Data	LCC 2 Data	UTD Data	Stanford Data
ICSI	**73.28**	**83.96**	**70.99**	**67.17**	51.14
LCC 1	35.87	76.33	42.74	35.11	54.19
LCC 2	51.14	56.48	51.90	49.61	45.03
UTD	58.77	59.54	59.54	59.54	57.25
Stanford	51.90	69.46	56.48	50.38	**63.35**

Our results, though, come with a number of caveats. We assumed gold standard text analysis, frame parsing, entity tagging, question analysis, and model invocation. We also had a number of relevant event models already available for these particular data sets. For any remaining event-related questions that we did not have models for, we created small, course-grain models that were sufficient for the analysis required.

For any other question, we backed-off to finding frame matches between the passage and question, and used positive matches as indication of a positive entailment. Due to the limited size of the total data set and the nature of the evaluation, it was possible to pre-construct much of the data required for analysis with our system.

Demo 2 – Question Answering

To better test our system's ability to use complex event structures, we participated in another public AQUAINT-related demonstration. Together with our partner teams, we tasked ourselves with trying to answer a series of questions about the complex scenario of: North Korea's development of biological weapons.

We designed our intricate Biological Weapons Production model, described in Section 4.6.3, to assist us in answering 15 selected questions, including these two examples:

Is there evidence North Korea is building a BW research laboratory?
What are the effects of North Korea acquiring a BW delivery system?

As compared with Demo 1, here we did not have access to gold-standard frame annotations of the data required to answer these questions. Instead, we received data at run-time from the UTD front-end system (as described in Section 6.1). At the time of the demo, though, the front-end system was not yet capable of accepting our expanded query of entity-restricted frames at the end of our system's first stage (recall Figure 6-2). Instead, it was only able to provide data passages initially retrieved based on the text of the question. Just as in the Answer Selection evaluation (Section 5.5), the vast majority of the passages were not applicable to the event question at hand,

and the frames of the passages would not match any of the slots in the Context Structure we computed for the questions. The limited relevant data we were able to extract from the passages provided by the front-end were typically insufficient to infer substantial new facts.

For example, the question, "Is North Korea capable of producing weaponized anthrax?", focuses the attention of the system on the action of 'production of anthrax' in the model. This action is only enabled at the culmination of many research and development steps, each with multiple dependencies. Receiving information on North Korea's development of a biology research lab, while being a relevant detail, is causally distant from the 'production' action. Without information on whether North Korea has acquired a strain of the anthrax virus, the model cannot infer further progress by the country towards full production of the weaponized form.

Two positive outcomes were achieved in this demonstration. Firstly, when, for any of our questions, we bypassed the passage input and directly added information requested by the system (empty slots in the Context Structure), we were able to infer significant data in the Answer Structure. Secondly, the passages retrieved by the front-end system that did match with our context structures, were relevant to the question asked.

Demo 3 – Question Answering

So as to show the ability of the system to answer complex questions about complex scenarios while relying on natural language input, we took part in a third demonstration. Here we developed a model about a specific event: the Lebanon War of 2006, between Israel and Lebanon/Hezbollah.

We selected six questions. Ahead of time, we ran them through the first half of our system (to the point of Query Expansion). We submitted the expanded query to our partner team, who returned 100 passages per queried Context Structure slot, receiving approximately 2400 passages in total. We filtered out passages that were obviously irrelevant and that would not produce frame annotations that would match with our model. From the remainder, we selected the top 10 to 15 results and hand annotated and entity tagged them. With this data, we attempted to answer the questions.

We were able to answer all six questions, two of which were used as examples in the preceding section (Section 6.7.1). The limited scope and deep, high-quality data set were instrumental in achieving this outcome.

7 Applying Event Reasoning to Pathway Classification

Classification of dynamic system pathways is a challenging and important research problem that can be addressed using event modeling and reasoning. As mentioned in Section 3.6, pathways are evolution trajectories of complex dynamic systems that serve a particular goal. To classify an unknown, partially-observable pathway requires comparing the behavior of the unknown pathway to the expected behavior of each pathway hypothesis. This task provides a valuable test of our event reasoning framework, because, unlike our previous examples in Question Answering and Answer Selection, here uncertainty exists about the actual structure and purpose of the events of interest.

In Chapter 3, we described our methodology for creating active event models; with Section 3.6, we detailed the analysis routines required to use the models in pathway classification tasks. In this chapter, we apply these techniques to a system built for pathway classification, and we demonstrate its capability on models an order of magnitude more complex than those used in Chapters 5 and 6.

In Section 7.1, we describe a classification system in which we created a central component, the Pathway Inference Engine, and we contrast this work with our Question Answering work. We then give an overview of our engine, highlighting its deliverables to the classification system (Section 7.2). We detail each module of the engine in Section 7.3, and we provide implementation details in Section 7.4. Finally, in Section 7.5, we describe multiple demonstrations of the system.

7.1 Task and System context

A **pathway** can be any set of activities, organized around temporal and causal structure, that proceed from a starting point to a terminal point in the service of a designated task. They can be complex processes that include redundancy and alternative options, with individual **segments** of a pathway creating and consuming resources and changing state. In real-world pathways, though, not all segments may be observable, and one may wish analyze a pathway to estimate its unobservable segments and/or the intent of those who manage it. We seek to facilitate these analyses.

7.1.1 PCLASS system

Our pathway inference research was conducted within the context of a larger project (PCLASS). The project's overall system, diagramed in Figure 7-1, is designed to classify a partially-observable real world production pathway as fitting one of a given set of hypotheses about it. (It thus attempts to answer *Hypothesis Disambiguation* questions, one of our objectives stated in Section 3.2.) Classification problems are well understood in the probabilistic modeling literature (Weiss and Kulikowski 1991; Michie, Spiegelhalter et al. 1994). Our task is to use pathway simulation and analysis techniques to compile hypotheses about complex stochastic, dynamic systems into a form where standard techniques can be applied (as described in Section 3.6.2).

The system is executed in three stages with respect to a particular pathway of interest and a probing strategy (described below). The first stage provides the data, given the probing strategy, that will later be used to train a classifier for each hypothesis class. The second stage records observations of the real pathway, given

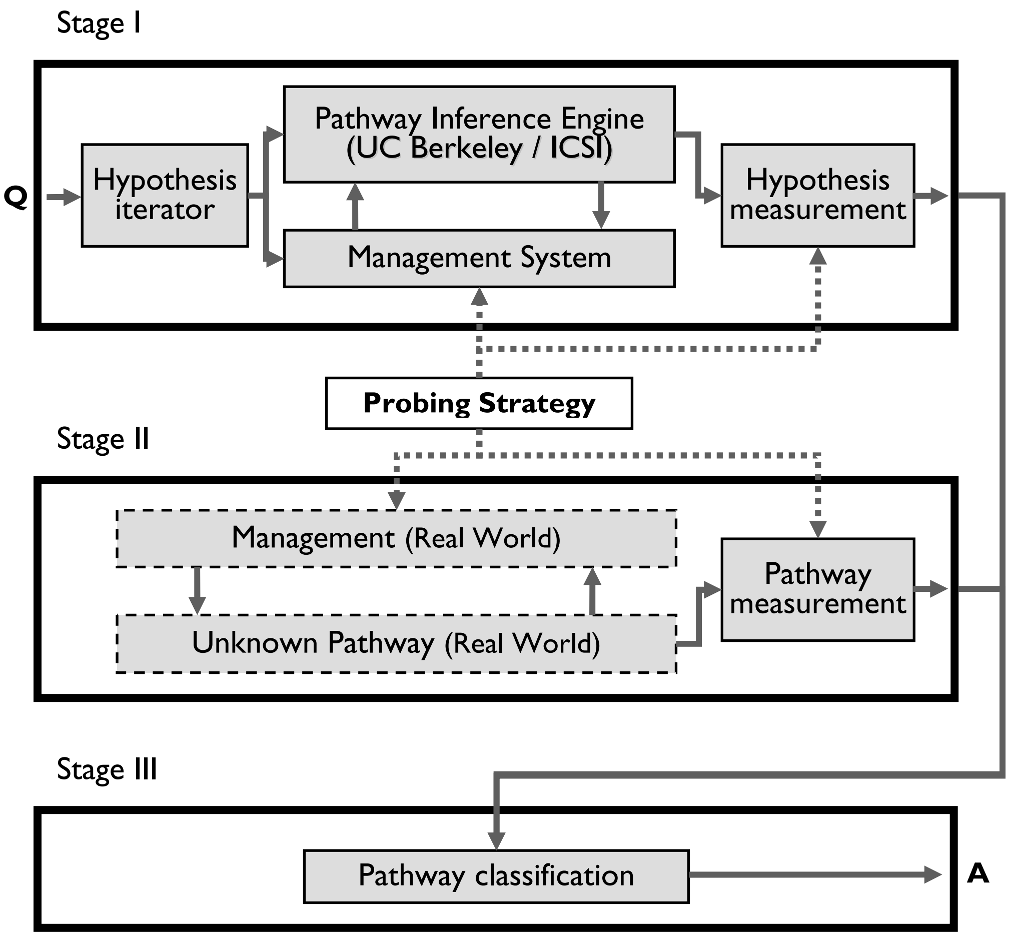


Figure 7-1: PCLASS system flowchart

the same probing strategy. And the third stage computes a posterior distribution over the hypotheses for the pathway being classified.

The first stage ("Stage I") has four major components. A query enters the system, specifying a set of hypotheses about the structure of the pathway, and its decision making (management's) structure and intent. Each hypothesis consists of a (partially known) stochastic, dynamic model of the full pathway, and a (hypothesized) model of its management. The "Hypothesis iterator" submits each hypothesis in turn to our "Pathway Inference Engine" and the "Management System". The Pathway Inference Engine and Management System simulate the evolution of the hypothesized models, with the Management System simulating the management model's dynamic allocation

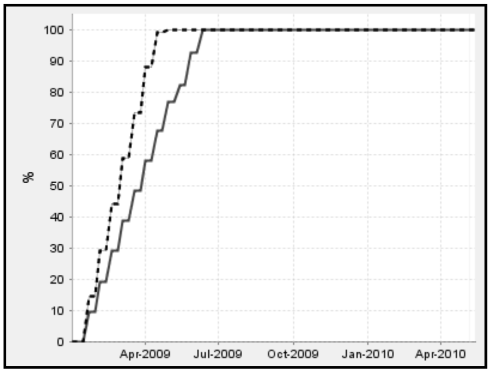


Figure 7-2: Change of Rate

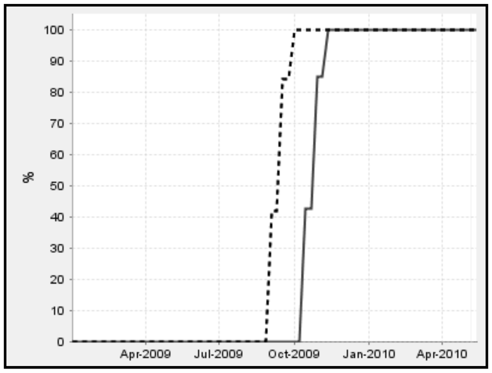


Figure 7-3: Segment Delay

of resources to the pathway model, and the Pathway Inference Engine simulating the pathway's use of those resources. As the simulation progresses, observable features of the pathway are sampled ("Hypothesis measurement"). These samples (with simulated measurement error) are the data used to train the classifier of the hypothesis.

The second stage ("Stage II") has three components. The observable features of the real-world "Unknown Pathway" are measured ("Pathway measurement") through a sampling process similar to that in Stage I. These observations are used to classify the pathway and its management ("Management") as fitting one of the hypotheses.

In the final stage ("Stage III"), the classification task is executed ("Pathway classification"), calculating the posterior distribution of the hypotheses given the observations. The system chooses the most probable hypothesis class from the distribution.

7.1.2 Probes

In certain scenarios (especially those where the different hypotheses share many pathway segments), the posterior distributions over hypotheses given observations will not provide a clear classification without intervention. A strong correlation may exist between the real pathway and multiple hypotheses with respect to the behavior of the observable segments.

As described in Section 3.6.3, a probe may be able to tease apart the hypotheses. Consider a pathway segment that uses a certain amount of resources to complete a task over a certain amount of time. By actively probing the segment, resources used by the segment can be slowed, stopped, or delayed. The effects of the probe will be manifested at multiple points. In Figure 7-2, we show the *direct* effect of a slowdown of resource inputs for a given segment on that same segment. Compared to the default, unprobed simulation (dotted line), the probed simulation (solid line) shows a negative change of rate to completion (x-axis is time, y-axis is percent completion). In Figure 7-3, we show how a delay in resources of one segment can affect the time to completion of another segment. By slowing down the completion of a segment that enables the one shown, this segment will have a delayed start (solid line) compared to the unprobed scenario (dotted line). Of course, the results of a probe can be even more dramatic if it fully stops the resources to a segment, leading all segments reliant on its completion to fail to start. If a probe can have varying, unique effects on the set of hypotheses in a question, classifications of a real pathway will be more clear cut.

Probe strategies

Optimal probe strategies for pathway classification are highly dependent on what type of pathway the user is trying to identify, and their means and constraints on manipulating resource allocations (active probes) or providing new observations or measurements (passive probes) for that pathway. While it is situation dependent, our system can facilitate the design of such strategies by providing the platform for evaluating the value of information of each candidate probe. In Section 3.6.3, we lay out in detail the process for calculating the entropy reduction provided by a probe for a set of hypotheses. In evaluations of probes for the PCLASS system, this entropy reduction value is calculated and submitted to a probe strategy tool that uses stochastic search algorithms to generate improved probes (this is outside the scope of our work).

7.1.3 System design scope

Our piece of the PCLASS system is the Pathway Inference Engine. Our focus is on modeling and simulating complex pathways, and providing meaningful measurements of their progress. To accomplish this, we fully exercise the ability of our event modeling framework to represent complex technology pathways, and extend it to handle dynamic data input. Our system design does not cover resource allocation and management, nor the components in Stages II or III. That work, implemented by our research partners, is outside the scope of our research and is not discussed further here.

Key differences with Question Answering research

There are a number of significant differences that we wish to highlight between this system and the Question Answering system (described in Chapter 6).

- The level of complexity of the models tested is significantly higher. This is possible because in this work we remove the dependency on external language processing systems that constrain the type and amount of data that can be input into our system. With greater and more direct access to data, we have been able to test complex single models, as well as multiple models interacting with one another. Those pathway models incorporate the event controller

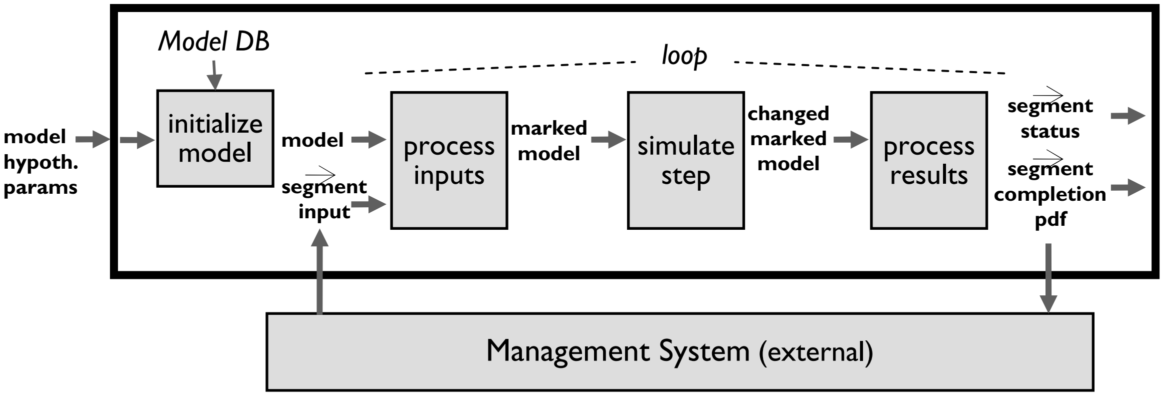


Figure 7-4: Pathway Inference Engine flowchart

(Section 3.4.2) for each segment, directly informing how we simulate and assess progress.

- The resource allocation to the models is interactive during a simulation. This significantly increases the dynamic range of the evolution of the pathway response. It also allows for more accurately modeling the ability of actors to change behavior during a simulation, based on incremental results.
- Real valued data can be used through input mapping functions. This increases the scope of models to be able to handle uncountably infinite data sets.
- The system solves a different type of question, *Hypothesis Disambiguation.*

7.2 System modules and process flow

Our Pathway Inference Engine has a simple sequence of modules with a loop, as shown in Figure 7-4. For each hypothesis, the PCLASS system activates our component to simulate the hypothesized pathway model over time. Snapshots of the status and level of completion for each of the segments of the pathway model are sent to the Management System at each simulation step. They are also sent to the greater PCLASS system in order to build the hypothesis classifier.

Specifically:

- The **model initialization** phase: The system loads the model structure for a given hypothesis and initializes it with a set of input parameters. These input parameters specify the resource requirements, minimum duration, and level of observation uncertainty of each segment.
- The **input processing** phase: The system goes into a simulation loop. At each step, it receives and processes sample resources allocated by the external Management System.
- The **step simulation** phase: The system then simulates forward for a set amount of time. Segments ready to start, start. Segments ongoing and with resources, continue or complete.
- The **result processing** phase: Finally, the system computes for each segment a) its new status; and b) a measurement of its level of completion. This information is relayed back to the Management System to help it determine how to allocate resources in the next step. It is also communicated to the Hypothesis Measurement module (see Figure 7-1) which stores the values for observable segments and later uses them to calculate the hypothesis classifier.

7.2.1 System outputs

The Pathway Inference Engine is responsible for generating simulated observation data for each segment of a given hypothesis model, for each time step, for a set of input data. The key measurement for each segment is its observed degree of completion (0% = not started, 100% = finished). Observations, though, can be noisy. As specified above, one of the model initialization parameters for each segment is the degree to which observations are rated to be uncertain (levels: deterministic, small uncertainty, large uncertainty). The engine generates a distribution (a probability density function) over the degree of completion of a segment, based on its level of uncertainty.

Specifically, pathway observations have the following features: (*adapted from an unpublished PCLASS design specification*)

- The degree of completion of a pathway segment is a percentage (0–100%).
- Each distribution over the degree of completion is discretized and represented as a histogram of 12 bins. Each bin represents the probability that the degree of completion (%) is within certain bounds:

Bin 1	0% (not started)
Bin 2	0.01 – 10.00% complete
Bin 3	10.01 – 20.00% complete
. . .	
Bin 11	90.01 – 99.99% complete
Bin 12	100% complete (finished)

- Different levels of known uncertainty associated with observations for each segment will produce distributions with different levels of variance in the estimate. (See example in Figure 7-5, for a segment "P9")
    - Deterministic (level "1"): The observation has no uncertainty; when the segment is active it is clearly observable.
    - Small uncertainty (level "2"): A small amount of uncertainty in observations as to when a segment is staring, active and stopping.
    - Larger uncertainty (level "3"): A large amount of uncertainty in observations as to when a segment is staring, active and stopping.

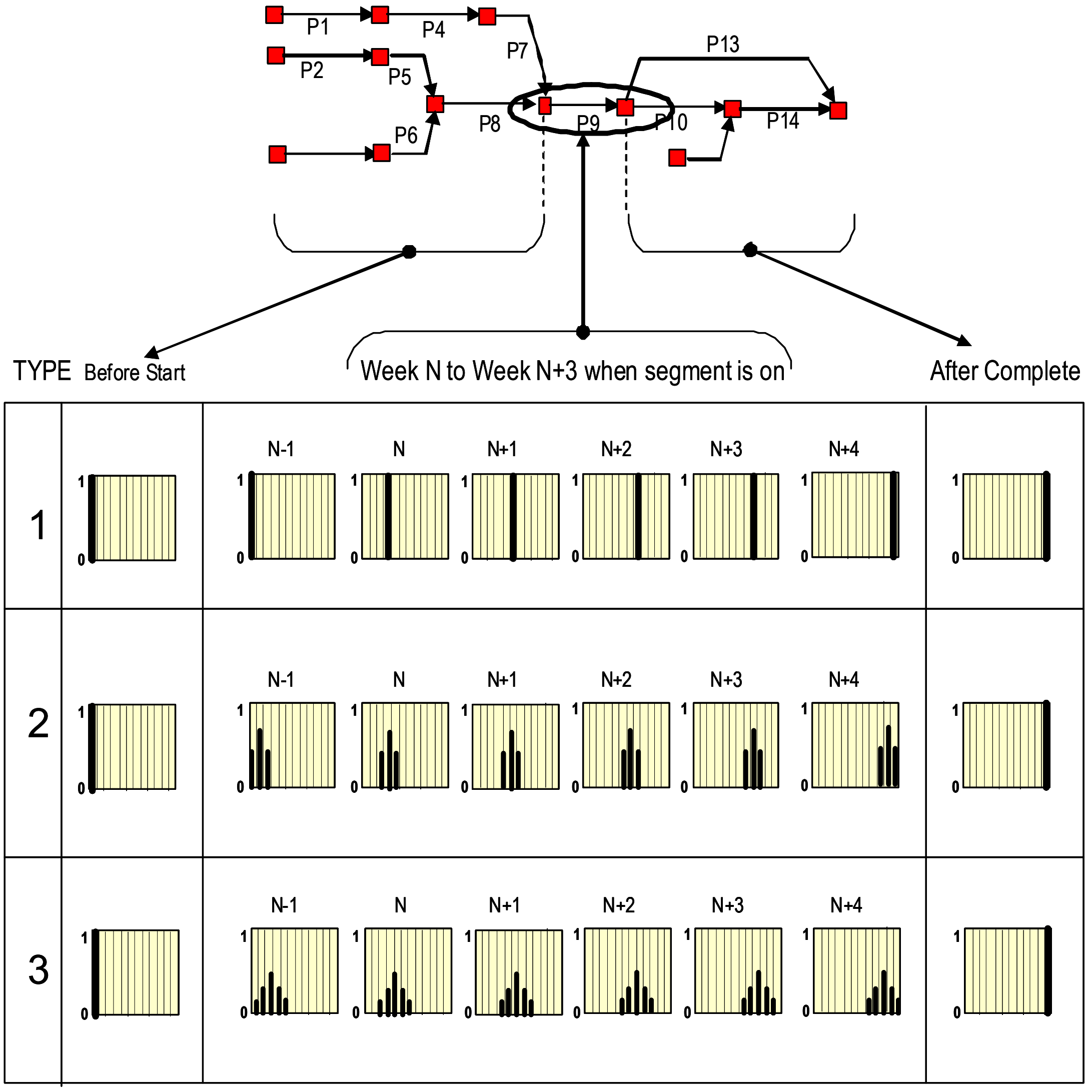


Figure 7-5: Pathway Probability Distribution Functions

A simulation thus creates a matrix of distributions: one per segment per simulation time step. This is the data used to create hypothesis classifiers.

7.3 Module details

In this section, we detail each module depicted in Figure 7-4.

7.3.1 Model Initialization

Pathway models are composed of interconnected segments. In Section 3.4.2, we discussed a general, common event controller (Figure 3-5), relating events to other events based on their trajectories. Here we are able to deploy that controller, using a subset of the relations, as shown in Figure 7-6. We replicate the structure for each segment, but must tune the resource requirements in the controller for the scenario.

The Model Initialization phase is activated by a set of hypothesis parameters. These parameters are composed of three principle pieces: 1) a model identifier; 2) a set of constraints for each segment of the model; and 3) the simulation step size.

- The identifier is a single field specifying which pathway model to load from an internal database. This pathway model is a dynamic event model (an X-net) depicting the interconnected structure of the pathway's segments.
- The constraints are specified with three parameters for each segment: 1) the amount of resources required to complete the segment, specified as a vector of [resource type, amount] pairs; 2) the amount of time (duration) required to complete the segment when operating with adequate resources; and 3) the observation level of uncertainty (as described in Section 7.2.1).
- The step size is a single value representing the period between simulation steps (the internal clock increment).

The system retrieves the pathway model structure from its database keyed by the given identifier, then tunes the behavior of the model's segments using the supplied parameters to match the desired behavior.

Tuning the model with the constraint parameters is accomplished primarily through the tuning of the functions mapping simulation step inputs to X-net tokens (explained in the next section) and the setting of certain Transition delays. Within the X-net, we treat the amount of resources required to complete an entire segment as a percentage – specifically 100%, represented by a required total of 100 tokens (we can

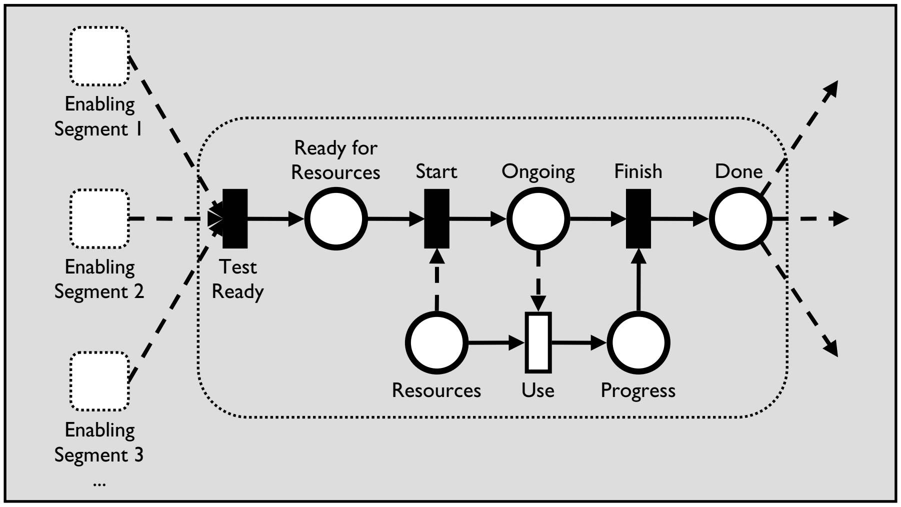


Figure 7-6: Pathway segment controller

use 1000 to increase in precision, if desired). The 'Progress'-'Finish' arc is thus fixed with a weight of 100. We assume that with adequate resources, progress in a segment is made uniformly. The value of ($100 * \frac{simulationStepSize}{sementDuration}$) specifies then the maximum number of tokens that can be consumed by the 'Use' Transition per simulation step. To achieve this, we set the delay on the 'Use' Transition to ($\frac{segmentDuration}{100}$).

As an example, if a segment representing an event that takes 4 weeks to accomplish, and the simulation step was 1 week, each week, at most 25% of the task could be completed (with our assumption of uniform progress). Thus each week, at max, 25 tokens could be consumed by 'Use' and created in 'Progress'. We achieve this in our model by setting the delay on 'Use' to (4 weeks/100), which entails up to 25 firings within the time of one simulation step. At the end of four weeks, 100 tokens should be in 'Progress', allowing 'Finish' to fire.

7.3.2 Input Processing (per step)

With a tuned model in hand, the system is prepared to accept resource inputs from the Management System. To support scenarios where interim outcomes can affect external decisions that can subsequently change long term trajectories, we require a system that can accept dynamic, incremental input. This moves beyond the single instance input system described in Section 4.4 and implemented in the Question Answering system of Chapter 6.

Each step of simulation begins with an input of the current state of the model, and the set of new resources allocated by the resource management system. The model is passed either from the Model Initialization module or from the result processing module after the last step of simulation. The new resources are passed from the Management System in the form of a set of $\mathbb{R}^+$ valued vectors, one vector per segment.

We can represent the resource requirement parameters of a segment as the vector $X_{1..N}$, and we can represent the resource inputs of that segment for the current simulation step as the vector $x_{1..N}$. The system maps these inputs to tokens to be placed in the 'Resource' Place of the segment (Figure 7-6). The function mapping the inputs calculates the number of tokens using the following equation: $\lceil \frac{100}{N} * \sum_{i=1..N} \frac{x_i}{X_i} \rceil$. The system verifies that, over all simulation steps thus far, the aggregate total of the i^{th} resource is not greater than X_i. For each segment in the pathway model, the system applies this input mapping.

There is an underlying simplifying assumption we use in our system: that the Management System distributes different types of resources to a segment uniformly, as a percentage of the total required amount of each resource. (An alternative interpretation also supported: the relative distribution of different resources for a particular simulation step doesn't affect that step's progress, in and of itself.) This allows our system to collapse all input resources into one group of tokens (which are untyped in our design). (Note that the system is capable of simulating more complex resource interactions within segments by using custom segment structures represented in the X-net. For this project, we did not require support for such detail.)

7.3.3 Simulation step

Instead of applying a reachability algorithm (Section 3.4.2), projecting all possible paths forward with the given set of resources, we require our system to simulate forward for a fixed amount of time, before receiving new resources and stepping again. (Note that while there is uncertainty in measurements of progress towards completion of a segment, the structure of pathway segments does not allow for competition between resources during simulation and thus there will not be multiple potential trajectories within a step; any resource competition is resolved in the Management System prior to allocation.)

Our X-net simulation and analysis engine, built to follow the simulation semantics described in Section 3.4.1, supports this need. After receiving new input for a simulation step, the system can simulate forward the progress enabled by those resources (and any leftover resources unused from previous steps). The simulation step size is the amount of time simulated. Immediate Transitions fire as they become enabled. For segments in the state of 'Ongoing' that have tokens in 'Resources', the 'Use' Transition will theoretically fire as many times as it can within the step period (refer back to Figure 7-6). For performance purposes, we collapse the potential numerous firings per simulation step into one bulk move of up to the maximum number of tokens consumed under normal firing semantics from 'Resources' to 'Progress'. This token count represents the mean of the stochastic (exponential) distribution. With the work of the segment simulated and the progress computed (displayed in the tokens in 'Progress'), the system fires any newly enabled Immediate Transitions ('Finish' and 'Test Ready' transitions, if possible).

7.3.4 Results Processing (per step)

At the conclusion of each simulation step, two vectors are computed. One is the estimated level of completion of each segment (as a discrete distribution over percentage complete, as described in Section 7.2.1). The second is the segment status: "Completed", "Ongoing", "Ready" to receive resources, or "Not ready". These vectors are passed externally to the Management System, as well as the Hypothesis Measurement module.

Status outputs are most useful to the Management System, providing it direction on which segments to provide with resources in the next step. When a simulation step is complete, retrieving the current status of each pathway segment is accomplished by analysis of the X-net marking. If there is a token in 'Done', the segment is "Complete". If not, but there is a token in 'Ongoing', the segment is "Ongoing". If not 'Ongoing', but there is a token in 'Ready for Resources', the segment is "Ready for Resources". If not, the segment is "Not ready".

The level of completion is also quickly read from the model. If the segment is "Complete", it is 100% complete. If not, the number of tokens in 'Progress' specifies the mean estimate of the percent complete.

The Management System always receives precise, complete information.

The Hypothesis Measurement receives a distribution reflecting the uncertainty of the simulated observation (Section 7.2.1). We use beta distributions for the non-deterministic observations, with a higher variance for those observations specified as having larger uncertainty.

7.4 Implementation details

The Pathway Inference Engine uses an underlying X-net simulation and analysis engine similar to the version used in the QA system described in Chapter 6. It uses a modified version of the Platform Independent Petri-net Editor 2 (PIPE2), version 2.4, with the modifications providing an ability to: simulate simultaneous transition firings, apply capacity restrictions to Places, create test and inhibitory arcs, and run coverability analysis. All code is in Java 6. Pathway models are stored in Petri Net Markup Language (PNML: ISO 2008), and resource requirements are stored in tab-separated format.

Due to the regularity of the control structure employed in the models (Figure 7-6), we were able to heavily optimize the firing functions of PIPE2, leading to a 115x improvement in runtime speed. This enabled a significant increase in the number of simulations our system could achieve during tests.

The Pathway Inference Engine connects to the rest of the PCLASS system through the Backplane engine of one of our research partners. The Backplane provides an API through which our system connects to the Management System and Hypothesis Measurement module.

7.5 Demonstrations

We evaluated this system in three multi-university demonstrations. The purpose of the demos was to test and show how active probes could increase diagnosticity, providing greater separation of hypotheses. This is an important step before classifying a real unknown pathway. The challenge is, with only nominal probes (probes that expose only observables in a model that are available without any significant effort), the observables available and the values obtained from the observables may make hypotheses statistically indistinguishable (the entropy given the observables will still be high). By perturbing the inputs and/or exposing additional observables, the hypothesis measurements may allow for a cleaner classification of an unknown pathway.

7.5.1 Demo Models

For each demo, we used elaborate pathway model(s) of technology development, with two hypotheses. The technology pathways include such stages as R&D, pilot production, testing, and full production, split amongst multiple segments.

Demo 1 model

In the first model, shown in Figure 7-7, a company may be making one or two products. Hypothesis 1: the company is just making "S" (the Safe product); Hypothesis 2: the company is making both "S" and "N" (the Not safe product).

The arcs shown in the figure are pathway segments, activities required to complete the production of either "S" or "N" (e.g. "S20" is for "S", "N29" is for "N"). Nodes represent synchronization points: arcs going into nodes represent all the segments that must complete to enable all the outgoing segments. The one major

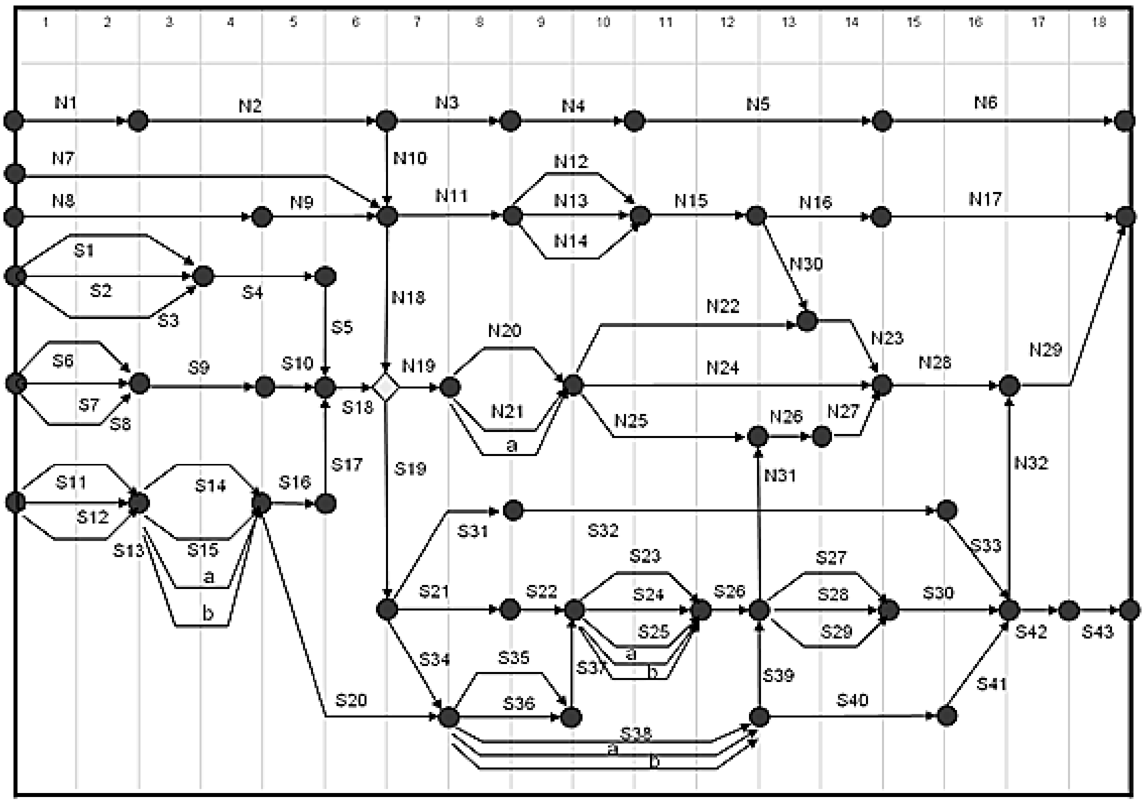


Figure 7-7: Demo 1 – Products S & N model

exception are arcs labeled "a" or "b", which are *alternative* segments. For example, segment "S15" has two alternative segments ("S15a" and "S15b"); completion of any of the three segments is sufficient to enable "S16" and "S20", if "S14" is also complete. (One minor clarification: at the diamond node, "N18" enables "N19", but not "S19".) At full speed, the entire pathway takes 18 months to complete (represented by the vertical lines). Arcs that are completely vertical are segments that complete in zero time.

The pathway has a dependency of "N" on "S", but not vice versa. (Note that any connections between "N" and "S" segments go from "S" to "N".) Furthermore, in this scenario, only a subset of "S" segments are observable; no "N" segments are. Thus, the company's choice to create "N" or not create "N" cannot be observed directly.

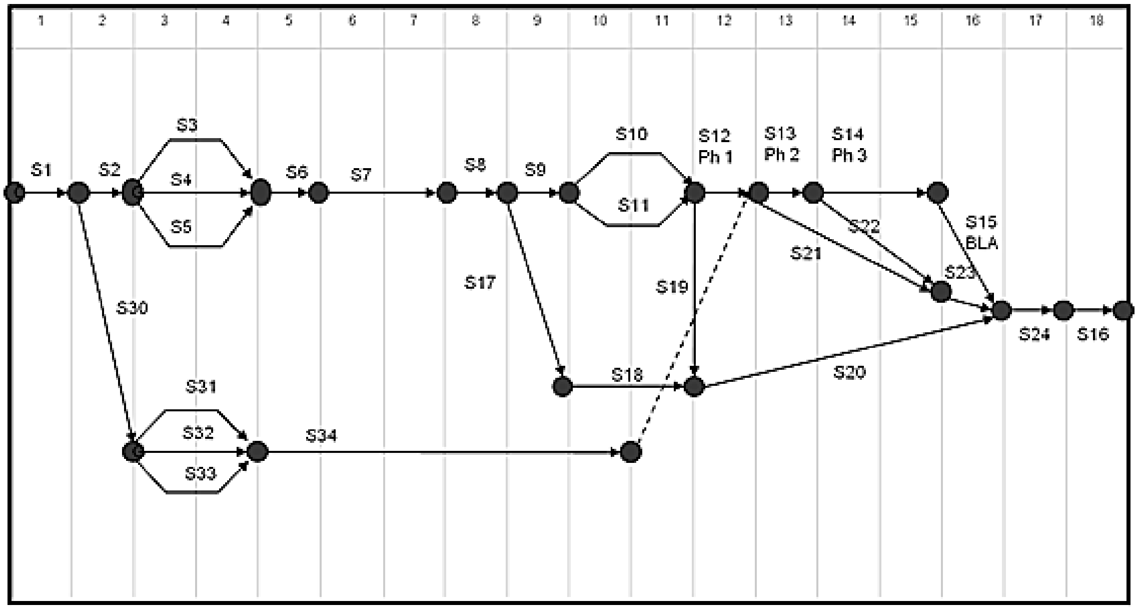


Figure 7-8: Demo 2 – Public and Private product model

Instead, it can only be surmised by any effects it may have on observable "S" segments.

The model has 82 segments in total, each of which translate into a pathway segment controller (Figure 7-6). In total, this model has 410 Places and 328 Transitions.

Demo 2 model

In the second demo, the scenario was somewhat similar. A company has two potential products, one publically observable (Product A), and another not (Product B). Each follows the same pathway structure, shown in Figure 7-8, but with different resource requirements. The development of the products run independently from one another, except for the allocation of resources which the management can shift between the projects. In this scenario, the private product, if it exists, starts seven months later than the public product.

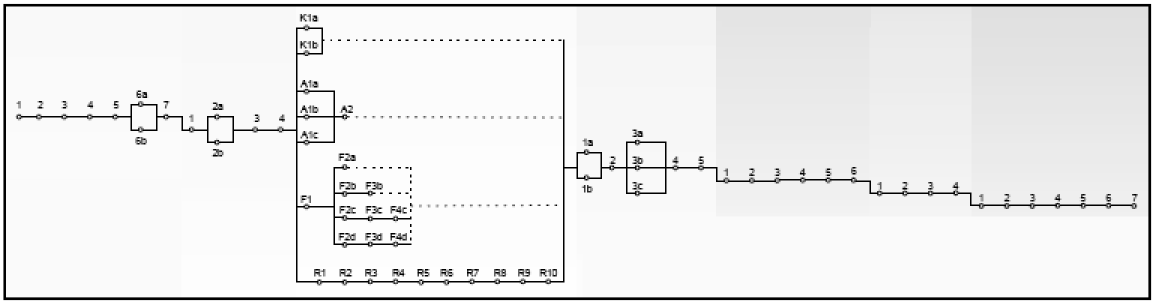


Figure 7-9: Demo 3 - Two public, one private product model

Hypothesis 1 speculates that the company is only producing Product A and all resources are allocated to it. Hypothesis 2 believes that the company is producing Product A and Product B, and resources have to be split between the two.

The complete model for each product has 29 segments, translating into 145 Places and 116 Transitions.

Demo 3 model

The third demo significantly extended the scenario in Demo 2. There are three potential products a company may be making: Products A, B, and C (the Product C pathway is shown in Figure 7-9). They each have seven stages, running from Hypothesis, to Production, to Testing. The pathways share structure for each stage except for the third stage, where Products A and B have mutually exclusive segments, and Product C has a combination of the segments of Products A and B. In addition, Product B has a few dependencies from Product A, and Product C has a few dependencies from both Product A and Product B. If the company is producing Products A or B, it will be observable. Product C, if produced, will not be directly observable.

Once again, there are two hypotheses. Hypothesis 1: the company only produces Products A and B. Hypothesis 2: the company produces all three products.

Individually, the Product A model has 67 segments, for a total of 335 Places and 268 Transitions. Product B has 56 segments, for a total of 283 Places and 224 Transitions. Finally, Product C has 79 segments, for a total of 397 Places and 316 Transitions. In sum, demo 3 required analyzing 1015 Places and 808 Transitions, which is the largest model set we tested on our event reasoning framework.

Variations

Successive demos increased the complexity of the models and analysis system.

- Demo 1: There was one model with two pathways. It was an early demo and our system was not required to produce distributions over the degree of completion outputs.
- Demo 2: There were two models with one pathway each. A distribution stub for each the degree of completion was produced and output for each segment at each time step.
- Demo 3: There were three, large models, with dependencies between each other. Our system computed full distributions (as described in Section 7.3.4) of the degree of completion output.

7.5.2 Results

The task in each of the three demos was to disambiguate between two hypotheses being considered. To do so, the probes used in each case had to be able to provoke a different response under the two hypothetical situations. As discussed in Section 7.1.2, this can be accomplished through actions that result in delays or changes of rate in pathway segment progress. In Demos 1 and 2, we were successful in analyzing the

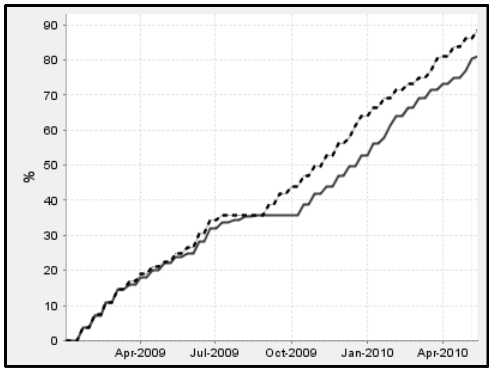


Figure 7-11: Hypothesis 1 (Prod A)

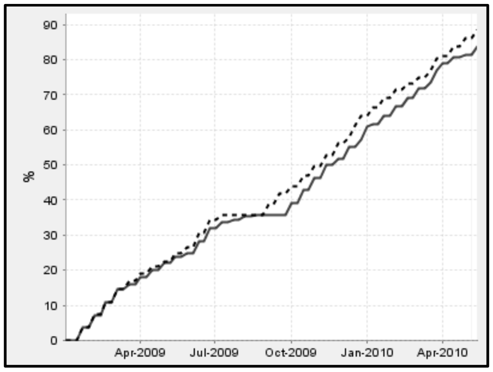


Figure 7-10: Hypothesis 2 (Prod A & B)

effects of probes that could provoke different responses under the two hypotheses. Demo 3 is ongoing work.

Demo 1

Figure 7-11 and Figure 7-10 show one result using the Demo 1 model. (In the figures, dotted lines are the baseline, unprobed scenario; solid lines are the probed scenario under the hypothesis labeled for the figure.) Hypothesis 1 imagines the Company producing only Product A, the production of which is partially observable. Hypothesis 2 imagines the Company producing both Product A and a not-observable Product B.

To disambiguate between the two models, we used a complex probe that combined two actions, spread out over a short period of time. The first action was to inhibit a subset of the workers needed to produce Product A from going to work; this happened early in the production cycle. The rate change not only threatened progress on the pathway segment that used those workers, it also threatened to delay every segment dependent on the outcome of that segment. Without additional personnel, nothing could be done to improve the situation. This was the case under Hypothesis 1.

The second action, that came soon after the first, would only affect the production of Product B (if it were being produced), slowing down one of its resources and leaving idle the personnel who would normally work on the resource. Under Hypothesis 2, the Company *would* be attempting to produce Product B and thus would now have idle workers after the second action of the probe. The Company, threatened by delays to Product A production, would reassign the idle workers from Product B to Product A, speeding the production of Product A back up. While nothing could be observed about Product B, affects on its production could manifest themselves in observable segments for Product A. This can be seen in Figure 7-11 and Figure 7-10 which show the overall progress of the pathway over time under the two hypotheses. Note that without the probe, entropy for the hypotheses was high (the dotted lines were identical, so classification of a real pathway would be impossible). With the probe, though, there is separation, entropy is lower, and classification will be easier.

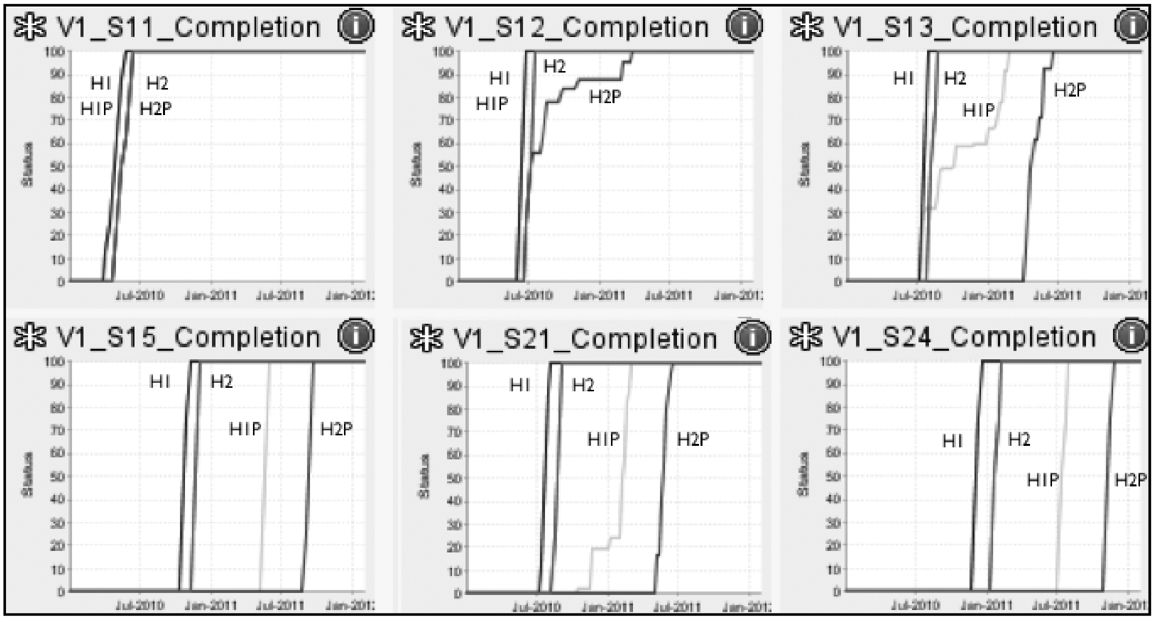


Figure 7-12: Demo 2 probe effects

Demo 2

Demo 2 combines finer grain distinctions from each of the individual pathway segments into an estimate of the expected behavior of each hypothesis under probed conditions. Figure 7-12 shows the completion progress over time of six segments of the model's 29 under four separate conditions (overlaid): Hypothesis 1 (H1), Hypothesis 2 (H2), Hypothesis 1 – Probed (H1P), and Hypothesis 2 - Probed (H2P). As can be seen, the probe starts affecting the pathway after Segment 11 (V1_S11) ; during Segment 11, under the four conditions, the behavior of the system is nearly identical, with H1 and H1P overlapping, and H2 and H2P overlapping. In Segment 12 (following Segment 11), greater discrepancies emerge between the behavior of H2P and the other cases. This discrepancy expands in Segment 13. By Segment 15, 21, and 24, the probe effect has materialized as a significant relative delay under the H2P case compared to the H1P case. This demonstration shows the exact time series effects that will distinguish Hypothesis 1 from Hypothesis 2 in the probed scenario, lowering the entropy of the system from the unprobed scenario.

Demo 3

Demo 3 integrates a stochastic search algorithm for learning optimal probes (briefly mentioned in Section 7.1.2) into the PCLASS system. This is done in the context of a multi-university team effort, where the mutual information measure (between model and probe, as described in Section 3.6.3) is integrated into a stochastic search algorithm that incrementally changes probe characteristics to better separate the pathway hypotheses in our demonstration domain. The pathway models for this demonstration are complete. The candidate probe search algorithm is currently being integrated at the time of this publication. Results are thus ongoing.

8 Conclusions and Future Work

In this dissertation, we attempted to demonstrate that computer systems need to apply detailed, dynamic models of events when attempting to automatically answer questions about complex events. To back up this claim, we 1) chose a representative set of event-related question types (a class of questions previously ignored in question answering systems because of their difficulty); 2) described an event model that has the features necessary to infer information targeted by those question types; 3) provided a set of tractable solutions that makes the models accessible for humans and computers to interface with; and 4) demonstrated end-to-end system implementations that utilize the features of the model to answer questions of our target types.

Specifically, we focused on five question types, each related to the inherent property of events to evolve over time based on their structure and their state and resource context. The modeling and inference framework that we created had an ability to simulate and reason about the events underlying these question types due to its 1) capturing contingent causal and temporal relations between events; 2) modeling contextual information about events; 3) handling uncertain and partially ordered trajectories; 4) representing sequentiality and concurrency; and 5) supporting asynchronous control. By exploiting the frame semantic structure of events, our model also showed a means of interfacing between event models and language. These features were combined when we showed: 1) analyzing the relations of questions in the context of expressive event models improving Answer Selection; 2) simulation and analysis of models providing inferences about missing information targeted by questions in our Question Answering task; and 3) dynamic simulation of pathways providing the means to do Pathway Classification and probe design. As a whole, these systems covered all five question types.

The results of our demonstrations were promising. However, in the case of Answer Selection and Question Answering, the language processing systems we relied on to setup the question and data for our system faced many long-standing hurdles in NLP research that have yet to be solved (Section 8.3). Without those solutions, the test sets we used to demonstrate our approach remained relatively small. The latest research on pathway classification does not require language input and is thus able to access larger test sets. Initial results have been successful, and work is ongoing (Section 7.5). We detail additional ongoing and future work, next.

8.1 The next generation of event modeling and reasoning

Within our research group, Leon Barrett is developing an extension of X-nets, called Coordinated Probabilistic Relational Models (CPRMs). CPRMs not only model the dynamic structure of events, but also the uncertainty in belief states about events. CPRMs will be able to replace X-nets in systems such as our Question Answering system of Chapter 6.

Currently, the QA system is dependent on external evidence supplied by an Information Retrieval system. The system makes an implicit assumption: if evidence about a state is not available, that state does not exist. The system can infer otherwise, but only if there is enough indirect evidence to show a causal chain to that state. Hence, the burden of proof is on the case for the existence of the state. With CPRMs, we can better integrate beliefs about the values of unknown states, where the beliefs are shaped by the evidence that is available. These beliefs can also be updated during simulation, based on the incremental evolution of the event. This will avoid the erroneous implicit assumption mentioned above and increase the amount of information the model can use to infer an answer.

Separately, in situations where multiple conflicting sources of evidence are available with varying levels of trustworthiness, the CPRM can integrate the data into a single estimate of the information. (Currently, our system uses the value from the first piece of evidence returned.)

The tradeoff is the increased complication in model design. However, the payoff of access to consistent data we believe would be significant. We look forward to exploring this in greater detail in the future.

8.2 Additional event-related question types

Counterfactuals

In our Question Answering work (Section 3.5, Chapter 6), we attempt to answer *Hypothetical* questions, but only the most basic kind and using only the most basic solution: those that hypothesize new positive evidence about a condition or resource. In Section 6.7.1, we describe an example of answering a *Hypothetical/Projection* question: "If Israel wished to conquer Lebanon, would Israel destroy Lebanon's airports?" The question hypothesizes "Israel wishes to conquer Lebanon", a condition that can be *added* to the model, and then asks a *Projection* question of that hypothesized state ("would Israel destroy Lebanon's airports?"). We would like to also be able to answer a more difficult hypothetical question, *Counterfactuals*. *Counterfactual* questions hypothesize a state that runs counter to the known facts about the scenario. It is challenging because, beyond the change to the condition or resource explicitly mentioned in the question, additional interventions into the model

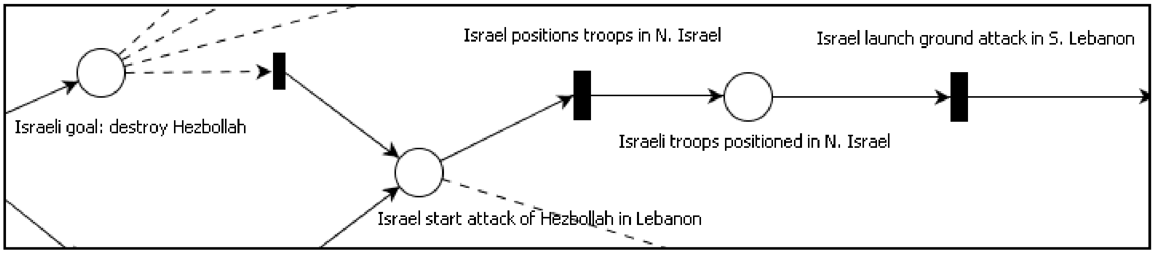

Israeli goal: destroy Hezbollah
Israel positions troops in N. Israel
Israel launch ground attack in S. Lebanon
Israeli troops positioned in N. Israel
Israel start attack of Hezbollah in Lebanon

Figure 8-1: Issues in models for counterfactuals

may be necessary to remove any historical influence of the now-changed condition or resource.

Take for example Figure 8-1 from our Lebanon War model, along with the question, "if Israel did not wish to destroy Hezbollah, would Israel launch a ground attack on south Lebanon?" If we knew in the baseline, real world case (no hypothesis) that Israel had troops stationed in northern Israel (which leads to a ground attack on southern Lebanon), should that remain a valid fact in the counterfactual scenario? It may have only been a fact because it resulted from a state that is now assumed untrue (Israeli goal to destroy Hezbollah).

A comprehensive computational treatment of counterfactuals culling data from natural language, social sciences, and from behavioral psychology is currently being completed (Narayanan *in preparation*). Details are outside the scope of this dissertation.

State Steering

In our Pathway Classification work (Section 3.6, Chapter 7), we described the use of probes to reveal information about an unknown pathway that might allow us to better disambiguate between hypotheses about the pathway (i.e. we were attempting to solve the *Hypothesis Disambiguation* question type). We do not, though, always seek to passively analyze a pathway. In some cases, we would like to be able to change the behavior of a pathway in accordance with a defined policy (i.e. we wish to *steer the* *system to a desired state*). With real world tests of *State Steering* quite possibly being expensive and maybe irreversible, we see an opportunity for our event reasoning framework to answer a different question type. We wish to extend our current pathway work to test whether we can use probes to steer the trajectory of a particular pathway towards a desirable outcome without reaching any undesirable outcomes. The relatively easy case will be for outcomes defined using out observable segment outputs. The more difficult case will be for outcomes defined using a mix of observable and hidden segment outputs.

8.3 Language processing support for Question Answering

The Natural Language Processing barriers are significant in the Question Answering task. To produce results, we made a number of idealistic assumptions about the form of the data our system would receive. Specifically, we expected gold-standard quality frame parses and entity tagging, with entity tags derived from fine-grained, hierarchical domain ontologies. With it, we achieved strong results. Unfortunately, it is not realistic at this time to have on-the-fly access to such high-quality language analysis. To move towards a fully automated system for Question Answering requires certain natural language processing support, which we briefly describe below.

Frame analysis

Currently, there are no good off the shelf solutions for frame parsing text. The systems by University of Texas, Dallas (Harabagiu, Bejan et al. 2005) and Saarbrucken University (Erk and Pado 2006) and the work by Gildea and Jurafsky (2002) and by Giuglea and Moschitti (2006), demonstrate the great strides made in the task in the last few years. Still, 65% precision and 61% recall, as Gildea and Jurafsky achieved, is not sufficient for finding the data needed to execute inference tasks with our event reasoning framework. We would like to explore improvements in this space.

Furthermore, FrameNet annotations recognize "null instantiated" frame elements (Ruppenhofer, Ellsworth et al. 2006), that the aforementioned systems do not attempt to fill. Null instantiations are conceptually salient frame elements (FEs) that are implied in the sentence annotated, but do not appear. Some null instantiated FEs exist in linguistic or discourse context (e.g. across sentence boundaries), but require additional analysis to fill (akin to anaphora resolution). While a full solution may not be easily achievable, some low hanging fruit may exist that can be exploited to decrease the effective number of null instantiations in frame parses, providing more relational information for our event reasoning system to use. We would like to explore this possibility as well.

Embodied Construction Grammar

Recent work in our research group has led to the development of a rich semantic representation of language that outstrips frames in its representational power. As an alternative to frame analysis, we believe using Embodied Construction Grammar (ECG: Bergen and Chang 2002) to analyze the semantic content of questions and passages may significantly increase our Question Answering system's processing power. New ECG analysis tools and demonstrations show the potential for of this approach (Bryant 2008; Mok 2008). Unfortunately, at this time ECG coverage is limited, making it difficult to analyze full domains that we may wish to target. If coverage is extended relatively broadly, we would like to integrate ECG-based descriptions of events into our event ontology and use ECG annotated questions and passages to further test our QA system. Our group has preliminary work on converting FrameNet frames into ECG representations (Chang, Narayanan et al. 2002), which could provide the semantic basis for a significant ECG grammar.

Entity analysis and relation uses

Quality, fine-grained entity extraction is critical to our system's performance (see Section 4.1.3). As such, we are experimenting with our own in-house tool that will link concepts found in text to the WordNet graph. We would like to eventually connect our ontologies to the SUMO/MILO ontologies, which have additional logical inference features (Niles and Pease 2001). In addition, we would like to explore and integrate solutions for automatically extracting fine-grain ontologies targeted to particular domains, linking these to SUMO/MILO, if possible. Each of these steps we believe would provide greater coverage for specifying constraints in event models using entity-restricted frames, providing higher precision and greater recall in linking to relevant questions and finding relevant data.

Entity references are frequently made using metonymies (e.g. "White House" to refer to the US President or his administration). We tested encoding some of these relationships directing in our domain ontology, explicitly specifying a "representative-of" relation between entities with a metonymic relation, and it helped in a few cases. We would like to further explore this and other solutions to this common problem.

Additional language cues

Natural language provides additional valuable cues required for accurate event inference, but not provided in frame or entity analysis. Specifically, we would like to explore solutions for automatically extracting polarity (i.e. negation) and aspect cues from sentences, allowing for more precise intervention on event models (Section 4.4.2) and more accurate placement of control tokens, respectively.

Another significant issue in analyzing events is resolving discrepancies between multiple pieces of evidence that refer to different stages in the evolution of an event. Blindly combining evidence such as this can produce partial markings that are potentially mutually exclusive from one another, producing inaccurate inferences (e.g. evidence that 'Joe has money' and 'Joe buys a car' – does Joe having money refer to the state after Joe bought the car or before? Does he have the money now?). To accurately infer a state at a particular time requires data to be tagged with the time it refers to. (In addition, as we demonstrated in our Pathway Classification task (Chapter 7), precisely timed input to our event reasoning system can enable inferences dependent on discrepancies in the timing of certain model output.) We would like to explore the use of TimeML-based systems for tagging and querying temporal information from text (e.g. Evita system: Sauri, Knippen et al. 2005).

Question analysis

In Section 4.2, we provided guidelines for the question analysis required for our applications. For our demonstrations, we frequently relied on hand-generated question classification. We would like to look at automated solutions for tagging questions with their question type.

In addition, event questions using qualifiers may require translation. A question, "can Joe buy the car *for $5000 or less*?", can be translated into a basic *Hypothetical*: "if Joe has $5000, can he buy the car?" We would also like to pursue solutions to this problem.

Information Retrieval

Information Retrieval technology used in supplying our Question Answering system with data also needs improvement. Currently, when our system sends out a list of entity-restricted frames to the front-end system for the purpose of retrieving relevant passages, the frame-based queries are converted into keyword-based IR queries, losing the relational content in the queries. We would like to look at solutions for indexing passage data using entity-bound frames. As this relies on more robust automated frame annotation, we see this as a long-term goal. Recent commercial work using relational information, though, like the Powerset search engine (http://www.powerset.com), gives us hope that a nearer-term solution may be forthcoming.

Event Ontology

In Section 4.6, we discuss a few preliminary efforts to enable model designers to populate an event ontology. To facilitate their efforts, we would like to use techniques in event extraction to automatically learn event primitives (single actions and constituent parameters). This would provide building blocks from which larger domain scenarios can be assembled by a designer. We would like to extend the work of (Bethard 2007; Chambers, Wang et al. 2007; Chambers and Jurafsky 2008) and others to encompass the larger range of event parameters described in Section 3.3.

8.4 Closing thoughts

Our work has taken place in a research domain that is in its early adolescence. We have demonstrated a number of valuable techniques for answering questions about complex events, showing their viability under engineered environments. In addition, we have shown many promising avenues of future research to make these solutions more robust and to extend their capabilities. We hope our work has set the agenda for future enhancements to help the field grow into its full maturity.

Bibliography

Agre, P. E. and D. Chapman (1987). Pengi: an implementation of a theory of activity. Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87), Morgan Kaufmann**:** 268-272.

Ajmone Marsan, M., G. Balbo, et al. (1995). Modelling with Generalized Stochastic Petri Nets, J. Wiley.

Akharware, N. (2005). PIPE2: Platform Independent Petri Net Editor.

Aouladomar, F. (2005). Answering Procedural Questions. Knowledge and Reasoning for Question Answering Workshop, IJCAI05, Edinburgh, Scotland.

Arkin, R. C. (1990). "Integrated Behavioral, Percentual, and World Knowledge in Reactive Navigation." Robotics and Autonomous Systems **6**: 105-122.

Astrom, K. J. (1965). "Optimal control of Markov decision processes with incomplete state estimation." J1 Math1 Anal1 Applic. **10**: 174-205.

Barker, K., V. Chaudhri, et al. (2004). A Question-Answering System for AP Chemistry: Assessing KR&R Technologies. 9th International Conf on Knowledge Representation and Reasoning (KR'04), Whistler, British Columbia, Canada, AAAI Press.

Barker, K., B. Porter, et al. (2001). A Library of Generic Concepts for Composting Knowledge Bases. 1st Int Conf on Knowledge Capture (K-Cap'01), Victoria, British Columbia, Canada.

Bause, F. and P. Kritzinger (1996). Stochastic Petri Nets: An Introduction to the Theory, Vieweg Verlag.

Bergen, B. K. and N. C. Chang (2002). Simulation-Based Language Understanding in Embodied Construction Grammar. Construction Grammar(s): Cognitive and Cross-language dimensions, John Benjamins.

Bethard, S. J. (2007). Finding event, temporal and causal structure in text: A machine learning approach. Computer Science. Boulder, CO, University of Colorado at Boulder. **Ph.D.**

Brooks, R. A. (1986). "A robust layered control system for a mobile robot." IEEE Journal of Robotics and Automation **2**: 14-23.

Bryant, J. (2008). Best-Fit Constructional Analysis. Computer Science. Berkeley, CA, University of California, Berkeley. **Ph.D.**

Busi, N. (2002). "Analysis issues in Petri nets with inhibitor arcs." Theoretical Computer Science **275**: 127-177.

Chambers, N. and D. Jurafsky (2008). Unsupervised Learning of Narrative Event Chains 46th Annual Meeting of the Association for Computational Linguistics (ACL-08). Ohio, USA.

Chambers, N., S. Wang, et al. (2007). Classifying Temporal Relations Between Events. 45th Annual Meeting of the Association for Computational Linguistics (ACL-07). Prague, Czech Republic.

Chang, N., S. Narayanan, et al. (2002). Putting Frames in Perspective. Proc. Nineteenth International Conference on Computational Linguistics (COLING 2002).

Chinchor, N. (1998). Overview of MUC-7. Seventh Message Understanding Conference (MUC-7).

Doddington, G., A. Mitchell, et al. (2004). The Automatic Content Extration (ACE) Program - Tasks, Data, and Evaluation. LREC 2004, Lisbon, Portugal.

Erk, K. and S. Pado (2006). Shalmaneser - a flexible toolbox for semantic role assignment. LREC 2006, Genoa, Italy.

Fellebaum, C. (1998). WordNet: An Electronic Database, MIT Press.

Fikes, R. E. and N. J. Nilsson (1971). "STRIPS: a new approach to the application of theorem proving to problem solving." Artificial Intelligence **2**(3-4): 189-208.

Fillmore, C. (1976). "Frame semantics and the nature of language." Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech **280**: 20-32.

Fillmore, C., C. R. Johnson, et al. (2003). "Background to FrameNet." International Journal of Lexicography **16**(3).

Fillmore, C. J. (1982). Frame Semantics. Linguistics in the Morning Calm. Seoul, Hanshin: 111-38.

Fliedner, G. (2004). Towards Using FrameNet for Question Answering. LREC 2004 Workshop "Building Lexical Resources from Semantically Annotated Corpora". Lisbon, Portugal: 61-65.

Fliedner, G. (2005). A Generalized Similarity Measure for Question Answering. 10th International Conference on Applications of Natural Language to Information Systems (NLDB), Alicante, Spain, Springer.

Fliedner, G. (2006). Towards Natural Interactive Question Answering. 5th International Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy.

FrameNet. (2008). "FrameNet online frame database." 1.3. from http://framenet.icsi.berkeley.edu.

Gelfond, M. and V. Lifschitz (1993). "Representing Action and Change by Logic Programs." Journal of Logic Programming **17**: 301-322.

Gildea, D. and D. Jurafsky (2002). "Automatic Labeling of Semantic Roles." Computational Linguistics **28**(3): 245-288.

Giuglea, A.-M. and A. Moschitti (2006). Shallow Semantic Parsing Based on FrameNet, VerbNet and PropBank. European Conference on Artificial Intelligence, Riva del Garda, Italy.

Harabagiu, S., C. Bejan, et al. (2005). Shallow Semantics for Relation Extraction. Nineteenth Internatioal Joint Conference on Artificial Intelligence, Edinburgh, Scotland.

ISO, I. O. f. S. (2008). Software and system engineering -- High-level Petri nets. Part 2: Transfer Format. **ISO/IEC FCD 15909-2**.

Kingsbury, P. and M. Palmer (2002). From Treebank to PropBank. Proc. 3rd International Conference on Language Resources and Evaluation (LREC-2002).

Klein, D., J. Smarr, et al. (2003). Named Entity Recognitino with Character-Level Models. Seventh Conference on Natural Language Learning at HLT-NAACL 2003, Edmonton, Canada.

Krishnan, V. and C. D. Manning (2006). An Effective Two-Stage Model for Exploiting Non-Local Dependencies in Named Entity Recognition. 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia.

Lachiche, N. and P. A. Flach (2003). Improving accuracy and cost of two-class and multi-class probabilistic classifiers under ROC curves. 20th International Conference on Machine Learning (ICML '03), Washington, DC, AAAI Press.

Lehnert, W. G. (1978). The Process of Question Answering: a computer simulation of cognition, Lawrence Erlbaum Associates.

Lenat, D. B. and R. V. Guha (1990). "Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project."

Lifschitz, V. (1989). Between circumscription and autoepistemic logic. Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann: 235-244.

Martin, D. L., M. Burstein, et al. (2004). "OWL-S: Semantic Markup for Web Services." from http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

McCarthy, J. (1958). Programs with Common Sense. Proceedings of the Symposium on Mechanisation of Thought Processes, Her Majesty's Stationery Office. **1:** 77-84.

McCarthy, J. and P. J. Hayes (1969). Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligenc1 4, Edinburgh University Press: 463-502.

McGuinness, D. L., F. Van Harmelen, et al. (2004). "OWL Web Ontology Language Overview." from http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Michie, D., D. J. Spiegelhalter, et al. (1994). Machine Learning, Neural and Statistical Classification, Ellis Horwood.

Moens, M. and M. Steedman (1988). "Temporal Ontology and Temporal Reference." Computational Linguistics **14**(2): 15-27.

Mok, E. (2008). Contextual Bootstrapping for Grammar Learning. Computer Science. Berkeley, CA, University of California, Berkeley. **Ph.D.**

Mossman, D. (1999). "Three-way ROCs." Medical Decision Making **19**: 78-89.

Murata, T. (1989). Petri Nets: Properties, Analysis, and Applications. Proc. IEEE-89. **77:** 541-576.

Narayanan, S. (1997). Knowledge-based Action Representations for Metaphor and Aspect (KARMA), Computer Science Division, University of California at Berkeley.

Narayanan, S. (1997). Talking the Talk is Like Walking the Walk: A Computational Model of Verbal Aspect. Proc. 19th Cognitive Science Society Conference.

Narayanan, S. (1999). Reasoning about Actions in Narrative Understanding. Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Morgan Kaufmann Press.

Narayanan, S. and S. McIllraith (2002). Simulation, Verification, and Automated Composition of Web Services. Proc. Eleventh International World Wide Web Conference (WWW2002).

Niles, I. and A. Pease (2001). Towards a Standard Upper Ontology. 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Ogunquit, Maine.

Nilsson, N. J. (1984). Shakey the robot, SRI International.

Palmer, M., D. Gildea, et al. (2005). "The Proposition Bank: A Corpus Annotated with Semantic Roles." Computational Linguistics **31**(1): 71-106.

Pasca, M. and S. Harabagiu (2001). High Performance Question/Answering. The 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), New Orleans, LA.

Pearl, J. (2001). Causality: Models, Reasoning, and Inference, Cambridge University Press.

Ramakrishnan, G., S. Chakrabarti, et al. (2004). Is question answering an acquired skill? 13th International Conference on World Wide Web (WWW '04), New York, NY, ACM.

Rickel, J. and B. Porter (1994). Automated modeling for answering prediction questions: Selecting the time scale and system boundary. Twelfth National Conference on Artificial Intelligence.

Rosenschein, S. J. (1985). "Formal theories of knowledge in AI and robotics." New Generation Computing **3**(4): 345-357.

Ruppenhofer, J., M. Ellsworth, et al. (2006). FrameNet II: Extended Theory and Practice.

Salton, G., A. Wong, et al. (1975). "A vector space model for automatic indexing." Communications of the ACM **18**(11): 613-620.

Sauri, R., R. Knippen, et al. (2005). Evita: A Robust Event Recognizer for QA Systems. Human Language Technology Conference / Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, B.C., Canada.

Sekine, S. and H. Isahara (2000). IREX: IR and IE Evaluation project in Japanese. LREC-2000, Athens, Greece.

Steedman, M. (1996). Temporality. Handbook of Logic and Language. North Holland, Elsevier.

Swets, J. A. (1988). "Measuring the accuracy of diagnostic systems." Science **240**: 1285-93.

Weiss, S. M. and C. A. Kulikowski (1991). "Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems."

Yin, L. (2004). Topic Analysis and Answering Procedural Questions. Technical Report Series, Information Technology Research Institute, University of Brighton.